Operating Instructions

PA-CONTROL

Programmer's Manual

from version 5.30

Edition: January 2016
Art.-No.: 1086949

IEF-Werner GmbH
Wendelhofstraße 6
78120 Furtwangen - Germany
Phone: 0049-7723-925-0
Fax: 0049-7723-925-100
www.IEF-Werner.de

Modification history:

Document code	Date	Action / Modification
MAN_EN_1086949_PAC_ProgrammersManual_R1a.doc	November 2009	Creation of the English document R1a
MAN_EN_1086949_PAC_ProgrammersManual_R1b.doc	June 2010	System-Flags, N/R-Registers added
MAN_EN_1086949_PAC_ProgrammersManual_R1c.doc	August 2015	Document has been updated according to the German document: MAN_DE_1071142_PAC_Programmierhandbuch_ R5m.doc and formally revised.
MAN_EN_1086949_PAC_ProgrammersManual_R1ddoc	January 2016	G19x command added (see chapter 1.16.2 on page 224)

The use of trademarks and trade names does not affect their legal status. These texts and examples were written with the utmost care. Nonetheless, errors cannot be ruled out. Special application cases were not considered for the examples given. The use of the examples printed requires close check on, for instance, the traverse distance or an acceleration value.

IEF-WERNER GmbH is neither legally responsible nor liable in any way for missing or erroneous information and its consequences.
IEF-Werner reserves the right to change or improve software or hardware or parts thereof, including the printed matter supplied or parts thereof, without prior notification.

All rights to copy (to photomechanical reproduction), even excerpts, are expressly reserved by IEF-WERNER GmbH.

We are always grateful for suggestions for improvements and notices of errors.

© January 2016 IEF-WERNER GmbH

Contents

1 Commands of the PA-CONTROL Family 11

1.1 Information about software releases 11

1.2 Programming elements 12

1.2.1 Inputs 12

1.2.2 Outputs 12

1.2.3 Flags 12

1.2.4 Times 12

1.2.5 Real number registers 12

1.2.6 Integer register 12

1.2.7 Strings 12

1.2.8 Axes 12

1.2.9 Label 12

1.2.10 Comment 13

1.2.11 Direct addressing 13

1.2.12 Indirect addressing 13

1.2.13 Display 13

1.2.14 Counter 13

1.2.15 Serial interface 13

1.2.16 AD 13

1.2.17 DA 13

1.2.18 Temperature controller 13

1.2.19 List of system flags 14

1.2.20 List of system N registers 23

1.2.21 List of system R registers 27

1.3 Overview of the command set 29

1.3.1 Program execution (automatic) 29

1.3.1.1 Sequence 29

1.3.1.2 Jumps and Branches 29

1.3.1.3 Call subroutines 30

1.3.1.4 Loops 30

1.3.1.5 Parallel runs 30

1.3.1.6 Error management 30

1.3.2 Commands for Axis 31

1.3.2.1 General Remarks 31

1.3.2.2 Modes of an axis 31

1.3.2.3 Move 31

1.3.2.4 Position and Parameter 31

1.3.2.5 Limit switches 31

1.3.2.6 Wait 31

1.3.2.7 Wait until po Encoder and following error 32

1.3.2.8 Interpolation 32

1.3.2.9 Special PA-CONTROL Commands 32

1.3.3 Digital Inputs, digital Outputs, Flags and Registers 33

1.3.3.1 Digital inputs 33

1.3.3.2 Digital outputs 33

1.3.3.3 Flags 33
1.3.3.4 Registers 33
1.3.4 Dwell Timer, monitoring time, date, time of the day 34
1.3.4.1 Wait 34
1.3.4.2 Monitoring time 34
1.3.4.3 Date, time of the day 34
1.3.5 Set time Mathematics 34
1.3.5.1 Addressing 34
1.3.5.2 Mathematic commands 34
1.3.5.3 Compare functions 35
1.3.5.4 Logical operations 35
1.3.5.5 Register operation 35
1.3.6 Communication 37
1.3.6.1 Serial ports, display and keyboard 37
1.3.6.2 CANopen bus 37
1.3.6.3 MODBUS 38
1.3.6.4 String commands 38
1.3.7 Analog / digital 38
1.3.8 File commands 38
1.3.9 Counter 39
1.3.10 Temperature control 39
1.3.11 Command groups of PA-CONTROL 39
1.3.11.1 Commands of the group G16x 39
1.3.11.2 Commands of the group G17x 39
1.3.11.3 Commands of the G19x-group 39
1.3.11.4 Commands of the group G2xx 39
1.3.11.5 Commands of the group G4xx 40
1.3.11.6 Commands of the group G5xx 40
1.3.11.7 Commands of the group G6xx 40
1.3.11.8 Commands of the group G7xx 41
1.4 Programming tips 42
1.4.1 The run interpreter 42
1.4.2 System functions 43
1.4.3 Control of the automatic mode via external inputs 45
1.4.3.1 Start of automatic operation with Cancel 45
1.4.3.2 Start of automatic operation with STOP and START AFTER STOP 46
1.4.3.3 PA-CONTROL is in automatic mode and a MALFUNCTION occurs 47
1.4.3.4 PA-CONTROL is in the initial position and a malfunction is present 48
1.4.4 Program structure 49
1.5 Explanations 50
1.6 Organisation of Program 51
1.6.1 Program sequence 51
1.6.1.1 BREAK - Cancel automatic mode 51
1.6.1.2 START - Start automatic run 52
1.6.1.3 STOP - Stop automatic run 53
1.6.1.4 Ni:=LINE - Request for the current line number of a program 54
1.6.1.5 End - ending 54
1.6.2 Jump and branches 55

1.6.2.1 JMP - General form of the unconditional jump 55
1.6.2.2 G21 - Conditional jump 56
1.6.2.3 Start of a program at a specified program line 57
1.6.2.4 CASE.JMP - Jump distributor 58
1.6.3 Call subroutines 60
1.6.3.1 SUB - Subroutine call 60
1.6.3.2 SUB.name - Standard subroutine call 61
1.6.3.3 SUB.S0 / SUB.Sn - Subroutine call, program name is in a string 63
1.6.3.4 SUB.Nn – Subroutine call, program name is in an integer register 64
1.6.3.5 G22 - Conditional Subroutine call 65
1.6.3.6 CASE.SUB - Subroutine distributor 66
1.6.4 Loops 68
1.6.4.1 INC - Loop with conditional jump 68
1.6.4.2 DEC - Loop with conditional jump 69
1.6.5 Parallel sequence 71
1.6.5.1 RUN - Start of a parallel run 71
1.6.5.2 SLEEP - Stopping a parallel run 73
1.6.5.3 CANCEL - Ending a parallel run 74
1.6.5.4 CASE.RUN - Starting parallel runs with CASE 75
1.6.5.5 CASE.SLEEP - Stopping of parallel runs with Case 77
1.6.5.6 CASE.CANCEL - Ending parallel runs with Case 79
1.6.5.7 PROGSTAT - Get program status 81
1.6.6 Error Management 82
1.6.6.1 ERROROFF / ERRORON 82
1.7 Commands for axis 84
1.7.1 General 84
1.7.1.1 The statuses of an axis 84
1.7.2 Modes 86
1.7.2.1 G140 - Change to DISABLED status 86
1.7.2.2 G141 - Change to OPERATIONAL status 87
1.7.2.3 OFF.An - Deactivate axis 88
1.7.2.4 ON.An - Activating axis 89
1.7.2.5 STOP.An - Interrupt readiness for travel for an axis 90
1.7.2.6 START.An - establish readiness for traversing for an axis 90
1.7.2.7 ABORT.An - Abort a stopped travel command 91
1.7.3 Move axis 92
1.7.3.1 G25 - Approach to reference point 92
1.7.3.2 A1 - Positioning of the axes 93
1.7.3.3 G90 - Absolute dimension system 94
1.7.3.4 G91 - Incremental dimension system 95
1.7.3.5 G100 - Specifying the acceleration 96
1.7.3.6 FAn - Traversing speed 97
1.7.3.7 G123 - Traverse as long as condition fulfilled 99
1.7.3.8 G123Q - Travel for as long as condition is satisfied 101
1.7.3.9 G150 - Traverse section with Start-Stop 102
1.7.3.10 RN:=FAN Request for the set or current speed 104
1.7.4 Position and Parameter 105
1.7.4.1 G26 - Set position to zero / set to position value 105

1.7.4.2 G29 - Set position to dimension 106
1.7.4.3 Rn:=An get axis position 108
1.7.5 Limit switches 109
1.7.5.1 G142 - Limit switch monitoring "OFF" [1] 109
1.7.5.2 G143 - Limit switch monitoring "ON" [1] 110
1.7.6 Encoder and following error 111
1.7.6.1 Ni:=PEAn / Ri:=PEAn - Get following error 111
1.7.6.2 ENC - Transfer encoder position [1] 112
1.7.6.3 Wait for position from the SSI-interface [4] 114
1.7.7 Parameter commands 115
1.7.7.1 PARAMETER.XX.Ai. - Write LV-servoTEC-Parameters [4] 115
1.7.7.2 Ri:=PARAMETER.XX.Ai - Reading LV-servoTEC-Parameters [4] 117
1.7.7.3 Ran.m:=Ri - Load register with axis parameters[3] 118
1.7.7.4 Ri:=Ran.m - Load axis parameter from a register 119
1.7.8 Interpolation 120
1.7.8.1 1.7.8.1 G144 - Switch LV-servoTEC axis to interpolation mode 120
1.7.8.2 G145 - Deactivate LV-servoTEC axis interpolation mode 121
1.7.8.3 G100.B.n - IPO path acceleration 122
1.7.8.4 FB:=n - IPO path speed 123
1.7.8.5 Defining IPO path speed, one axis running too 123
1.7.8.6 G01 - Linear interpolation with 4 from 16 axes [3] 124
1.7.8.7 Linear interpolation with following axis/axes 126
1.7.8.8 IPOEND – Abort linear interpolation 128
1.7.8.9 Rn:=FB - Get current path speed 129
1.7.8.10 Rn:=FBAn - Get the current speed of an axis during an interpolation 130
1.7.8.11 G16x - Operation of outputs during interpolation 131
1.7.8.12 G160 - Activate outputs before starting the interpolation 131
1.7.8.13 G161 / G162 - Activate outputs during interpolation 132
1.7.8.14 JMP-LINE-IPO.Ni - Start of an interpolation at a specified program line133
1.7.9 Motor current commands 134
1.7.9.1 G101 - Changing the motor current and limiting G101 [1, 4] 134
1.7.9.2 G101 - Current limitation changes in a PA-CONTROL-MP[1] 135
1.7.9.3 G101 – changing current limitations at a LV-servoTEC axis 136
1.7.9.4 G101 - Current limitation changes in a LV-servoTEC S2 137
1.8 Inputs, Outputs, Flags 138
1.8.1 Inputs 138
1.8.1.1 In.m - Waiting for logical status of input 138
1.8.2 Outputs 139
1.8.2.1 On.m - Waiting for logical status of output 139
1.8.2.2 On:=m - Set / reset output 140
1.8.3 Flags 141
1.8.3.1 Mn:=m - Set / reset flag 141
1.8.3.2 M!2:= - indirect or direct addressing of flags 141
1.8.3.3 Mn.m - Waiting for then logical status of a flag 143
1.8.3.4 > = < - Comparisons 144
1.8.3.5 Write flag number to an N register after comparison 145
1.9 Timer, Date and Time of day 146
1.9.1 Wait / Timer 146

1.9.1.1 Tn - Hold-up time 146
1.9.2 Date and time of day 147
1.9.2.1 DATE:= - Set the date from the user program 147
1.9.2.2 TIME:= - Set the time from the user program 148
1.9.2.3 Read-out of the real-time clock 149
1.10 Mathematics 150
1.10.1 Organisation of commands 150
1.10.1.1 Direct and indirect addressing 150
1.10.1.2 Assignments 150
1.10.2 Mathematic commands 151
1.10.2.1 Rn/Nn:= Load register 151
1.10.2.2 Load register with a binary value 152
1.10.2.3 Load register with a hexadecimal value 152
1.10.2.4 Rn:=RM+K - Addition 153
1.10.2.5 Rn:=Rm-K - Subtraction 154
1.10.2.6 Rn:=Rm*K - Multiplication 155
1.10.2.7 Rn:=RM/K - Division 156
1.10.2.8 Trigonometrical functions 157
1.10.2.9 SIN/ASIN – trigonometric functions 157
1.10.2.10 COS/ACOS - Cosine functions 158
1.10.2.11 TAN/ATAN - Tangent functions 159
1.10.2.12 SQRT - Root function 160
1.10.2.13 INT - Integer component 160
1.10.2.14 FRAC - Decimal component of a real number 161
1.10.2.15 ABS - Absolute value of a real number / integer 161
1.10.3 Compare 162
1.10.3.1 Compare - Comparison operations 162
1.10.3.2 Compare: complex examples 163
1.10.4 Logical operations 164
1.10.4.1 Introduction 164
1.10.4.2 LD/AND/OUT/NOT – Logical AND operation 168
1.10.4.3 LD/OR/OUT/NOT - Logical OR operation 169
1.10.4.4 SET/RES - Supplemental instructions for logic operations 170
1.10.4.5 XOR logical - operation (Exclusive-OR) 173
1.10.4.6 Multi-level logical AND operation 174
1.10.4.7 OR-LD - Multi-level logical OR operation 175
1.10.4.8 Complex logical operation 176
1.10.4.9 Bit-by-bit processing of N registers 177
1.10.5 Register operations 178
1.10.5.1 Ni.n - Waiting for the logical status of an N register 178
1.10.5.2 Nn:=SNn - Load content of a system N register into an integer register 179
1.10.5.3 Rn:=SRn - Load content of a system R register into a real number register r 180
1.10.5.4 Bit-by-bit processing of N registers 181
1.11 Communication 182
1.11.1 Communication by MODBUS 182
1.11.2 Reading values from a MODBUS participant 182
1.11.3 Writing a value to a MODBUS participant 183

1.11.4 Communication via control console 184
1.11.4.1 G11 - Switch display on / off 184
1.11.4.2 Nn:=CHN - Check whether character transfer in the background has been completed 185
1.11.5 String commands 186
1.11.5.1 Sn - Copying character strings 186
1.11.5.2 S0:=CHN - Copying character strings 186
1.11.5.3 POS - Search for character position in local character buffer S0 187
1.11.5.4 Copy - Convert characters from the local character buffer 188
1.11.5.5 Sn:=COPY - get substring 190
1.11.5.6 Si:=Sn+Sm - Merging strings 191
1.11.5.7 LENGTH - Get the length of a string 191
1.11.5.8 Create XOR checksum from a string 192
1.11.5.9 Sn:=Ni / Sn:=Ri - Write the content of a register (number) to a string 192
1.11.5.10 GET - Transfer the content of the local character buffer S0 to an integer register 193
1.11.5.11 GETI - Transfer the content of the local character buffer S0 to an integer register in INTEL format 194
1.11.5.12 PUT - Transfer the content of an N register to the local character buffer195
1.11.5.13 PUTI - Transfer the content of an N register in the INTEL format to the local character buffer S0 196
1.12 Analog / Digital 197
1.12.1 Analog/digital converter - general description 197
1.12.1.1 Get A-D values 197
1.12.2 Digital-to-analog converter 199
1.12.2.1 Dai - Output of DA values 200
1.13 File commands 201
1.13.1 Commands of the G17X-Group 201
1.13.1.1 G170 - Read character from PTX file 201
1.13.1.2 G171 - Write character to PTX file 202
1.13.1.3 G172 - Write line from PTX file to string (Sn) 203
1.13.1.4 G173 - Store string (Sn) in a line of a PTX file 204
1.13.1.5 STORE –Storage of current values in a program file 205
1.13.1.6 STOREEND / STORESTART - Deactivating / activating assignment with constants 207
1.13.1.7 CASE.STORE.N - Storage of values if content of N-register 208
1.14 Counter 210
1.14.1 General principles 210
1.14.2 Hardware 210
1.14.3 Commands 211
1.14.3.1 Initialising counters 211
1.14.3.2 Setting counters 212
1.14.3.3 Reading counters 212
1.14.3.4 Comparing counters 213
1.15 Temperature control 214
1.15.1 General principles 214
1.15.1.1 Parameters 214
1.15.2 Commands for temperature controls 215
1.15.2.1 Writing parameter value 215

1.15.2.2 Reading parameter value 216
1.16 Command groups of the PA-CONTROL 217
1.16.1 Commands of the G18x group 217
1.16.1.1 G18x - Acquisition of several values 217
1.16.1.2 G180 - A-D values synchronous to the axis movement 217
1.16.1.3 G181 - Enter A-D values in a defined time grid 219
1.16.1.4 G182 – A-D values in synch with counter value (CNT0) 220
1.16.1.5 G183 - Obtain A-D values in synch with counter value (CNT0) 222
1.16.2 Commands of the G19x Group 224
1.16.2.1 General statements to the commands of the G19x Group 224
1.16.2.2 G190 – Get the number of inserted bus terminals 225
1.16.2.3 G191 – Read the content of a register 226
1.16.2.4 G192 – Describes a register with a value 227
1.16.2.5 Example G19x commands 228
1.16.3 Commands of the G2xx group 229
1.16.3.1 Basic explanations of G2xx commands group 229
1.16.3.2 G210 - Activate the start positioning mode 230
1.16.3.3 G211 - Position-conditional jump 231
1.16.3.4 G212 - Position-conditioned subroutine call 233
1.16.3.5 G213 - Activate the standard positioning mode 234
1.16.3.6 G221 - Position-conditional jump (current position) 234
1.16.3.7 G222 - Position-conditional subroutine call (current pos.) 237
1.16.3.8 G230 - Wait until current position </> than value 239
1.16.3.9 G231 - Wait until the residual distance of current positioning is reached or undershot 240
1.16.4 Commands of the G4xx-Group 242
1.16.4.1 G4xx - Time monitoring commands 242
1.16.4.2 G421 - Time monitoring with a conditional jump 243
1.16.4.3 G422 - Time monitoring with conditional subroutine call 244
1.16.4.4 G423 - with conditional subroutine call (return in same line) 245
1.16.4.5 G401 - Reset of the time condition 246
1.16.5 Commands of the G5xx-Group 247
1.16.5.1 G5xx - Text and value output via the current data channel 247
1.16.5.2 G500 - Selection of the data channel / Initialisation of the ports 247
1.16.5.3 G501 - Clear the display 249
1.16.5.4 G502 - Clear up to end of line 250
1.16.5.5 G503 / G504 - Positioning the cursor 251
1.16.5.6 G510 - Text output 252
1.16.5.7 G511 - Text output 253
1.16.5.8 G512 - Output of control characters 254
1.16.5.9 G515 - Text output 255
1.16.5.10 G520 - Output of value 256
1.16.5.11 G521 - Output of value 258
1.16.5.12 G531 - Value transfer 260
1.16.5.13 G532 - Character transfer to the local character buffer S0 261
1.16.5.14 G533 - Character transfer in the background 262
1.16.5.15 G534 - Character transfer in the background 264
1.16.5.16 G540 - Check if a key has been actuated 265

1.16.5.17 G541 - Get a character from the keyboard 266
1.16.5.18 G542 - Entry of a value via the keyboard 267
1.16.6 Commands of the G6xx-Group 269
1.16.6.1 G6xx - Image to register / Image from register 269
1.16.6.2 G600 - Binary notation to outputs 270
1.16.6.3 G601 - BCD notation to outputs 271
1.16.6.4 602 - Binary notation to flags 272
1.16.6.5 G603 - Inputs in binary format to register 273
1.16.6.6 G604 - Flags in binary format to register 274
1.16.7 Commands of the G7xx-Group 275
1.16.7.1 General 275
1.16.7.2 Network management commands 276
1.16.7.3 G701 - Start Remote Node 276
1.16.7.4 G702 - Stop Remote Node 277
1.16.7.5 G703 - Reset Remote Node 277
1.16.7.6 G704 - Enter Pre-Operational State 278
1.16.7.7 G705 - Reset Communication 278
1.16.7.8 G711 - Functional monitoring of the devices 279
1.16.7.9 G72x - Commands for checking device errors 279
1.16.7.10 G721 / G722 - Check whether a device error message is present 280
1.16.7.11 Commands for processing service data objects 280
1.16.7.12 G730 - Read the content of a service data object (Initiate Domain Upload Rq) 281
1.16.7.13 G731 - Write the content of a service data object (Initiate Domain Upload Rq) 281
1.16.7.14 Commands for the processing of process data objects 282
1.16.7.15 G74x - Read process data 284
1.16.7.16 Send process data 285
1.16.7.17 Typical applications of command group G7xx 286

2 PA-CONTROL key code 288

3 INDEX 292

1 Commands of the PA-CONTROL Family

1.1 Information about software releases

G25

In Version 4.xx, the “G25.A0” command could be used to activate the axis for travel without referencing it.
In Version 5.00 and higher, this command is no longer available. Command G26 should be used instead.

G140 / G141

In Version 4.xx, the G140 / G141 commands were used to activate or deactivate MEASURING MODE.
in Version 5.00 and higher, if an axis has several status conditions, the MEASURING MODE status has been replaced by the DISABLED status. The function and usability of the G140 / G142 commands has therefore changed slightly.

1.2 Programming elements

Various elements are used for programming in PA-CONTROL.

1.2.1 Inputs

An input can be polled for its logical status *(*see chapter *1.8, page 138*). 2048 inputs are implemented.

1.2.2 Outputs

An output can be assigned with the logical status 0 or 1 (see chapter *1.8.2.2, page 140*). Its logical status can be polled for conditional jumps (see chapter *1.8.2.1, page 139*). There are 2048 outputs implemented.

1.2.3 Flags

A flag can be assigned with the logical status 0 or 1 (see chapter *1.8.3.1, page 141*) and its logical status can be polled. 8192 flags are implemented (see chapter *1.8.3.3, page 143*).

1.2.4 Times

Times are used to create defined waiting times (see chapter *1.9, page 146*).

1.2.5 Real number registers

Real number registers (1- 8192) are special memory areas for real numbers (+/- 8,000,000.000). The contents of the registers can be used for positioning as well as for calculation and comparison operations. Up to 3 decimal places may be used (see chapter *1.10.2.14, page 161*).

1.2.6 Integer register

Integer registers (1-8192) are special memory areas for whole numbers

(+/-231, i.e. 2,147,483,648). The contents of the registers can be used for times as well as for count, calculation and comparison operations. No decimals may be used (see chapter *1.10.2.1, page 151*).

1.2.7 Strings

Strings are used to store up to 80 ASCII characters received e.g. via the serial port.
Global strings S1 to S16 are available for access by each process. Each process also has its individual local S0. The contents of the global strings remain in the controller until they are overwritten. The local strings are reset at each start of the automatic mode.

1.2.8 Axes

Axes are stepping motor or servomotor axes and are used for positioning. They are addressed by the names A1, A2, A3, A4, A5, A6, A7, A8 (only unless they exist) (see chapter *1.7.3.2, page 93*).

1.2.9 Label

Labels are orientation points and serve as targets for conditional jumps and for unconditional jumps. Example of a label: $REDBLUE.

1.2.10 Comment

A comment is introduced by a semicolon (;) and is used for program documentation (see examples in the command explanations) and operator prompting in the automatic mode.

1.2.11 Direct addressing

In direct addressing, a new value is transferred directly to registers, respectively a new status is assigned to flags.

1.2.12 Indirect addressing

Indirect addressing is identified by an "!" after the designator (R!5), M!6). In the indirect addressing of registers, there is a pointer to an additional register which is addressed by the contents of the called register and the assigned value is transferred to this register. In the case of indirect flag addressing, the flag points to the contents of the integer register with the same index and sets the contents of the integer register as an index of the flag to be addressed. Only available in PNC programs!

1.2.13 Display

Three options to display are available for PA-CONTROL:

- 2-line display devices with its own keyboard
- 2-line display inside of the IEF-operating panel and
- Representation at the SÜTRON-operating panel via CANopen bus.

To display and control of both first options the commands of G5xx command group are primarily for use (see chapter *1.16.5, from page 247 onwards*). To display and control the panels via CANopen bus it can be used flags and registers of PA-CONTROL.

1.2.14 Counter

To the application, counter are added which can be connected via CANopen bus (see chapter *1.14, from page 210*).

1.2.15 Serial interface

Maximum 4 interfaces COM1 to COM4 are possible with the PA-CONTROL. Applicable are the G5xx commands of the group of commands *(see chapter 1.16.5, from page 247 onwards*).

1.2.16 AD

The processing of analog signals for example: temperatures or press, can be possible with optional used IEF modules but as well with special CANopen bus module.

For signal processing it can be used the commands in chapter *1.12*, *page 197.*

1.2.17 DA

Up to 8-DA converters can be connected via the CANopen-bus (see chapter *1.12.2 page 199*).

1.2.18 Temperature controller

With the help of the CANopen-Bus the field bus terminal “**Ksvario**” can be connected. The terminal can be used e.g. as a temperature controller.

1.2.19 List of system flags

The internal system flags (logical status 0 or 1) can only be requested by the user (see following list of system flags).

No.:	Function:
1	Power On: The flag is set by PA-CONTROL after switch-on and after polling (LD SM1, G21 SM1.n, ...)
2	A keyboard is connected to PA-Control and ready for use.
3	The key switch on the PA-CONTROL keyboard is in the "manual" position (horizontal, key cannot be removed)
4	Not assigned
5	The "G542 command", input of a register value, has been concluded with ENTER
6	The "G171 command", write characters to a PTX file, was not executed correctly
7	Battery OK
8	Axis distance is greater than interpolation path distance
9	Interpolation running (0 = interpolation not active, 1 = interpolation running)
10	Flag status always "**0**"
11	Axis 1 was stopped by the G123 function
12	Axis 2 was stopped by the G123 function
13	Axis 3 was stopped by the G123 function
14	Axis 4 was stopped by the G123 function
15	Axis 5 was stopped by the G123 function
16	Axis 6 was stopped by the G123 function
17	Axis 7 was stopped by the G123 function
18	Axis 8 was stopped by the G123 function
19	Axis 9 was stopped by the G123 function
20	Axis 10 was stopped by the G123 function
21	Axis 11 was stopped by the G123 function
22	Axis 12 was stopped by the G123 function
23	Axis 13 was stopped by the G123 function
24	Axis 14 was stopped by the G123 function
25	Axis 15 was stopped by the G123 function
26	Axis 16 was stopped by the G123 function
27	CANOpen control console "Sütron" ID 63 is active
28	CANOpen control console "Sütron" ID 62 is active
29	CANOpen control console "Sütron" ID 61 is active
30	CANOpen control console "Sütron" ID 60 is active
31	Axis 1 is ready / stand-by input is hot
32	Axis 2 is ready / stand-by input is hot
33	Axis 3 is ready / stand-by input is hot
34	Axis 4 is ready / stand-by input is hot

No.:	Function:
35	Axis 5 is ready / stand-by input is hot
36	Axis 6 is ready / stand-by input is hot
37	Axis 7 is ready / stand-by input is hot
38	Axis 8 is ready / stand-by input is hot
39	Axis 9 is ready / stand-by input is hot
40	Axis 10 is ready / stand-by input is hot
41	Axis 11 is ready / stand-by input is hot
42	Axis 12 is ready / stand-by input is hot
43	Axis 13 is ready / stand-by input is hot
44	Axis 14 is ready / stand-by input is hot
45	Axis 15 is ready / stand-by input is hot
46	Axis 16 is ready / stand-by input is hot
47	Flashing flag 0, 160ms cycle (pulse plus pause)
48	Flashing flag 1, 320ms cycle (pulse plus pause)
49	Flashing flag 2, 640ms cycle (pulse plus pause)
50	Flashing flag 3, 1280ms cycle (pulse plus pause)
51	Axis 1 has been referenced
52	Axis 2 has been referenced
53	Axis 3 has been referenced
54	Axis 4 has been referenced
55	Axis 5 has been referenced
56	Axis 6 has been referenced
57	Axis 7 has been referenced
58	Axis 8 has been referenced
59	Axis 9 has been referenced
60	Axis 10 has been referenced
61	Axis 11 has been referenced
62	Axis 12 has been referenced
63	Axis 13 has been referenced
64	Axis 14 has been referenced
65	Axis 15 has been referenced
66	Axis 16 has been referenced
67	Base programme processed an END-command
68	You may RUN a new TASK
69	Reserve
70	Reserve

Part 3 List of system flags

No.:	Function:
71	Axis 1 is released for traverse (G25.A0)
72	Axis 2 is released for traverse (G25.A0)
73	Axis 3 is released for traverse (G25.A0)
74	Axis 4 is released for traverse (G25.A0)
75	Axis 5 is released for traverse (G25.A0)
76	Axis 6 is released for traverse (G25.A0)
77	Axis 7 is released for traverse (G25.A0)
78	Axis 8 is released for traverse (G25.A0)
79	Axis 9 is released for traverse (G25.A0)
80	Axis 10 is released for traverse (G25.A0)
81	Axis 11 is released for traverse (G25.A0)
82	Axis 12 is released for traverse (G25.A0)
83	Axis 13 is released for traverse (G25.A0)
84	Axis 14 is released for traverse (G25.A0)
85	Axis 15 is released for traverse (G25.A0)
86	Axis 16 is released for traverse (G25.A0)
87	Reserve
88	Reserve
89	Reserve
90	Reserve
91	Axis 1 is in state ACTIVE
92	Axis 2 is in state ACTIVE
93	Axis 3 is in state ACTIVE
94	Axis 4 is in state ACTIVE
95	Axis 5 is in state ACTIVE
96	Axis 6 is in state ACTIVE
97	Axis 7 is in state ACTIVE
98	Axis 8 is in state ACTIVE
99	Axis 9 is in state ACTIVE
100	Axis 10 is in state ACTIVE
101	Axis 11 is in state ACTIVE
102	Axis 12 is in state ACTIVE
103	Axis 13 is in state ACTIVE
104	Axis 14 is in state ACTIVE
105	Axis 15 is in state ACTIVE

Part 4 List of system flags

No.:	Function:
106	Axis 16 is in state ACTIVE
107	Reserve
108	Reserve
109	Reserve
110	Reserve
111	Axis 1 is in state HALT
112	Axis 2 is in state HALT
113	Axis 3 is in state HALT
114	Axis 4 is in state HALT
115	Axis 5 is in state HALT
116	Axis 6 is in state HALT
117	Axis 7 is in state HALT
118	Axis 8 is in state HALT
119	Axis 9 is in state HALT
120	Axis 10 is in state HALT
121	Axis 11 is in state HALT
122	Axis 12 is in state HALT
123	Axis 13 is in state HALT
124	Axis 14 is in state HALT
125	Axis 16 is in state HALT
126	Axis 16 is in state HALT
131	Axis 1 is in state IDLE
132	Axis 2 is in state IDLE
133	Axis 3 is in state IDLE
134	Axis 4 is in state IDLE
135	Axis 5 is in state IDLE
136	Axis 6 is in state IDLE
137	Axis 7 is in state IDLE
138	Axis 8 is in state IDLE
139	Axis 9 is in state IDLE
140	Axis 10 is in state IDLE
141	Axis 11 is in state IDLE
142	Axis 12 is in state IDLE
143	Axis 13 is in state IDLE
144	Axis 14 is in state IDLE

Part 5 List of system flags

No.:	Function:
145	Axis 15 is in state IDLE
146	Axis 16 is in state IDLE
147	Reserve
148	Reserve
149	Reserve
150	Reserve
151	Axis 1 is in state SAFE
152	Axis 2 is in state SAFE
153	Axis 3 is in state SAFE
154	Axis 4 is in state SAFE
155	Axis 5 is in state SAFE
156	Axis 6 is in state SAFE
157	Axis 7 is in state SAFE
158	Axis 8 is in state SAFE
159	Axis 9 is in state SAFE
160	Axis 10 is in state SAFE
161	Axis 11 is in state SAFE
162	Axis 12 is in state SAFE
163	Axis 13 is in state SAFE
164	Axis 14 is in state SAFE
165	Axis 15 is in state SAFE
166	Axis 16 is in state SAFE
171	Axis 1 is in state IDLE or SAFE
172	Axis 2 is in state IDLE or SAFE
173	Axis 3 is in state IDLE or SAFE
174	Axis 4 is in state IDLE or SAFE
175	Axis 5 is in state IDLE or SAFE
176	Axis 6 is in state IDLE or SAFE
177	Axis 7 is in state IDLE or SAFE
178	Axis 8 is in state IDLE or SAFE
179	Axis 9 is in state IDLE or SAFE
180	Axis 10 is in state IDLE or SAFE
181	Axis 11 is in state IDLE or SAFE
182	Axis 12 is in state IDLE or SAFE
183	Axis 13 is in state IDLE or SAFE
184	Axis 14 is in state IDLE or SAFE

Is set when the axis has reached "DISABLED" or "SAFE" status as a result of an OFF command

No.:	Function:	
185	Axis 15 is in state IDLE or SAFE	
186	Axis 16 is in state IDLE or SAFE	
191	Axis 1	These flags are set if an axis in the IDLE / SAFE status is moved by more than the permissible parameter value and the axis is switched into OPERATIONAL or HALT status, otherwise the flag is reset during the ON axis function. The flag is set before the axis has reached OPERATIONAL or HALT status. If this flag is set, other actions are influenced according to "activation travel mode". The flag is reset by a query or by exiting readiness for travel.
192	Axis 2	
193	Axis 3	
194	Axis 4	
195	Axis 5	
196	Axis 6	
197	Axis 7	
198	Axis 8	
199	Axis 9	
200	Axis 10	
201	Axis 11	
202	Axis 12	
203	Axis 13	
204	Axis 14	
205	Axis 15	
206	Axis 16	
207 208 209 210	The WINPAC program can be used to combine the 16 possible axes into groups. This is done using the setting options under the “Axis parameters" register. A maximum of 4 groups can be formed for which the 207, 208, 209 and 210 system flags are assigned. If the system flag for the axis is set, the group flag is set too. For as long as one of these two flags is set, the axis cannot be traversed. The flags are cancelled by: ■ queries in automatic mode (LD SM191..., G21 SM191..., LD SM207... ■ AUTOMATIC or ONLINE being aborted **NOTE**: A group flag cannot be cancelled directly. The group flag is reset by cancelling all the flags for this group..	
211	Axis 1 is busy with an interpolation	0 = Axis is not traversed by an interpolation 1 = Axis is traversed by an interpolation
212	Axis 2 is busy with an interpolation	
213	Axis 3 is busy with an interpolation	
214	Axis 4 is busy with an interpolation	
215	Axis 5 is busy with an interpolation	
216	Axis 6 is busy with an interpolation	

Part 7 List of system flags

No.:	Function:	
217	Axis 7 is busy with an interpolation	0 = Axis is not traversed by an interpolation 1 = Axis is traversed by an interpolation
218	Axis 8 is busy with an interpolation	
219	Axis 9 is busy with an interpolation	
220	Axis 10 is busy with an interpolation	
221	Axis 11 is busy with an interpolation	
222	Axis 12 is busy with an interpolation	
223	Axis 13 is busy with an interpolation	
224	Axis 14 is busy with an interpolation	
225	Axis 15 is busy with an interpolation	
226	Axis 16 is busy with an interpolation	
227	Reserve	
228	Reserve	
229	Reserve	
230	Reserve	
231	Axis 1 is FAULT	
232	Axis 2 is FAULT	
233	Axis 3 is FAULT	
234	Axis 4 is FAULT	
235	Axis 5 is FAULT	
236	Axis 6 is FAULT	
237	Axis 7 is FAULT	
238	Axis 8 is FAULT	
239	Axis 9 is FAULT	
240	Axis 10 is FAULT	
241	Axis 11 is FAULT	
242	Axis 12 is FAULT	
243	Axis 13 is FAULT	
244	Axis 14 is FAULT	
245	Axis 15 is FAULT	
246	Axis 16 is FAULT	
247	Reserve	
248	Reserve	
249	Reserve	
250	Reserve	

Part 8 List of system flags

No.:	Function:
251	Axis 1 is IN POSITION
252	Axis 2 is IN POSITION
253	Axis 3 is IN POSITION
254	Axis 4 is IN POSITION
255	Axis 5 is IN POSITION
256	Axis 6 is IN POSITION
257	Axis 7 is IN POSITION
258	Axis 8 is IN POSITION
259	Axis 9 is IN POSITION
260	Axis 10 is IN POSITION
261	Axis 11 is IN POSITION
262	Axis 12 is IN POSITION
263	Axis 13 is IN POSITION
264	Axis 14 is IN POSITION
265	Axis 15 is IN POSITION
266	Axis 16 is IN POSITION
271	Gantry-Axis 1 has been referenced
272	Gantry-Axis 2 has been referenced
273	Gantry-Axis 3 has been referenced
274	Gantry-Axis 4 has been referenced
275	Gantry-Axis 5 has been referenced
276	Gantry-Axis 6 has been referenced
277	Gantry-Axis 7 has been referenced
278	Gantry-Axis 8 has been referenced
279	Gantry-Axis 9 has been referenced
280	Gantry-Axis 10 has been referenced
281	Gantry-Axis 11 has been referenced
282	Gantry-Axis 12 has been referenced
283	Gantry-Axis 13 has been referenced
284	Gantry-Axis 14 has been referenced
285	Gantry-Axis 15 has been referenced
286	Gantry-Axis 16 has been referenced
291	Axis 1 : set Reference-Offset to Gantrysystem is possible
292	Axis 2 : set Reference-Offset to Gantrysystem is possible

Part 9 List of system flags

No.:	Function:
293	Axis 3 : set Reference-Offset to Gantrysystem is possible
294	Axis 4 : set Reference-Offset to Gantrysystem is possible
295	Axis 5 : set Reference-Offset to Gantrysystem is possible
296	Axis 6 : set Reference-Offset to Gantrysystem is possible
297	Axis 7 : set Reference-Offset to Gantrysystem is possible
298	Axis 8 : set Reference-Offset to Gantrysystem is possible
299	Axis 9 : set Reference-Offset to Gantrysystem is possible
300	Axis 10 : set Reference-Offset to Gantrysystem is possible
301	Axis 11 : set Reference-Offset to Gantrysystem is possible
302	Axis 12 : set Reference-Offset to Gantrysystem is possible
303	Axis 13 : set Reference-Offset to Gantrysystem is possible
304	Axis 14 : set Reference-Offset to Gantrysystem is possible
305	Axis 15 : set Reference-Offset to Gantrysystem is possible
306	Axis 16 : set Reference-Offset to Gantrysystem is possible

1.2.20 List of system N registers

No.:	Function:
1	System run time of PA-CONTROL in seconds (the run time is only reset after a new operating system has been loaded)
2	System run time of PA-CONTROL in tenths of a second (the run time is only reset after a new operating system has been loaded)
3	Measurement of run times with high resolution from V4.74G register SN3 is reset to 0 in the event of polling and each millisecond is incremented by 1 by PA-CONTROL. As a result, the time in milliseconds between two polls can be determined by means of register SN3, e.g. for measurement of the run time.
4	Reserve
5	PA-CONTROL version
6	PA-CONTROL serial number
7	Reserve
8	Reserve
9	Reserve
10	System error number
11	Run error number
12	Number of the axis or of one axis from an axis group (concerned in an error)
13	Number of the parallel sequence with the error
20	PA-CONTROL was stopped by <SN20> = 1 → STOP by control console (IEF or CANOpen) <SN20> = 2 → STOP by external STOP input <SN20> = 3 → STOP by diagnosis port (WINPAC) <SN20> = 4 → STOP by Profibus DP <SN20> = 5 → STOP by RS232-ONLINE CMD port <SN20> = 6 → STOP by Interbus-S <SN20> = 7 → STOP by PAB program
21	PA-CONTROL is in stop mode If PA-CONTROL is stopped in automatic mode, various steps, which can be polled in SN21, are completed in the STOPMODE. <SN21> = 0 → No STOP <SN21> = 1 → STOP detected <SN21> = 2 → Wait until all axes are at standstill <SN21> = 3 → Start program after STOP <SN21> = 4 → Process program after STOP <SN21> = 5 → Stopped <SN21> = 6 → Stopped and wait for Continue <SN21> = 7 → START after STOP detected, start stopped axes <SN21> = 8 → Process program START after STOP <SN21> = 9 → Activate Process all programs

Part 2 List of system N registers

No.	Function
31	Axis 1 / Counter absolute positioning system
32	Axis 2 / Counter absolute positioning system
33	Axis 3 / Counter absolute positioning system
34	Axis 4 / Counter absolute positioning system
35	Axis 5 / Counter absolute positioning system
36	Axis 6 / Counter absolute positioning system
37	Axis 7/ Counter absolute positioning system
38	Axis 8 / Counter absolute positioning system
39	Axis 9 / Counter absolute positioning system
40	Axis 10 / Counter absolute positioning system
41	Axis 11 / Counter absolute positioning system
42	Axis 12 / Counter absolute positioning system
43	Axis 13 / Counter absolute positioning system
44	Axis 14 / Counter absolute positioning system
45	Axis 15 / Counter absolute positioning system
46	Axis 16 / Counter absolute positioning system
47	Reserve
48	Reserve
49	Reserve
50	Reserve
51	Axis 1 / Number of A-D values actually got by the G180 command
52	Axis 2 / Number of A-D values actually got by the G180 command
53	Axis 3 / Number of A-D values actually got by the G180 command
54	Axis 4 / Number of A-D values actually got by the G180 command
55	Axis 5 / Number of A-D values actually got by the G180 command
56	Axis 6 / Number of A-D values actually got by the G180 command
57	Axis 7 / Number of A-D values actually got by the G180 command
58	Axis 8 / Number of A-D values actually got by the G180 command
59	Axis 9 / Number of A-D values actually got by the G180 command
60	Axis 10 / Number of A-D values actually got by the G180 command
61	Axis 11 / Number of A-D values actually got by the G180 command
62	Axis 12 / Number of A-D values actually got by the G180 command
63	Axis 13 / Number of A-D values actually got by the G180 command

Part 2 List of system N registers

No.:	Function:		
64	Axis 14 / Number of A-D values actually got by the G180 command		
65	Axis 15 / Number of A-D values actually got by the G180 command		
66	Axis 16 / Number of A-D values actually got by the G180 command		
67	Reserve		
68	Number of free TASKS for a RUN-CMD		
69, 70	Reserve		
71	State of the axis 1		
72	State of the axis 2		
73	State of the axis 3		
74	State of the axis 4		
75	State of the axis 5		
76	State of the axis 6		
77	State of the axis 7		
78	State of the axis 8		
79	State of the axis 9		
80	State of the axis 10		
81	State of the axis 11		
82	State of the axis 12		
83	State of the axis 13		
84	State of the axis 14		
85	State of the axis 15		
86	State of the axis 16		
91	CAN-IO ID25 Type number of the Module (on V5.01B)		
92	CAN-IO ID26 Type number of the Module (on V5.01B)		
93	CAN-IO ID27 Type number of the Module (on V5.01B)		
94	CAN-IO ID28 Type number of the Module (on V5.01B)		
101	on V. 4.77A	COM 1	Number of received signs
102		COM 2	
103		COM 3	
104		COM 4	
105		COM 5	
106		COM 6	
107		COM 7	
108		COM 8	

Possible axis statuses:
0 : INIT
1 : DISABLE
2 : SAFE
3 : SAFE_WHILE_DISABLE
4 : SAFE_WHILE_OPERATIONAL
5 : SAFE_WHILE_ACTIVE
6 : IDLE
7 : IDLE_WHILE_DISABLE
8 : IDLE_WHILE_OPERATIONAL
9 : IDLE_WHILE_ACTUVE
10 : OPERATIONAL
11 : ACTIVE
12 : HALT
13 : HALT_WHILE_DISABLE
14 : HALT_WHILE_OPERATIONAL
15 : HALT_WHILE_ACTIVE
16 : FAULT

Part 4 List of system N registers

No.:	Function:
110	Counter 0 : actual Value
111	Counter 1 : actual Value
112	Counter 2 : actual Value
113	Counter 3 : actual Value
114	Counter 4 : actual Value
115	Counter 5 : actual Value
116	Counter 6 : actual Value
117	Counter 7 : actual Value
118	Counter 8 : actual Value
119	Counter 9 : actual Value
120	Counter 10 : actual Value
121	Counter 11 : actual Value
122	Counter 12 : actual Value
123	Counter 13 : actual Value
124	Counter 14 : actual Value
125	Counter 15 : actual Value
126	Counter 16 : actual Value
130	Counter 0 : Number of AD-Values get with the G182-CMD
131	Counter 1 : Number of AD-Values get with the G182-CMD
132	Counter 2 : Number of AD-Values get with the G182-CMD
133	Counter 3 : Number of AD-Values get with the G182-CMD
134	Counter 4 : Number of AD-Values get with the G182-CMD
135	Counter 5 : Number of AD-Values get with the G182-CMD
136	Counter 6 : Number of AD-Values get with the G182-CMD
137	Counter 7 : Number of AD-Values get with the G182-CMD
138	Counter 8 : Number of AD-Values get with the G182-CMD
139	Counter 9 : Number of AD-Values get with the G182-CMD
140	Counter 10 : Number of AD-Values get with the G182-CMD
141	Counter 11 : Number of AD-Values get with the G182-CMD
142	Counter 12 : Number of AD-Values get with the G182-CMD
143	Counter 13 : Number of AD-Values get with the G182-CMD
144	Counter 14 : Number of AD-Values get with the G182-CMD
145	Counter 15 : Number of AD-Values get with the G182-CMD
146	Counter 16 : Number of AD-Values get with the G182-CMD

1.2.21 List of system R registers

No.:	Function:
1...9	Reserve
10	Reference path speed for interpolation
11...30	Reserve
31	Axis A1 / Position in the absolute system
32	Axis A2 / Position in the absolute system
33	Axis A3 / Position in the absolute system
34	Axis A4 / Position in the absolute system
35	Axis A5 / Position in the absolute system
36	Axis A6 / Position in the absolute system
37	Axis A7 / Position in the absolute system
38	Axis A8 / Position in the absolute system
39	Axis A9 / Position in the absolute system
40	Axis A10 / Position in the absolute system
41	Axis A11 / Position in the absolute system
42	Axis A12 / Position in the absolute system
43	Axis A13 / Position in the absolute system
44	Axis A14 / Position in the absolute system
45	Axis A15 / Position in the absolute system
46	Axis A16 / Position in the absolute system
47	Reserve
48	Reserve
49	Reserve
50	Reserve
51	Position of the axis 1, as the axis reached state IDLE or SAFE
52	Position of the axis 2, as the axis reached state IDLE or SAFE
53	Position of the axis 3, as the axis reached state IDLE or SAFE
54	Position of the axis 4, as the axis reached state IDLE or SAFE
55	Position of the axis 5, as the axis reached state IDLE or SAFE
56	Position of the axis 6, as the axis reached state IDLE or SAFE
57	Position of the axis 7, as the axis reached state IDLE or SAFE
58	Position of the axis 8, as the axis reached state IDLE or SAFE
59	Position of the axis 9 as the axis reached state IDLE or SAFE
60	Position of the axis 10, as the axis reached state IDLE or SAFE
61	Position of the axis 11, as the axis reached state IDLE or SAFE
62	Position of the axis 12, as the axis reached state IDLE or SAFE

No.:	Function:
63	Position of the axis 13, as the axis reached state IDLE or SAFE
64	Position of the axis 14, as the axis reached state IDLE or SAFE
65	Position of the axis 15, as the axis reached state IDLE or SAFE
66	Position of the axis 16, as the axis reached state IDLE or SAFE
67	Reserve
68	Reserve
69	Reserve
70	Reserve
71	Target position of axis 1 with the last drive command
72	Target position of axis 2 with the last drive command
73	Target position of axis 3 with the last drive command
74	Target position of axis 4 with the last drive command
75	Target position of axis 5 with the last drive command
76	Target position of axis 6 with the last drive command
77	Target position of axis 7 with the last drive command
78	Target position of axis 8 with the last drive command
79	Target position of axis 9 with the last drive command
80	Target position of axis 10 with the last drive command
81	Target position of axis 11 with the last drive command
82	Target position of axis 12 with the last drive command
83	Target position of axis 13 with the last drive command
84	Target position of axis 14 with the last drive command
85	Target position of axis 15 with the last drive command
86	Target position of axis 16 with the last drive command

1.3 Overview of the command set

Some commands of PA-CONTROL can only be used with specific PA-CONTROL types.

Command [1] → [1] = PA-CONTROL MP

Command [2] → [2] = PA-CONTROL STEUER with PLS7 / PLS9

Command [3] → [3] = PA-CONTROL COMPACT with PLS7

Command [4] → [4] = PA-CONTROL LV-servoTEC

The superscripted number in the descriptions of the commands identifies the PA-CONTROL type in which the command can only be used.

NOTE: all page references contain of the following command overview contain a link.

1.3.1 Program execution (automatic)

1.3.1.1 Sequence

Cancel ..... BREAK 51
Stop ..... STOP 53
Continue ..... START 52
Get the number of the line of the program ..... Ni:=LINE
..... Ni:=LINE.NAME
..... Ni.=LINE.S1 54
Ending ..... END 54

1.3.1.2 Jumps and Branches

Always jump ..... JMP CYCLE 58
Start the program at the defined line ..... JMP-LINE.Ni 57
Jump if (logical) condition of an input G21 I1.1 56
Jump if (logical) condition of an output G21 O1.1 56
Jump if (logical) condition of a flag G21 M1.1 56
Jump if logical condition is fulfilled ..... G21 M5.1 drilling 56
Jump if contents of N register CASE.JMP.N8 58
Jump if monitoring time has elapsed ..... G421.1.500 SAVE_1 243
Jump if axis at standstill / moving ..... G211.A1.0 WAIT 231
Jump if current position of the axis ..... G221.A2.1 READY 234
Jump if error message is received ..... G721.49 EMERGENCY 280
Start interpolation at the defined line ..... JMP-LINE-IPO.Ni 133

1.3.1.3 Call subroutines

Call subroutine ..... SUB LAMBLI 60
Call subroutine, name of the program is defined in N-Register. SUB.N8.....60
Call subroutine, name of the program is defined in N-Register
Name of program has fix character length SUB.N7.8 60
Call subroutine, program is checked of right type SUB.N6.PNC 60
Call if (logical) condition of an input ..... G22 I1.1..... 65
Call if (logical) condition of an output G22 O1.1 65
Call subroutine when conditions are meet..... G22 M5.1 SAWING 65
Call if (logical) condition of an flag ..... G22 M1.1.....65
Call if contents of N register CASE.SUB.N8 66
Call if monitoring time has elapsed ..... G422.1.500 ERROR 244
Call if time monitoring has elapsed with repetition ..... G423.1.200 ERROR_2..... 245
Call if axis at standstill / moving ..... G212.A0.0 ELAPS 233
Call if current position of the axis ..... G222.1.A2.1500 PASTE 237
Call if error message is received ..... G722.50 GRIPPER.....280
Call name of program is in a string..... SUB.S0, SUB.SN 63

1.3.1.4 Loops

Increment counter..... INC .N2.N3 START 68
Decrement counter ..... DEC .N3 CYCLE 69

1.3.1.5 Parallel runs

Start parallel run ..... RUN..... 71
Delete parallel run ..... SLEEP 74
Stop parallel run ..... CANCEL 73
Start if parallel run is dependent ..... CASE.RUN. 75
Delete if parallel run is dependent ..... CASE.SLEEP 79
Stop if parallel run is dependent ..... CASE.CANCEL 77
Get program status ..... PROGSTAT.....81

1.3.1.6 Error management

ERROROFF..... ERROROFF 82
ERRORON ..... ERRORON 82

1.3.2 Commands for Axis

1.3.2.1 General Remarks

1.3.2.2 Modes of an axis

Activate start positioning mode G210.A3 230
Activate normal positioning mode G213.A3 234
Disable, activate the function “OFF-Axis” G140 86
Enable, activate the function “ON-Axis” G141 87
Disable axis(IDLE, SAFE) ..... OFF.An 88
Enable (OPERATIONAL, ACTIVE) ..... ON.An 89
Stop moving of the axis (HALT) ..... STOP.An 90
Start moving if the axis has stopped (OPERATIONAL, ACTIVE) ..... START.An 90
Abort moving of the axis which was stopped (HALT) ..... ABORT.An 91
Wait till current position value </> G230.1.A1.456 239
Wait till remaining distance is exceeded or fell short of the Current position G231.A4.34 240

1.3.2.3 Move

Approach to reference point G25.A1 92
Traversing ..... A1:=20 # A1:=R3 93
Absolute traversing ..... G90.A0 # G90.A1 94
Relative traversing ..... G91.A0 # G91.A1 95
Acceleration G100.A1.15 96
Traversing speed ..... FA1:=200 # FA2:=R4 97
Move with START-STOP-Traversing speed G150 102
Traverse as long as condition is fulfilled G123.A1 I3.1 99
Request for the set or current speed ..... Rn:=FAn 104
Request for path speed ..... Rn:=FB 129
Get the current speed of an axis during an interpolation ..... Rn:=FBAn 130

1.3.2.4 Position and Parameter

Set position ..... G26.A1 # G26.A1.R1 105
Set position to size (set to new value) ..... G29.A.1.345 # G29.A2.R3 106
Load axis position ..... R1:=A1, N1:=A1 108
Get axis parameter (load) ..... R1:=A1, N1:=1 108

1.3.2.5 Limit switches

Switch off limit switch monitoring G142 109
Switch on limit switch monitoring G143 110

1.3.2.6 Wait

Wait until the position is bigger or less than value G230.1.A1.456 239
Wait until current position </> than value G230.1.A1.456 239

Wait until distance exceeds or falls below remaining distance for current positioning G231.A4.34 240

1.3.2.7 Wait until po Encoder and following error

Get following error ..... Ni:=PEAn, Ri:=PEAn 111
Get encoder position (load) [1] ..... R1:=ENC1 # N1:=ENC1 112
Wait until position of SSI-Interface SSI1.1.R12, SSI1.0.N4 114

1.3.2.8 Interpolation

Linear interpolation [2,3] ..... G01 A1:=40 A2:=60 124
Define acceleration during interpolation G100.B.1500 96
Define velocity of the interpolation ..... FB:=2000, FB:=R4 123
Define IPO velocity, one axis is free-moving ..... FB.2:=1234 # FB.3:=R7 123
Set output during an interpolation G160.20.O123.1 131
..... G161.1.O333.1 132
..... G162.5.O338.1 132
Reset output during an interpolation G160.20.O123.0 131
..... G161.1.O333.0 132
..... G162.5.O338.0 132
Start interpolation at the defined line ..... JMP-LINE-IPO.Ni 133
IPO determine path acceleration G100.B.1500 122
IPO determine path acceleration ..... FB:=2000, FB:=R4 123
Cancel linear interpolation ..... IPOEND 126
Linear interpolation, an axis is free-moving ..... G01 A1:=50 A2:=30 A3:=FB*R6 126
IPO path acceleration, an axis is free-moving ..... FB.2:=1234 # FB.3:=R7 123

1.3.2.9 Special PA-CONTROL Commands

Change current of motor G101.A1.50 134

1.3.3 Digital Inputs, digital Outputs, Flags and Registers

1.3.3.1 Digital inputs

Wait for status of input I1.1 138
Logical AND operation with input ..... LD / AND 168
Logical OR operation with input ..... LD / OR 169
Image of inputs to a register G603.R3.2.8 273

1.3.3.2 Digital outputs

Set / Reset output ..... O1:=1# O2:=0 140
Wait for output status O1.1 139
Logical AND operation with output ..... LD / AND / OUT 168
Logical OR operation with output ..... LD / OR / OUT 169
Set, reset, negate output ..... SET / RES / NOT 170
Extended functions of the commands SET and RES ..... SET / RES.01.016 170
Image of register to outputs (binary) G600.R3.5.8 270
Image of register to outputs (BCD) G601.N2.2.4 271

1.3.3.3 Flags

Direct or indirect addressing ..... M1.1 # M!10.1 141
Set / Reset flag ..... M1:=1 # M2:=0 141
Wait for flag status M1.1 143
Logical AND operation with flag ..... LD / AND / OUT 168
Logical OR operation with flag ..... LD / OR / OUT 169
Set, reset, negate flag ..... SET / RES / NOT 170
Extended functions of the commands SET and RES ..... SET / RES.M1.M256 170
Image of register to flag G602.R3.4.8 272
Image of flags to a register G604.N4.2.8 274
Write flag number to an N register after comparison ..... M3:=GEM.1.128.1 145
Assign the comparison result ..... M36:=R1<56 144

1.3.3.4 Registers

Direct and indirect addressing ..... N1, N!2 151
Load register with value (constant) ..... N1:=23, R1:=33,5 151
Load register with value (constant) , binary ..... N1:=2#00110000 152
Load register with value (constant) , hex ..... N4:=16#3E8 152
Wait for status of an N register ..... Ni.n, Ni.Nn, N!i.n, N!i.N!n 178
Load contents of system N register into an integer system ..... Nn:=SNn 179
Load contents of system R register into a real number register ..... Ri:=SRn 180
Bit by bit processing of N register ..... Ni:=Nn&Nm
..... Ni:=Nn&constant
..... Ni:=Nn|Nm
..... Ni:=Nn|constant
..... Ni:=Nn^Nm
..... Ni:=Nn^constant 177

1.3.4 Dwell Timer, monitoring time, date, time of the day

1.3.4.1 Wait

Delay program execution until time has elapsed T10, TN1 146

1.3.4.2 Monitoring time

Start monitoring time with jump in program ..... G421.1.200 ERROR_1 243
Start monitoring time with call of subroutine ..... G422.1.500 ERROR_2 244
Start monitoring time with repeat function ..... G423.1.300 CRASH 245
Delete (reset) monitoring time G401.1 246

1.3.4.3 Date, time of the day

Read real time clock ..... S0:=TIME.HOUR.MIN 149
Set date ..... Date:=N1.N2.N3.N4 147

1.3.5 Set time Mathematics

1.3.5.1 Addressing

Direct and indirect addressing ..... N1, N!2 150

1.3.5.2 Mathematic commands

Load register ..... Rn/Nn:=1234, N:=16#3E8 151
..... N3:=2#00110000 153
Addition ..... R2:=R2+57 153
..... R1:R1+R2 153
Subtraction ..... R4:=R4-33 154
..... R5:=R5-R6 154
..... R23:=1000-R7 154
Multiplication ..... N2:=N1*12 155
..... R3:=R2*R4 155
Division ..... N5:=N5/12 156
..... R6:=R6/R7 156
..... N7:=1000/N8 156
Sinus functions ..... Rn:=SIN.K 157
..... Rn:=SIN.Ri 157
..... Rn:=ASIN.K 157
..... Rn:=ASIN.Ri 157
Cosine functions ..... Rn:=COS.K 158
..... Rn:=COS.Ri 158
..... Rn:=ACOS:K 158
..... Rn:=ACOS.Ri 153
Tangent functions ..... Rn:=TAN.K 159
..... Rn:=TAN.Ri 159
..... Rn:=ATAN.K 159
..... Rn:=ATAN.Ri 159

Root function ..... R99:=SQRT.R43 160
Integer component ..... R45:=INT.R70 160
Decimal component of real number ..... R43:=FRAC.R55 161
Absolute value of real number / integer ..... N55:=ABS.N22 161

1.3.5.3 Compare functions

Compare command syntax 162
Examples 163
Greater ..... M11:=N22>12 144
Equal to ..... M12:=R21=R22 144
Smaller ..... M13:=R23<R24 144
Not equal to ..... M113:=N100<>123 144

1.3.5.4 Logical operations

LD ..... LD I3.1 168
AND ..... AND I4.0 168
OR ..... OR I5.1 168
SET ..... SET O6 170
SET ..... SET M92 170
SET SET.O1.O6 170
SET SET.M2.M96 170
NOT ..... NOT O7 168
RESET ..... RES O32 170
RESET ..... RES M16 170
RESET RES.O1.O16 170
RESET RES.M56.M94 170
XOR logical – operation (exclusive OR) ..... XOR In.i 173
Create XOR checksum from a string and store it into integer register ..... N1:=XOR.S0 192
Multilevel AND (brackets) ..... AND-LD 174
Multilevel OR (brackets) ..... OR-LD 175
Logical functions with N register ..... Ni:=Nn&Nm 178
..... Ni:=Nn&constant
..... Ni:=Nn|Nm
..... Ni:=Nn|constant
..... Ni:=Nn^Nm
..... Ni:=Nn^constant

1.3.5.5 Register operation

Load register with value (constant) ..... N1:=23, R1:=33,5 151
Load register with value (constant) , binary ..... N1:=2#00110000, ... 152
Load register with value (constant) , hex ..... N3:=16#3E8 152
Wait for status of a N register ..... Ni.n, Ni.Nn, N!i.n, N!i.N!n 179
Load contents of system N register into an integer system ..... Nn:=Sn 180
Load contents of system R register into a real number register ..... Ri:=SNn 180

Bit by bit processing of N register ..... Ni:=Nn&Nm
..... Ni:=Nn&constant
..... Ni:=Nn|Nm
..... Ni:=Nn|constant
..... Ni:=Nn^Nm
..... Ni:=Nn^constant 178

1.3.6 Communication

1.3.6.1 Serial ports, display and keyboard

Switch off run displays ... G11.0 ... 184
Initialize ... G500.0, G500.1.6.1.0 ... 247
Clear display ... G501 ... 249
Delete line ... G502 ... 250
Position cursor ... G503.4.1 # G504.N2.N6 ... 251
Output text ... G510.MACHINE ERROR ... 252
Output text with CR and LF ... G511.PROCEDURE ... 253
Output special character ... G512.36 ... 254
Output text line from a PTX file ... G515.3.ERROR TEXT ... 255
Output of content of register ... G520.R2 # G520.R3.3.1 ... 256
Output of status of flags, output or input ... G520.I7 ... 256
Output contents of register with CR and LF ... G521.N5 # G521.N7.8 ... 258
Output of status of flags, output or input with CR and LF ... G521.M25 ... 258
Take over content of register ... G531.R7 ERROR_1 ... 260
Take over flag or initial condition ... G531.M34.DATA ... 260
Take over character string ... G531.R2 ERROR ... 260
Take over characters to the local character buffer S0 ... G532.13.U_ERROR ... 261
Take over a character string in the background ... G533.13 ... 262
Take over a number of defined characters ... G534.4 ... 264
Check if interface has been taken over ... N5:=CHN ... 185
Check if key has been pressed ... G540.N6 ... 265
Get character (key code) ... G541.N2 ... 266
Input register ... G542.30.1.3.N1 ... 267
Query key switch ... LD SM3.0 ... 14

1.3.6.2 CANopen bus

Network management commands ... general ... 275
Start Remote Node ... G701.n ... 276
Stop Remote Node ... G702.n ... 277
Reset Remote Node ... G703.n ... 277
Enter Pre-Operational-State ... G704.n ... 278
Reset communication ... G705.n ... 278

Node guarding ... G711.49.12 ERROR ... 279
Commands for checking device errors, general ... G72x ... 279
Jump if error message is received ... G721.(ID) (MARK) ... 280
Call if error message is received ... G722.(ID) (MARK) ... 280

Commands for processing service data objects ... G73x ... 280
Read service device object ... G730 ... 281
Write service device object ... G731 ... 281
Commands for processing service data objects general ... G74x, G75x ... 282
Read PDO1 ... G741.49.4 ... 284
Read PDO2 ... G742.50.2 ... 285
Send PDO1 ... G751.49.4 ... 285
Send PDO2 ... G752.50.2 ... 285

1.3.6.3 MODBUS

Read data from MODBUS ..... Ni:=MODBUS 182
Write data to MODBUS..... MODBUS 183

1.3.6.4 String commands

Sn - copy of strings ..... Sn:=Si # Sn:=”Hallo Welt”..... 186
S0 - copy of strings ..... S0:=CHN 186
Input via keyboard G542.10.2.3.N1 267
Value transfer from current data channel ..... G531.R1 ERROR 260
Value transfer from current data channel ..... G532.13 FINE..... 261
Value transfer from current data channel ..... G531.13..... 260
Search for position of character in local character buffer S0 .... N2:=POS1.35 187
Convert characters out of the local character buffer ..... N3:=COPY.2.5 C_ERROR 188
Get substring ..... S3:=COPY.N5.N7..... 190
Merging strings ..... Si:=Sn+SM 191
Get the length of a string ..... N6:=LENGTH.S0 191
Write the content of register (number) to a string..... S4:=R6.N3.N7 192
Transfer the content of the local character buffer S0
to an integer register ..... N4:=GET.3.2 193
Transfer the content of the local character buffer S0 in
INTEL format to an integer register ..... N1:=GETI.1.2 195
Transfer the content of N register to the local character buffer. PUT.1.4.N1..... 195
Transfer the content of an N register to the INTEL format
into local character buffer S0..... PUTI.1.4.N1..... 196
Create XOR-Checksum from a string..... N3:=XOR.S1..... 192

1.3.7 Analog / digital

Analog and digital converter ..... general description 197
Get A-D values ..... N1:=AD1 # R1:=AD1 197
Acquisition of several values G18x 217
A-D measuring series synchronous to axis movement ..... G180.A2.R22.R23.1.100.25 ... 217
A-D measuring series with time grid G181.5.1.100.50 219
A-D-measuring synchronous via counter ..... G183.0.N10.N11.1.33.10..... 222
Analog and digital converter ..... further information 199
Set D-A value (increments)..... DA1:=N2 200

1.3.8 File commands

Get N register from PTX file G170.N1.N2.N10.TEXTS 201
Store N register in PTX file G171.N2.N3.N20.ERR 204
Write line from PTX file to string ..... G172.4.S2.ERROR TEXT 203
Store string in a line of PTX file ..... G173.N2.S1.ERROR_1..... 204
Storage of current values in a program file ..... STORE TA2:=PE_001..... 205
Storage of values if content of N-register CASE.STORE.N3 208

Deactivating / activating assignment with ..... STOREEND.....207
Begin with store command ..... STARTSTART.....207

1.3.9 Counter

Initialize Counter..... CNT4.INIT.0 211
Set counter to value..... CNT1:=N1 # CNT3:=25.....212
Read counter ..... N7:=CNT8 212
Comparing counters with constant CNT3.0.1234 213
Comparing counters with content of integer register..... CNT4.1.N7.....213

1.3.10 Temperature control

General principles 214
Write value (read parameter) from temperature controller ..... TC3.SP:=25.5 215
Read value (write Parameter) to temperature controller ..... R27:=TC1.X_EFF.....216

1.3.11 Command groups of PA-CONTROL

1.3.11.1 Commands of the group G16x

Set outputs during an interpolation..... G160.20.O123.1 131
Activate outputs before starting the interpolation G161.1.O333.1 131
..... G162.5.O338.1 131
Reset output during an interpolation G160.20.O123.0 132
..... G161.1.O333.0 132
..... G162.5.O338.0 132

1.3.11.2 Commands of the group G17x

Get N register from PTX file ..... G170.N1.N2.N10.TEXT.....201
Store N register in PTX file G171.N2.N3.N20.ERR 208
Write line from PTX file in string (Sn) G172.4.S2. 204
Store string in a line of PTX file ..... G173.N2.S1.ERROR_1.....205

1.3.11.3 Commands of the G19x-group

Get the number of inserted bus terminals G190.17.N8000 225
Read the content of a register G191.17.6.0.8.N8001 226
Describes a register with a value..... G192.17.6.0.32.N8003 227

1.3.11.4 Commands of the group G2xx

Activate start positioning mode..... G210.A3 230
Activate normal positioning mode..... G213.A3 234
Jump if axis at standstill / moving ..... G211.A1.0 WAIT 231
Jump if current position of the axis ..... G221.A2.1 READY..... 234
Call if axis at standstill / moving ..... G212.A0.0 ELAPS 233
Call if current position of the axis ..... G222.1.A2.1500 PASTE 237

Wait until the position is bigger or less than value G230.j.An.constant 239
Wait until distance exceeds or falls below remaining
Distance for current positioning G231.An.constant 240

1.3.11.5 Commands of the group G4xx

Start time monitoring with a conditional jump ..... G421.1.200 ERROR_1 243
Time monitoring with conditional subroutine call ..... G422.1.500 ERROR_2 244
Call if time monitoring has elapsed with repetition ..... G423.1.300 CRASH 245
Delete (reset) monitoring time G401.1 246

1.3.11.6 Commands of the group G5xx

Initialize G500.0, G500.1.6.1.0 247
Clear display G501 249
Delete line G502 250
Position cursor ..... G503.4.1 # G504.N2.N6 251
Output text ..... G510.MACHINE ERROR 252
Output text with CR and LF G511.INTERVENTION 253
Output special characters G512.36 254
Output text line from a PTX file ..... G515.3.ERROR MESSAGE 255
Output contents of register G520.R2 256
Output status of flags, output or input G520.I7 256

Output of content of register with CR and LF G521.N5 258

Output of states of flags, input, output wit CR and LR G521.M25 258

Value transfer ..... G531.R7 ERROR_1 260

Take over status of flags or output G531.M34.DATA 260
Take over character string ..... G531.R2 ERROR 260
Take over character in local character buffer S0 ..... G532.13.U_ERROR 261
Take over character string in background G533.13 262
Take over a defined number of characters G534.4 264
Check if key has been pressed G540.N6 265
Get character (key code) G541.N2 266
Input register G542.30.1.3.N1 267
Input string G542.10.2.10.S0 267

1.3.11.7 Commands of the group G6xx

Image of register to outputs (binary) G600.R2.2.4 270
Image of register to outputs (BCD) G601.N3.2.4 271
Image of inputs to a register G603.R4.2.8 273
Image of register to flag G602.R3.17.32 272
Image of flags to a register G604.R4.2.8 274

1.3.11.8 Commands of the group G7xx

Network management commands ..... General 276
Start Remote Node G701.n 276
Stop Remote Node G702.n 277
Reset Remote Node G703.n 277
Enter Pre-Operational-State G704.n 278
Reset communication G705.n 278

Node guarding ..... G711.49.12 ERROR 279
Commands for checking device errors G72x 279
Branch if error message is received ..... G721.(ID) (MARK) 280
Call if error message is received ..... G722.(ID) (MARK) 280

Commands for processing service data objects G73x 280
Read service device object G730. 281
Write service device object G731 281

Commands for processing of a service data objects G74x, G75x 282
Read process data PDO1 ..... G741.49.4 284
Read process data PDO2 ..... G742.50.2 284
Send process data PDO1 ..... G751.49.4 285
Send process data PDO2 ..... G752.50.2 285

1.4 Programming tips

1.4.1 The run interpreter

PA-CONTROL is a sequence controller. It processes one command after another (e.g. traverse axis, set output, wait for input, etc.). Once a command has been processed, the next command in the program is executed.

The sequence controller has been designed for parallel sequence control, in order to be able to control parallel processes in a machine.

The runtime interpreter is able in the initial position to execute the program.

In this initial position are the following commands or commands group not allowed:

- Movement command
- Interpolation command
- Subroutine call
- RUN

However the following operations within the initial program are allowed:

- Loading the screen of CANopen control panel
- Changing outputs in subject to inputs
- Starting automatic mode whenever predefined conditions have been fulfilled

The run interpreter can process up to 31 parallel processes simultaneously. Each parallel process can call subroutines up to a nesting depth of 16 programs. This is referred to as multitasking.

PNC, PNX und PAB programs are available as executable programs. All commands can be used irrespective of the program type.

Text data can be saved in an available PTX programme (PTX files). In this programme presentable ASCII-signs should be used only.

NOTE Please do not use the ASCII-signs 01hex (SOH), 02hex (STX), 03hex (ETX) and 04hex (EOT), as they disturb the transfer protocol between the PA-CONTROL and WINPAC.

The difference between PNC, PNX und PAB programs can only be noticed when PA-CONTROL is stopped (STOP_MODE).

When activating a parallel process (RUN, …), the program type is stored, so that the parallel process can be handled appropriately in the STOP MODE.

PNC processes: Axes are stopped, commands will not be further processed.

PNX processes: Axes are stopped, commands will not be further processed.

PAB processes: If a traverse command is currently active, the axis is stopped and the command is not processed any further. Otherwise all the other commands are processed as if STOP were not active.

This difference enables e.g. operator prompting or communication via serial ports to be continued in STOP MODE.

Each parallel process has an individual pointer to an actual data channel and an individual (local) S0 character buffer (max. 80 characters), which is deleted at the start of the automatic mode.

The system does not change any outputs on entering the automatic mode (START). The system resets all outputs when the automatic mode is exited (CANCEL).

1.4.2 System functions

The following definitions can be made for system functions:

- Program at START (Start program)
- Program at STOP (Stop program))
- Program at START after STOP (START after STOP program))
- Program at MALFUNCTION)
- Program at initial position

The above system functions can be activated in different ways:

	System function				
Activated by	START	STOP	START after STOP	END CANCEL	MAL-FUNCTION
Keyboard	X	X	X	X	-
Digital inputs (external)	X	X	X	X	-
"Start program"	X	X	X	X	
PAB process	-	X	X	X	-
PNC process/PNX process	-	-	-	X	-
Diagnosis port	X	X	X	X	-
RS232 Online	X	X	X	X	-
Interbus-S command	X	X	X	X	-
Profibus-DP command	X	X	X	X	-
RS232 monitoring	-	-	-	-	X
Interbus-S monitoring	-	-	-	-	X
Profibus-DP monitoring	-	-	-	-	X
Axis monitoring	-	-	-	-	X
Program at MALFUNCTION	X				

Table 1: System functions

System function START

The program defined as "program at START“ (start program) is registered in the first parallel run and processed. Further parallel runs can be executed as required by means of the RUN commands.

When the start program reaches the END command, the automatic operation is ended.

System function STOP

All active positioning operations are stopped and all parallel runs started as PNC/PNX programs are not processed any further (like SLEEP). After all the positioning operations have been stopped, the stop program is processed (if defined)

The parallel runs started as PAB programs are only affected by the stop operation in that the positioning operations are stopped. Otherwise they are further processed.

System function START after STOP

The START after STOP program is processed after the end of the stop program (if defined). All stopped positioning operations are then restarted and all parallel runs further processed.

System function END-CANCEL

All positioning operations are stopped. Parallel runs are not processed any further, and the automatic mode of the controller is exited (System function MALFUNCTION). A stop program (if defined) is not processed.

System function MALFUNCTION

All positioning operations are stopped. Parallel runs are not processed any further. The "Program at “MALFUNCTION“ will be processed (if defined). The automatic mode is only exited after confirmation.

The following confirmations are possible:

- BREAK command
- ESC key on the keyboard
- "External START" via digital input without "External STOP"
- CANCEL command from port (Profibus-DP, RS232-Online, etc.)

After exiting the "AUTOMATIC mode", an attempt is made to reset the malfunction.

The system malfunctions are archived in the controller. Up to 50 malfunctions with the following information content can be stored:

- Time (from operating hours counter)
- Error number (with text)

If more than 50 system errors have occurred, the oldest messages will be overwritten.

1.4.3 Control of the automatic mode via external inputs

Definition:

I [START]	Start input from the system parameters (push button NO contact, output from a master controller)
I [STOP]	Stop input from the system parameters (push button NO contact, output from a master controller)
O [READY]	Ready output from the system parameters (signal lamp, input from a master controller)
O [AUTOMATIC ACTIVE]	Is set by a command (On: =) at the beginning of the program (signal lamp, input from a master controller)
O [AUTOMATIC STOPPED]	Is controlled by commands in the programs "STOP", "START AFTER STOP" and "MALFUNCTION" (signal lamp, input from a master controller)

Table 2: Definition of external inputs

1.4.3.1 Start of automatic operation with Cancel

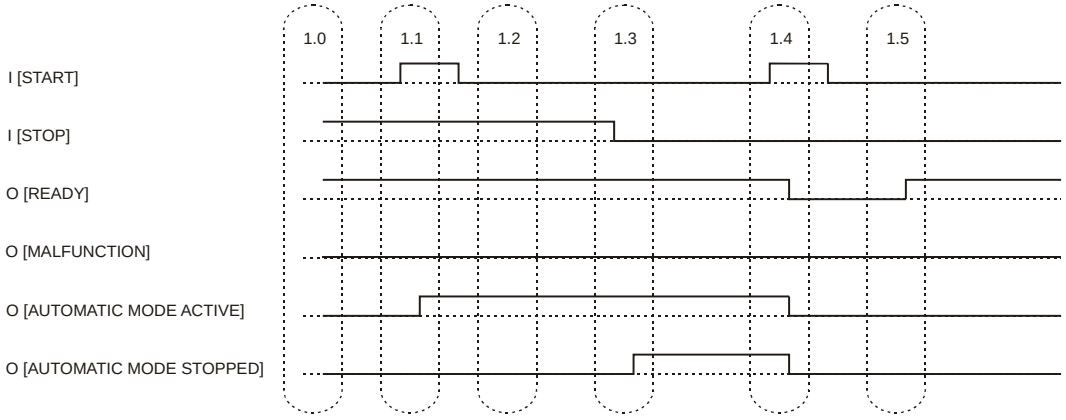


Bild311E

Flow diagram 1: Start automatic mode

Description of the statuses:

1.0 → PA-CONTROL is in the initial position, no error
1.1 → Automatic mode is started
1.2 → PA-CONTROL is in automatic mode, no error
1.3 → PA-CONTROL is stopped
Waiting time of "START" t1 ≥ 10 ms
1.4 → PA-CONTROL aborts the automatic mode and switches to the initial position
1.5 → PA-CONTROL is in the initial position, no error

1.4.3.2 Start of automatic operation with STOP and START AFTER STOP

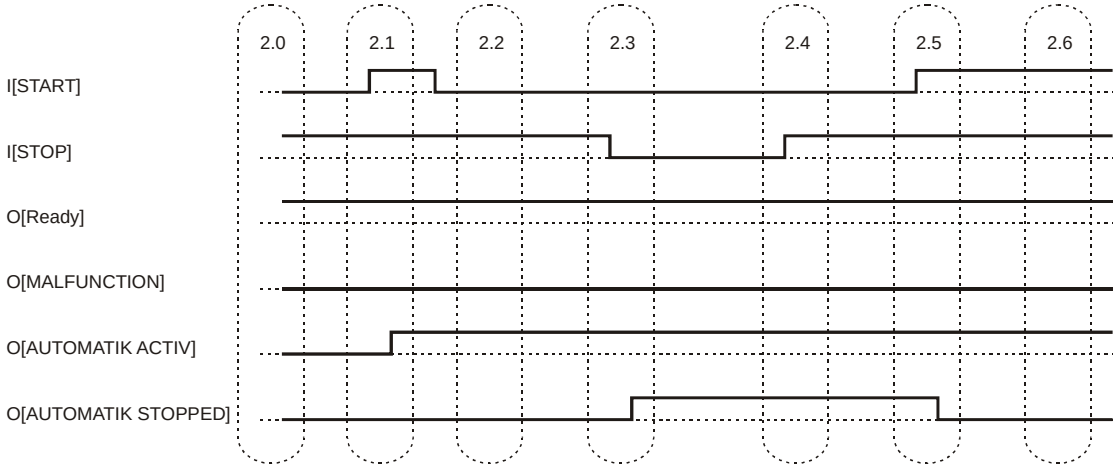


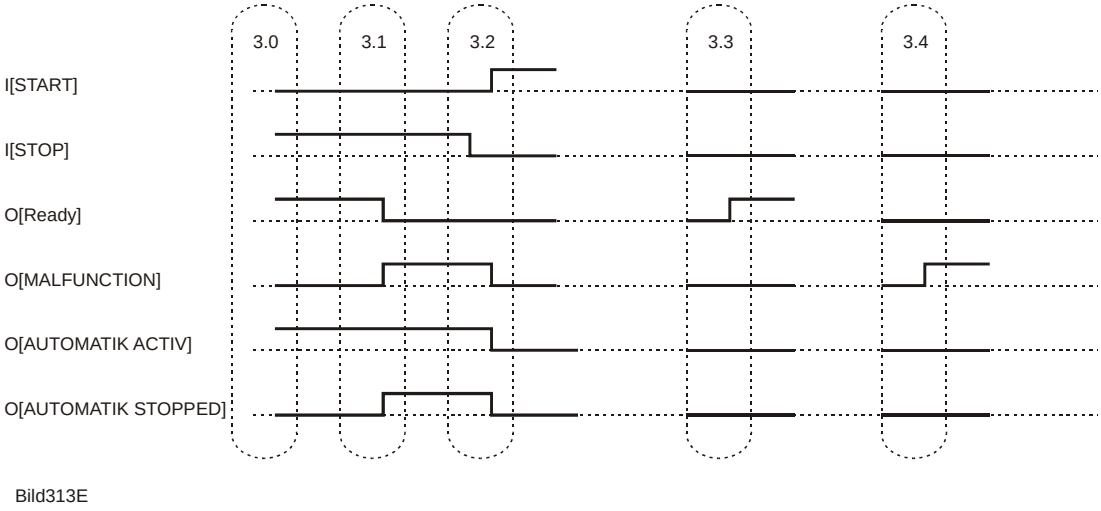
Bild312E

Flow diagram 2: Start automatic mode with STOP and START after STOP

Description of the statuses:

2.0 → PA-CONTROL is in the initial position, no error
2.1 → Automatic mode is started
2.2 → PA-CONTROL is in automatic mode, no error
2.3 → PA-CONTROL is stopped
2.4 → STOP input is de-activated
Waiting time $t \geq 2ms$
2.5 → START input is activated, PA-CONTROL continues in automatic mode
2.6 → PA-CONTROL is in automatic mode, no error

1.4.3.3 PA-CONTROL is in automatic mode and a MALFUNCTION occurs



Flow diagram 3: Sequence of malfunction in automatic mode

Description of the statuses:

3.0 → PA-CONTROL is in AUTOMATIC mode, no error

3.1 → Malfunction occurs

3.2 → Malfunction is acknowledged, AUTOMATIC mode is aborted, PA-CONTROL switches to the initial position

3.3 → PA-CONTROL is in the initial position and the malfunction is corrected, possible errors:

- Limit switch actuated
- Value too high
- Rotation monitoring
- Power section not ready
- Temperature error

3.4 → PA-CONTROL is in the initial position, the malfunction is still present, possible cause:

- Error in the supply voltage
- etc.

NOTE PA-CONTROL must be switched off and the hardware error corrected!

1.4.3.4 PA-CONTROL is in the initial position and a malfunction is present

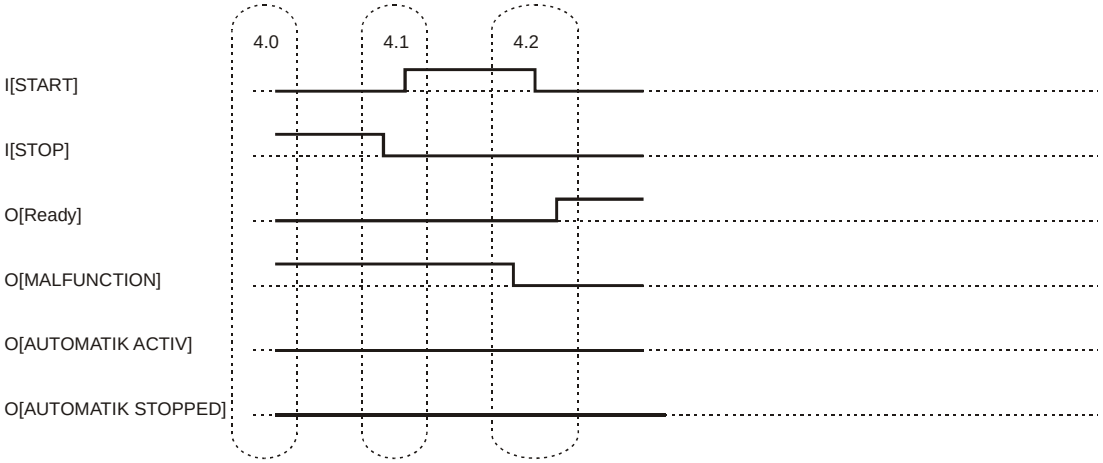


Bild314E

Flow diagram 4: Sequence of malfunction, PA-CONTROL in the initial position

Description of the statuses:

4.0 → PA-CONTROL is in the initial position, a malfunction is present
waiting time $t_1 \geq 80ms$
4.1 → The error is reset
4.2 → The error has been reset, PA-CONTROL signals READY

1.4.4 Program structure

A program has a program name and an extension. The extension identifies the program type.

The program name consists of a maximum of 20 alphanumeric characters (0-9, A-Z). The only special characters which are legal in the name are the hyphen "-" and underscore "_".
The program type consists of three letters.

A program consists of single commands which are processed one after the other and must be concluded with the END command.

A program can call a further program as a subroutine by means of the SUB command (up to 16 nesting levels are possible) or activate a further parallel run by means of the RUN command (up to 31 parallel run programs).

Conditional jumps and unconditional jumps to labels can be executed within a program. The label names are only known within a program. The same label names can therefore be used in different programs.

Jump commands from one program to another program are not possible.

A label name consists of a maximum of 20 alphanumeric characters (0-9, A-Z).
The only special characters which are legal in the name are the hyphen "-" and underscore "_". The label name is concluded by a blank character, semicolon or the line end. A label must be defined in the first column of a line headed by the character "$". A label can only be defined once within a program. The label can be jumped to from different positions in the program.

1.5 Explanations

Command code

Description:

The command is described in general terms in this section.

NOTE Here we will point out particular features and provide you with help by means of cross references.

Command format:

mmmm nnnnn

The commands are divided into an operand and an operator, in other words, a command code and a number.

NOTE The spaces between the operand and the operator have been included in the command format section for the sake of clarity. They are illegal for input into PA-CONTROL!

Several commands can be written in one program line. Commands in one line must be separated by at least one space.

Application:

This section explains the practical reference of the command.

CAUTION

The examples have been prepared without knowledge of the actual mechanical system and must always be checked to ensure their application is appropriate.

Example

1	*I8.1*	*The controller waits until input 8 is logically 1*
2	*I4.0*	*The controller waits until input 4 is logically 0*
3	*END*	

The numbers at the beginning of the lines are the line numbers generated by the editor in PA-CONTROL and are only used for orientation in the program editor.

The examples can be transferred directly. As in PA-CONTROL, comments are separated by a semicolon and can also be transferred to the program provided they do not exceed the maximal line length of 80 characters. As a general principle, all programs are concluded with an END command.

NOTE When transferring a program from the PC to PA-CONTROL, you can specify in the selection window for the program which is to be transferred whether the program should be transferred to the controller with or without comments.

If two semicolons are written in front of the comment during programming, the comment is always transferred to the controller!

1.6 Organisation of Program

The commands of the program group “organisation of programs” are able to implement branches and loops, starting sub programs, starting, end and stop parallel processes and are responsible for the error management.

1.6.1 Program sequence

1.6.1.1 BREAK - Cancel automatic mode

BREAK

Description:

The BREAK command cancels the automatic mode of PA-CONTROL. PA-CONTROL returns to the main menu.

Command format:

BREAK

Application:

The BREAK command is used to discontinue the automatic cycle of PAC when e.g. a machine error has occurred in a subroutine. Using the BREAK command avoids returning to the program structure, in order to return to the main program from several subroutine levels and from there to branch to the end of the program.

NOTE If the BREAK command is used in "Program at malfunction", the user will not receive a message via error number "EXXX".
PA-CONTROL goes to the initial position. Error analysis cannot be carried out.
If a "Program at STOP" has been specified, it will not be processed!

Example

```
1   $LOOP
2   I10.1                ;Start
3   SUB GET              ;Subroutine for get part
4   SUB PROCESS          ;Subroutine for process part
5   SUB DROP             ;Subroutine for drop part
6   G21 I11.1 LOOP       ;Continue production
7   END
```

Program: GET.PNC

```
1   G421.1.100 ERROR     ;Error, if "get part" operation not completed in 1 s
2   O1:=1                ;Open gripper
3   I1.0 I2.1            ;gripper open
4   O2:=1                ;Lower gripper
5   I3.0 I4.1            ;Gripper down
6   O1:=0                ;Close gripper
7   I2.0 I1.1            ;Gripper closed
8   O2:=0                ;Lift gripper
9   I4.0 I3.1            ;Gripper up
10  G401.1
11  END
12  $ERROR
13  I12.1                ;ACKNOWLEDGE ERROR FOR GET PART
14  BREAK
```

1.6.1.2 START - Start automatic run

START

Description:

The **START** command starts programs and positioning commands which have been stopped and cancels the Stop command. This means that traversing commands which have been stopped are restarted and ended at their specified position.

Command format:

START

Application:

The **START** command restarts axes which have been stopped and continues the processing of PNC/PNX processes which have been stopped (started by RUN).

NOTE The START command can only be used in a PAB program.

Example

```
1   $LOOP
2   I10.1                ;Start
3   RUN LOOK             ;Start of a PAB program
4   RUN CONTINUE         ;Start of a PAB program
5   SUB PROCESS          ;Subroutine for processing the part
6   SUB DROP             ;Subroutine for dropping the part
7   G21 I11.1 LOOP       ;Continue production
8   END
```

Program: LOOK.*PAB*

```
1   G421.1.100 Error     ;Error if "get part" operation not completed in 1 s
2   I1.0 I2.1            ;Gripper open
3   I3.0 I4.1            ;Gripper down
4   I2.0 I1.1            ;Gripper closed
5   I4.0 I3.1            ;Gripper up
6   G401.1
7   END
8   $ERROR
9   STOP                 ;Activate system function STOP
10  END
```

Program: CONTINUE.*PAB*

```
1   $LOOP                ;Beginning of loop
2   G21 I3.1 Start       ;To start if input 3 is hot
3   JMP LOOP             ;Go to beginning
4   $Start               ;Jump label START
5   Start                ;Activate system function START after STOP
6   END
```

1.6.1.3 STOP - Stop automatic run

STOP

Description:

The STOP command stops all axes, irrespective of whether a PNC or a PAB program was started. All processes behave as specified in RUN. This means that PAB programs continue to run.

Command format:

STOP

Application:

The STOP command is used to stop all positioning commands, axes and PNC processes started by RUN. For example, if an application error occurs, the program can be stopped in an elegant way without being aborted.

NOTE The STOP command can only be used in a PAB program.

Example

```
1   $LOOP
2   I10.1
3                          ;Start
4   RUN GRIPPER            ;Start of a PAB program
5   M11.1
6   SUB PROCESS            ;Subroutine for processing the part
7   SUB DROP               ;Subroutine for dropping the part
8   G21 I11.1 LOOP         ;Continue production
9   END
```

PAB program: GRIPPER.*PAB*

```
1   M10.1                  ;Wait for finished flag
2   G421.1.200 Error       ;Error if "get part" operation not completed in 2 s
3   O1:=0 O2:=1            ;Open gripper
4   I1.0 I2.1              ;Gripper open
5   O3:=0 O4:=1            ;Lower gripper
6   I3.0 I4.1              ;Gripper down
7   O1:=1 O2:=0            ;Close grippe
8   I1.1 I2.0              ;Gripper closed
9   O3:=1 O4:=0            ;Raise gripper
10  I4.0 I3.1              ;Gripper up
11  G401.1
12  M11:=1
13  END
14  $ERROR
15  STOP                   ;Activate system function STOP
16  END
```

1.6.1.4 Ni:=LINE - Request for the current line number of a program

Ni:=LINE

Description:

The line number of a program can be determined by means of the LINE command, in order to subsequently branch back to this point in the program by means of the command "JMP-LINE...".

The command "Ni:=LINE" determines the line number of its own program, i.e. the line in which this command is located.

The commands "Ni:=LINE.name" or Ni:=LINE.Si search for the program name in all TASKS and their sub-program levels.

If the program is not active, the result is 0. However, if the program does not exist, error message "E513=Program does not exist" is output.

NOTE Only S1 to S16 can be used as a string!

Command format:

Ni:=LINE

Ni:=LINE.Name

Ni:=LINE.Si

1.6.1.5 End - ending

1.6.2 Jump and branches

1.6.2.1 JMP - General form of the unconditional jump

JMP

Description:

An "unconditional jump" to the defined label is executed in the program by means of the function JMP.

Command format:

JMP (Label)

Application:

For program loops

Example

```
1   $ENDLESS
2   I1.1              ;Wait until input 1 is hot
3   O1:=1
4   T100
5   O1:=0
6   JMP ENDLESS       ;Jump to the label "ENDLESS"
7   END
```

NOTE A jump target (label) or a program name can be denoted both in uppercase and lowercase characters. Numeric characters and letters can be used.
PA-CONTROL works internally only with uppercase letters.
The jump targets "BEGINNING" and "beginning" are identical!

1.6.2.2 G21 - Conditional jump

G21

G21 (Condition) (Label)

Description:

A "conditional jump" is executed by means of the function **G21**. The condition defined in the operand can be the logical status of an input, output, flag or system flag. If the condition is fulfilled, a jump to the label defined in the operator is executed, and the program is continued there. If the condition is not fulfilled, the next line is processed.

Command format:

G21 I1.0 Label
G21 O2.1 Label
G21 M4.1 Label
G21 M!3.1 Label
G21 SM51.1 Label

Application:

Program branch depending on the logical status of an input, output, flag or system flag.

Example

```
1    G21 I1.1 ON_5          ;If input 1 is hot, the processing of the program is
                            continued at the label "ON_5", otherwise with the
                            next command
2    O1:=1 T100 O1:=0       ;Output 1 set for 1 second
3    $EIN_5
4    T10
5    G21 O5.1 FLAG_31       ;If output 5 is set to 1, the program is continued at
                            the label "FLAG_31", otherwise with the next
                            command
6    O1:=1 T100 O1:=0       ;Output 1 set for 1 second
7    $FLAG_31
8    T10
9    G21 M31.0 END          ;If flag 31 is set to logical 0, the program is continued
                            at the label "END", otherwise with the next command
10   O1:=1 T100 O1:=0       ;Output 1 set for 1 second
11   $END
12   T10
13   END
```

1.6.2.3 Start of a program at a specified program line

JMP-LINE.Ni

Description:

A specified line in the program can be jumped to by means of this command.

NOTE If the jump target is in an interpolation (second, third G01 command) or if the command jumped to is a G161, G162 or FB command, error "E577=incorrect jump target" is output.

NOTE The line number of an FB command cannot be determined via the functions "Ni:=LINE.Name" or "Ni:=LINE-Sn". As a result, it is not actually possible to jump to this command using the JMP-LINE (or JMP-LINE-IPO) command.

Example

Application: <N2> are modified through a PAP-program for example

```
1     N2:=2
2     $A
3     JMP-LINE.N2
4     JMP A
5     ;
6     O1:=1 T10 O1:=0 T10
7     JMP A
8     ;
9     O1:=2 T10 O2:=0 T10
10    JMP A
11    ;
12    O3:=1 T10 O3:=0 T10
13    JMP A
14    ;
15    END
```

CASE.JMP.N

CASE.JMP.(Variable)

Description:

The CASE.JMP command is a program branch depending on the contents of an integer register. Jump distributors can be easily created using this command.

PA-CONTROL checks the contents of the integer register and jumps to the labels in accordance with the value. If the content of the integer register is less than 1 or greater than the number of elements in the label table (in this example: (RED), (BLUE); (YELLOW), (PINK)), a jump to the label after ELSE is executed. Otherwise a jump to the corresponding label is executed.

Command format:

CASE.JMP.Nn
(Label1)
(Label2)
...
(Label_i)
ELSE Label_e

Or with indirect addressing of the register
CASE.JMP.N!n
(Label1)
(Label2)
...
(Label_i)
ELSE Label_e

Application:

Program branches in the case of more than two possibilities, e.g. various manufacturing types, branch by reason of a counter content, etc.

Example

```
1    $GREY
2    N8:=N13                ;Read value via register
3    CASE.JMP.N8            ;Check contents of N8 and branch if:
4    (RED)                  ;N8=1 : Jump to label "RED"
5    (BLUE)                 ;N8=2 : Jump to label "BLUE"
6    (YELLOW)               ;N8=3 : Jump to label "YELLOW"
7    (PINK)                 ;N8=4 : Jump to label "PINK"
8    ELSE GREY              ;Jump to label "GREY" if N8<1 or N8>4 (value of
                            N8 is not within the specified range of 1 to 4)
9    $RED
10   O1:=1 T100 O1:=0       ;Set output 1 for 1s
11   JMP END
12   $BLUE
13   O2:=1 T100 O2:=0       ;Set output 2 for 1s
14   JMP END
15   $YELLOW
16   O3:=1 T100 O3:=0       ;Set output 3 for 1s
17   JMP END
18   $PINK
19   O4:=1 T100 O4:=0       ;Set output 4 for 1s
20   $END
21   END
```

1.6.3 Call subroutines

1.6.3.1 SUB - Subroutine call

SUB

Description:

The subroutine call **SUB** enables the user to create a well-structured program. Thus, functions which are required several times can be stored in a subroutine. The programs thereby remain manageable and the subroutines can be tested individually.

The subroutine call **SUB** executes a jump to a program stored in PA-CONTROL. The name can be explicitly stated (see chapter *1.6.3.2, page 61*) or taken from a string (see chapter *1.6.3.3, page 63*), or an integer register *(see chapter 1.6.3.4, page 64*).

NOTE A subroutine may not call itself. Subroutines must be concluded with the END command. Up to 15 subroutine calls may be nested.

NOTE A jump target (label) or a program name can be denoted both in uppercase and lowercase characters. Numeric characters and letters can be used. PA-CONTROL works internally only with uppercase letters. The jump targets "BEGINNING" and "beginning" are identical!

Command formats:

SUB.name	(see chapter *1.6.3.2, page 61*)
SUB.S0	(see chapter *1.6.3.3, page 63*)
SUB.S1	(see chapter *1.6.3.3, page 63*)
SUB.N1	(see chapter *1.6.3.4, page 64*)
SUB.N2.PNX	(see chapter *1.6.3.4, page 64*)

G22 I1.1 Name
G22 O2.1 Name
G22 M2.0 Name
G22 M!3.1 Name
G22 SM6.0 Name (see chapter *1.6.3.5, page 65)*

CASE.SUB.(Variable) (see chapter *1.6.3.6, page 66*)

1.6.3.2 SUB.name - Standard subroutine call

SUB.name

Description:

A jump to a stored program with the stated name is executed by the subroutine call **SUB.name**.

Command format:

SUB.name

Example on next page

Example

```
1       I2.1
2       SUB LAMBLI          ;Subroutine "LAMBLI" is called and processed
                            ;After the END command in the subroutine has
                            been detected, the next command in the calling
                            program is processed (in this case I3.1)
3       I3.1
4       I4.1
5       SUB LAMBLI          ;Subroutine "LAMBLI" is called and processed,
                            ;After the END command in the subroutine has
                            been detected, the next command in the calling
                            program is processed (in this case I5.1)
6       I5.1
7       END
```

Program: LAMBLI

```
I1      O1:=1               ;Output 1 is set for 1s, then is reset
2       T100
3       O1:=0
4       END
```

Examples of nesting:

Program : Example

```
1   I2.1
2   SUB LAMBLI      →→→
3   I5.1            ←←←
4   END
```

Program : LAMBLI

```
1   O1:=1
2   T100
3   O1:=0
4   END
```

Program: Example

```
1   I2.1
2   SUB TEST        →→→
3   I5.1            ←←←
4   END
```

Program: TEST

```
1   T10
2   SUB LAMBLI      →→→
3   I6.1            ←←←
4   END
```

Program: LAMBLI

```
1   O1:=1
2   T100
3   O1:=0
4   END
```

SUB.Sx

Command format:

SUB.S0*[.Program type]*

SUB.Sn*[.Program type]*

SUB.S0 / SUB.Sn / SUB.S0.<*Program type*> / SUB.Sn.<*Program type*>

The program name of the program to be called is in a string. If the program does not exist, error message "E513 – Program does not exist" is output.

The statement of the program type (PNC, PAB or PNX) is a command option. If a program type is not specified, the subroutine is called without a check of the program type.

If the program type, e.g. SUB.S0.PNX, is stated, a check is made as to whether the program exists and corresponds to the requested type. If there is no match, error message "E513 – Program does not exist" is output.

Example

Content of an S0 string in ASCII format: **BLINK**

1	*$A*	
2	***N2:=PROGSTAT.S0***	*Status check, program name from string S0*
3	*T1*	
4	*M1:=N2>0*	*Program exists*
5	*G21 M1.0 A*	
6	***SUB.S0***	*Call of the program whose name is in the S0 string*
7	*JMP A*	
8	*END*	

Program: BLINK

I*1*	*$BEGINNING*	
2	*O10:=1*	*Set output 10*
3	*T10*	*Wait 100ms*
4	*O10:=0*	*Reset output 10*
5	*T10*	*Wait 100ms*
6	*END*	

NOTE When executing the SUB.S0 and SUB.Sn commands the content of the string is taken over up to a maximum length of 20 characters or up to the first blank!

1.6.3.4 SUB.Nn – Subroutine call, program name is in an integer register

SUB.Nn

Command format:

SUB.Nn*[.Program type]*

SUB.Nn.i*[.Program type]*

SUB.Nn / SUB.Nn.i / SUB.Nn.<Program type> / SUB.Nn.i<Program type>

The content of the stated integer register is converted to a string consisting of numerical characters. This string provides the program name of the desired subroutine. At the command SUB.Nn.i, the N register is converted to a string with "i" characters. If the number in the N register is smaller than the specified number of slots, leading zeros are added (see following example).

The statement of the program type (PNC, PAB or PNX) is a command option.
If a program type is not specified, the subroutine is called without a check of the program type.

If the program type, e.g. SUB.N24.PNC, is stated, a check is made as to whether the program exists and corresponds to the requested type. If there is no match, error message "E513 – Program does not exist" is output.

Example

1	*$A*	
2	*N5:=4567*	*Load the number 4567 to integer register N5*
...		
10	*SUB.N5*	*Call of the program 4567 without check of the program type*
...		
20	*SUB.N5.PNC*	*Call of the program 4567 and check of the program type PNC*
...		
30	*SUB.N5.7*	*Call of the program 0004567 without check of the program type. The content of the register is converted with three leading zeros, i.e. 7 characters.*
...		
40	*SUB.N5.10.PNX*	*Call of the program 0000004567 and check of the program type. The content of the register is converted with six leading zeros, i.e. 10 characters.*
41	*END*	

Program: 4567

***I**1*	*$BEGINNING*	
2	*O10:=1*	*Set output 10*
3	*T10*	*Wait 100ms*
4	*O10:=0*	*Reset output 10*
5	*T10*	*Wait 100ms*
6	*END*	

1.6.3.5 G22 - Conditional Subroutine call

G22

G22 (Condition) (Name)

Description:

A "Conditional subroutine call" is executed by means of the function **G22**. The condition defined in the operand can be the logical status of an input, output, flag or system flag. If the condition is fulfilled, the subroutine call defined in the operand is executed, and the program is continued there. As soon as the subroutine has been processed or if the condition is not fulfilled, the next command in the calling program is processed.

Command format:

G22 I1.1 Name

G22 O2.1 Name

G22 M2.0 Name

G22 M!3.1 Name

G22 SM6.0 Name

Application:

A subroutine is executed depending on the logical status of an input, output, flag or system flag.

Example

```
1   G25.A1 G90          ;Approach to reference point A1 axis, absolute
                        dimension system
2   G22 I1.1 OUTP_2     ;If input 1 is hot, the subroutine "OUTP_2" is
                        executed and processing continued with the next
                        command in the calling program
3   G22 M21.1 OUTP_2    ;If flag 21 is set, the subroutine "OUTP_2" is
                        executed and processing continued with the next
                        command in the calling program
4   G22 O7.1 OUTP_2     ;If output 7 is set, the subroutine "OUTP_2" is
                        executed and processing continued with the next
                        command in the calling program
5   END
```

Program: OUTP_2

```
1   O2:=1 T100 O2:=0 END    ;Set output 2 for 1s
```

CASE.SUB.N

CASE.SUB.(Variable)

Description:

The CASE.SUB command is a program branch depending on an integer register. Program distributors can be easily created using this command. PA-CONTROL checks the contents of the integer register and executes a subroutine call in accordance with the value. If the content of the integer register is less than 1 or greater than the number of elements in the names table (in this example: SMALL, MEDIUM, LARGE), a jump to the label defined after ELSE is executed. Otherwise the respective subroutine is called up and processed. Processing is continued with the program line after the ELSE branch (in this example: Line 8 with O12:= 0).

The following assignments apply:

Value in register	Name in name table
1	1st Name
2	2nd Name
....	....
N	nth Name

NOTE The subroutines must exist; otherwise the error message "subroutine not found" is displayed.

Subroutines may not activate themselves.

Command format:

CASE.SUB.Nn
(Name1)
(Name2)
...
ELSE Label_e

Or with indirect addressing of the register:

CASE.SUB.N!n
(Name1)
(Name2)
...
ELSE Label_e

Application:

Program branch in the case of more than two possibilities: e.g. various manufacturing types, variants of traversing frequencies.

Example

```
1    $ERROR
2    N8:=N13                ;Read value via register
3    CASE.SUB.N8            ;Check contents of N8 and call subroutine if:
4    (SMALL)                ;N8=1: process subroutine "SMALL"
5    (LARGE)                ;N8=2: process subroutine "LARGE"
6    (MEDIUM)               ;N8=3: process subroutine "MEDIUM"
7    ELSE ERROR             ;Jump to label "ERROR", if the value of N8 is outside
                            the specified limits (here if N8<1 or N8>3)
8    O12:=0
9    I1.1
10   END
```

Program: SMALL

```
1    O1:=1 T100 O1:=0       ;Set output 1 for 1s
2    END
```

Program: LARGE

```
1    O2:=1 T100 O2:=0       ;Set output 2 for 1s
2    END
```

Program: MEDIUM

```
1    O3:=1 T100 O3:=0       ;Set output 3 for 1s
2    END
```

1.6.4 Loops

1.6.4.1 INC - Loop with conditional jump

INC

INC.(Counter).(Target value) (Label)

Description:

The INC command is used to create program loops. It can be compared with the Pascal command Repeat...Until.

The INC command checks the contents of the integer register Nn (example N2). If the contents of the integer register are less than the default value of the integer register or number, the content (N2) is incremented by 1 and a jump to the label is executed. Otherwise processing is continued with the next command (next line).

NOTE The INC command is only applicable for integer registers.

Command format:

INC.Nn.Nn Label

INC.Nn.n Label

Application:

For the creation of program loops whose number of runs is stored as a number or in an integer register, e.g. process "n" parts, traverse "n" times, etc.

Example

```
1   N3:=5                   ;Number of loops equals 5
2   N2:=1                   ;Loop counter
3   $BEGINNING              ;Beginning of loop
4   O1:=1
5   T100
6   O1:=0
7   T100
8   INC.N2.N3 BEGINNING     ;Repeat or jump to "BEGINNING", provided
                            N2<N3.
```

Or:

```
8   INC.N2.5 BEGINNING      ;Repeat or jump to "BEGINNING", provided N2<5.
9   END
```

1.6.4.2 DEC - Loop with conditional jump

DEC

DEC.(Counter) (Label)

Description:

The DEC command is used to create program loops. It can be compared with the Pascal command "Repeat...until" or the C command "Do... while".

The DEC command checks the contents of the integer register. If the contents of the integer register are greater than 0, it is decremented by 1 and a jump to the label executed. Otherwise the next command (next line) is processed.

NOTE The DEC command is only applicable for integer registers.

Command format:

DEC.Nn Label

Application:

For the creation of program loops whose number of runs is stored in an integer register, e.g. process "n" parts, traverse "n" times, etc.

Examples of loops using the DEC command:

Example 1

```
1    N2:=8                    ;Number of loops
2    $BEGINNING
3    O1:=1
4    T100
5    O1:=0
6    T100
7    DEC.N2 BEGINNING         ;All returns executed? no, then jump to
                              "BEGINNING"
8    END
```

Example 2

The loop program section is passed through at least once (Repeat...Until, Do...While), i.e. the program section is always processed. Afterwards a check is made (DEC.) as to whether the program section has to be processed once again.

```
1    N3:=10
2    $DO_STH
3    O1:=1
4    T100
5    O1:=0
6    T100
7    DEC.N3 DO_STH
8    END
```

Example 3

The loop program section does not run through depending on the value (While), i.e. the register is checked first (DEC.N3), and the program section is then processed depending on the contents of the register.

```
1    N3:=15
2    $LOOP
3    DEC.N3 DO_STH
4    JMP DONE
5    $DO_STH
6    O1:=1
7    T100
8    O1:=0
9    T100
10   JMP LOOP
11   $DONE
12   END
```

1.6.5 Parallel sequence

1.6.5.1 RUN - Start of a parallel run

RUN

Description:

Further tasks (processes) can be started in parallel by means of the command RUN.

During processing of the RUN command, the program type (*.PNC/ *.PNX / *.PAB) is checked and the parallel run started accordingly.

The difference between the run of a PNC/PNX program and that of a PAB program can only be seen if a stop is executed in AUTOMATIC mode.

The following applies for STOP in automatic mode:

For the PNC/ PNX run (task):

- Axes are stopped
- Programs are stopped (like SLEEP) and not processed any further

For the PAB run (task):

- Programs are further processed
- If a traversing command is being processed, or a traversing command is to be activated, the run stops at this command

The different operating principles of these program runs offer an advantage. If PA-CONTROL is stopped, output to the display, communication via the serial port and/or the flashing of signal lights can be implemented by means of PAB runs.

NOTE The program is processed until it is ended by an END command or CANCEL.

NOTE A program can only be registered once in the run interpreter. Up to 31 programs (tasks) can be processed in parallel in the process interpreter.

Command format:

RUN *NAME*

RUN.Sn (from version 5.24)

RUN.Sn.*Programmtype* (from version 5.24)

Application:

Start of parallel sequencing programs.

Example 1

PNC program

```
1    M1:=0                ;Reset flag 1
2    M2:=0                ;Reset flag 2
3    RUN FLASHING         ;Start PAB program "FLASHING"
4    RUN OPEN_DOOR        ;Start program "OPEN_DOOR"
5    M1.1                 ;Wait until flag 1 has been set to logical "1"
6    M2.1                 ;Wait until flag 2 has been set to logical "1"
7    O3:=1                ;Set output 1
8    END
```

PAB program "OPEN_DOOR.PAB"

```
1    O30:=1               ;Set output 31, open door
2    I30.1                ;Door is open
3    O30:=0               ;Reset output 30
4    M2:=1                ;Set flag 2, door open
5    END
```

PAB program "FLASHING.PAB"

```
1    $B_CYCLE
2    O2:=1                ;Set output 2
3    T20                  ;Waiting time 200ms
4    O2:=0                ;Reset output 2
5    M1:=1                ;Set flag 1
6    JMP B_CYCLE          ;Jump to label "B_CYCLE"
7    END
```

While the PAB program "OPEN_DOOR" is terminated after the opening and therefore deleted from the parallel run interpreter, the PAB program "FLASHING" runs until it is stopped through another process by the command "CANCEL".

1.6.5.2 SLEEP - Stopping a parallel run

SLEEP

Description:

Operational programs can be stopped by means of the SLEEP command. The program is placed in a sleep mode by the SLEEP command of PA-CONTROL, i.e. it is no longer processed and remains at the current command. This program can be continued from the current command by means of a RUN command.

NOTE The SLEEP command can only be applied to programs which are in RUN mode.

Command format:

SLEEP *name*
SLEEP.Sn (from version 5.24)
SLEEP.Sn.*program type* (from version 5.24

Application:

Stopping of parallel run programs, e.g. if a safety door is opened.

Example

PNC program

```
1   M1:=0          ;Reset flag 1
2   M2:=0          ;Reset flag 2
3   RUN OFF1       ;Start program "OFF1"
4   M1.1           ;Wait until flag 1 has been set to logical "1"
5   SLEEP OFF1     ;Stop program "OFF1"
6   M1:=0          ;Reset flag 1
7   I1.1           ;Wait until input 1 has been set to logical "1"
8   RUN OFF1       ;Continue program "OFF1"
9   M2.1           ;Wait until flag 2 has been set to logical "1"
10  O3:=1          ;Set output 1
11  END
```

PAB program: "OFF1.PAB"

```
1   O1:=1          ;Set output 1
2   M1:=1          ;Set flag 1
3   M1.0           ;Wait until flag 0 has been set to logical "1"
4   T10            ;Waiting time 100ms
5   O1:=0          ;Reset output 1
6   M2:=1          ;Set flag 2
7   END            ;End program "OFF1"
```

1.6.5.3 CANCEL - Ending a parallel run

CANCEL

Description:

The CANCEL command terminates operational programs.

NOTE The CANCEL command can also be used for programs that are not in the *RUN* or *SLEEP* mode.

The program will be aborted at the command which is currently being processed, i.e. elements which the command is controlling may have to be maintained (flag, outputs, registers).

NOTE If the program addressed by the CANCEL command does not exist in the controller, the controller stops with error E513, Program does not exist.

Command format:

CANCEL *name*

CANCEL.Sn (from version 5.24)

CANCEL.Sn.*program type* *(*from version 5.24

Application:

Immediate termination of operational programs. This creates space in the process interpreter for a new program, e.g. when changing the operating mode of a manufacturing facility.

Example

```
1   M1:=0              ;Reset flag 1
2   M2:=0              ;Reset flag 2
3   RUN OFF1           ;Start program "OFF1"
4   I7.1               ;Wait until input 7 is hot
5   CANCEL OFF1        ;End program "OFF1"
6   END
```

Program: AUS1.PAB

```
1   $BEGINNING
2   O1:=1              ;Set output 1
3   T10                ;Waiting time 100ms
4   O1:=0              ;Reset output 1
5   T10                ;Waiting time 100ms
6   JMP BEGINNING      ;Jump to label "BEGINNING"
7   END
```

CASE.RUN

Description:

The CASE.RUN command starts parallel runs depending on an integer register. Program distributors can be easily set up using this command.

PA-CONTROL checks the contents of the integer register and starts a parallel run in accordance with the value. If the contents of the integer register are less than 1 or greater than the number of elements in the name table (in this example: PROG-1, PROG-2, PROG-3), a jump to the label defined after ELSE is executed. Otherwise the respective PAB program is called and continued with the program line after the ELSE branch (in this example: Line 8 with O12:= 0).

The following assignments apply:

Value in register	Name in the name table
1	1st Name
2	2nd Name
...	...
n	nth Name

NOTE The programs must exist; otherwise the error message "program does not exist" is output.

Once a parallel run has been started, it may not be started again. Otherwise the error message "program already running" is output.

Command format:

CASE.RUN.Ni
(Name1)
(Name2)
...
ELSE Label_e

Application:

Depending on e.g. different production types, various processing operations can take place parallel to other operations. These operations can be variably initiated by means of CASE.RUN.

Example

```
1    G11.0                      ;Switch off run display
2    G500.0                     ;LC display is current data channel
3    G501                       ;Clear display
4    $ERROR                     ;
5    G503.1.1                   ;Position cursor
6    G510.INPUT PROG. NUMBER    ;Output of user request
7    G542.24.1.2.N8             ;Input value via the keyboard
8    CASE.RUN.N8                ;Check contents of N8 and start PAB program
                                at:
9    (PROG-1)                   ;N8=1 : start program PROG-1
10   (PROG-2)                   ;N8=2: start program PROG-2
11   (PROG-3)                   ;N8=3: start program PROG-3
12   ELSE ERROR                 ;Jump to label "ERROR if value of N8 is out of
                                the specified range (in this case if N8<1 or
                                N8>3)
13   O12:=0
14   I1.1
15   END
```

Program: PROG-1.PAB

```
1    O1:=1 T100 O1:=0           ;Set output 1 for 1s
2    END
```

Program: PROG-2.PAB

```
1    O2:=1                      ;Set output 2
2    I1.1                       ;Wait for input 1
3    O2:=0                      ;Reset output 2
4    END
```

Program: PROG-3.PAB

```
1    O3:=1 T100 O3:=0           ;Set output 3 for 1 s
2    END
```

CASE.SLEEP

Description:

Parallel sequences can be stopped depending on an integer register by means of the CASE.SLEEP command. Program distributors can be easily set up using this command.

PA-CONTROL checks the contents of the integer register and starts a parallel run in accordance with the value. If the contents of the integer register are less than 1 or greater than the number of elements in the name table (in this example: PROG-1, PROG-2,) a jump to the label defined after ELSE is executed. Otherwise the respective program is activated and continued with the program line after the ELSE branch.

The following assignments apply:

Value in register	Name in the name table
1	1st Name
2	2nd Name
...	...
N	nth Name

NOTE The programs must exist; otherwise the error message "program does not exist" is the output.

Once a parallel run has been stopped, it may not be stopped again. Otherwise the error message "program already in sleep (see *page 73*) mode" is output.

Command format:

CASE.SLEEP.Ni
(Name1)
(Name2)
...
ELSE Label_e

Application:

Depending on e.g. different production types, various processing operations can take place in parallel to other operations. These operations can be variably stopped by means of CASE.SLEEP.

Example

```
1     RUN PROG-1                ;Start of the program PROG-1
2     RUN PROG-2                ;Start of the program PROG-2
3     $BEGINNING
..    ...
19    G11.0                     ;Switch off run display
20    G500.0                    ;LC display is current data channel
21    G501                      ;Clear display
22    $ERROR                    ;
23    G503.1.1                  ;Position cursor
24    G510.INPUT PROG. NUMBER   ;Output of user request
25    G542.24.1.2.N10           ;Input value via the keyboard
26    CASE.SLEEP.N10            ;Checks the contents of N10 and stops the
                                corresponding PAB program at N10:
27    (PROG-1)                  ;N10=1: stops program PROG-1.PAB
28    (PROG-2)                  ;N10=2: stops program PROG-2.PAB
29    ELSE ERROR                ;Jump to label "ERROR if value of N10 is out
                                of the specified range (in this case if N10<1
                                and N10>2).
..    ...
100   END
```

Program: PROG-1.PNC

```
1     $BEGINNING                ;Jump label
2     O1:=1                     ;Set output 1
3     I1.1                      ;Wait until input 1 is hot
4     O1:=0                     ;Reset output 1
5     T100                      ;Wait 1s
6     JMP BEGINNING             ;Go to label BEGINNING
7     END
```

Program: PROG-2.PAB

```
1     $BEGINNING                ;Jump label
2     O2:=1                     ;Set output 2
3     I2.1                      ;Wait until input 2 is hot
4     O2:=0                     ;Reset output 2
5     T50                       ;Wait 0.5s
6     JMP BEGINNING             ;Go to label BEGINNING
7     END
```

CASE.CANCEL

Description:

Parallel sequences can be ended depending on an integer register by means of the CASE.CANCEL command.

PA-CONTROL checks the contents of the integer register and ends a parallel run on the basis of the value stored in the integer register.

If the contents of the integer register are less than 1 or greater than the number of elements in the name table (in this example: PROG-1, PROG-2), a jump to the label defined after ELSE is executed. After the CANCEL command has been executed without error, processing is continued in the line which follows the ELSE *Label_e*.

NOTE The CASE.CANCEL command can also be executed for programs which are not running (see also *chapter 1.6.5.3, page 74*).

NOTE If the program addressed in the CASE.CANCEL command does not exist in the controller, the controller stops with error E513, Program does not exist.

The following assignments apply:

Value in register	Name in the name table
1	1st Name
2	2nd Name
...	...
n	nth Name

Command format:

CASE.CANCEL.Ni
(Name1)
(Name2)
...
ELSE Label_e

Application:

Depending on e.g. different production types, various processing operations can arise which run in parallel or are dormant. The undesired parallel processes can be ended by means of CASE.CANCEL.

Example:

```
1     RUN PROG-1                ;Start of the program PROG-1
2     RUN PROG-2                ;Start of the program PROG-2
3     $BEGINNING
..    ...
19    G11.0                     ;Switch off run display
20    G500.0                    ;LC display is current data channel
21    G501                      ;Clear display
22    $ERROR                    ;
23    G503.1.1                  ;Position cursor
24    G510.INPUT PROG. NUMBER   ;Output of user request
25    G542.24.1.2.N20           ;Input value via the keyboard
26    CASE.CANCEL.N20           ;Checks the  contents of N20 and stops the
                                corresponding PAB program at:
27    (PROG-1)                  ;N20=1: stops program PROG-1.PAB
28    (PROG-2)                  ;N20=2: stops program PROG-2.PAB
29    ELSE ERROR                ;Jump to label "ERROR if value of N10 is out
                                of the specified range (in this case if N10<1
                                and N10>2).
..    ...
100   END
```

Program: PROG-1.PNC

```
1     $BEGINNING                ;Jump label
2     O1:=1                     ;Set output 1
3     I1.1                      ;Wait until input 1 is hot
4     O1:=0                     ;Reset output 1
5     I2.1                      ;Wait until input 2 is hot
6     JMP BEGINNING             ;Go to label BEGINNING
7     END
```

Program: PROG-2.PAB

```
1     $BEGINNING                ;Jump label
2     O2:=1                     ;Set output 2
3     T50                       ;Wait 0.5s
4     O2:=0                     ;Reset output 2
5     T50                       ;Wait 0.5s
6     JMP BEGINNING             ;Go to label BEGINNING
7     END
```

1.6.5.7 PROGSTAT - Get program status

Ni:=PROGSTAT

Ni:=PROGSTAT.xxx

Description:

The command checks whether the program has been activated as a task (by means of RUN or CASE.RUN). If the program cannot be found as a current task in the run interpreter, a check is made to ascertain that the program exists in PA-CONTROL.

After the check, the following information is stored in register Ni:

Contents of Ni		Status
0	→	Program does not exist in PA-CONTROL
1	→	Program exists in PA-CONTROL, but is not an active TASK
2	→	Exists as a task, is being processed
3	→	Exists as a task, is not being processed (SLEEP)

Command format:

Ni:= PROGSTAT.Name
N!i:=PROGSTAT.Name
Ni:= PROGSTAT.S0
Ni:= PROGSTAT.Sn
Ni:= PROGSTAT.Nn
Ni:= PROGSTAT.Nn.i

Example:

1	*N10:=PROGSTAT.DOOR_OPEN*	*Is the program "DOOR_OPEN" active or does it exist in PA-CONTROL? Storage of the result in integer register N10.*
2	*$A*	
3	*N5:=128*	*Load the number 128 to integer register N5*
...		
10	*PROGSTAT.S0*	*Inquire the status of the program whose name is in string S0*
20	*PROGSTAT.S1*	*Inquire the status of the program whose name is in string S1*
...		
30	*PROGSTAT.N10*	*Inquire the status of the program whose name is in integer register N10*
...		
40	*PROGSTAT.N5.7*	*Inquire the status of program 0000128. The command PROGSTAT adds four leading zeros to the three-figure number from integer register N5*
41	*END*	

1.6.6 Error Management

1.6.6.1 ERROROFF / ERRORON

ERROROFF / ERRORON from V5.07

Description

The “ERROROFF” and “ERRORON” command is used to activate or deactivate monitoring of a status which results in a system error.

NOTE In principle (default status), PA-CONTROL monitoring is activated.

Command format:

ERROROFF<error number>.<axis number>

ERRORON<error number>.<axis number>

Example of application for a PA-CONTROL MP:

During PA-CONTROL MP, a motor voltage of 24 V DC is supplied from an external source. However the internal monitoring for this voltage requires at least 60 V DC.

The “MonitorMotorVoltage” flag for the PA-CONTROL-MP is activated when the PA-CONTROL is switched on (and only then!).

The “ERROROFF.103.A1” command can be used to deactivate monitoring. This can be done in the “Normal position program”.

The “When changing to readiness for travel mode” setting would be the correct setting for approval mode.

List of possible system errors with the respective axis types:	
Error number	Axis types
E103	PLS-MP

Example

Normal position.pnc

```
1     ERROROFF.123.A1        Deactivate monitoring error E103
2     T10                    Wait 100ms
3     END
```

Start.pnc

```
1     R1:=RA1.1
2     R4:=RA1.6
3     $A
4     G21 I9.1 REFG26
5     G21 I10.1 REFG25
6     G21 I7.1 MOVE NEGATIVE
7     G21 I8.1 MOVE POSITIVE
8     G21 I6.1 MOVE
9     JMP A
10    $REFG26
11    G26.A1.R1
12    I9.0
13    JMP A
…     ;
14    $REFG25
15    G25.A1
16    I10.0
17    JMP A
18    $MOVE NEGATIVE
19    A1:=0
20    I7.0
21    JMP A
…     ;
22    $MOVE POSITIVE
23    G123.A1.I8.1
24    A1:=R2
25    I8.0
26    JMP A
…     ;
27    $MOVE
28    FA1:=R1
29    G100.A1.R4
30    A1:=100
31    A1:=0
32    JMP A
…     ;
33    END
```

1.7 Commands for axis

1.7.1 General

1.7.1.1 The statuses of an axis

Each of the 16 possible axes of a PA-Control may be in different statuses (*see Figure 1, below, for explanations see Table 3: Explanations for statuses of an axis, page 85).*

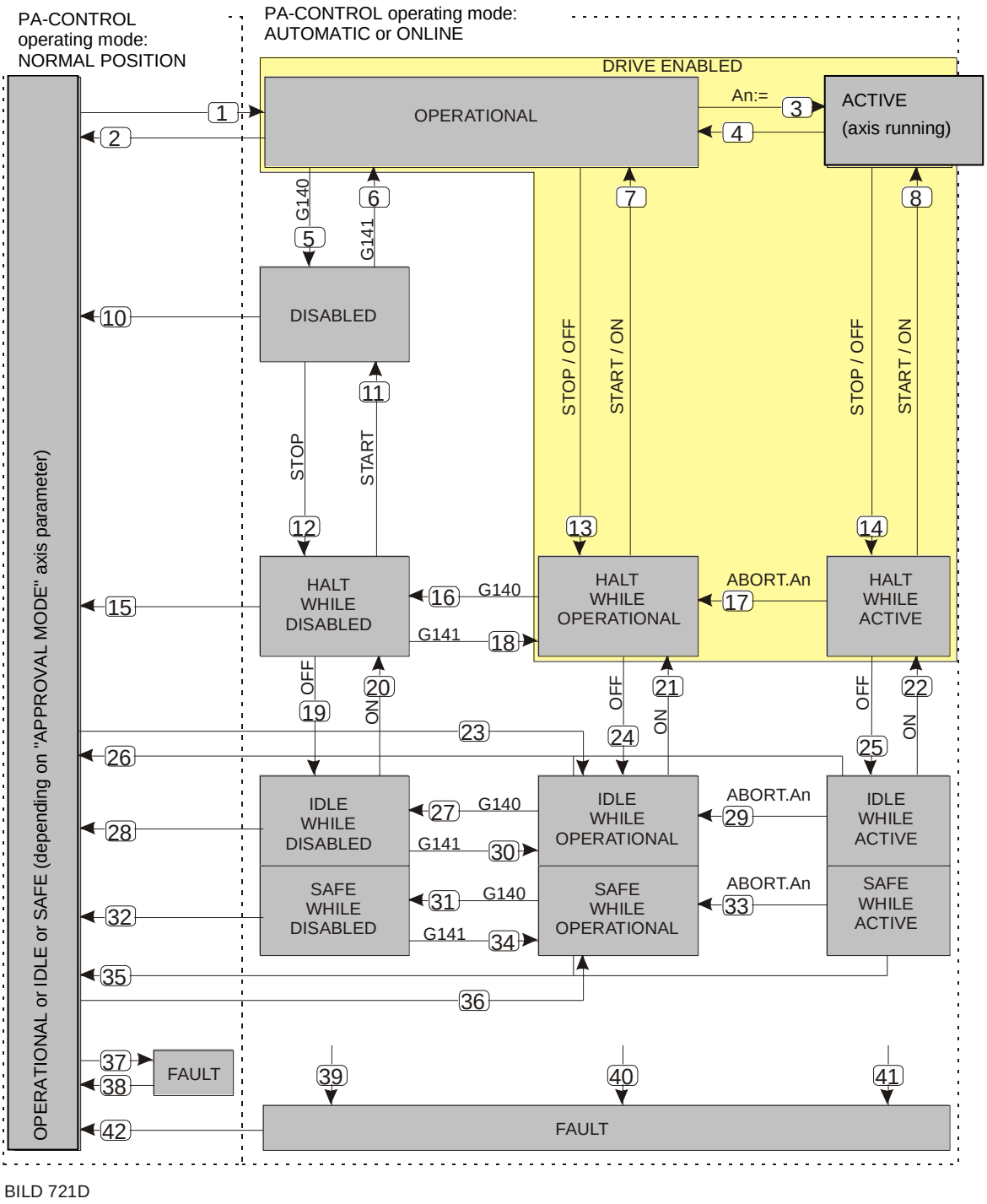


Figure 1: Status of an axis

Explanation of terms in *Figure 1:*

Status	Description
OPERATIONAL (1), (4)	The motor is energised and in its position.
ACTIVE (3), (8)	The motor is running, the axis is moving.
HALT WHILE OPERATIONAL (13), (17)	The motor is energised and is in position.
HALT WHILE ACTIVE (14), (22)	The motor is energised and is stationary. A traverse order has been stopped.
DRIVE ENABLED (1), (6), (18), (21), (22)	The motor is energised. The axis is either in OPERATIONAL, ACTIVE or HALT status.
IDLE WHILE OPERATIONAL (24), (29), (39)	The motor is switched to de-energise, (DRIVE DISABLED).
IDLE WHILE ACTIVE (25), (26)	A traverse order for the axis has been stopped and the motor has then been switched to de-energise (DRIVE DISABLED).
SAFE WHILE OPERATIONAL (33), (34), (36)	The motor is switched to de-energise. (DRIVE DISABLED). The “Safe halt” function has been activated via the “PUT-SAFE” output with the start inhibit on the amplifier.
SAFE WHILE ACTIVE	A traverse order for the axis has been stopped and the motor has then been switched to de-energise. (DRIVE DISABLED). The “Safe halt” function has been activated via the “PUT-SAFE” output with the start inhibit on the amplifier.
DISABLED (5), (11)	The motor is switched to de-energise, (DRIVE DISABLED).
HALT WHILE DISABLED (12), (16), (20)	The motor is switched to de-energise, (DRIVE DISABLED).
IDLE WHILE DISABLED (19), (27)	The motor is switched to de-energise, (DRIVE DISABLED).
SAFE WHILE DISABLED (31)	The motor is switched to de-energise, (DRIVE DISABLED). The “Safe halt” function has been activated via the “PUT-SAFE” output with the start inhibit on the amplifier.
FAULT (37), [(39), (40), (41)]	The axis or drive has an error.

Table 3: Explanations for statuses of an axis

For more information, please refer to the following chapters:

1.7.2.1 G140 - Change to DISABLED status, page 86,
1.7.2.2 G141 - Change to OPERATIONAL status, page 87
1.7.2.3 OFF.An - Deactivate axis, page 88
1.7.2.4 ON.An - Activating axis, page 89
1.7.2.5 STOP.An - Interrupt readiness for travel for an axis, page 90
1.7.2.6 START.An - establish readiness for traversing for an axis, page 90
1.7.2.7 ABORT.An - Abort a stopped travel command, page 91

1.7.2 Modes

1.7.2.1 G140 - Change to DISABLED status

G140

Description:

As the command is executed, the axis changes to DISABLED status. Previously the command activated measuring mode.

Function:

The G140.An (A0, A[1..16]) command activates the "OFF-AXIS" function. The axis does not however change after HALT and then after IDLE or SAFE, but after DISABLED ((5)).

NOTE

- If the axis is in the ACTIVE status, the "E523 – axis still running" process error is set.
- If the axis is in the HALT WHILE ACTIVE status, the "E584 – axis is in HALT status" process error is set.
- If the axis is in the DISABLED, HALT WHILE DISABLED, IDLE WHILE DISABLED or SAFE WHILE DISABLED status and a travel command is activated on this axis, the"E571 – command not possible with this axis" process error is produced.

Actions during activation:

- Brake is activated (option),
- Current zeroing is activated in the output stage,
- Monitoring for speed sensor and power element readiness is deactivated.

Result:

- Output stage can be switched off (operational safety),
- Axis can be traversed manually,
- The axis can no longer be traversed by the control (Positioning commands are blocked and produce an error message).

Command format:

G140.An [A1....A16]

EXAMPLE

Please, *see chapter 1.7.2.2, page 87*

NOTE Not useful for PA-CONTROL with PLS-6/ -7/ -8/ -9

1.7.2.2 G141 - Change to OPERATIONAL status

G141

Description:

As the command is executed, the axis changes to OPERATIONAL status. Previously the command deactivated measuring mode.

Actions with PA-CONTROL MP:

- Motor is energised again,
- Brake is deactivated (option),
- Synchronisation to speed sensor is undertaken (option),
- Axis position is taken from speed sensor position (option),
- Monitoring for speed sensor and power element readiness is activated.

Actions for LV-servoTEC:

- Approve controller,
- Adopt values for speed sensor position

Result:

The axis can again be traversed by the control.

NOTE With a PA-CONTROL-MP, when exiting the DISABLED status and with synchronisation to the speed sensor, the axis is traversed a maximum of 4 full motor steps in a positive direction.

Command format:

G141.An [A1....A16]

NOTE Not useful for PA-CONTROL with PLS-6/ -7/ -8/ -9.

EXAMPLE

```
1   G25.A1      ;Approach reference point with A1 axis
2   A1:=100     ;Move to position 100
3   I5.1        ;Wait until setup mode is active, protective
                circuit bridged
4   G140.A1     ;Switch A1 axis to DISABLED status, axis
                can be moved manually.
5   I5.0        ;Protective circuit active again, setup mode
                is blocked
6   G141.A1     ;Axis A1 executes actions as described
                above
7   A1:=1000    ;Move to position 1000 with A1 axis
8   END
```

1.7.2.3 OFF.An - Deactivate axis

OFF.AN	from V5.07

Description:
(also refer to application details “APP5004_DE_1070300_PAC_OeffnenDerSchutztuer_LV-servoTEC.pdf”)
The OFF.An (A[0...16]) command activates the "OFF AXIS" function for one axis.

Function:
Starting from different initial situations, the command must have three different impacts.

- If the axis is in OPERATIONAL status, it changes to HALT WHILE OPERATIONAL status (*see Figure 1: Status of an axis on page 84 path (13)).* The "OFF axis" function is then active and the axis is switched to the "DISABLED / SAFE" status (*see Figure 1: Status of an axis on page 84 path (24)).*
- When the axis moves, positioning is stopped immediately and the axis changes to the HALT WHILE ACTIVE status ((14). The "OFF axis" function is then active and the axis is switched to the "DISABLED / SAFE" status (see *Figure 1: Status of an axis on page 84 path (25)*.

If the axis is in the HALT status, it changes to the IDLE or even SAFE status.

Example of the OFF.An and ON.An commands

```
1    $W_ON
2    I3.1
3    G21 I4.0 W_ON
4    ON.A1                    Axis 1 ON
5    ON.A2                    Axis 2 ON
6    ;
7    $W_OFF
8    G21 I3.0 M_OFF
9    G21 I4.0 M_OFF
10   JMP W_OFF
11   ;
12   $M_OFF
13   OFF.A1                   Axis 1 OFF
14   OFF.A2                   Axis 2 OFF
15   JMP W_ON
```

A parallel program runs in AUTOMATIC mode which monitors input 3. If the input is not energised, the axes are switched off (IDLE/SAFE) using the “OFF.A1” and “OFF.A2” commands. Through these commands firstly active positioning is stopped and then the axes are switched off (IDLE).

The staff protection restart inhibit (KSI+/-) in the LV-servoTEC amplifier is activated using the “PUT-SAFE” axis parameter via outputs 4 and 5. This then bridges the door switches with the “KSO” contacts.

From the standpoint of the axes, the work room is safe and the door can be opened (SAFE).

1.7.2.4 ON.An - Activating axis

ON.An	from V5.07

Description:

The ON.An (A[0...16]) command activates the "ON AXIS" function for an axis selected in the command (*see example on page 88*)

Function:

Starting from different initial situations, the command must have two different impacts.

- If the command is issued and the axis is in IDLE or SAFE status, the axis is switched into HALT status according to the activation traverse mode axis parameter (*see Figure 1: Status of an axis on page 84 paths (21) and (22)*).
- Positioning has been interrupted during "OFF AXIS" function. If permitted by the "activation traverse mode", positioning is continued (see *Figure 1: Status of an axis on page 84* path (22)) once the axis has switched to HALT WHIL ACTIVE (see *Figure 1: Status of an axis on page 84* path (8)).

NOTE

- The "ON AXIS" function can only take effect if the "OFF AXIS" function is not subject to activation (e.g.: "OFF AXIS" function not energised*), (see Figure 2: "Axis position" diagnosis window*
- If the command is issued for an axis in the HALT ACTIVE or OPERATIONAL status, there is no change or impact.

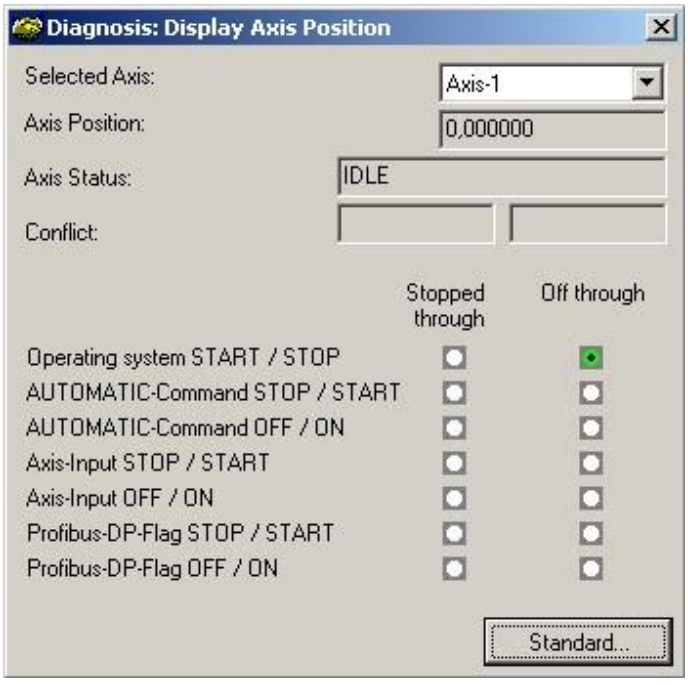


Figure 2: "Axis position" diagnosis window

1.7.2.5 STOP.An - Interrupt readiness for travel for an axis

STOP.An	from V5.07

Description and function:

- The STOP.An (STOP.A[0...16]) command can be used to interrupt a travel command. The axis is stopped and switches to the HALT WHILE ACTIVE status (*see Figure 1: Status of an axis on page 84 path (14)*.

NOTE If the STOP.An command is used with a stationary axis, the axis switches to HALT WHILE OPERATIONAL status (13). A travel command for this axis is then only started when the axis has switched to OPERATIONAL status via a START.An (*see Figure 1: Status of an axis on page 84 path (7).*

Use:

This can be used in all program types (*.PNC, *.PAB, *.PNX)

1.7.2.6 START.An - establish readiness for traversing for an axis

START.An	from V5.07

Description and function:

- The "START-An" (START.A1[...16]) command results in an axis, which has previously been stopped using the "STOP.An" command and is therefore in HALT WHILE ACTIVE status, fully executing the interrupted travel command (see *Figure 2: "Axis position" diagnosis window on page 89 (8)*
- If the axis is in HALT WHILE OPERATIONAL status, it changes to OPERATIONAL status *(see Figure 2: "Axis position" diagnosis window on page 89 path (7))*

NOTE If the "START-An" command is applied to an axis which is currently in position (OPERATIONAL) or is active (ACTIVE), the command has no effect on the axis and the axis status, *see Figure 2: "Axis position" diagnosis window on page 89*.

Use:

This can be used in all program types (*.PNC, *.PAB, *.PNX).

1.7.2.7 ABORT.An - Abort a stopped travel command

ABORT.An from V5.07

Description and function:

- The "ABORT.An" (A0,A1-A16) command can be used to abort the travel command for an axis which has been stopped using the "STOP.An" or “OFF.An” commands. The STOP condition is cancelled and the current position of the axis adopted as the actual and target position. The axis switches from the HALT WHILE ACTIVE status to the HALT WHILE OPERATIONAL status *(see Figure 1, page 84 path (17), (29) and (33)).*

EXAMPLE

Start.pnc

```
1    $A
2    I9.1
3    RUN REFERENZ_A1
4    T20
5    $LOOP
6    G21 SM51.1 REF_OK
7    G21 I16.1 REF_ABORT
8    JMP LOOP
9    ;
10   ;
11   $REF_OK
12   T500
13   JMP A
14   ;
15   $REF_ABORT
16   STOP.A1
17   CANCEL_REFERENZ_A1
18   ABORT.A1
19   JMP A
20   ;
21   END
```

REFERENZ_A1.pnc

```
1    G25.A1
2    END
```

1.7.3 Move axis

1.7.3.1 G25 - Approach to reference point

G25.An

G25.(Axis)

Description:

After the command **G25.An** has been started, PA-CONTROL carries out an approach to reference point with the respective axis (The variable "n" stands for the relevant axis here). The position reached after the approach to reference point is effective as a zero point for the axis (see *chapter 1.7.7, page 115*)

NOTE The parameter settings for the reference speed and the acceleration must be adjusted to the excursion (distance between the reference switch and the mechanical end stop)

For axes with stepper motor the parameter setting of the reference speed and the acceleration must be adapted to the overrun (spacing between reference switch and mechanical stop).

In version 4.xx it was possible to activate every axis with the command G25.A0 for running, without referencing of the axes.

From version 5.00 it can be used the command G26 to run the axes without a reference drive.

With the system flag SM51 (A1) to SM66 (A16) it can be checked if the axes are referenced.

Command

G25.An

The addressed axis, e.g. G25.A7 for axis 7, moves towards the negative limit switch until the latter opens, changes direction and clears the switch at the start/stop frequency. If the enabling bit has been set, the position counter is also set to zero.

Application:

After every restart, e.g. after a power failure or the switch-off of a PA-CONTROL, an approach to reference point is normally required. This is the only way to guarantee that the controller executes its program in the defined manner. Command format 1 is provided for this task.

Example 1

```
1   G25.A1    ;Approach to reference point is carried out for the A1 axis
2   G25.A2    ;Approach to reference point is carried out for the A2 axis
3   G25.A0    ;Setting of the enabling bit for all other axes
4   END
```

1.7.3.2 A1 - Positioning of the axes

A1

A1:=(Operator)

Description:

The positioning command allows the axes to be traversed in the positive and negative direction, regarded from the zero point. The position is the absolute value in the absolute dimension system (G90) and the relative traverse path in the incremental dimension system **(G91)**.

NOTE The end position of the movement is dependent upon the programmed system of units, either the absolute dimension (G90) or the incremental dimension (G91).

Command format:

A1:=nnnn

A1:=Rn

Application:

Positioning of axes

Example

```
1   G25.A1      ;Carry out approach to reference point with A1 axis
2   G90.A1      ;Absolute dimension system
3   A1:=200     ;Axis A1 200 in positive direction
4   A1:=198     ;Axis A1 by 2 in negative direction to absolute position
                198
5   R3:=678
6   A1:=R3      ;Axis A1 in positive direction to absolute position 678
7   G91.A1      ;Incremental dimension system
8   R3:=-200
9   A1:=-10     ;Traverse axis A1 by 10 in negative direction
10  A1:=R3      ;Traverse axis A1 by -200 in negative direction
11  R15:=3
12  A1:=R!15    ;Traverse axis A1 by -200 in negative direction
13  END
```

1.7.3.3 G90 - Absolute dimension system

G90

G90.(Axis)

Description:

After the **G90.An** command has been invoked, all subsequent positioning movements ensue in the absolute dimension system until the incremental dimension system is switched over by means of the command **G91.An**. The absolute dimension method is the standard dimension system.

Command format:

G90.An

G90.A0

Application:

For dimensions which are all measured from a zero point.

Example

```
1   G25.A1      ;Approach to reference point
2   G90.A1      ;Subsequent positioning commands are executed for axis 1 in the absolute dimension system
3   A1:=10      ;Position 10 is approached with the A1 axis, path = 10 mm [Sketch: 1.]
4   A1:=450     ;Position 450 is approached with the A1 axis, path = 440 mm [Sketch: 2.]
5   A1:=30      ;Position 30 is approached with the A1 axis, path = -420 mm [Sketch: 3.]
6   G90.A0      ;Subsequent positioning commands are executed for all axes in the absolute dimension system
7   END
```

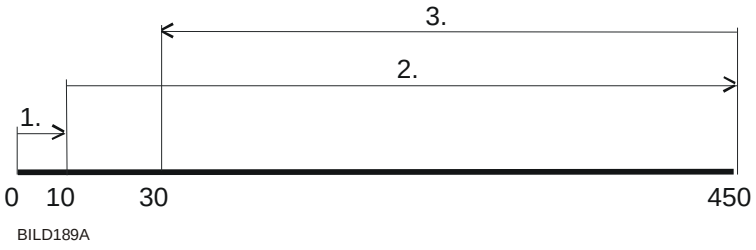


Figure 3: Absolute dimension system

G91

G91.(Axis)

Description:

After the **G91.An** command has been invoked, all subsequent positioning movements ensue in the incremental dimension system until the absolute dimension system is switched over by means of the command **G90.An**.

Command format:

G91.An

G91.A0

Application:

For dimensions which are not referenced to a zero point.

For distances which successively occur several times and can be repeated by means of a program loop (DEC.Ni).

Example

```
1   G25.A1      ;Approach to reference point
2   G91.A1      ;Subsequent positioning commands are executed for the A1 axis in the incremental dimension system
3   A1:=10      ;Position 10 is approached with the A1 axis, path = 10 mm [Sketch:  1.]
4   A1:=450     ;Position 460 is approached with the A1 axis, path = 450 mm [Sketch: 2.]
5   A1:=30      ;Position 490 is approached with the A1 axis, path = 30 mm [Sketch: 3.]
6   G90.A0      ;Subsequent positioning commands are executed for all axes in the incremental dimension system
7   END
```

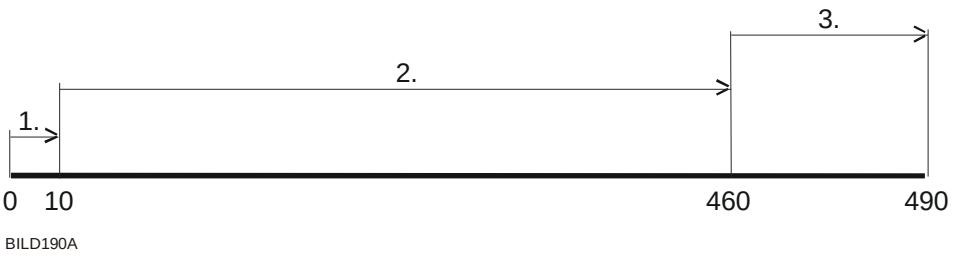


Figure 4: Incremental dimension system

1.7.3.5 G100 - Specifying the acceleration

G100

G100.(Axis).(Value)

Description:

"An" stands for the designation of the axis and can assume the name of all existing axes in multi-axis modules (number of axes varies depending on the equipment).

G100.B is used to set the path acceleration of the interpolation. The path speed **FB** must be set beforehand.

NOTE If the value is higher or lower than the limits, the error message "Value too high" or "Value too low" is the output.

The command cannot be executed, if the axis has the following condition: “HALT”, “IDLE” or “SAFE”. The program sequence stops at this command and only if the axis has the condition “OPERATINAL” the command will be executed.

Command format:

G100.An.15

G100.An.R7

Application:

Deviating from the values for acceleration stored in the parameter values, a mode of operation tailored to the situation can be achieved by means of the command G100. The specification is effective until the next time the command G100 is used or until automatic mode is exited. If the command G100 is not used, the stored parameter values are effective.

Example

```
1   G25.A1          ;Approach to reference point A1-Axis
2   R7:=20
3   G91.A0          ;Incremental dimension system
4   FA1:=250
5   G100.A1.15      ;Specification of the acceleration for the A1
                    axis
6   A1:=250         ;Travel command for the A1 axis
7   G100.A2.R7      ; Specification of the acceleration for the A2
                    axis in accordance with the contents of real
                    number register 7
8   A2:=500         ;Travel command for the A1 axis
9   END
```

1.7.3.6 FAn - Traversing speed

F

F(Axis):=(Speed)

Description:

The command **F** prompts the controller to traverse at the traversing speed specified in the operator in the further program run. All axes which are actuated by operand F can be traversed at various speeds.

The axis parameters specified in the parameter list can also be loaded via registers.

FB is used to set the path speed during interpolation.

NOTE The command cannot be executed, if the axis has the following condition: “HALT”, “IDLE” or “SAFE”. The program sequence stops at this command and only if the axis has the condition “OPERATINAL” the command will be executed

If a speed F has not been programmed in the program run, all traversing commands are executed at the traversing speed defined in the parameter field.

A speed demand higher than determined in the parameter field, results in the error message “value beyond reach”(E532)

An error message is not the output if the speed value is too low. The lowest possible speed is set instead.

Command format for changing the speed of an axis:

FAn:=200

FAn:=R4

Command format for changing the speed of a path:

FBAn:=200;

FBAn:=R4

Application:

Changing speeds for traversing and moving axes during the program run.

Example 1

```
1   G25.A1          ;Execute approach to reference point with A1 axis
2   G91.A1          ;Set incremental dimension system
3   FA1:=200        ;Traversing speed A1 axis 200 AE/s
4   A1:=1000
5   R4:=500
6   FA1:=R4         ;Traversing speed A1 axis 500 AE/s
7   A1:=10
8   R23:=4
9   FA1:=R!23       ;Traversing speed A1 axis from register R4
10  END
```

Particular features of the PA-CONTROL MP / LV-servoTEC:

In contrast to the other variants of the controller, the speed can be changed in the PA-CONTROL MP and LV-servoTEC while positioning is in progress.

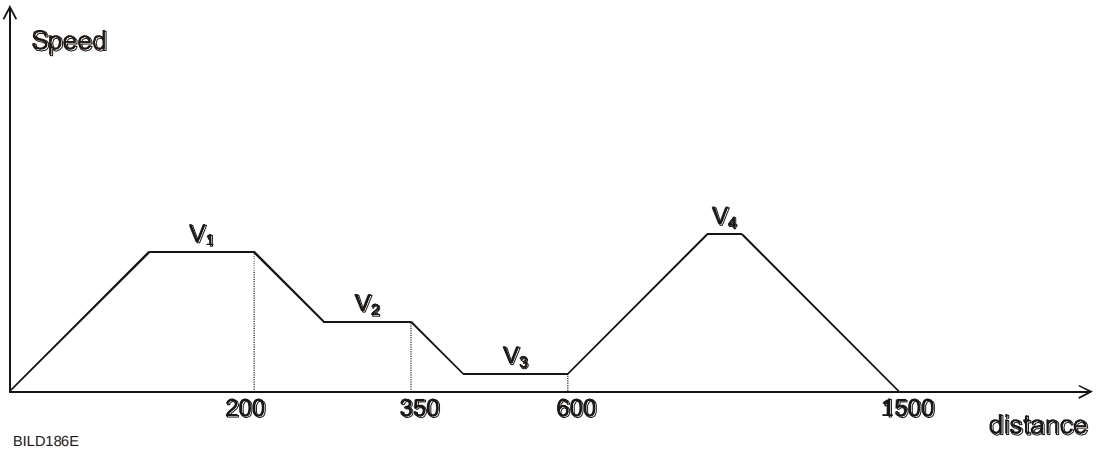


Figure 5: Speed profile

Example: only for PA-CONTROL MP and LV-servoTEC

1	*G90.A0*	*;Positioning ensues in the absolute dimension system*
2	*G25.A1*	*;Approach to reference point*
3	*G100.A1.100*	*;Specify acceleration of the A1 axis*
4	*G210.A0*	*;Process commands beyond line limit*
5	*FA1:=1000*	*;Traversing speed A1 axis v_1: 1000AE/s*
6	*A1:=1500*	*;Positioning of the A1 axis*
7	*G230.1.A1.200*	*;Wait until the current absolute position of the A1 axis is higher than 200*
8	*FA1:=500*	*;Traversing speed A1 axis v_2: 500AE/s*
9	*G230.1.A1:=350*	
10	*FA1:=150*	*;Traversing speed A1 axis v_3: 150AE/s*
11	*G230.1.A1.600*	
12	*FA1:=1200*	*;Traversing speed A1 axis v_4: 1200AE/s*
13	*G213.A0*	
14	*END*	

1.7.3.7 G123 - Traverse as long as condition fulfilled

G123

G123.(Axis) (Condition)

Description:

In the function **G123**, the next traversing command of the respective axis is executed until the condition **n.m** of the operand is effective. If the condition is not fulfilled, the traversing command is aborted and the next command executed. **G123** only acts on the axis defined in the command. If the condition is not fulfilled on starting the axis, the positioning is not started. The program is immediately continued with the next command.

NOTE

The current absolute position is considered for subsequent positioning in the case of aborted traversing commands. The system flag (SM) is then set for the relevant axis.

Only one G123 can be active for each axis. Logical operations of several conditions are possible via the logical operation commands.

The following assignment applies:

SM11: Axis 1 (SM = system flag)

SM12: Axis 2

.....

SM26: Axis 16

Per axis only one G123 command can be active. Links with several conditions are possible with the logical operations commands.

Command format:

G123.An I3.1
G123.An M3.1
G123.An O4.1

Application:

- A work piece is traversed until it is detected. On detection, a sensor or similar transmits a signal back to PA-CONTROL, so that the program can be continued.
- For the teach procedure (manual approaching of a position): the axis is driven as long as the “teach button” is chosen.

Example

```
1    G25.A1 G90.A1          ;Carry out approach to reference point with A1
                            axis and switch over to absolute dimension
                            system
2    R2:=1000               ,Load speed to register R2
3    FA1:=R2                ;Select slow speed
4    G123.A1 I3.0           ;The following traversing command is effective as
                            long as the sensor of input 3 is cold
5    A1:=100                ;Traversing command is executed
6    G21SM11.0 NO_PART      ;No work piece present → the condition (system
                            flag of axis 1, SM11, not set) is fulfilled → a
                            conditional jump to the label NO_PART follows.
                            If a work piece is present → condition not fulfilled
                            → program is continued in the next line.
7    R1:=A1                 ;Real number register is loaded (value 100
                            corresponds to e.g. the edge of the work piece)
8    R1:= R1+10             ;Real number register is increased by 10 (value
                            110 corresponds to the point of the workpiece at
                            which the drill hole is to be set – e.g. centre of the
                            work piece)
9    A1:=R1                 ;Traverse e.g. to the centre of the work piece
10   SUB DRILLHOLE          ;Jump to subroutine/execute
11   $;NO_PART              ;Label NO_PART
12   END
```

Program: DRILLHOLE

```
1    O10:=1 T10             ;Switch on drill machine
2    O16:=1                 ;Switch on pneumatic feed
3    I15.0 I16.1            ;Stop position reached
4    T50                    ;Wait 0.5 seconds
5    O16:=0 O15:=1          ;Pneumatic feed off,
                            backward movement
6    I16.0 I15.1            Idle position reached again
7    O15:=0                 Switch off backward movement
8    O16:=0                 Switch off the drill machine
9    END
```

Explanation of the program example:

In the example, the work piece is traversed until a sensor detects that it is present. If the work piece is present, the sensor is switched on and the program is continued in the next line.

If no work piece is detected, a jump to the label **$NO_PART** takes place.

The system flag (SM) is only set if traversing commands are aborted.

1.7.3.8 G123Q - Travel for as long as condition is satisfied

G123Q from V5.16

G123Q.(axis) (condition)

Description:

With the **G123Q** function, the next travel command for the corresponding axis is executed for as long as the operand's **n.m** condition is valid. If the condition is not satisfied, the travel command is aborted and the next command executed.

The axis is braked. Please use the "OFF ramp" value set in the axis parameters.

G123Q only impacts the axis defined in the command. If the condition is not satisfied when starting the axis, positioning is not started. The system moves on to the next command straight away.

Application:

Monitoring a "crash sensor"

NOTE If travel commands are aborted, the current absolute position for subsequent positioning is taken into account. The system flag (SM) for the corresponding axis is then set.

The following assignment applies:
SM11: axis 1 (SM = system flag)

Only one G123Q can be active per axis. Links between several conditions are possible via the commands for logical links.

The G123Q command can only be used for the LV-servoTEC and LV-servoTEC S2 axis types.

Example

```
1   G25.A1 G90.A1      ;Approach reference point with A1 axis and
                       switch to absolute measuring system
2   R2:=1000           ;Load speed into R2 register
3   FA1:=R2            ; Select slow speed
4   G123QA1 I3.0       ;Following travel command applies for as long as
                       sensor of input 3 is not energised
5   A1:=100            ;Travel command is executed
    END
```

1.7.3.9 G150 - Traverse section with Start-Stop [1]

G150

Description:

The G150 command provides the programmer with the possibility of initiating the braking operation earlier in the next positioning of the relevant axis and to position the section defined by G150 at the Start-Stop speed. The transition from the braking ramp to the Start-Stop speed (section) is continuous.

NOTE The G150 command cannot be used for a LV-servoTEC axis. The syntax check checks the axis type and puts out an appropriate message.

NOTE The number after the axis name in the G150 command is a section and is interpreted unsigned and non-directionally. The set gear factor is considered. The command is only considered for the next positioning of this axis. If the section to be traversed by the next positioning is less than the section defined in the G150 command, the G150 command is ignored and the positioning proceeds as usual.

Command format:

G150.An.200
G150.An.Rn

Application:

For positioning operations, in which the last section is positioned in the start stop mode.
Example: Jointing operation: faster reaction to G123 commands.

[1] Not for PAC-LV-servoTEC

Example

```
1    G25.A1          ;Carry out approach to reference point with A1 axis
2    G91.A1          ;Switch over to incremental dimension system
3    G150.A1.200     ;Traverse the last 200 increments with the A1 axis at the Start-Stop speed in the next positioning
4    A1:=1000        ;Position 1000 increments with the A1 axis. The braking operation is initiated in such a way that the start-stop speed is reached 200 increments before the target position. The last 200 increments are then positioned at the start-stop speed
5    END
```

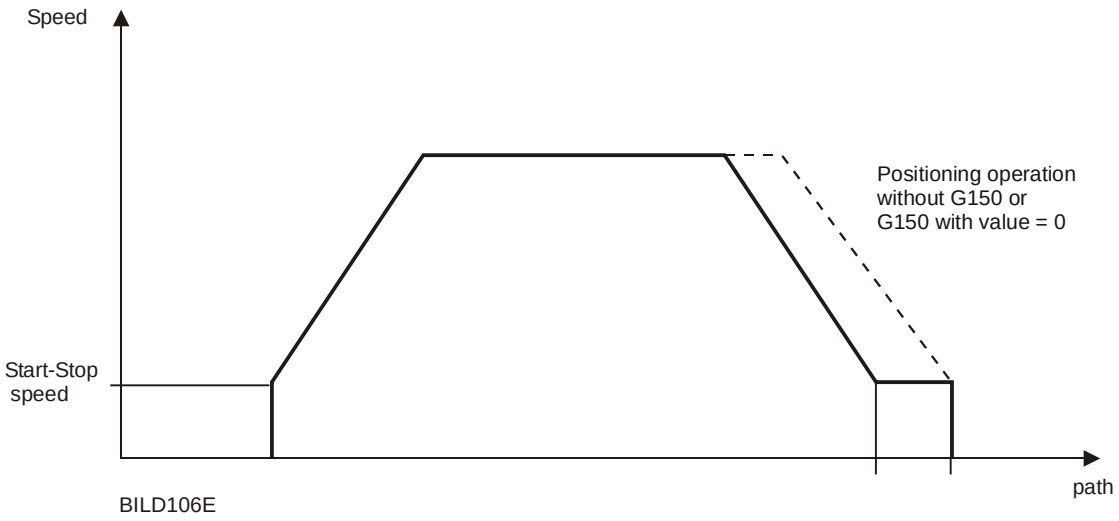


Figure 6: Speed progression when using G150

1.7.3.10 RN:=FAN Request for the set or current speed

Rn:=FAn

Command format:

This command supplies the last set speed of the axis. It does not provide information as to whether the axis is currently traversing at this speed (axis at standstill, axis accelerating, axis traversing or axis braking).

Command format:

Rn:=FAn

.

1.7.4 Position and Parameter

1.7.4.1 G26 - Set position to zero / set to position value

G26

G26.(Axis)
G26.(Axis).konst
G26.(Axis).Rn

Description:

The absolute position counter of each axis can be set to zero by means of the function G26.An and to the position value by means of the functions G26.An.konst and G26.An.Rn. The variable “An” of the operand specifies the selected axis. No range check takes place. The axis can therefore be set to a position outside the traversing range limits.

NOTE

The software limit switches which were specified by the range in the parameter level are transferred unchanged. If this command is repeated, positioning can be carried out practically infinitely in one direction, since no counter overflow and no overtravel of the range limits is possible.

Important: In the PA-CONTROL Version 4.xx and servo axes, driven by LV-servoTEC the counter overflow is possible.

The remainders are deleted in case of gear factors with decimal places not equal to 0, i.e. they cannot be considered during subsequent positioning.

During execution of the command G26 the reference flag is set. Herefore the axis is referenced.

Command format:

G26.An

Application:

An appropriate application case for this PAC command is for continuous movements, e.g. use of the DT140 turntable in only one direction of rotation.

In exchange for the reference run when the position comes from a measure system.

Example 1

```
1   G26.A1      ;Set absolute position of the A1 axis to zero
2   G26.A2      ;Set absolute position of the A2 axis to zero
3   END
```

Command format:

G26.An.konst

G26.An.Rn

Application:

An appropriate application case for this PAC command is the specification of the axis position e.g. on the basis of stored or transferred values.

Example 2

```
1   R113:=342.5
2   G26.A1.234.7      ;Set absolute position of the A1 axis 234.7
3   G26.A2 .R113      ;Set absolute position of the A2 axis 342.5
4   END
```

1.7.4.2 G29 - Set position to dimension

G29

G29.(Axis)

Description:

The position counter of each axis can be set to a new value by means of the function **G29**. The operator specifies the selected axis. The software switches are also changed accordingly.

Command format:

G29.Ai.constant	Sets the current axis position to the value of the constant
G29.Ai.Rn	Sets the current axis position to the value of the R register
G29.Ai Rn.Rm	Sets the axis position so that the axis is positioned on the value of the R register Rn for the position Rm.

NOTE The G29 commands of the versions prior to version 4.71R are rejected completely. The axis position after the approach to reference point G25, G26 is used as a reference.

In case of a servo drive (LV-servoTEC) only the command “G29.Ai.Rn.Rm.” should be used, because by using other command variants the positions may drift away, as they are set in the amplifier.

If the command G29.A1.Rn is used the point of reference run (G25, G26) gets lost. The user has to know how often and in which position the axis was moved so that it can be moved back to the zero position.

With the G29.Ai.Rn.Rm command, a position of an axis can as often be placed or changed as wished. When the G29.Ai.Rm.Rm command is used at last, the offset from the G29 commands will be rejected and the axis can continue without problems with the ‘original coordinates" (from / to G25 command).

Example 1

Machining of three parts

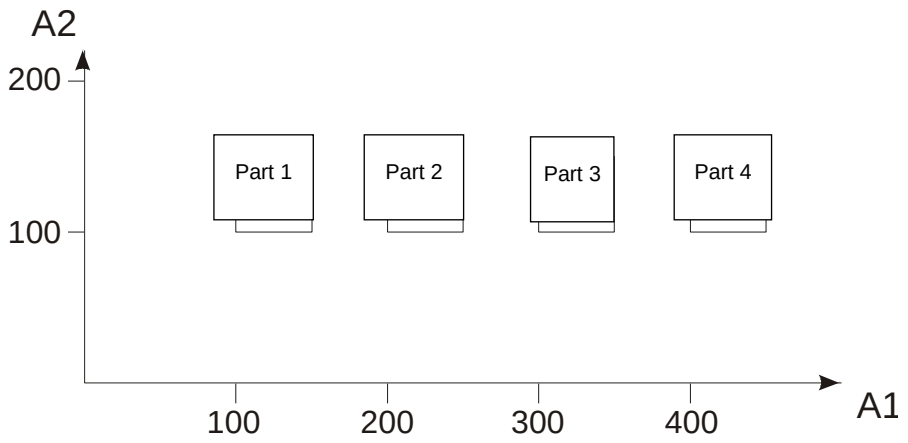


Figure 7: Setting the dimension

Program: START.PNC

```
1     G25.A1 G25.A2
2     R1:=100                  ;A1 position of part 1
3     R2:=200                  ;A1 position of part 2
4     R3:=300                  ;A1 position of part31
5     R4:=0
6     ;
7     $LOOP
8     G29.A1.R4.R1             ; Set axis position for machining part 1
9     SUB MACHINE_PART
10    G29.A1.R4.R2             ; Set axis position for machining part 2
11    SUB MACHINE_PART
12    G29.A1.R4.R3             ; Set axis position for machining part 3
13    SUB MACHINE_PART
14    G29.A1.R4.R4             ; Set axis position for traverse to
                               initial position
15    A1:=0 A2:=0
16    ...
17    JMP LOOP
18    END
19
```

Program: MACHINE_PART

```
1     A!.=0 A2:=100            ; Approach start position for machining part
2     ...
3     ...
4     END
```

1.7.4.3 Rn:=An get axis position

Rn:=An

Nn:=An

Description:

The current axis position is transferred to an N or R register.

The position is available in the R register according to the set unit of measurement.
The step counter is transferred to an N register.

Command format:

R1:=An;

N1:=An

Application:

Storage of axis positions e.g. in the teach program.

Display the axis position and step monitoring in the program.

Example

```
1   N226:=0.01          ;Maximum deviation of the step counter to the encoder position
2   $LOOP               ;Approach position 100 at reduced torque
3   R222:=A1            ;Get axis position
4   R223:=ENC1          ;Get encoder position
5   R25:=R222-R223      ;Calculate difference
6   M221:=R225>R226     ;Compare
7   G21 M221.0 LOOP
8   M220:=0             ;STOP e.g. of an axis started with G123
9   END
```

1.7.5 Limit switches

1.7.5.1 G142 - Limit switch monitoring "OFF" [1]

G142	only for axis type LE3-80 LE8-80

G142.(Axis)

Description:

After the G142.A0 or G142.An command has been invoked, the limit switch monitoring is switched off for one axis (all axes), until the limit switch monitoring for one axis (all axes) is switched on again by the G143.A0 or G143.An command.

Note After each change into the automatic mode, the switch monitoring is activated. Please recognize this note when programming.

Command format:

G142.An

Application:

The command is of particular importance for round axes, which are often positioned beyond the 360° limit.

Example

```
1   G25.A1      ;Approach to reference point
2   G91.A1      ;The following positioning commands are
                executed for axis 1 in the incremental
                dimension system
3   G142.A1     ;Limit switch monitoring for axis 1 OFF
4   A1:=10      ;Position 10 is approached with the A1 axis
5   A1:=450     ;Position 460 is approached with the A1 axis
6   A1:=30      ;Position 490 is approached with the A1 axis
7   G90.A1      ;The following positioning commands are
                executed for axis 1 in the absolute dimension
                system
8   G143.A1     ;Limit switch monitoring for axis 1 ON
9   END
```

Axes control	command implemented
PLS6	Yes
PLS7	Yes
MP	Yes
LV-servoTEC	No
intelliMOT	No
flexmoTEC	No

[1] not for LV-servoTEC

1.7.5.2 G143 - Limit switch monitoring "ON" [1]

G143

G143.(Axis)

Description:

After the G142.A0 or G142.An command has been invoked, the limit switch monitoring is switched off for one axis (all axes), until the limit switch monitoring for one axis (all axes) is switched on again by the G143.A0 or G143.An command.

Command format:

G143.A0

G143.An

Application:

The command is of particular importance for round axes, which are often positioned beyond the 360° limit.

Example

```
1   G25.A1      ;Approach to reference point
2   G91.A1      ;The following positioning commands are
                executed for axis 1 in the incremental
                dimension system
3   G142.A1     ;Limit switch monitoring for axis 1 OFF
4   A1:=10      ;Position 10 is approached with the A1 axis
5   A1:=450     ;Position 460 is approached with the A1 axis
6   A1:=30      ;Position 490 is approached with the A1 axis
7   G90.A1      ;The following positioning commands are
                executed for axis 1 in the absolute dimension
                system
8   G143.A1     ;Limit switch monitoring for axis 1 ON
9   END
```

Axes control	Command implemented
PLS6	Yes
PLS7	Yes
MP	Yes
LV-servoTEC	No
intelliMOT	No
flexmoTEC	No

[1] not for LV-servoTEC

1.7.6 Encoder and following error

1.7.6.1 Ni:=PEAn / Ri:=PEAn - Get following error

Ni:= PEAn

Ri:= PEAn

This applies only:

PLS7-9	PLSMP	LV-servoTEC	LV-servoTEC S2
_	X	X	X

Description:

The current following error of axes which are driven by an LV-servoTEC or a PA-CONTROL MP can be transferred to an N or R register. Following errors of a servo-axis are transferred from the LV-servoTEC to an N or R register. Following errors of the PA-CONTROL MP are transferred from the internal counter to an N or R register.

In this connection, there are two fundamentally different forms:

- Following errors of the axis n in SI units, i.e. in increments or steps, are transferred to an integer register.
- Following errors of the axis n in display devices are transferred to real number registers.

Command format:

Ni:= PEAn

Ri:= PEAn

Application:

Recording of the "following error" in the case of screwing or force limitation

NOTE Approximately 8-10ms is required for execution of the command. This means that a following error can only be acquired at a data rate of approx. 10ms.

Example

```
1   N26:=150          ; Max. following error
2   M22:=0            : Delete cancel flag
3   $LOOP
4   N25:=PEA1         ; Get following error
5   N25:=ABS.N25      ; Generate absolute value unsigned
6   M21:=N25>N26      ; Difference too large
7   G21 M21.0 LOOP
8   M22:=1            ; Cancel
9   END
```

1.7.6.2 ENC - Transfer encoder position [1]

Rn:=ENCn

Ni:=ENCn

(ENCAxisNo)

Description:

The encoder position can be transferred to an N or R register.

NOTE If the encoder value is stored in an R register, the rotary encoder position, which is comparable with the axis position, is stored.

If the encoder value is stored in an N register, the rotary encoder increments are stored adjusted to the steps of the stepping motor.

Command format:

R1:=ENCn

N1:=ENCn

Application:

Step deviations can be detected in the program, e.g. for screwing, for which a specified current can be assigned to a specified torque. When the torque is reached, the motor stops, because the previously set current has been exceeded. The difference between the encoder position and the step counter verifies that the screw has been correctly tightened.

[1] = PA-CONTROL MP

Example 1

```
1    R226:=0.01          ;Maximum deviation of the axis position to the
                         encoder position
2    $LOOP
3    R222:=A1            ;Get axis position
4    R223:=ENC1          ;Get encoder position
5    R224:=R222-R223     ;Calculate difference
6    R225:=ABS.R224      ;Calculate sum
7    M221:=R226>R225     ;Compare
8    G21 M221.0 LOOP
9    M220:=1             ; For triggering a stop e.g.
10   END
```

Example 2

```
1    N226:=32            ;Maximum deviation of the step counter to the
                         encoder position counter. 32 corresponds to the
                         maximum following error of a 2 phase stepping
                         motor of 2 full steps
                         (2 full steps with 16x resolution→32).
2    $LOOP
3    N222:=A1            ;Get step counter
4    N223:=ENC1          ;Get encoder position counter
5    N224:=N222-N223     ;Calculate difference
6    N225:=ABS.N224      ;Calculate sum
7    M221:=N225>N226     ;Compare
8    G21 N221.0 LOOP
9    M220:=0             ; For triggering a stop e.g.
10   END
```

1.7.6.3 Wait for position from the SSI-interface [4]

SSIni.Rm

SSIn.i.Nm

(SSIn.i.<Register>)

Description:

Waits until the position or the count value of the SSI module is larger or smaller than the value of the register (real number or integer register).

- Wait until larger : I = 1
- Wait until smaller : I = 0

Command format:

SSIn.i.Rm

SSIn.i.Nm

Application:

Is used for setting or resetting outputs in the case of high-speed LV-servoTEC axes, in which the repeat accuracy must be less than 10ms.

Example

```
1       R12:=123.45
2       N33:=446
...
n       SSI1.1.R12      ;Waits until the position of the SSI module is
                        higher than the contents of R2, i.e. higher than
                        123.45
...
m       SSI1.0.N33      ; Waits until the counter value of the SSI
                        module is less than the contents of N33s, i.e.
                        less than 446
```

[4] = LV-servoTEC, flexmoTEC

1.7.7 Parameter commands

1.7.7.1 PARAMETER.XX.Ai. - Write LV-servoTEC-Parameters [4]

PARAMETER.XX.Ai.

Parameter. <Axis type>.<Axis>.<Parameter>:=<Value>

Description:

Specific parameters can be sent to a specified LV-servoTEC by means of this command.

CAUTION

The use of this command requires more comprehensive knowledge of amplifiers and experience in their optimisation.

IEF-Werner GmbH excludes any responsibility for the consequences of improper application of the command.

NOTE Please refer to the document "DE_EN_LV-servoTEC_ascii20.chm" or the help function of the program "DRIVE.EXE" for a list of the possible parameters including their meaning.

A list of the possible parameters, including their meaning, can be taken from the enclosed manual for the amplifier "LV flexmoTEC"!

The parameter list for the LV-servoTEC S2 is printed on the next page.

Command format:

PARAMETER.St.Ai.Parameter name:=Constant	(LV-servoTEC S2)
PARAMETER.St.Ai.Parameter name:=Nn	
PARAMETER.St.Ai.Parameter name:=Rn	
PARAMETER.FT.A1.Parameter name:=Nn	(flexmoTEC)
PARAMETER.S2.A1.Parameter name:=Nn	(LV-servoTEC S2)

NOTE The use of an integer or real number register depends on the type of parameter.

List of the parameters of the LV-servoTEC S2:

SYNCHRONISATION_MAIN
SYNCHRONISATION_ENCODER_SELECTION
POWER_STAGE_TEMPERATURE
MOTOR_TEMPERATURE
ENCODER_EMULATION_RESOLUTION
ENCODER_EMULATION_OFFSET
FOLLOWING_ERROR_WINDOW
FOLLOWING_ERROR_TIME_OUT
TORQUE_ACTUAL_VALUE
CURRENT_ACTUAL_VALUE
ENCODER_X10_RESOLUTION
ENCODER_X10_NUMERATOR
ENCODER_X10_DIVISOR
ENCODER_X10_COUNTER

The use and scaling of the parameters can be found in the CANopen manual or the LV-servoTEC S2 amplifier.

Program: start_send.pnc

```
1     $A
2     I9.1
3     N2:=1
4     R3:=0.5
5     PARAMETER.ST.A1.DIS:=N2          ; Send parameter DIS to LV-servoTEC axis 1
                                       from integer register N2
6     PARAMETER.ST.A1.IPEAK:=R3        ; Send parameter IPEAK to LV-servoTEC axis
                                       1 from real number register R3
7     PARAMETER.ST.A1.O1:=1            ; Send parameter O1 to LV-servoTEC axis 1,
                                       the definition ensues with the constant "1"
8     T200
9     I10.1
10    N2:=0
11    R3:=1.5
12    PARAMETER.ST.A1.EN:=N2           ; Send parameter EN to LV-servoTEC axis 1
                                       from integer register N2
13    PARAMETER.ST.A1.IPEAK:=R3        ; Send parameter IPEAK to LV-servoTEC axis
                                       1 from real number register R3
14    PARAMETER.ST.A1.O1:=0            ; Send parameter O1 to LV-servoTEC axis 1,
                                       the definition ensues with the constant "0"
15
16    PARAMETER.FT.A8.RunModeSel
      ection :=N25
17
18    T200
18    JMP A
20    END
```

[4] =LV-servoTEC, flexmo TEC

1.7.7.2 Ri:=PARAMETER.XX.Ai - Reading LV-servoTEC-Parameters [4]

Ri:=PARAMETER.XX.Ai

<Register>:=Parameter <Axis type>.<Axis>.<Parameter>

Description:

Specific parameters can be retrieved from a specified LV-servoTEC by means of this command.

NOTE Please refer to the document "DE_EN_LV-servoTEC_ascii20.chm" or the help function of the program "DRIVE.EXE" for a list of the possible parameters including their meaning.

For the amplifier “LV-flexmoTEC“ please use the relevant manuals that have been supplied.

For LV-servoTEC S2 see the parameter list with the command send PARAMETER.S2.AN.Parametername

Command format:

Ni:=Parameter.St.An.Parameter name

Ri:=Parameter.St.An.Parameter name

Ni:= PARAMETER.FT.An.Parameter name

NI:=PARAMETER.S2.AN.Parameter name

NOTE The use of an integer or real number register depends on the type of parameter.

Program: start_get.pnc

```
1  $A
2  R1:=PARAMETER.ST.A1.PFB     ; Get parameter PFB from LV-servoTEC axis 1 and enter it in real number register R1
3  N2:=PARAMETER.ST.A1.IN1     ; Get parameter IN1 from LV-servoTEC axis 1 and enter it in integer register N2
4  N3:=PARAMETER.ST.A1.ACC     ; Get parameter ACC from LV-servoTEC axis 1 and enter it in integer register N3
5  R2:=PARAMETER.ST.A1.IPEAK   ; Get parameter IPEAK from LV-servoTEC axis 1 and enter it in real number register R2
6  T10
7  JMP A
8  END
```

[4] = LV-servoTEC, flexmo TEC

1.7.7.3 Ran.m:=Ri - Load register with axis parameters[3]

Ran.m:=Ri

Description:

The axis parameters can be loaded into the registers by means of this command.

General format : RA(*number of axis).(number of parameter)*:=R (*number of register)*

The selection of the desired parameter "m" ensues through input as per the following table: It includes a selection of the most important axis parameters.

7	→	Gear factor
8	→	min. traversing range
9	→	max. traversing range
23	→	Sequence of reference
24	→	Direction of rotation
28	→	Axis limit switch replaced
62	→	Reference offset

Command format:

RAn.m:=Ri

Example

```
1   R2:=200
2   RA1.7:=R2        ;set the gear factor of axis 1 to content of
                     register 2
3   R7:=10
4   RA2.8:=R7        ;set the minimum traversing range of axis 2
                     for content of register 7
5   R23:=1000
6   RA12.9:=R23      ;set the maximum traversing range of axis 12
                     to the content of register 23
```

NOTE This command does not overwrite only the current (local) parameters but also the stored parameters, so that loading of the parameters of PA-CONTROL the modified (new) parameter is read.

NOTE If you change the gear ratio, you should call the commands "FAn: = ..." and "G100.An.., before the axis is moved in order for the velocity and the acceleration corresponding to the new gear ratio.

1.7.7.4 Ri:=Ran.m - Load axis parameter from a register

Ri:=Ran.m

Description:

With the help of this command values of different axis parameters can be written into a R-register.

In this case the following applies:

Ri	→	Real number register i
RA1.i	→	„i“-tens axis parameter axis 1
RA2.2	→	„2“-tens axis parameter axis 2
...	→	
RAn.i	→	„i“-tens axis parameter axis n

HINWEIS The number of the axis parameters can be taken out of the description of axis parameter in chapter 5.

Command format:

Ri:=RAn.m

Example

Program: achs_para.pnc

```
1
2     R101:= RA1.1        ; “1st“ axis parameter of axis 1 (traversing velocity)
3     R102:=RA1.7         ; „7th“ axis parameter of axis 1 (gear factor)
4     R103:=RA1.8         ; „ 8th“ axis parameter of axis 1 (min.travel range)
5
6     R201:= RA2.1        ; “2nd” axis parameter of axis 2 (traversing velocity)
7     R202:=RA2.7         ; „7th“ axis parameter of axis 1 (gear factor)
8     R203:=RA2.9         ; „9th“ axis parameter of axis 1 (max. travel range)
9     END
```

1.7.8 Interpolation

1.7.8.1 1.7.8.1 G144 - Switch LV-servoTEC axis to interpolation mode

G144	from V5.13

G144. A<Nr-LV-servoTEC-Axis>.A<Nr-PLS7-Axis>

Description:

Interpolation in the PA-CONTROL takes place in the PLS7 card. The position specification is undertaken via the pulse direction interface by the PLS7 card on the LV-servoTEC amplifiers (see application details “APP5014...”).

At the start of interpolation, the G144- command switches the LV-servoTEC axis into “Interpolation mode”, and adopts the current position of the LV-servoTEC axis in the PLS axis (sets PLS axis to this position).

NOTE The command can only be used for LV-servoTEC S2 axes and PLS7 axes.

The command G144 activates the X10 input at the LV-servoTEC S2 controller.

A G29 offset of the LV-servoTEC S2 axis is also taken over with the acquisition of the position in the PLS7 axis and then taken into account in the interpolation.

During interpolation the travel range limits are only monitored in the PLS7 axes. The PLS7 axes should have the same travel range limits as the associated LV-servoTEC axes.

Example

See G145 command on next page.

1.7.8.2 G145 - Deactivate LV-servoTEC axis interpolation mode

G145	from V5.13

G145. A<Nr-LV-servoTEC-Axis>.A<Nr-PLS7-Axis>

Description:

Interpolation in the PA-CONTROL takes place in the PLS7 card. The position specification is undertaken via the pulse direction interface by the PLS7 card on the LV-servoTEC amplifiers (see application details “APP5014...”).

At the end of interpolation, the G145- command deactivates the LV-servoTEC axis “interpolation mode”.

NOTE

The command can only be used for LV-servoTEC S2 axes and PLS7 axes.

The command G145 deactivates (blocks) the X10 input of the LV-servoTEC S2 controller.

A G29 offset, which is taken over with the G144 command of the LV-servoTEC S2 axis in the PLS7 axis, remains unchanged in the PLS7 axis

A G29 offset, which is executed with the command G29 onto the PLS7 axis, is not taken over to the LV-servoTEC axis.

Example

```
1   G25.A1 G25.A2 G90.A0
2   A1:=10          ;Move A1 axis to position 10
3   A2:=20          ;Move A2 axis to position 20
4   G100.B.100      ;Define path acceleration for the complete
                    contour
5   G144.A1.A13     ; Switch A1 into interpolation mode, A13 used as
                    source for pulse and direction
6   G144.A2.A14     ; Switch A2 into interpolation mode, A14 used as
                    source for pulse and direction
7   FB:=1000        ;Set path speed to 1000AE/s
8   G01 A1:=40 A2:=30   ;Linear interpolation with the A1 and A2 axis from
                    current position to position A1=40 and A2=30
9   FB:=800         ;Change path speed to 800AE/s for the imminent
                    interpolation stages with the ramp of the specified
                    acceleration
10  G01 A1:=100 A2:=100 ;Interpolation is continued at a different speed
                    without stopping
11  G01 A1:=10 A2:=20   Contour is ended
12  G145.A1.A13     ; Deactivate interpolation mode of A1
13  G145.A2.A14     ;Move A1 axis to position 10
14  END
```

1.7.8.3 G100.B.n - IPO path acceleration[2]

G100.B.n

G100.B.n

Description:

G100.B.n is used to set the path acceleration during interpolation. The path speed **FB** must be set in advance.

NOTE If the value is greater than or less than the limit values, the “value too high” or “value too low” error message is issued.

Command format:

G100.B.1500

G100.B.R2

Application:

A way of working adapted to the situation can be achieved using the G100.B.n command. This may deviate from the values saved for acceleration in the parameter values. The definition used applies until the G100.B command is next used or until automatic mode is exited from.

NOTE If path acceleration is not defined, an error message is generated!

[2] only suitable for PA-CONTROL controller (Steuer) equipped with PLS7 / PLS9.

1.7.8.4 FB:=n - IPO path speed[3]

FB:=n

FB:=1200
FB:=R4

During linear interpolation, travel to the specified target position takes place in a straight line at path speed. Movements can be carried out parallel to the axis and at any angle.

The path speed is set before the actual **G01** interpolation command using the **FB:=n** command. If several interpolation commands follow in succession, they are processed without a break. The path speed can be changed during a complex process without stopping.

NOTE This command can only be used in conjunction with PA-CONTROL equipped with PLS7 / PLS9.

If interpolation commands in the program follow one another directly (with no other commands in between), these interpolation commands are processed in one go. Path speed and linear interpolation commands and commands from the G16i group (operating outputs during interpolation) may then be mixed.

NOTE Before the interpolation is executed, the path acceleration and path velocity have to be defined (see example 1 on page *125*).

1.7.8.5 Defining IPO path speed, one axis running too

FB.i:=n

FB.i:=Rn

FB.2:=1234
FB.3:=R7

In applications with the PA-CONTROL, during interpolation, one axis often has to run in parallel with the interpolation. This axis should not however influence the resulting "IPO path" of the other axes.

Application:

- The flux should be transported at a fixed ratio to path speed.
- During interpolation an axis should be traversed at particular points

[3] only suitable for PA-CONTROL COMPACT equipped with PLS7 / PLS9.

1.7.8.6 G01 - Linear interpolation with 4 from 16 axes [3]

G01

G01 A1:=(POS) A2:=(POS).....

Description:

In linear interpolation, traversing to a defined target position is executed on a straight line at path speed. Axially parallel movements and movements traversing at any angle can be executed. The path speed is set before G01 using the FBn command. Several interpolation commands defined in succession will be processed without stopping. The path speed can be altered during a complex run without stopping.

The transition to the new path speed is carried out at the pre-set path acceleration. It is set by means of the command G100.B.n and cannot be altered within a contour.

NOTE This command is only possible in combination with a PA-CONTROL equipped with PLS7 / PLS9.

Every axis, which takes part of an interpolation, has to be controlled via a PLS card.

If interpolation commands follow each other in succession (without other commands in-between), they are processed in one piece. Path speed and interpolation commands as well as commands of the G16x group (operation of outputs during interpolation) can be combined.

NOTE The path acceleration and the path speed must always be specified before execution of the interpolation (see example 1 on the next page).

Command format:

G01 → AiAn

G01 → Ari ARn

Application:

Simultaneous traversing of 2 to 4 axes moving along a straight line between the start and target position.

[1] only suitable for PA-CONTROL COMPACT equipped with PLS7 / PLS9.

Example 1

```
1    G25.A1 G25.A2 G90
2    A1:=10              ;Traverse A1 axis to position 10
3    A2:=20              ;Traverse A2 axis to position 20
4    G100.B.100          ;Specify path acceleration for complete
                         contour
5    FB:=1000            ;Set path speed to 1000AE/s
6    G01 A1:=40 A2:=30   ;Linear interpolation with axes A1 and A2 from
                         current position to position A1=40 and A2=30
7    FB:=800             ;Change of the path speed to 800AE/s for
                         next interpolation sections with the ramp of
                         the pre-set acceleration
8    G01 A1:=100 A2:=100 ;Interpolation is continued at the changed path
                         speed without stopping
9    G01 A1:=10 A2:=20   Contour is completed
10   END
```

Example 2

Complex example of multiple linear interpolations in succession to trace a contour at a constant path speed. Due to the fact that the interpolation commands are defined in succession (no other command in-between), the distances "a“ to "f“ are traced without stopping.

```
1    O6:=0 I5.1          ;Tool up
2    G25.A1  G25.A2. G90 ;Approach to reference point with axis 1 and 2
3    A1:=30 A2:=20       ;Approach edge of work piece
4    O6:=1               ;Lower tool
5    I6.1                ;Wait until tool is down
6    FB:=500             ;Set path speed to 500AE/s
7    G100.B.30           ;Set path speed to 30AE/ss
8    G01 A1:=40 A2:=40   ;Trace lateral edge "a"
9    G01 A1:=60 A2:=40   ;Trace lateral edge "b"
10   FB:=300             ;Set path speed to 300AE/s
                         ;no stop is executed here, the transition to the
                         new path speed ensues with the ramp of the
                         pre-set acceleration
11   G01 A1:=70 A2:=30   ;Trace lateral edge "c"
12   G01 A1:=70 A2:=25   ;Trace lateral edge "d"
13   G01 A1:=60 A2:=20   ;Trace lateral edge "e"
14   G01 A1:=30 A2:=20   ;Trace lateral edge "f"
15   O6:=0               ;Tool up
16   A1:=5 A2:=5         ;Approach initial position
17   END
```

1.7.8.7 Linear interpolation with following axis/axes

G01 A1:=(POS) A2:=(POS)……

G01 A1:=100 A2:=100 A3:=50
G01 A1:=10 A2:=5 A3:=FB*R7
G01 A1:=20 A2:=10 A3:=0

NOTE The route taken by axes which do not influence the path (FB.2:=..., An:=FB*Ri) should be less than or equal to the resulting path length as otherwise the IPO interval is extended until the path of these axes has been traversed.
In this case, the SM8 system flag is set.

NOTE Path influence (FB.i:=...) can only be set starting with the first axis in ascending order.
For example: A1, or A1 & A2, or A1 & A2 & A3, or all.
A2 & A3, or A1 & A3 are not possible for example.

NOTE With FB*Rn the direction for this axis is taken from Rn
Rn > 0 → positive direction
Rn < 0 → negative direction

NOTE If multiplying together the axis route from the path route, factor and gear factor (An:=FB*Ri) produces a number of axis steps which is greater than the number of path steps, system flag SM8 is set.

NOTE If the gear factor of an axis which is traversed at path speed (An:=FB**Ri) is greater than the largest factor for an axis involved in interpolation, the axis route may be greater than the path route. In this case, the SM8 system flag is also set.

Example 1

```
1   R7:=0.5
2   R8:=-0.5
3
4   G01 A1:=10 A2:=5 A3:=FB*R7     ; Axis 3 is traversed in a positive direction at
                                     half the path speed
5   G01 A1:=20 A2:=10 A3:=FB*R8    ; Axis 3 is traversed in a negative direction at
                                     half the path speed
6
```

Example 2

```
1   G25.A1 G25.A2 G25.A3          ; Approach reference point with all three axes
2   G90.A0                        ; Absolute measuring system for all three axes
3   FB.2:=1000                    ; Set path speed to 1000mm/s, only the first two
                                  axes influence the path
4   R4:=0.1                       ; Set speed for axis 3 to a tenth of the path
                                  speed
5   G01 A1:=100 A2:=100 A3:=50    ; The resulting path length = 141.42mm, axis 3
                                  is traversed 50mm during this interval
6   G01 A1:=200 A2:=200 A3:=0     ; The resulting path length = 141.42mm, axis 3
                                  is not traversed during this interval
7   G01 A1:=300 A2:=300           ; The resulting path length = 141.42mm, axis 3
    A3:=FB*R4                     is traversed 100mm in a positive direction
                                  during this interval
8   .....
```

1.7.8.8 IPOEND – Abort linear interpolation

IPOEND

IPOEND.Counter.Limit

IPOEND.Ni.Nn

IPOEND.N!i.N!n

Description:

The IPOEND command enables the user to abort an interpolation.

Function:

A check is carried out for each IPO command (G01) as to whether the function IPOEND has been activated. If this is not the case, the programme is continued as described. If this is the case, a check is carried out as to whether the counter is less than the limit (Ni<Nn). In this case, the IPO command is executed.

If the counter is the same as the limit or even higher than the limit, processing of the IPO command is discontinued and the program is continued after the last IPO command.

Please note:

- IPOEND is normally inactive and is deactivated at the end of interpolation.
- The counter and the limit cannot be changed during interpolation.
- IPOEND sets the counter to 0 (at the beginning of processing).
- At least one IPO command (G01) is executed.

Example

Program 1

```
1       N7:=3
2       SUB TRAVERSE_IPO
        ...
6       N7:=5
7       SUB TRAVERSE_IPO
8       END
```

Subprogram

```
1       $ TRAVERSE_IPO
2       IPOEND.N2.N7            Activate IPOEND, that means 3 or 5 IPO
                                commands are executed
3       G100.B.100              ;path acceleration is determined for the complete
                                contour
4       FB:=100                 ;path velocity is set of 100AE/s e
5       G01 A1:=R1 A2:=R2       ;Linear interpolation with the A1- and  A2 axis
6       G01 A1:=R3 A2:=R4
7       G01 A1:=R5 A2:=R6
8       G01 A1:=R7 A2:=R8
9       G01 A1:=R9 A2:=R10
10      G01 A1:=R11 A2:=R12
11      END
```

1.7.8.9 Rn:=FB - Get current path speed

Rn:=FB

Command format:

Rn:=FB

Description:

The current path speed can be retrieved and stored in a real number register in parallel to an interpolation in progress from another task (parallel run) by means of this command.

This information enables the application to affect components of the plant on the basis of the current path speed.

Action	Result
Interpolation activated and traversing	Current path speed
Acceleration or braking during interpolation	Current path speed in the ramp
Interpolation was stopped (STOP PA-CONTROL)	0
No interpolation active	-1

Example

```
1       G26.A1 G25.A2
...
11      RUN GET_CURRENT_FB      Start PAB program GET_CURRENT_FB
12      R1:=1000
13      R2:=5000
14      R3:=3000
15      R4:=1000
16      $A
17      I8.1
18      G100.B.10000
19      FB:=R1
20      G01 A1:=1000 A2:=2000
21      FB:=R2
22      G01 A1:=2000 A2:=4000
23      FB:=R3
24      G01 A1:=3000 A2:=6000
25      FB:=R4
26      G01 A1:=4000 A2:=8000
27      T100
28      A1:=0  A2:=0
29      T100
30      JMP A
31      END
```

Program GET_*CURRENT*_FP

```
1       $A
2       R5:=FB          Get the current path speed and store the value in
                        real number register 5.
3       JMP A
4       END
```

1.7.8.10 Rn:=FBAn - Get the current speed of an axis during an interpolation

Rn:=FBAn

Command format:

Rn:=FBAn

Description:

The command supplies the current speed of the selected axis during an interpolation. The speed is returned according to the path speed and with the proportional ratio of the axis.

1.7.8.11 G16x - Operation of outputs during interpolation

G16?

Description:

The G160 commands are used to control outputs or flags synchronous to the interpolation command.

General:

G16x.[Time].[Element].[Status]

[Time]	:	Time specification for the action in n*10ms
[Element]	:	O → output, M → flag
[Status]	:	0 → is reset, 1 → is set

NOTE

- Only one storage location is provided for a G160, G161 or G162 command in an interpolation interval.
- Only logical commands and parallel runs can be used to control multiple outputs.
- If the duration of an interpolation interval is shorter than specified in the G161 or G162 commands, the element is controlled by the G161 command at the end of the interval and by the G162 command at the beginning of the interval.

1.7.8.12 G160 - Activate outputs before starting the interpolation

Example 1

n	*G160.20.O123.1*	*Set output O123 to 1 and wait 200ms before starting the subsequent interpolation*
n+1	*G01 A1:=10 A2:=20*	*;Linear interpolation with axes A1 and A2 from the current position to position A1=10 and A2=20*
...	*G01 A1:=30 A2:=30*	*;Linear interpolation with axes A1 and A2 from the current position to position A1=30 and A2=30. The second interpolation directly follows the first without any interruption!*

1.7.8.13 G161 / G162 - Activate outputs during interpolation

Example 2

n	*G01 A1:=10 A2:=20*	
n+1	*G161.1.O333.1*	*Set output O333 to "1" 10ms after start of the subsequent interpolation interval.*
n+2	*G01 A1:=30 A2:=30*	*;Linear interpolation with axes A1 and A2 from the current position to position A1=30 and A2=30*
...	*G01 A1:=60 A2:=40*	*;Linear interpolation with axes A1 and A2 from the current position to position A1=60 and A2=40*
...	*G162.5.O333.0*	*Set output O333 to "0" 50ms before the next interpolation interval is completed.*
...	*G01 A1:=90 A2:=50*	

Example 3

n	*G01 A1:=10 A2:=20*	
n+1	*G161.1.M34.1*	*Set flag M34 to "1" 10ms after the subsequent interpolation interval was started.*
...	*G162.5.M34.0*	*Set flag M34 to "0" 50ms before the next interpolation interval is completed.*
...	*G01 A1:=300 A2:=250*	

Example 4

n	*G01 A1:=10 A2:=20*	
n+1	*G161.1.O333.1*	*Set output O333 to "1" 10ms after the subsequent interpolation interval was started.*
n+2	*G01 A1:=30 A2:=30*	*;Linear interpolation with axes A1 and A2 from the current position to position A1=30 und A2=30*
...	*G162.5.O334.1*	*Set output O334 to "1" 50ms before the next interpolation interval is completed.*
...	*G01 A1:=60 A2:=40*	
...	*G162.5.M3.1*	*Set flag M3 334 to "1" 50ms before the next interpolation interval is completed.*
...	*G01 A1:=90 A2:=50*	

JMP-LINE-IPO.Ni

Description:

A specified line in the program can be jumped to and the interpolation started from there.

Prerequisite:

- No interpolation can be active.
- A G01, FB or G160 command must be located in the programme line which is stated as a jump target. If this condition is not complied with, error "E577=incorrect jump target" is output.

NOTE The path acceleration (G100.B...) and the path speed (FB...) must be set to appropriate values before execution of the command "JMP-LINE-IPO", since otherwise an error Exxx is output.

NOTE An FB command is processed with JMP-IPO-LINE, and then it changes to next IPO command.

NOTE G16x commands are not processed, and it is changed to the next IPO command. The probability that a line number of a G16x command is recognised is very low, since the G16x commands are implemented in the IPO interval during the execution of the IPO commands.
A G160 command is only recognised via a line number if the interpolation is aborted during the start phase (loading phase of the G01 intervals).

1.7.9 Motor current commands

1.7.9.1 G101 - Changing the motor current and limiting G101 [1, 4]

G101

(G100.Axis.Value)

Description:

The motor current can be changed in percentage terms in relation to the nominal current set in the parameters. The maximum torque of the motor is almost linear in the relation to the motor current.

The command G101 can only be used for the following axes types	
Device	**Precentral change compared to the parameter value [%]**
PA-CONTROL EP	10 to 100
PA-CONTROL MP	10 to 300
LV-servoTEC (S2)	1 to 300

Command format:

G101.An.n

G101.An.Nn

Application:

- Screwing with a stepping motor or servomotor with a defined torque.
- Force limitation of spindle axes or motor-driven presses.
- Force limitation of axis motion with risk of collision

NOTE If you do not leave the operating mode AUTOMATIC, the change by the G101 command is retained.

Only when you leave and restart the operating mode AUTOMATIC will the change by the G101-command be rejected and the axis is set for the parameter value of axis or drive parameters.

1.7.9.2 G101 - Current limitation changes in a PA-CONTROL-MP[1]

G101

(G100.axis.value)

Description:

The motor current of a PA-CONTROL MP can be changed by percentage between 10% and 100% to the setting parameter value “phase current”.

NOTE The current setting is only active while driving.
During standstill the motor of the PA-CONTROL MP is energized together with the set motor parameters “phase current during standstill”

Example 1

```
1   G101.A1.50      ;50% of the nominal current
2   A1:=100         ;Approach position 100 at reduced torque
3   END
```

Example 2

```
1   N1:=50          ;50% of the nominal current
2   G210.A1         ;Does not activate the start positioning mode
                    for axis 1 until the axis has arrived at the stop
                    position.
3   A1:=100         ;Approach position 100
4   G230.1.A1.50    ;Wait until the position >=50 has been
                    reached
5   G101.A1.N1      ;Set the motor current according to N1
6   G213.A1         ;Wait until axes are in position and activate
                    the standard positioning mode
7   END
```

[1]only for PA-CONTROL MP, LV-servoTEC; [4] for PA-CONTROL LV-servoTEC

1.7.9.3 G101 – changing current limitations at a LV-servoTEC axis

G101

(G100.axis.value)

Description:

With an LV-servoTEC axis the G101 command changes peak current IPEAK. The peak current is adjusted to the value in percentage of the rated current IRMS. The possible change can be between 1% and 300%.

NOTE If the peak current IPEAK should be reset on the parameter value after the change with the G101 command, this has to be realized with another G101 command (see example).

Whether the G101 command with 300% (G101.A1.300) harms the engine lies in the responsibility of the operator.

Settings of the axis parameters for the example:

Rated current IRMS : 1,5 A

Peak current IPEAK : 2,0 A

Example

```
1   N1:=20          ;IPEAK on 20% of IRMS → 0,3 A
2   G210.A1         ;activates the starting position mode for axis 1
3   A1:=100         ;move to position 100
4   G230.1.A1.50    ; wait until position >=50 is reached
5   G101.A1.N1      ; limit the motor current to 0.3 A according to N1
6   G213.A1         ; wait until the axis is in position and activate
                    regular positioning mode
7   G101.A1.133     ; IPEAK on 133% of IRMS
                    → set on 2,0 A  (reverse limitation)
8   END
```

1.7.9.4 G101 - Current limitation changes in a LV-servoTEC S2

G101

(G100.axis.value)

Description:

For a LV-servoTEC S2 axis, the current has the parameters (drive parameters)

- Maximum current [A]
- Rated current [A]
- Current limitation [A]

The G101 command affects the parameter "current limitation“. The G101-command sets the parameter "current limitation" to the percentage of the rated current in the LV-servoTEC S2. The possible change may be between 1% and 300%.

NOTE If the current limitation is deactivated again by the G101-command, this must be done by another G101 command (see example). Since the G101 command does not change the parameters for rated current and maximum current, this is possible by the handover of 300%. The LV-servoTEC S2 takes the percentage or the maximum current of the amplifier if the percentage exceeds the maximum current.

Settings of the axis parameters for the example LV-servoTEC S2 105:

Rated current	: 2.5 A
maximum current	: 4.0 A
Current limitation	: 5.0 A (LV-servoTEC S2 105)

Example

1	*N1:=20*	*;current limitation to 20% of the rated current → peak current = 0.5 A*
2	*G210.A1*	*;activates the starting position mode for axis 1*
3	*A1:=100*	*;Move to position 100*
4	*G230.1.A1.50*	*;wait until position >=50 is reached*
5	*G101.A1.N1*	*;limit the motor current to 0.5 A according to N1*
6	*G213.A1*	*;wait until the axis is in position and activate regular positioning mode*
7	*G101.A1.300*	*;current limitation to 300% of the rated current → set 7.5 A* *Since the amplifier can only drive 5.0 A maximum current, the current limitation is set to 5.0 A.*
8	*END*	

1.8 Inputs, Outputs, Flags

1.8.1 Inputs

1.8.1.1 In.m - Waiting for logical status of input

In.m

I (Number).(Status)

Description:

The controller checks by means of the command I (input) whether the input n defined in the operand has the status m. If the poll result is not true, the controller waits until the condition is provided in operator m.

Command format:

I12.1

I12.0

Application:

Polling of proximity switches, push buttons, switches, etc. …

Example

1	*I8.1*	*Controller waits until input 8 is logical 1*
2	*I4.0*	*Controller waits until input 4 is logical 0*
3	*END*	

1.8.2 Outputs

1.8.2.1 On.m - Waiting for logical status of output

On.m

O (Number).(Status)

Description:

The controller checks by means of the command O (output) whether the output n defined in the operand has the status m. If the polling result is not true, the controller waits until the condition is provided in operator m.

NOTE This command is only appropriate in case of multiple program runs in parallel, because only another parallel process can change the output while this process is waiting for the output.

Command format:

O12.1

O12.0

Application:

Polling of outputs

Example

```
1   i   O8:=1      ;The controller sets output 8
                   (e.g. for the accurately timed start of a parallel run)
2       O7.1       ;Controller waits until output 7 is logical 1
                   (is set by a program operating in parallel)
3       END
```

1.8.2.2 On:=m - Set / reset output

On:=m

O (Number):=(Status)

Description:

The available outputs can be manipulated by means of the command O (output). If output n is set to the status m = 1, it completes the circuit, i.e. the transistor is conductive and remains switched on until it is reset by m = 0 and the transistor is off.

Command format:

On:=1	
On:=0	
On:=Mm	Status of flag m is transferred for output n
On:=Im	Status of input m is transferred for output n
On:=Om	Status of output m is transferred for output n

Application:

Switching of relays or pneumatic valves, signal transmission to other electrical devices

Example

```
1   O8:=1          ;Output 8 is set to 1
2   O7.0           ;Controller waits until output 7 is logical 0
3   END
```

1.8.3 Flags

1.8.3.1 Mn:=m - Set / reset flag

Mn:=m

M (Number):=(Status)

Description:

The available flags can be controlled by means of the command M. If flag n is set to the logical status m=1, it remains in this status until it is reset with m=0 (see also description of commands for comparison operations).

Command format:

M1:=0	Reset flag 1
M1:=1	Set flag 1
Mn:=Mm	Status of flag m is transferred for flag n
Mn:=Im	Status of input m is transferred for flag n
Mn:=Om	Status of output m is transferred for flag n
M!n:=M!m	Indirect addressing
Mn:=SMm	Condition of system flag is taken over for flags (from V5.17)

Application:

- Storage of statuses, processing situations
- Synchronization of parallel runs

1.8.3.2 M!2:= - indirect or direct addressing of flags

Flags can be addressed indirectly from version 3.72 and higher. For indirect addressing, the flag is identified by an exclamation mark after the designator M. The flag points to the contents of the integer register with the same index.

Command format:

M!2:=0	→	Indirect addressing when setting flags

The indirect addressing must be declared before application!

N15:=10	→	The content of integer register N15 is 10
M!15:=1	→	The number 15 after the exclamation mark points to the contents of integer register **N**10 whose content =5. Flag **M**10 is set to "1".

Application:

- Parameter setting for programs

Example

```
1    I8.1        ;Controller waits until input 8 is logical 1
2    M12:=1      ;Flag 12 is set to logical 1
3    M10:=0      ;Flag 10 is set to logical 0
4    N15:=10     ;Load register N15 for indirect addressing
5    M!15:=1     ;M10 is set to logical 1 by the indirect addressing of M15
6    T100        ;Wait 1 second
7    N16:=10     ;Load register N16 for indirect addressing
8    M!16:=0     ;M10 is set to logical 0 by the indirect addressing of M16
9    END
```

1.8.3.3 Mn.m - Waiting for then logical status of a flag

Mn.m

M (Number).(Status)

Description:

The controller checks by means of the command M (flag) whether the flag n defined in the operand has the status m. If the polling result is not true, the controller waits until the condition is provided in operator m.

The flags can be addressed indirectly. The flag for indirect addressing is identified by an exclamation mark after the designator M. The number n after the exclamation mark points to the integer register with the same address. The content of the register provides the actual flag address. See the following example.

N10:=5	→	The contents of integer register **N**10 is 5
M!10.1	→	The number 10 after the exclamation mark points to the contents of integer register **N**10 whose content =5. The controller now waits until the status of the flag **M**5 is equal to "1"
M5.1	→	The command **M**5.1 is identical to the previous example with indirect addressing

Command format:

M1.1		Set flag 1 to status logical 1
M1.0		Reset flag 1
M!3.1	→	Indirect addressing for flag polling

Application:

Synchronization of parallel runs

Example

```
1   O8:=1      ;The controller sets output 8 (e.g. triggering for
               parallel run control)
2   M7.1       ;Controller waits until input 7 is logical 1 (was set by
               parallel run control)
3   N10:=5     ;Load N register for indirect addressing
4   M!10.1     ;Wait until M5 has the logical status 1
5   END
```

1.8.3.4 > = < - Comparisons

> = <

Command format:

Direct addressing			Indirect addressing	
Mn:=Rm=K	Mn:=Nm=K		M!n:=Rm=K	M!n:=Nm=K
Mn:=Rm>K	Mn:=Nm>K		M!n:=Rm>K	M!n:=Nm>K
Mn:=Rm<K	Mn:=Nm<K		M!n:=Rm<K	M!n:=Nm<K
Mn:=Rm=Ri	Mn:=Nm=Ni		M!n:=Rm=Ri	M!n:=Nm=Ni
Mn:=Rm>Ri	Mn:=Nm>Ni		M!n:=Rm>Ri	M!n:=Nm>Ni
Mn:=Rm<Ri	Mn:=Nm<Ni		M!n:=Rm<Ri	M!n:=Nm<Ni
Mn:=R!m=Ri	Mn:=N!m=Ni		M!n:=R!m=Ri	M!n:=N!m=Ni
	Mn:=Nn<>K			

NOTE See also chapter *1.10.3.1, page 162.*

1.8.3.5 Write flag number to an N register after comparison

GETM from V5.07

Command format:

Ni:=GETM.n.m.l

Ni:=GETM.<As of flag no>.<Up to flag no>.<Comparison status>

N3:=GETM.1.128.1

Application:

The command writes to the N register (Ni) the number of the first flag from the flag range to be checked (<As of flag no>.<Up to flag no>) which corresponds to the comparison status

NOTE A maximum of 128 flags can be checked.
(<As of flag no>.<Up to flag no>)

If no flags corresponding to the comparison status are found in the checked range, the N register is set to “0”.

Example

```
1    LD SM10.0
2    RES M1.M48
3    $A
4    N3:=GETM.2.39.1
5    O1:=1
6    T10
7    O1:=0
8    T10
9    JUMP A
10   END
```

1.9 Timer, Date and Time of day

1.9.1 Wait / Timer

1.9.1.1 Tn - Hold-up time

T

T (Time in 10/milliseconds)

Description:
The command T causes the controller to hold for the time defined in the operand.
The following is applied here:

the shortest time	n	=	1	=	10 ms
the longest time	n	=	2^{31} =2.147.483.648	=	21.474.836.480 ms

Command format:
T1000
TN12

Application:
Waiting times, keeping outputs set for defined times.

Example

```
1    T1      ;Hold-up time 10ms
2    T300    ;Hold-up time 3s
3    TN2     ;Hold-up time according to the contents of N2
3    END
```

1.9.2 Date and time of day

1.9.2.1 DATE:= - Set the date from the user program

DATE:=

DATE:=[Day].[Month].[Year].[Weekday]

Description

The date and the week day of the real-time clock on the CPU4 can be set from the user program by means of this command.

Command format:

Date:=Ni.Nj.Nn.Nm

NOTE On execution of the command, a check is carried out to ensure that the values for the date are legal. If this is not the case, error E532 "Value outside range" is set.

Number	Weekday
0	Sunday
1	Monday
2	Tuesday
3	Wednesday
4	Thursday
5	Friday
6	Saturday

Example

```
1   N1:=15
2   N2:=8
3   N3:=2003
4   N4:=5
5   DATE:=N1.N2.N3.N4        Sets the date to Friday 15th August 2003
```

1.9.2.2 TIME:= - Set the time from the user program

TIME:=

TIME:=[Hours].[Minutes].[Seconds]

Description

The time of the real-time clock on the CPU4 can be set from the user program by means of this command.

Command format:

TIME:=Ni.Nj.Nn

NOTE On execution of the command, a check is carried out to ensure that the values for the time are legal. If this is not the case, error E532 "Value outside range" is set.

Example

1	*N1:=14*	
2	*N2:=50*	
3	*N3:=23*	
4	*TIME:=N1.N2.N3*	*Sets the time to 14:50:23*

1.9.2.3 Read-out of the real-time clock

S0:=TIME

S0:=TIME[.YEAR][.MONTH][.DAY][.HOUR][.MIN][.SEC][.WD][.YD][.W]

Description

This command enables access to the real-time clock of the controller. In the CPU4 this is available as real-time clock component on the circuit board.

If there is an IEF touch screen control panel (see PA-CONTROL Manual, chapter 'Options') connected via CANopen-Bus, the time from the control panel can be cyclically sent to the PA-CONTROL and can be further processed with this command.

The Information is always output to the global S0 string. The elements in the [ ] are optional and may be mixed. The elements can be separated by means of a dot, hyphen or colon. The separator used is transferred to the same position in the S0 string.

The following input examples refer to the 4th December 2002, 08:41:53

Command format	**Output**
S0:=TIME.DAY.MONTH.YEAR	04.12.2002
S0:=TIME.DAY.MONTH.YEAR-HOUR:MIN:SEC	04.12.02-08:41:53
S0:=TIME.HOUR:MIN:SEC	08:41:53
S0:=TIME.W	49
S0:=TIME.W-WD	49-3

Example

```
1   S0:=TIME.DAY.MONTH.YEAR   ;Get day, month and year from the real-time clock and write this information to the global string S0
2   G11.0                     ;Switch off run displays
3   G500.0                    ;Switches the G500 commands over to the current display medium (LC display or the serial port with "Simulation of the front plate")
4   G501                      ;Clears the display and places the cursor in the upper left corner (1st. column, 1st line)
5   G510.CURRENT DATE         ;Output of text "CURRENT DATE"
6   G503.17.1                 ;Position the cursor in the 17th column, 1st line
7   G520.S0                   ;Display the contents of the global string
8   T500                      ;Wait 5 seconds
9   END
```

1.10 Mathematics

1.10.1 Organisation of commands

1.10.1.1 Direct and indirect addressing

Description:

Registers can be addressed directly and indirectly when assigning values and for types of basic arithmetic.

Direct addressing:

- A number (1-8192), identifying the desired register, follows the letter R(N).

Indirect addressing:

- A "!" and then a number (1-8192), referring to the register whose content indicates the desired register, follows the letter R(N).

1.10.1.2 Assignments

The following applies generally for assignments:

Objective		Source
Register	→	Constant (K)
		→ Register
		→ Absolute position of an axis (only with real number registers)

The following applies generally for arithmetic operations:

Target operand		Source operand 1		Operation		Source operand 2
Register	→	Register	→	+ - * /	→	Constant without prefix Register

NOTE The result depends on the operation and prefix of the register content.

K (constant) is used both when describing integers and for real numbers. Only the permissible number type can be used for the real application.

The source operands remain unchanged. One exception is when the target operand is also the source operand.

Example

Example 1:

R1:=5
R2:=3
R3:= R1+R2

The registers then have the following values: R1=5, R2=3, R3=8

Example 2:

R1:=R1+R2

The registers then have the following values: R1=8, R2=3.

1.10.2 Mathematic commands

1.10.2.1 Rn/Nn:= Load register

Rn:=
Nn:=

Description:
Values can be loaded into the registers by means of these commands.

NOTE No spaces are permitted in the command.

Command format: (selection from the possible combinations)

Nn:=K	N!n:=K	Nn:=Nm	Nn:=Rm	Nn:=Ai	Nn:= SNn
Rn:=K	R!n:=K	Rn:=Rm	Rn:=Nm	Rn:=Ai	Rn:=SRn

Example

```
1   R2:=45          ;Load real number register 2 with 45
2   R3:=R2          ;Load real number register 3 with value of real
                    number register 2
3   R!2:=100.79     ;Load real number register 45 with 100.79
5   R5:=A1          ;Load real number register 5 with absolute
                    position of the A1 axis
6   N7:=87          ;Load integer register 7 with value 87
7   N65:=N7         ;Load integer register 65 with value from
                    integer register 7
8   N1:=R45         ;Load integer register 1 with value from real
                    number register 45, value is rounded
9   R25:=N65        ;Load real number register 25 with value from
                    integer register 65
10  N2:=4           ;Load integer register 2 with 4
11  N!2:=17         ;Load integer register 4 (N2=4) with 17
12  N24:=SN12       ;Transfer of the number of the defective axis
13  R12:=SR32       ;Transfer of the absolute position of axis A2 to
                    real number register R11
14  END
```

1.10.2.2 Load register with a binary value

Nn:=2#xxxxxxx

Description:

These commands can be used to load a constant into the integer register in the form of a binary value.

Command format:

N2:=2#0011000

N3:=2#1010

N4:=N2+2#1010

NOTE No spaces are permitted within the command.

1.10.2.3 Load register with a hexadecimal value

Nn:=16#yyy

Description:

These commands can be used to load a constant into the integer register in the form of a binary value.

Command format:

N2:=16#64

N3:=16#3E8

N4:=N2+16#0A00

N5:=N2&16#000A

N6:=N4&N5

NOTE No spaces are permitted within the command.

1.10.2.4 Rn:=RM+K - Addition

+

Command format:

Rn:=Rm+K
R!n:=Rm+K
Rn:=Rm+Ri
Rn:=R!m+Ri
Nn:=Nm+K
Nn:=Ni+Nm
N!n:=Nm+K
Nn:=N!m+Ni

Example

```
1   R2:=R2+57          ;Adds 57 to the content of R2
2   R2:=R4+R3          ;Adds the content of R4 and the content of R3
                       and assigns the result to R2
3   R1:=5
4   R5:=100
5   R24:=R!1+R2        ;Adds the content of R5 and the content of R2
                       and assigns the result to register R24. The
                       content of R1 = 5. The content of R5 is thus
                       accessed by indirect addressing. The content
                       of R5 and R2 is added and the result
                       assigned to R24. R1, R5 and R2 remain
                       unchanged.
6   N3:=N3+87          ;Adds 87 to the content of N3
7   N3:=N3+N5          ;Adds the content of N5 to the content of N3
8   END
```

1.10.2.5 Rn:=Rm-K - Subtraction

-

Command format:

Rn:=Rm-K
R!n:=Rm-K
Rn:=Rm-Ri
Rn:=R!m-Ri
Nn:=Nm-K
Nn:=Nm-Ni
N!n:=Nm-Ni
Nn:=N!m-Ni
Nn:=K-Nm
Rn:=K-Rm

Example

```
1    R2:=R2-57          ;Subtracts 57 from the content of R2
2    R2:=R2-R3          ;Subtracts the content of R3 from the content
                        of R2
3    R1:=5
4    R5:=100
5    R24:=R!1-R2        ;Subtracts the content of R2 from the content
                        of R5 and assigns the result to register R24.
                        The content of R1 = 5. The content of R5 is
                        thus accessed by indirect addressing. The
                        content of R2 is now subtracted from this
                        value and the result assigned to R24. R1, R5
                        and R2 remain unchanged.
6    N23:=N23-99        ;Subtracts 99 from the content of N23
7    N23:=N23-N4        ;Subtracts the content of N4 from the content
                        of N23
8    END
```

1.10.2.6 Rn:=Rm*K - Multiplication

*

Command format:

Rn:=Rm*K
R!n:=Rm*K
Rn:=Rm*Ri
Rn:=R!m*Ri
Nn:=Nm*K
Nn:=Nm*Ni
N!n:=N!m*Ni

Example

```
1    R2:=R2*4          ;Multiplies the content of R2 by 4 and assigns
                       the result to R2
2    R4:=R2*R3         ;Multiplies the content of R2 by the content of
                       R3 and assigns the result to R4.
3    R1:=5
4    R5:=100
5    R24:=R!1*R2       ;Multiplies the content of R5 by the content of
                       R2 and assigns the result to register R24.
6    N2:=N2*4          ;Multiplies the content of N2 by 4 and assigns
                       the result to N2.
7    N2:=N3*N6         ;Multiplies the content of N3 by the content of
                       N6 and assigns the result to N2.
8    END
```

1.10.2.7 Rn:=RM/K - Division

/

NOTE A system error occurs in the case of division by 0. The message "E005" is put on to the display.

Command format:

Rn:=Rm/K
R!n:=Rm/K
Rn:=Rm/Ri
Rn:=R!m/Ri
Nn:=Nm/K
N!n:=Nm/K
Nn:=Nm/Ni
Nn:=N!m/Ni
Rn :=K/Rm
Nn :=K/Nm

Example

```
1   R2:=R2/3        ;Divides the register content of R2 by 3 and assigns the result to R2
2   R4:=R2/R3       ;Divides the register content of R2 by the register content of R3 and assigns the result to R4
3   R1:=5
4   R5:=100
5   R24=R!1/R2      ;Divides the register content of R5 by the register content of R2 and assigns the result to R24
6   N34:=N34/2      ;Divides the content of N34 by 2 and assigns the result to N34, the result is rounded (if necessary)
7   N25:=N67/N88    ;Divides the content of N67 by the content of N88 and assigns the result to N25, the result is rounded (if necessary)
8   END
```

1.10.2.8 Trigonometrical functions

SIN COS TAN

Introduction:

The trigonometrical functions sine, cosine and tangent and their inverse functions arc sine, arc cosine and arc tangent are available in the PA-CONTROL. The value is put in in degrees. The trigonometrical functions can be used in conjunction with R registers and constant values.

1.10.2.9 SIN/ASIN – trigonometric functions

SIN / ASIN

Command format:

Rn:=SIN.K
Rn:=SIN.Ri
Rn:=SIN.R!i
Rn:=ASIN.K
Rn:=ASIN.Ri
Rn:=ASIN.R!i

Example

```
1   R43:=70
2   R70:=45
3   R100:=SIN.10          ;Write sine of the angle 10° to register R100 (R100= 0.174)
4   R101:=SIN.R43         ;Write sine of the angle 70° to register R101 (R101=0.940)
5   R102:=SIN.R!43        ;Write sine of the angle 45° to register R102 (R102=0.707)
6   R200:=ASIN.R100       ;Write arc sine of the register content of R100 to register R200 (R200:=10)
7   END
```

COS / ACOS

Command format: (selection from the possible combinations)

Rn:=COS.K
Rn:=COS.Ri
Rn:=COS.R!i
Rn:=ACOS.K
Rn:=ACOS.Ri
Rn:=ACOS.R!i

Example

```
1   R43:=70
2   R70:=-210
3   R100:=COS.10          ;Write cosine of the angle 10° to register R100 (R100= 0.985)
4   R101:=COS.R43         ;Write cosine of the angle 70° to register R101 (R101=0.342)
5   R102:=COS.R!43        ;Write cosine of the angle -210° to register R102 (R102=-0.866)
6   R200:=ACOS.R101       ;Write arc cosine of the register content of R101 to register R200 (R200:=70)
7   END
```

TAN / ATAN

NOTE The value range for tangent must be defined by the user himself. Non-permissible values (90°+ x * 180°, for x = 0, 1, 2,...) result in an undefined register value.

Command format: (selection from the possible combinations)

Rn:=TAN.K
Rn:=TAN.Ri
Rn:=TAN.R!i
Rn:=ATAN.K
Rn:=ATAN.Ri
Rn:=ATAN.R!i

Example

```
1   R43:=70
2   R70:=-210
3   R100:=TAN.10        ;Write tangent of the angle 10° to register
                        R100 (R100= 0.176)
4   R101:=TAN.R43       ;Write tangent of the angle 70° to register
                        R101 (R101=2.747)
5   R102:=TAN.R!43      ;Write tangent of the angle -210° to register
                        R102 (R102=-0.577)
6   R200:=ATAN.R102     ;Write arc tangent of the register content of
                        R101 to register R200 (R200:=-30)
7   END
```

1.10.2.12 SQRT - Root function

SQRT

NOTE The numerical value being squared may not be negative! The numerical value for the calculation must always be taken from a register!

Command format:

Rn:= SQRT.Ri

Rn:= SQRT.R!i

Example

```
1   R43:=70
2   R70:=210
3   R101:=SQRT.R43     ;Write square root of the number 70 to register R101 (R101=8.367)
4   R102:=SQRT.R!43    ;Write square root of the number 210 to register R102 (R102=14.491)
5   END
```

1.10.2.13 INT - Integer component

INT

Command format:

Rn:=INT.Rn

Example

```
1   R43:=70
2   R70:=210.23445
3   R10:=INT.R70       ;Stores the integer component of real number register R70 in real number register R10 (Result: 210)
4   END
```

1.10.2.14 FRAC - Decimal component of a real number

FRAC

Command format:

Rn:=FRAC.Rn

Example

```
1    R43:=70
2    R70:=210.23445
3    R10:=FRAC.R70        ;Stores the decimal component of real number
                          register R70 in real number register R10
                          (Result: 0.23445)
4    END
```

1.10.2.15 ABS - Absolute value of a real number / integer

ABS

Command format:

Rn:=ABS.Rn

Nn:=ABS.Nn

Example

```
1    R70:=-210.23445
2    R10:=ABS.R70         ;Stores the absolute value of real number
                          register R70 in real number register R10
                          (Result: |210.23445|)
3    N23:=-452
4    N11:=ABS.N23         ;Stores the absolute value  of integer register
                          N23 in integer register N11 (Result: |452|)
5    END
```

1.10.3 Compare

1.10.3.1 Compare - Comparison operations

>, =, <, <>

Direct addressing		Indirect addressing	
Mn:=Rm=K	Mn:=Nm=K	M!n:=Rm=K	M!n:=Nm=K
Mn:=Rm>K	Mn:=Nm>K	M!n:=Rm>K	M!n:=Nm>K
Mn:=Rm<K	Mn:=Nm<K	M!n:=Rm<K	M!n:=Nm<K
Mn:=Rm=Ri	Mn:=Nm=Ni	M!n:=Rm=Ri	M!n:=Nm=Ni
Mn:=Rm>Ri	Mn:=Nm>Ni	M!n:=Rm>Ri	M!n:=Nm>Ni
Mn:=Rm<Ri	Mn:=Nm<Ni	M!n:=Rm<Ri	M!n:=Nm<Ni
Mn:=R!m=Ri	Mn:=N!m=Ni	M!n:=R!m=Ri	M!n:=N!m=Ni
	Mn:=Nn<>K		

Example

```
1    M34:=R1=56.77      ; Setting flag 34, if the content of R1  is equal
                        to 56.77, otherwise the flag 34 is reset
2    M35:=R1>56.34      ;Setting flag 35, if the content of R1 is greater
                        than 56.3, otherwise the flag 35 is reset
3    M36:=R1<56         ;Setting flag 36, if the content of R1 is smaller
                        than 56, otherwise the flag 36 is reset
4    R2:=10
5    M12:=R!2=0         ;Setting flag 12, if the content of R10 is equal
                        0, otherwise the flag 12 is reset
6    M34:=N1=56         ;Setting flag 34, if the content of N1 is equal to
                        56, otherwise the flag 34 is reset
7    M35:=N1>56         ;Setting flag 35, of the content of N1 is greater
                        than , otherwise the flag 35 is reset
8    M36:=N1<56         ;Setting flag 36, if the content of N1 is smaller
                        than 56, otherwise the flag 36 is reset
9    N37:=255
10   M!37:=N100=510     Flag 255 is set through indirect addressing of
                        flag 37, if the content of N100 is equal to 510 ,
                        otherwise the flag 100 is reset
11   END
```

1.10.3.2 Compare: complex examples

Example 1

Load real number register of 10 to 20 with 0 during program start (reset of counter)

```
1   R1:=10                ;R1 leads to indirect addressing
2   $loop
3   R!1:=0.0              ;Register, displays the content of R1 to assign
                          the value
4   R1:=R1+1              ;Next register
5   M1:=R1<20             ;All 10 register (R10-R20) load with value?
6   G21 M1.1 loop         ;Jump, if not all register are load
7   END
```

Example 2

Production process of a tray with unequal grid

```
1   R11:=1
2   G11.0                 ;switch off display
2   G500.0                ;LC-display is the current data channel
3   G501                  ;delete display
4   $READINGIN
5   G503.1.1              ;Positioning the cursor
6   G510. INPUT PALLET POS.:  ;text output „INPUT PALLET POSITION:“
    G542.25.1.6.R!11      ;Input of the pallet position via front plate into
                          the indirect addressed register R11. The input
                          field appears in column 25, first line and
                          contains 6 signs
7   R11:=R11+1            ;Setting pointer to the next pallet position
8   M1:=R11>5             ;Are all pallet positions read in?
9   G21 M1.0 READING      ;No!
10  R11:=1                ;Position pointer for processing to position 1
11  $NEXT
12  A1:=100               ;Move to the fetching position
13  SUB GETTINGPARTS      ;pick up the piece
14  A1:=R!11              ;moving to placing position
15  SUB DROPPINGPARTS     ;drop the piece
16  R11:=R11+1            ;Position pointer to the next pallet position
17  M1:=R1>6              ;All pallet positons  used?
18  G21 M1.0 NEXT         ;No !
```

Program: GETTINGPARTS

```
1   I1.1                  ;wait until piece is in fetching position
...                       Process for  picking up pieces
n   END
```

Program : DROPPINGPARTS

```
1   I7.1                  Pallet  present
...                       Procedure for dropping pieces
N   END
```

1.10.4 Logical operations

1.10.4.1 Introduction

A logical operation must always start with an LD command and end with an OUT command. Only commands of logical operations may be applied within a logical chain (starting with LD, ending with OUT).

Logical operations are processed according to the following functional mechanism.

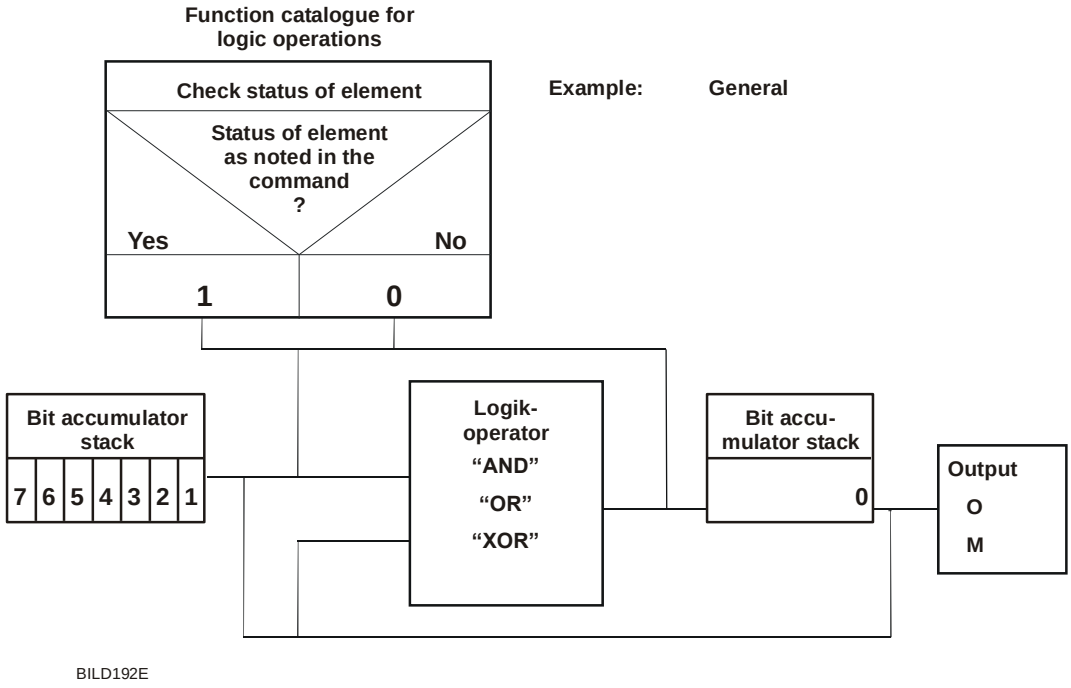


Figure 8: General logical operation mechanism

In order to be able to also apply the notation of the commands "Wait for the status of an element“ (e.g. I3.1, M12.0) to logical operations, both the element, such as inputs, flags, system flags or outputs, and the number, as well as the desired status of the element were assigned to the **LD**, **AND** and **OR** commands. If the current status of the element (I, M, SM, O) corresponds to the status of the element for the logical command, processing is continued with a logical "1“after the command (LD, AND, OR) has been converted. Otherwise processing is continued with a logical "0“.

The result of the logical operations LD, AND and OR is stored in the bit accumulator. In an LD command, the status of the bit-accu is first of all transferred to the bit accumulator stack (a depth of 7 bits) and the result of the check of the element is then stored in the bit accumulator.

The logical operations **AND-LD** and **OR-LD** link the bit accumulator and the bit accumulator stack and store the result in the bit accumulator.

The **OUT** command transfers the status of the bit accumulator to the corresponding element. If the status of the bit accumulator is logical "1“, the output or the flag is set, otherwise it is reset. Multiple OUT commands can directly follow each other.

The programmer can invert the status of the bit accumulator and transfer it to the element, output or flag to be controlled by means of the **NOT** command, i.e. the logical status "1" leads to a RESET of the addressed element.

Each parallel run has its own bit accumulator and bit accumulator stack with a stack depth of 7 bits for complex logical operations.

Command : LD

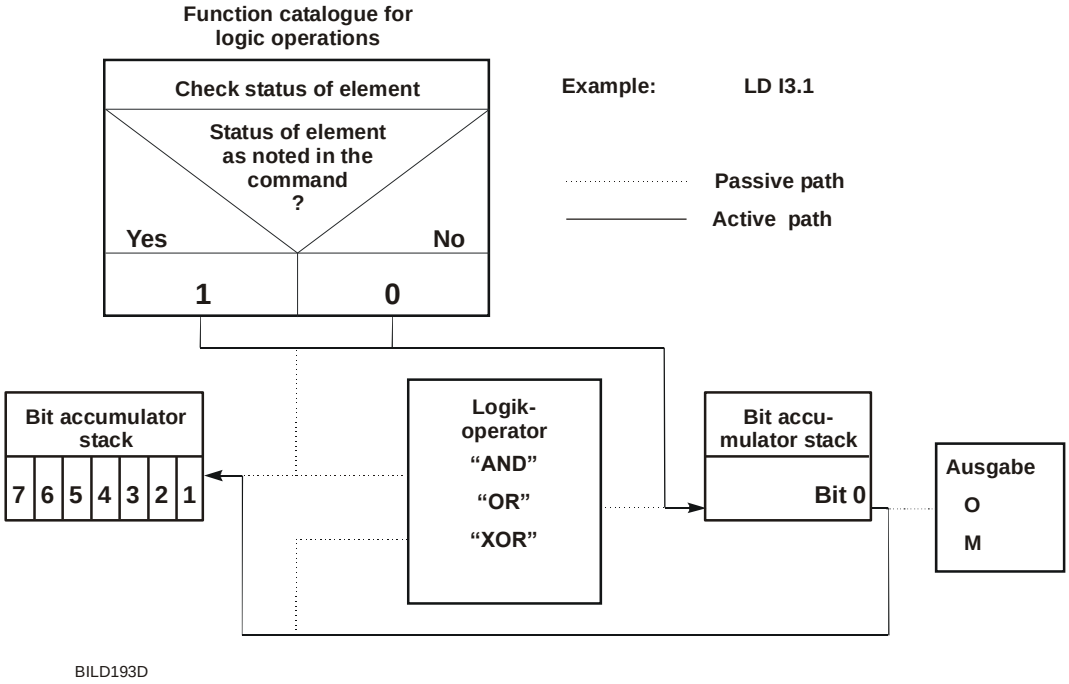


Figure 9: Logical operation mechanism for the command LD

Command : AND or OR

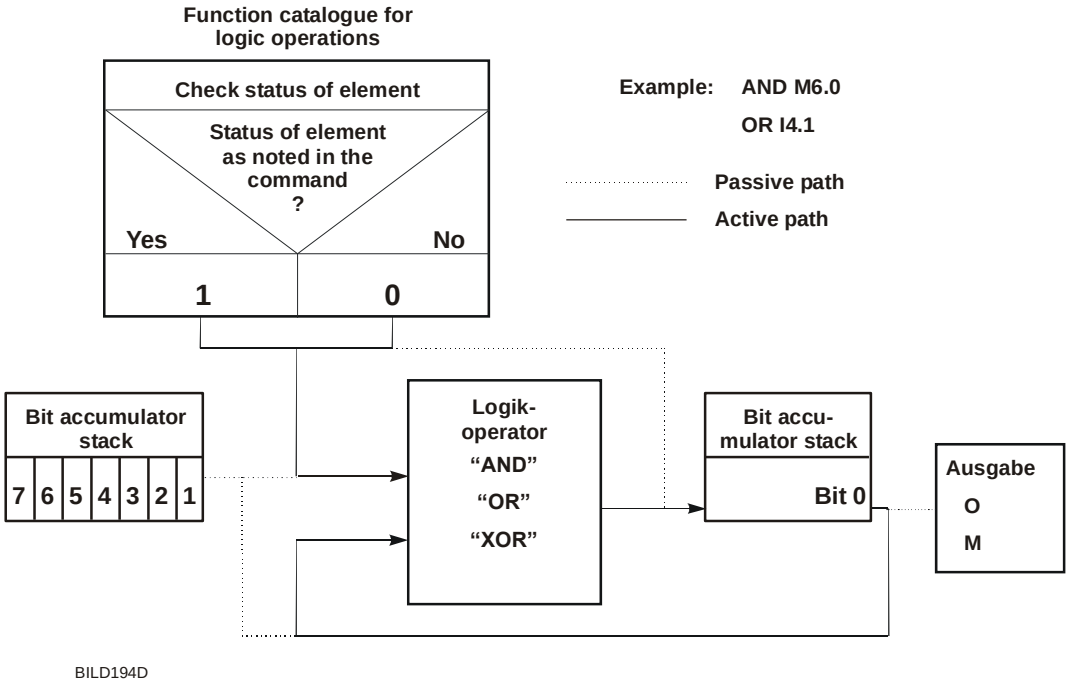


Figure 10: Logical operation mechanism for the commands AND and OR

Command: AND-LD or OR-LD

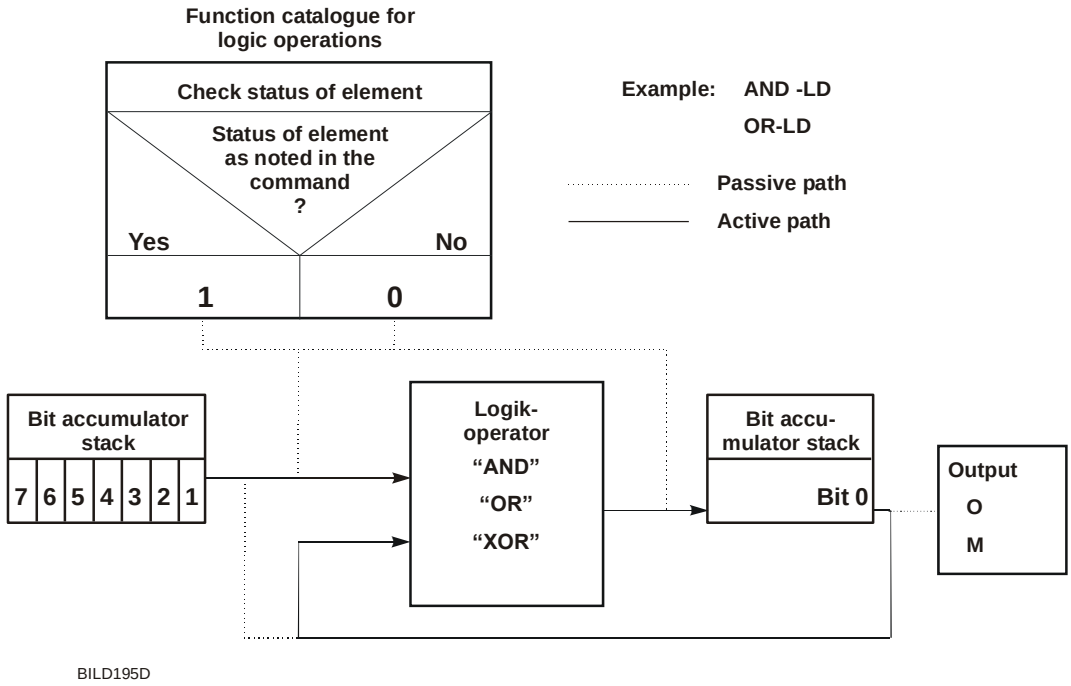


Fig. 11: Logical operation mechanism for the commands AND-LD and OR-LD

The logical operations AND-LD and OR-LD retrieve the last stored bit from the bit accumulator stack, i.e. bit 1 of the bit accumulator stack is moved to the logical operator. This bit is now linked with bit 0 of the bit accumulator stack. The result of this logical operation is then in bit 0 of the bit accumulator stack.

NOTE When bit 1 is read from the bit accumulator stack, the complete bit accumulator stack is moved one position to the right.
Since a maximum of 7 bits can be stored in the bit accumulator stack, the commands AND-LD or OR-LD can only be practically applied a maximum 7 times in succession!

Command : OUT

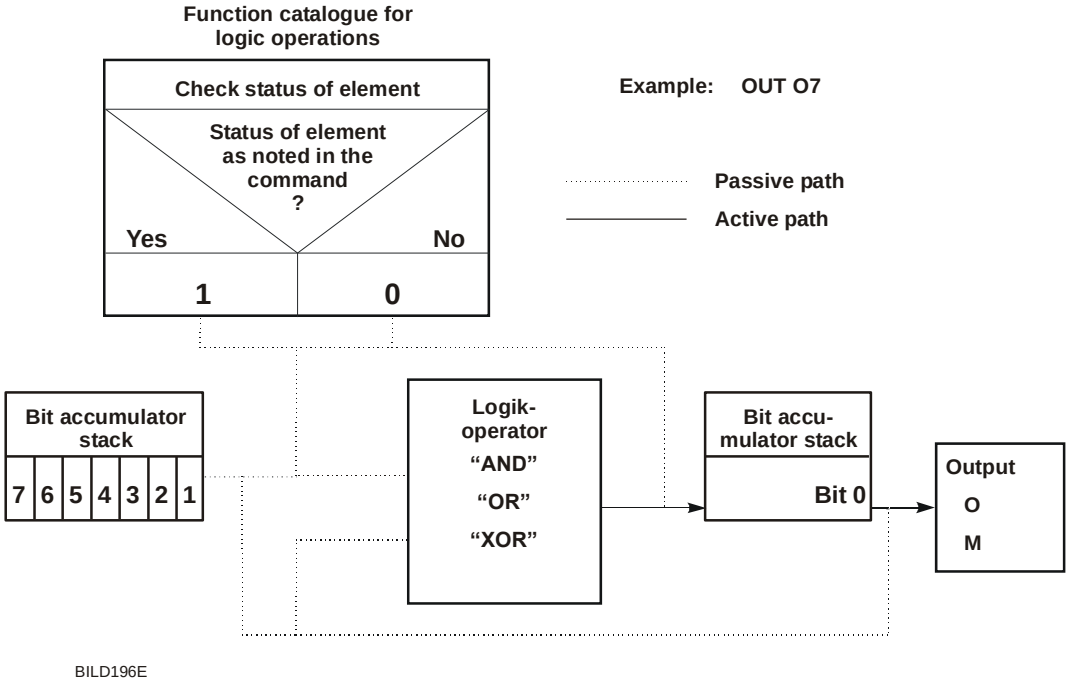


Figure 12: Logical operation mechanism for the command OUT

LD, AND, OUT, NOT

Command format:

AND Ii.n
AND Oi.n
AND Mi.n
AND M!i.n
AND SMi.n

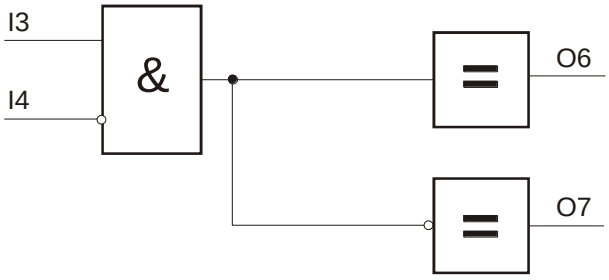
Application:

Creation of logical AND operations between inputs, flags, system flags and outputs. The result of the logical operation can be assigned to a flag or an output.

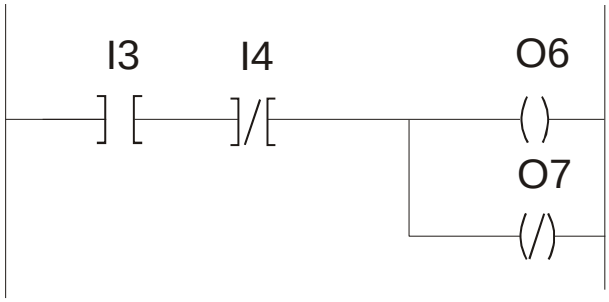
Example

```
1    LD I3.1
2    AND I4.0
3    OUT O6
4    NOT O7
5    END
```

Implementation as a function plan (FUP)



or as a contact plan (COP)



LD, OR, OUT, NOT

Command format:

OR Ii.n
OR Oi.n
OR M!i.n
OR Mi.n
OR Smi.n

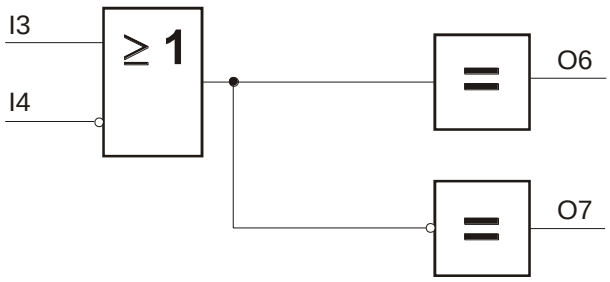
Application:

Setting up logical OR operations between inputs, flags, system flags and outputs. The result of the logical operation can be assigned to a flag or an output.

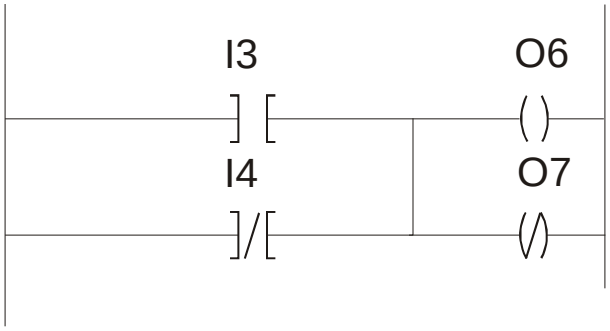
Example

```
1    LD I3.1
2    OR I4.0
3    OUT O6
4    NOT O7
5    END
```

Implementation as a function plan (FUP)



or as a contact plan (COP)



SET / RES

Command format:

SET Mn
RES Mn
SET On
RES On

Extended functions (from output/flag to output/flag # A maximum of 1024 outputs or flags can be influenced).

SET On.Om
RES On.Om
SET Mn.Mm
RES Mn.Mm

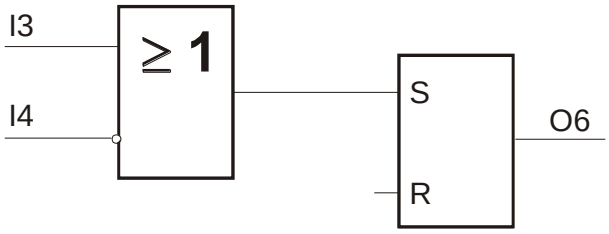
Application:

Simpler handling of results of logical operations arising from logical commands.

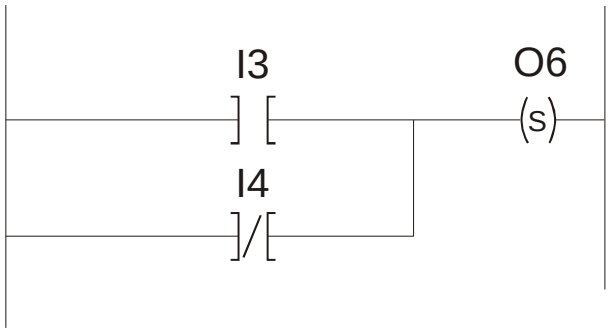
Example 1

```
1    LD I3.1
2    OR I4.0
3    SET O6
4    END
```

Implementation as a function plan (FUP)



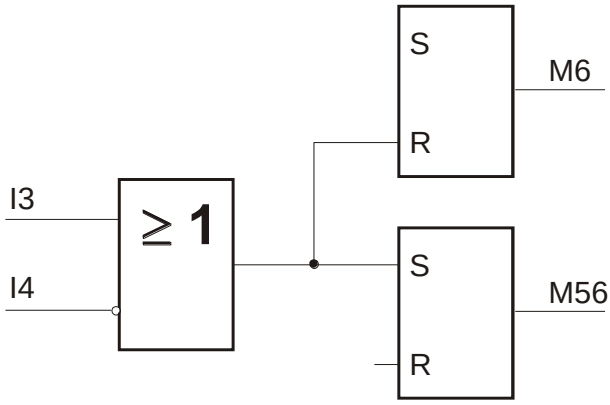
or as a contact plan (COP)



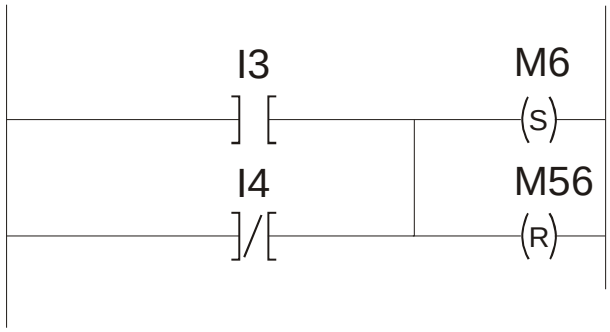
Example 2

```
1    LD I3.1
2    OR I4.0
3    RES M6
4    SET M56
5    END
```

Implementation as a function plan (FUP)



or as a contact plan (COP)



Example 3

The following example shows, how complete blocks of flags may be influenced.

```
1     $A
2     G21 I9.1 SETM3M9
3     G21 I10.1 SETM3M39
4     G21 I11.1 RESM3M9
5     G21 I12.1 RESM3M39
6     JMP A
7     ;
8     $ SETM3M9
9     LD I16.1
10    SET M3.M9                 Set the flags from M3 to M9
11    T10 I9.0
12    JMP A
13    ;
14    $ SETM3M39
15    LD I16.1
16    SET M3.M39                Set the flags from M3 to M39
17    T10 I10.0
18    JMP A
19    ;
20    $ RESM3M39
21    LD I16.1
22    RES M3.M9                 Reset the flags from M3 to M9
23    T10 I11.0
24    JMP A
25
26    $ RESM3M39
27    LD I16.1
28    RES M3.M39                Reset the flags from M3 to M39
29    T10 I12.0
30    JMP A
31    ;
32    END
```

1.10.4.5 XOR logical - operation (Exclusive-OR)

XOR

Command format:

XOR In.i
XOR On.i
XOR Mn.i
XOR SMn.i

Example 1 with truth table

	States			
LD I1.1	untrue	true	untrue	true
XOR I2.1	untrue	untrue	true	true
OUT O1	0	1	1	0

Example 2 with truth table

NOTE In the case of XOR logical operations with more than two variables, the EOR logical operation of the two first variables is first of all implemented and the result is then implemented with the next XOR logical operation (see the next example), and so on.

	States							
LD I1.1	untrue	true	untrue	true	untrue	true	untrue	true
XOR I2.1	untrue	untrue	true	true	untrue	untrue	true	true
XOR I3.1	untrue	untrue	untrue	untrue	true	true	true	true
OUT O1	0	1	1	0	1	0	0	1

1.10.4.6 Multi-level logical AND operation

AND-LD

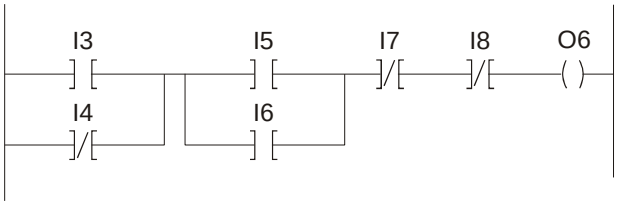
Command format:

AND-LD

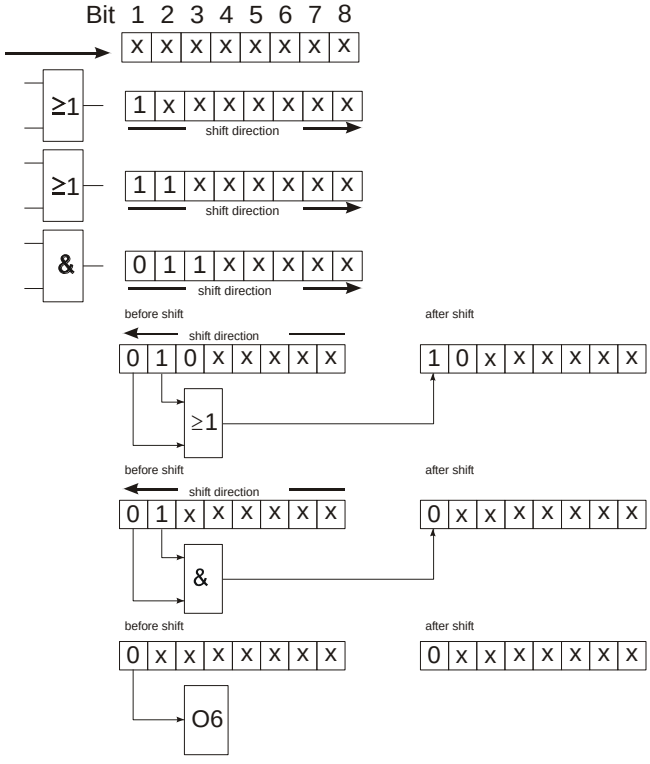
Example

Input	I3	I4	I5	I6	I7	I8
Log. Status	1	0	1	0	1	0

Implementation as a contact plan (COP)



Status of bit accumulation stack



1 L D I3.1
2 OR I4.0
3 LD I5.1
4 OR I6.1
5 LD I7.0
6 AND I8.0
7 OR-LD
8 AND-LD
9 OUT O6
10 END

Application:

Creation of complex (multi-level) logical AND operations between inputs, flags, system flags and outputs. The bit accumulator stack has a depth of 7 bits.

The result of the logical operation can be assigned to a flag or an output.

1.10.4.7 OR-LD - Multi-level logical OR operation

OR-LD

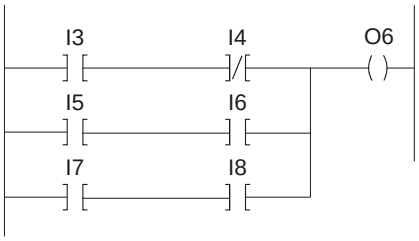
Command format:

OR-LD

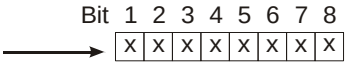
Example

Input	I3	I4	I5	I6	I7	I8
Log. Status	1	1	1	1	1	0

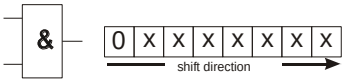
Implementation as a contact plan (COP)



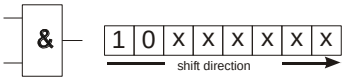
Status of bit accumulation stack



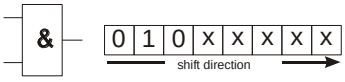
1 *LD I3.1*
2 *AND I4.0*



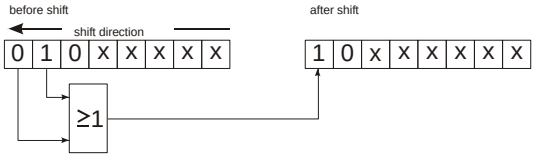
3 *LD I5.1*
4 *AND I6.1*



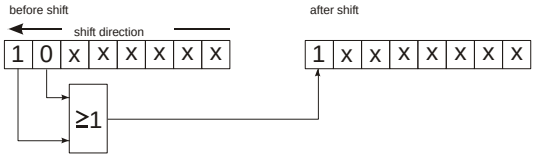
5 *LDI7.1*
6 *AND I8.1*



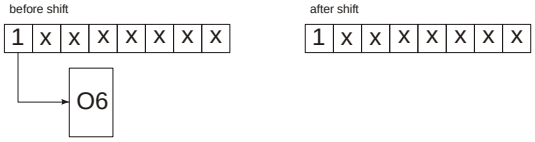
7 *OR-LD*



8 *OR-LD*



9 *OUT O6*



10 *END*

Application:

Creation of complex (multi-level) logical OR operations between inputs, flags, system flags and outputs. The bit accumulator stack has a depth of 7 bits.

The result of the logical operation can be assigned to a flag or an output.

1.10.4.8 Complex logical operation

Description:

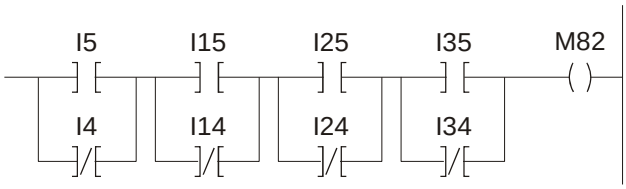
If a multi-level logical operation is executed, the result of a logical operation must be temporarily stored, since it is repeatedly overwritten in the bit accumulator stack. The bit accumulator stack has a depth of 8 bits.

The example shows a multi-level logical operation. If the programming is executed without buffering in flag 82, the bit accumulator stack is overwritten after the second LD command and the result of the logical operation is incorrect.

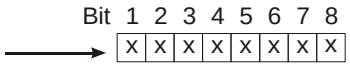
Example

Input	I4	I5	I14	I15	I24	I25	I34	I35
Log. Status	1	1	0	1	1	0	1	1

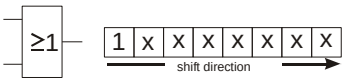
Implementation as a contact plan (COP)



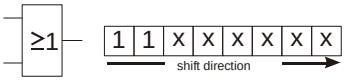
Status of bit accumulation stack



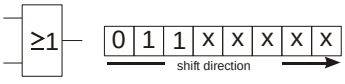
1 *LD I5.1*
2 *OR I4.0*



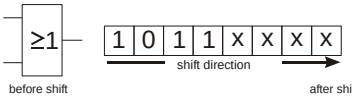
3 *LD I15.1*
4 *OR I14.0*



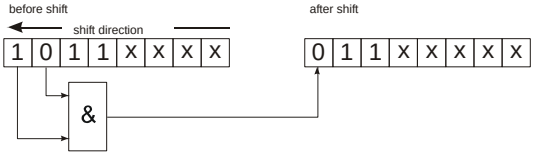
5 *LD I25.1*
6 *OR I24.0*



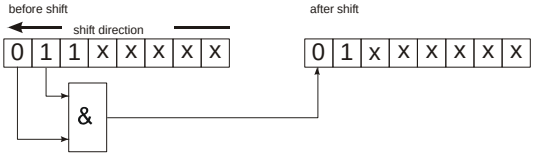
7 *LD I35.1*
8 *OR I34.0*



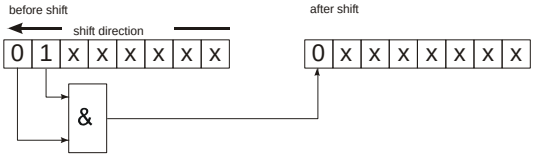
9 *AND-LD*



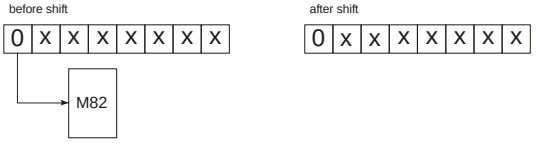
10 *AND-LD*



11 *AND-LD*



12 *OUT M82*



13 *END*

1.10.4.9 Bit-by-bit processing of N registers

AND, OR, EXCLUSIVE-OR

The possibilities for the use of N registers have been extended by the following bit-by-bit operating functions:

Function		Operator
Bit-by-bit	AND	&
Bit-by-bit	OR	|
Bit-by-bit	EXCLUSIVE-OR	^

The commands have been extended for:

Command formats

Ni:=Nn&Nm
Ni:=Nn&Constant

Ni:=Nn|Nm
Ni:=Nn|Constant

Ni:=Nn^Nm
Ni:=Nn^Constant

NOTE The N registers are stored in PA-CONTROL with 32 bits (signed long int). The bit-by-bit processing takes place for all 32 bits!

The following notation results from this:

Dec	Hex	Binary
0	0000 0000	0000 0000 0000 0000 0000 0000 0000 0000
1	0000 0001	0000 0000 0000 0000 0000 0000 0000 0001
100	0000 0064	0000 0000 0000 0000 0000 0000 0110 0100
-1	FFFF FFFF	1111 1111 1111 1111 1111 1111 1111 1111
-2	FFFF FFFE	1111 1111 1111 1111 1111 1111 1111 1110

Example 1

N2:=6	; Binary	0000 0000 0000 0000 0000 0000 0000 0110
N3:=3	; Binary	0000 0000 0000 0000 0000 0000 0000 0011
N12:=N2&N3	; Binary (Result=2)	0000 0000 0000 0000 0000 0000 0000 0010

Example 2

N4:=3	; Binary	0000 0000 0000 0000 0000 0000 0000 0011
N6:=10	; Binary	0000 0000 0000 0000 0000 0000 0000 1010
N3:=N4|N6	; Binary (Result=11)	0000 0000 0000 0000 0000 0000 0000 1011

Example 3

N4:=3	; Binary	0000 0000 0000 0000 0000 0000 0000 0011
N6:=10	; Binary	0000 0000 0000 0000 0000 0000 0000 1010
N3:=N4^N6	; Binary (Result=9)	0000 0000 0000 0000 0000 0000 0000 1001

1.10.5 Register operations

1.10.5.1 Ni.n - Waiting for the logical status of an N register

Ni.Value

N (Number).(Value)

Description:

Delays the processing of the following program section until the status of the N register is equal to the constant. If the polling result is not true, the controller waits until the condition is provided in operator n.

The register value is changed e.g. in a program operating in parallel.

Command format:

Ni.n

Ni.Nm

N!i.n

N!i.N!n

Application:

Polling of program sections running in a cyclic time frame.

Example

1	*$A*	
2	*N2.5*	*Controller waits until the content of register N2 is equal to 5*
3	*O1:=1 T10 O1:=0 T10*	*Flashing of a lamp*
4	*JMP A*	*Jump to A*
5	*END*	

1.10.5.2 Nn:=SNn - Load content of a system N register into an integer register

Nn:=SNn

Description:

System integer registers which are reserved for a wide range of tasks are integrated in the PA-CONTROL (*see section List of system N registers, page 23)*.

In order to gain a statement about the system in the event of errors, faults or particular situations, these registers can be read out and then processed, e.g. shown in the display.

NOTE The integer registers inside the system can only be queried by the user!

Command form:

N100:=SN6

N!20:=SN21

Example 1

```
1    N2:=21                ;
2    N124:=SN102           ; Transfers the number of characters received via
                           the COM 2 interface into the N124 integer
                           register
3    N!2:=SN1              ;Transfers the system runtime of the PA-
                           CONTROL (in seconds) into the N21 integer
                           register
4
5    END
```

1.10.5.3 Rn:=SRn - Load content of a system R register into a real number register r

Rn:=SRn

Description:

46 system real number registers which are reserved for a wide range of tasks are currently integrated in the PA-CONTROL (*see section List of system R registers, page 27)*.

In order to gain a statement about the system in the event of errors, faults or particular situations, these registers can be read out and then processed, e.g. shown in the display.

NOTE The real number registers inside the system can only be queried by the user!

Command format:

R1028:=SR10

R!27:=SR31

Example 1

```
1    R27:=524                ;
2    R12:=SR10               ; Transfers the nominal interpolation speed into
                             real number register 12
3    R!27:=SR31              ;Transfers the absolute position of axis 1 into the
                             real number register R524
4
5    END
```

1.10.5.4 Bit-by-bit processing of N registers

AND, OR, EXCLUSIVE-OR

The possibilities for the use of N registers have been extended by the following bit-by-bit operating functions:

Function		Operator
Bit-by-bit	AND	&
Bit-by-bit	OR	|
Bit-by-bit	EXCLUSIVE-OR	^

The commands have been extended for:

Command formats

Ni:=Nn&Nm
Ni:=Nn&Constant

Ni:=Nn|Nm
Ni:=Nn|Constant

Ni:=Nn^Nm
Ni:=Nn^Constant

NOTE The N registers are stored in PA-CONTROL with 32 bits (signed long int). The bit-by-bit processing takes place for all 32 bits!

The following notation results from this:

Dec	Hex	Binary
0	0000 0000	0000 0000 0000 0000 0000 0000 0000 0000
1	0000 0001	0000 0000 0000 0000 0000 0000 0000 0001
100	0000 0064	0000 0000 0000 0000 0000 0000 0110 0100
-1	FFFF FFFF	1111 1111 1111 1111 1111 1111 1111 1111
-2	FFFF FFFE	1111 1111 1111 1111 1111 1111 1111 1110

Example 1

N2:=6	; Binary	0000 0000 0000 0000 0000 0000 0000 0110
N3:=3	; Binary	0000 0000 0000 0000 0000 0000 0000 0011
N12:=N2&N3	; Binary (Result=2)	0000 0000 0000 0000 0000 0000 0000 0010

Example 2

N4:=3	; Binary	0000 0000 0000 0000 0000 0000 0000 0011
N6:=10	; Binary	0000 0000 0000 0000 0000 0000 0000 1010
N3:=N4|N6	; Binary (Result=11)	0000 0000 0000 0000 0000 0000 0000 1011

Example 3

N4:=3	; Binary	0000 0000 0000 0000 0000 0000 0000 0011
N6:=10	; Binary	0000 0000 0000 0000 0000 0000 0000 1010
N3:=N4^N6	; Binary (Result=9)	0000 0000 0000 0000 0000 0000 0000 1001

1.11 Communication

1.11.1 Communication by MODBUS

The AnyBus Communicator can be connected to the CANopen bus of the PA-CONTROL.

This converter allows the PA-CONTROL to communicate with devices which are equipped with a MODBUS interface (e.g.: temperature controller ...)

For more detailed information, please refer to the description "APP5002_DE_1070171_AnyBusCommunicator_MODBUS.pdf".

1.11.2 Reading values from a MODBUS participant

Ni/Ri:=MODBUS	from V5.07

Command format:

Ni:=MODBUS.<Address>.<Function>.<Register number>

Ri:=MODBUS.<Address>.<Function>.<Register number>

Application:

Reading process data, parameters or configuration data

Example

```
1   N23:=MODBUS.2.3.3180    ;Obtains the content (16-bit integer) of the 3180 register of the MODBUS participant with address 2
2   R4:=MODBUS.1.3.39128    ;Obtains the content (32-bit float) of the 39128 register of the MODBUS participant with address 1
3   END
```

1.11.3 Writing a value to a MODBUS participant

MODBUS	from V5.07

Command format:

MODBUS.<Address>.<Function>.<Register number>:= Ni / constant

MODBUS.<Address>.<Function>.<Register number>:= Ri / constant

Application:

Writing process data, parameters or configuration data

Example

```
1   MODBUS.1.6.3180:=5          ;Writes the value 5 to register 3180 in the
                                MODBUS participant with address 1

2   MODBUS.1.16.39128:=12.3     ;Writes the value 12.3 to register 39128 in the
                                MODBUS participant with address 1
3   MODBUS.5.6.3180:=N4         ;Writes the content of N4 to register 3180 in the
                                MODBUS participant with address 5
4   MODBUS.4.16.39128:=R5       ;Writes the content of R5 to register 39128 in the
                                MODBUS participant with address 4
5   END
```

1.11.4 Communication via control console

1.11.4.1 G11 - Switch display on / off

G11

G11.n

Command format:

G11.0

G11.1

Application:

For processes, in which the display is used for operator prompting and messages by means of the commands of the G500 command group.

Description:

The display of the commands during the program flow is controlled by means of the function G11. The display is switched off by means of the command G11.0 and switched on by means of G11.1.

NOTE The last executed command is displayed irrespective of the program version.

Following commands are never displayed: JMP, M1:=, etc.

Only by this means is it possible to display a previous program line when programing loops.

Example

1	*G11.0*	*;Switch off display*
2	*G500.0*	*;Selects the LC display of PA-CONTROL*
3	*G501*	*;Clears the display and replaces the cursor in the 1st column of the 1st line*
4	*G510.Machine error!*	*;puts out the text "Machine error" to the current display medium, the LC display*
5	*G11.1*	*;Switch on display*
6	*END*	

1.11.4.2 Nn:=CHN - Check whether character transfer in the background has been completed

Nn:=CHN

Description:

The receive status of a G533 or G534 is retrieved by means of the Nn:=CHN command. Branching can then be executed in the program depending on the result.

The result in the integer register is interpreted as follows:

Content of Ni		Description
0	→	End criterion not yet received
1	→	End criterion received
2	→	Transmission error at the serial port (parity, …)
3	→	Receive buffer full, but end criterion not yet received

Command format:

Nn:=CHN

1.11.5 String commands

1.11.5.1 Sn - Copying character strings

Sn:= Si

Description:

Strings are used to store up to 80 ASCII characters which are received e.g. via a serial port. Each process can access the global character buffers S1 to S16 (strings). In addition, each process has its own local character buffer S0.

The contents of S0 can be copied to a global character buffer by means of the command Sn:=S0. It is also possible to copy from a global character buffer to the local character buffer S0.

Command format:

S0:=Sn

Sn:=S0

1.11.5.2 S0:=CHN - Copying character strings

S0:= CHN

Description:

Strings are used to store up to 80 ASCII characters which are received e.g. via a serial port. Each process can access the global character buffers S1 to S16 (strings). In addition, each process has its own local character buffer S0.

The local character buffer S0 is used for processing and analysing character strings (POS, COPY, etc.). The command S0:=CHN copies the contents of the serial receive buffer of the current data channel to the local character buffer. The end criterion for the transfer is the character 0.

The two command formats differ in that the command S0:=CHN is used to copy the contents of the entire serial receive buffer, in the other case, a defined number of characters is copied.

Command format:

S0:=CHN

S0:=CHN.i

Sn:=S0

1.11.5.3 POS - Search for character position in local character buffer S0

POS

POS.(Search position).(Character 0-255)

Description:

A specific character ("m“) can be searched for in the local character buffer S0 by means of the **POS.** command.

"m“ is a number greater than 0 and less than 255 and represents the corresponding ASCII character.

The search in the local character buffer S0 begins at the position "n“. If the character is found, the position (1 to n) is stored in the integer register. If the character is not found, the value "0“is stored in the integer register.

Command format:

Ni:=POS.n.m

Ni:=POS.Nn.m

Ni:=POS.Nn.Nm

Example

```
1   G500.2.6.1.0                ;Initialisation of serial port 2
2   G532.13 U_ERROR             ;Receives characters in the background at the
                                serial port and stores them in the local
                                character buffer S0 until the end criterion 13 =
                                0Dhex = CR has been received
3   N2:=POS.1.35                ;Search in local character buffer S0 for the
                                character 35= 23hex =  # and store the
                                position in N2
4   M1:=N2<1                    ;Character found?
5   G21 M1.1 Z_NO               ;No!
6   N3:=COPY.N2.5 C_ERROR       ;Convert the next 5 characters starting from
                                the value of N2 from the local character buffer
                                S0 to a number and store this number in
                                integer register 3
7   JMP END
8   $U_ERROR
9   SUB Error_1
10  JMP END
11  $C_ERROR
12  SUB Error_2
13  JMP END
14  Z_NO
15  SUB Error_3
16  $END
17  END
```

1.11.5.4 Copy - Convert characters from the local character buffer

COPY

Copy.(Start position).(No. of characters)[(Label)]

Description:

A maximum of 80 characters from the local character buffer (S0) can be converted to a number and stored in an integer or real number register by means of the command COPY.

COPY starts from the "nth" character and considers the following "m" characters provided that there are characters available in the local character buffer (S0). If there are less than "m" characters available, the conversion process is ended without an error.

If an error occurs, the conversion process is aborted. The register is unchanged and the program is continued at the label.

Possible errors:

- "n" is less than 1 or greater than 80
- "m" less than 1 or greater than 80

No legal characters were found in the conversion process.

Legal characters	N Register	R Register
Leading blank characters	X	X
0...9	X	X
Decimal separator "."	X	X
E	-	X, If there is at least one number between 0...9 in front, or two numbers separated by the decimal separator ".". At least one number (1...9) must follow the E! (Number format for the R register)

NOTE Leading blank characters are included for determination of the "nth" character!

Command formats:

Ni:=COPY.n.m Label
Ri:=COPY.n.m Label

Ni:=COPY.Nn.Nm Label
Ri:=COPY.Nn.Nm Label

Ni:=COPY.N!n.N!m Label
Ri:=COPY.N!n.N!m Label

Example

```
1    G500.2.6.1.0          ;Initialisation of serial port 2
2    G532.13 U_ERROR       ;Receives characters in background at the
                           serial port and stores them in the local
                           character buffer S0 until the end criterion 13 =
                           0Dhex = CR has been received
3    N3:=COPY.2.5 C_ERROR  ;Converts the next 5 characters starting from
                           the 2nd character from the local character
                           buffer S0 to a number and stores this number
                           in integer register 3
4    JMP END
5    $U_ERROR
6    SUB Error_1
7    JMP END
8    $C_ERROR
9    SUB Error_2
10   $END
11   END
```

1.11.5.5 Sn:=COPY - get substring

Sn:=COPY

Sn:=Copy.(Star position).(No. of characters)

Description:

An S0 string is also copied to another string (S0 ... S16) by means of the COPY command.

Command formats:

Sn:=COPY.(Start position).(No. of characters)

Examples:

S1:=COPY.4.5
S3:=COPY.N5.N7
S5:=COPY.N!3.N!9

Example

```
1   G500.2.6.1.0        ;Initialisation of serial port 2
2   G532.13 U_ERROR     ;Receives characters in background at the
                        serial port and stores them in the local
                        character buffer S0 until the end criterion 13 =
                        0Dhex = CR has been received
3   S3:=COPY.2.5        ;Converts the next 5 characters starting from
                        the 2nd character from the local character
                        buffer S0 to a number and stores this number
                        in string 3
4   JMP END
5   $U_ERROR
6   SUB Error_1
7   $END
8   END
```

1.11.5.6 Si:=Sn+Sm - Merging strings

Si:=Sn+Sm

The commands of the following command group are used to create a string or to merge two strings.

Description:

- Two strings are merged to form one string
- A string and a number are merged to form one string

No format specifications can be made for the conversion of numbers. N registers are simply converted; R registers are converted by means of the function "ConvertFloatTo ASCII" (see G500 commands).

If the command results in a string which is longer than 80 characters, system error "E754, String too long" is displayed.

Command format:

Si:=Sn+SM
Si:=SN+Nm
Si:=Sn+Rm
Si:=Nm+Sn
Si:=Rm+Sn

Examples:

S0:=S1+S3
S0:=S0+S7
S4:=S3+S4

S1:=S3+N6
S3:=S0+R7
S0:=N4+S0

1.11.5.7 LENGTH - Get the length of a string

LENGTH

The length of a string, i.e. the current number of characters, can be determined by means of the following command.

Command format:

Ni:=LENGTH.Sn

Examples:

N6:=LENGTH.S0
N!7:=LENGTH.S4

1.11.5.8 Create XOR checksum from a string

Ni:=XOR.Sn from V5.07

When executing this command, the XOR checksum is produced from a string (S0, S1 to S16) and saved in the defined N register.

Command format:

Ni:=XOR.Sn

N23:=XOR.S0

N34:=XOR.S3

1.11.5.9 Sn:=Ni / Sn:=Ri - Write the content of a register (number) to a string

Sn:=Ni / Sn:=Ri

A numerical value is converted to a string by means of this command.

- The number of places and the number of decimal places is optional (as in G520)
- If the numerical value is less than the number of specified places, leading blank characters are put out.
- If the number is greater than the number of places, the string becomes longer

Command format:

Sn:= Ni
Sn:= Ni.(No. of places)

SN:=Ri
SN:=Ri.(No. of places).(No. of decimal places)

Examples:

S0:=N7

S0:=N8.5
S2:=N!7.N9

S4:=R2

S5:=R5.9.2
S4:=R6.N3.N7
S7:=R1.N!3.N!7

1.11.5.10 GET - Transfer the content of the local character buffer S0 to an integer register

GET

GET.(Start Position).(No. of Characters)

Description:

Up to four characters can be transferred to an integer register from the local character buffer S0 by means of the command GET.

GET begins from the "Start Position" specified in the command and considers the "Number of Characters" defined in the command.

Possible errors:

- **"Start Position"** is less than 1 or greater than 80
- **"No. of Characters"** is less than 1 or greater than 4
- "**Start Position" + "No. of Characters**" is greater than 80

Command format:

Ni:=GET.n.m

Ni:=GET.Nn.Nm

Ni:=GET.N!n.N!m

Example

1 *N1:=GET.1.2* *;Transfer two characters of the local character buffer S0 starting from the first place and store them in N1*

n-1

n *END*

Local character buffer S0	Byte 1	Byte 2	Byte 3	Byte 4	.....	Content of N1
$3E8_{hex}$	03	E8	00	00		1000
64_{hex}	00	64	00	00		100

1.11.5.11 GETI - Transfer the content of the local character buffer S0 to an integer register in INTEL format

GETI

Ni:=GETI.(Start position).(No. of characters)

Description:

Up to four characters can be transferred to an integer register from the local character buffer S0 by means of the command GETI.

GETI begins from the "Start Position" specified in the command and considers the "No. of Characters" defined in the command.

Possible errors:

- **"Start Position"** is less than 1 or greater than 80
- **"No. of Characters"** is less than 1 or greater than 4
- **"Start Position"** + **"No. of Characters"** is greater than 80

Application:

Process data objects can be transferred in INTEL format via the CANopen bus. The stored information has to be converted for further processing.

Command format:

Ni:=GETI.n.m

Ni:=GETI.Nn.Nm

Ni:=GETI.N!n.N!m

Example

```
1       N1:=GETI.1.2        ;Transfer two characters of the local character
                            buffer S0 starting from the first place from the
                            INTEL format and store them in N1

n-1     ...
n       END
```

Local character buffer S0	Byte 1	Byte 2	Byte 3	Byte 4	.....	Content of N1
$3E8_{hex}$	E8	03	00	00		1000
64_{hex}	64	00	00	00		100

1.11.5.12 PUT - Transfer the content of an N register to the local character buffer

PUT

PUT.(Start position).(No. of characters).(N Register)

Description:

The contents of an N register can be transferred byte-by-byte to the local character buffer by means of the command **PUT**. The characters are stored in the local character buffer starting from the **"Start Position"**. The number of characters specified as the **"No. of Characters"** is stored.

Possible errors:

- **"Start Position"** is less than 1 or greater than 80
- **"No. of Characters"** is less than 1 or greater than 4
- **"Start Position" + "No. of Characters"** is greater than 80

Command format:

PUT.n.m.Ni

PUT.Nn.Nm.Ni

PUT.N!n.N!m.Ni

PUT.n.m.Ri

Example

1	*N1:=100*	*;Load the number 100 to integer register 1*
2	*PUT.1.4.N1*	*;Write the contents of N1 from the first place as a binary number with a length of 4 places (4 bytes) to the local character buffer S0*
n-1	*.....*	
n	*END*	

Example

local character buffer S0	1	2	3	4
$100_{dec} \rightarrow 64_{hex}$	00	00	00	64
$1000_{dec} \rightarrow 3E8_{hex}$	00	00	03	E8
$100000_{dec} \rightarrow 186A0_{hex}$	00	01	86	A0

1.11.5.13 PUTI - Transfer the content of an N register in the INTEL format to the local character buffer S0

PUTI

PUTI.(Start position).(No. of characters).(N Register)

Description:

The contents of an N register can be transferred byte-by-byte to the local character buffer by means of the command **PUTI**. The characters are stored in the local character buffer starting from the **"Start Position"**. The number of characters specified as the **"No. of Characters"** is stored. The storage of numbers in the INTEL format is thereby considered.

Possible errors:

- **"Start Position"** is less than 1 or greater than 80
- **"No. of Characters"** is less than 1 or greater than 4
- **"Start Position" + "No. of Characters"** is greater than 80

Application:

The fact that process data objects may be transferred in the INTEL format via the CANopen bus must be considered when accessing the S0 string.

Command format:

PUTI.n.m.Ni

PUTI.Nn.Nm.Ni

PUTI.N!n.N!m.Ni

PUTI.n.m.Ri

Example

```
1       N1:=100          ;Load the number 100 to integer register 1
2       PUTI.1.4.N1      ;Write the contents of N1 from the first place
                         as a binary number with a length of 4 places
                         (4 bytes) to the local character buffer S0

n-1     .....
n       END
```

Example

Local character buffer S0	1	2	3	4
$100_{dec} \rightarrow 64_{hex}$	64	00	00	00
$1000_{dec} \rightarrow 3E8_{hex}$	E8	03	00	00
$100000_{dec} \rightarrow 186A0_{hex}$	A0	86	01	00

1.12 Analog / Digital

1.12.1 Analog/digital converter - general description

As an option the PA-CONTROL can be fitted with A-D converters. A distinction must be made here between the analog/digital converters designed as IEF modules and those connected via CANopen bus.

(1) 1-way / 8-way IEF module A-D converter. An A-D converter module can be used in the PA-CONTROL.

(2) Use of up to a maximum of 32 A-D converters via CANopen bus
- ID 17: AD1...AD4
- ID 18: AD5...AD8
- ID24: AD31…AD32

(For more information, please refer to the documentation APP5019_DE_1085270_PAC_mit_Ein-Ausgaengen_am_CAN-Bus.pdf)

1.12.1.1 Get A-D values

ADi

Description:

The value of the A-D converter can be transferred directly. The user can transfer the bit value of the A-D converter to an integer register or transfer the result to a real number register parameterized to the full-scale deflection.

Command format:

Nn:=Adi

Rn:=Adi

	IEF-Modul 527451 (12-Bit-transfer)	IEF-Modul 1077914 (12-Bit-transfer)	BK 5120 (15-Bit-transfer
Ni:=ADj	-4096….0….+4096	-4096….0….+4096	-32767 0 +32767
Ri:=ADj	-10VDC...0...+10VDC	-5VDC...0...+5VDC	-10VDC...0...+10VDC

Application:

Acquisition of temperatures, forces, speeds, etc.

Example

```
1   $BEGINNING
2   N100:=2250                  ;Specify limit
3   $LOOP                       ;
4   N101:=AD1                   ;The bit value is transferred to register N101.
5   M100:=N100<N101
3   G21 M100.1 LOOP
4   O8:=1                       ;Error lamp on
5   I8.1 I8.0                   ;Acknowledgement key for errors
6   O8:=0                       ;Error lamp off
7   R1:=AD1                     ;The value converted to the specified full-
                                scale deflection (voltage which corresponds to
                                a pressure value, temperature, etc.) is
                                transferred to real number register R1
8   JMP BEGINNING               ;
9   END
```

1.12.2 Digital-to-analog converter

Brief description

As an option, PA-CONTROL can be equipped with 8 D-A converters on a CANopen bus. Every 4 DA-Converter are summarised to one group (hardware module).

CAN-ID 17 :	DA 1	to	DA 4
CAN-ID 18 :	DA 5	to	DA 8
CAN-ID 32 :	DA 61	to	DA 64

The whole group is addressed in a telegram via CANOpen bus and a fixed PDO-addresses (PDO = process data objects).

For more information check: “APP5019_DE_1085270_PAC_mit_Ein-Ausgaengen_am_CAN-Bus.pdf“

NOTE If the automatic mode is exited, the D-A converters are set to the value 0!

Diagnosis

Diagnosis of the D-A converter is possible via PA-CONTROL front plate or IEF control console.

NOTE

The output values of the D-A converters are set to 0 each time this menu item is entered.

The input field is opened by means of the enter key and new D-A value can be entered.

1.12.2.1 Dai - Output of DA values

DAi

Command format:

DAi:=Nn

DAi:=Rn

Application:

Output of analog values

Example

```
1    $BEGINNING
2    N2:=32767          ;Load the number 32767 to the N2 register
3    DA1:=N2            ;Output contents of N2 as an analog value,
                        here the number 32767 results in an output
                        voltage of +10VDC at a resolution of
                        15Bit/10V
4                       ;
5    R2:=5.2            ;Load the numerical value 5.2 to the real
                        number register R2
6    DA2:=R2            ;Output contents of R2 as an analog value,
                        here the numerical value 5.2 results in an
                        output voltage of 5.2V at a scaling of 10V as a
                        maximum output value of the converter
n    END
```

1.13 File commands

1.13.1 Commands of the G17X-Group

1.13.1.1 G170 - Read character from PTX file

G170

G170.(Line). (Column).(Character).(File)

Description:

The G170 command reads a character from a PTX file and stores the determined ASCII value of the character in the integer register Nm (see Table "PAC key code + ASCII character set", *chapter 2, page 288)*.

The exact position of the character in the file is determined via the integer registers Ni and Nn. Ni specifies the line number and Nn the place in the selected line.

NOTE The value of the integer registers Ni and Nn must be greater than 0.

If the program file does not exist, the message "Program does not exist " is put out. If line Ni does not exist in the PTX file or the selected line is shorter than Nn, register Nm is set to 0 and the system flag SM6 is set.

A check can be carried out as to whether the command has been executed without error by means of the system flag. SM6=0 → command has been correctly executed, SM6=1 command execution with error.

Command format:

G170.Ni.Nn.Nm.TEXTS

G170.N!i.N!n.N!m.NAMES

G170.Ni.Nn.Nm.Sx

G170.N!i.N!n.N!m.Sx

Example

```
1   N1:=5                  ;Line number in file TEXTS.PTX
2   N2:=3                  ,3rd character in the 5th line
3   ; N10                  ;Register N10 contains the read ASCII value
4   G170.N1.N2.N10.TEXTS   ;Read 3rd character in 5th line of file
                           TEXTS.PTX and write the ASCII value to
                           register N10
5   END
```

Program: TEXTS.PTX

```
1   MONDAY
2   TUESDAY
3   WEDNESDAY
4   THURSDAY
5   FRIDAY
6   SATURDAY
7   SUNDAY
```

End of the assignment

1.13.1.2 G171 - Write character to PTX file

G171

G171.(Line).(Column).(Character).(File)

Description:

The **G171** command writes a character to a PTX file. The exact position of the character in the file is determined via the integer registers Ni and Nn. Ni specifies the line number and Nn the place in the selected line.

The numerical value from the integer register Nm is converted to ASCII code (*see Table "PAC key code + ASCII character set; see chapter PA-CONTROL key code, page 288*). Illegal characters are: 0x00=NUL, 0x0A=LF and ox0D=CR.

NOTE The value of the integer registers Ni and Nn must be greater than 0.

If the program file does not exist, the message "Program does not exist " is put out. If line Ni does not exist in the PTX file or the selected line is shorter than Nn, the system flag SM6 is set.

A check can be carried out as to whether the command has been executed without error by means of the system flag. SM6=0 → command has been correctly executed, SM6=1 command execution with error.

Command format:

G171.Ni.Nn.Nm.TEXTS
G171.N!i.N!n.N!m.NAMES
G171.Ni.Nn.Nm.Sx
G171.N!i.N!n.N!m.Sx

Example

```
1   N1:=5                ;Line number in file TEXTS.PTX
2   N2:=3                ,3rd character in 5th line
3   ; N10                ;Register N10 contains the read ASCII value
4   G171.N1.N2.N10.TEXTE ;Overwrites the 3rd character in 5th line of file
                         TEXTS.PTX with the ASCII value (value from
                         N10)
5   END
```

Program: TEXTS.PTX

```
1   MONDAY
2   TUESDAY
3   WEDNESDAY
4   THURSDAY
5   FRIDAY
6   SATURDAY
7   SUNDAY
```

End of the assignment

1.13.1.3 G172 - Write line from PTX file to string (Sn)

G172

G172.(Line).(String).(File)

Description:

The **G172** command copies the line specified in the command from a PTX file to a character string. In this connection, the line is specified directly in the command or addressed via a register. The target address can be either a global string or the local string.

NOTE If the file does not exist, the error message "Program does not exist" will be put out. If the addressed line does not exist or the line is longer than the maximum string length, the system flag "SM6" is set.

A check can be carried out as to whether the command has been executed without error by means of the system flag. SM6=0 → command has been correctly executed, SM6=1 command execution with error.

Command format:

G172.4.S2.ERRORTEXTS
G172.N8.S7. MENUTEXTS
G172.N!5.S6. MESSAGETEXTS
G172.4.S2.S0
G172.N8.S7.S1
G172.N!5.S6.S2

Example 1

```
1   G172.5.S2.MENUTEXTS     ;Copies the 5th line of the file
                            "MENUTEXTS.PTX" to the string "S2"
2   END
```

Example 2

```
1   N8:=12                  ;Line number in file ERRORTEXTS.PTX
2   G172.N8.S0.ERRORTEXTS   ;Copies the 12th line from the file
                            "ERRORTEXTS.PTX" to the local character
                            buffer S0
3   END
```

Example 3

```
1   S1:="Settings_00"
2   N8:=12                  ; Line No. of the *.PTX file
3   G172.N8.S0.ERRORTEXTS   ; copies line 12 out of the file
                            „EINSTELLUNGEN_00" into the local S0
                            character buffer.
4   END
```

1.13.1.4 G173 - Store string (Sn) in a line of a PTX file

G173

G173.(Line).(String).(File)

Description:

The **G173** command stores the content of the string "Sn" in a selected line of a PTX file. In this connection, the line is specified directly in the command or addressed via a register. The target address can be either a global string or the local string.

NOTE If the file does not exist, the error message "Program does not exist" is put out. If the addressed line does not exist or the string is longer than the line, the error message "STORE command could not be implemented correctly" is put out.

If the line of the PTX file is longer, the rest of the line is overwritten with blank characters. A line should not be longer than 80 characters; otherwise the message "Line too long" is put out when the command G172 is applied.

Command format:

G173.6.S0. MESSAGES
G173.N2.S1.ERROR_1
G173.N!9.S2.ERROR_2
G173.6.S0.S1
G173.N2.S1.S0
G173.N!9.S2.S11

Example 1

```
1   G173.6.S0.MESSAGES      ;Stores the contents of the local string "S0" in
                            the 6th line of the file "MESSAGES"
2   END
```

Example 2

```
1   N2:=21                  ;LINE number of the file ERROR_1.PTX
2   G173.N2.S1.ERROR_1      ;Stores the contents of the global string "S1"
                            in the 21st line of the file "ERROR_1.PTX".
3   END
```

Example 3

```
1   S0:="ERROR_INFO"
2   N2:=21                  ; Line No. of the *.PTX file
3   G173.N2.S1.S0           ;saves the content of the global string „S1" into
                            the 21st line of the file „FEHLER_INFO.PTX".
4   END
```

1.13.1.5 STORE –Storage of current values in a program file

STORE

Store.(Name)

Description:

All the current values of N registers, R registers, strings and flags are stored in a PNC, PNX or PAB program by means of the command STORE.

When executing the STORE command, the invoked PNC program is checked for assignments. If a register, flag or string assignment is found, the current value is entered in the assignment. Sufficient blank characters must be available in the PNX, PNC or PAB programs for this purpose.

- Maximum required number of placeholders for registers: 9 places
- Maximum required number of placeholders for flags: 1 place
- Maximum required number of placeholders for strings: current string length

HINWEIS

The **STORE command** must not call up the program in which the command is situated itself.

The **STORE command** may only call up programs that are **not** active at that time.

If the **STORE command** is in a subroutine, the superior program must not be called up.

Before executing the STORE command			
1	R10:=0	→	; A1 pick-up position
2	R11:=0	→	; A1 intermediate position
3	N10:=0	→	; Number of products to be processed
4	M101:=0	→	; Product flag for product 1
5	S1:=”000000“	→	; Greeting
6	END		

After executing the STORE command			
1	R10:=145.345	→	; A1 pick-up position
2	R11:=34565.789	→	; A1 intermediate position
3	N10:=645	→	; Number of products to be processed
4	M101:=1	→	; Product flag for product 2
5	S1:=”HELLO“	→	; Greeting
6	END		

NOTE The invoked PNC/PNC/PAB program must exist. A program line may only contain one assignment.

Sufficient zeros or blank characters must be available for overwriting after the assignment up to ";" (comment) or "CR" (end of line).

Calculation assignments are **not** allowed in the program which is to be overwritten.

If the STORE instruction is not successful because of a lack of free space, error E569 is set.

Command format:

STORE.(Name)

Application:

Storage of current values in various program files, e.g. traversing positions (R register) or status flags can be stored for different product types and called up again as required.

Example

```
1   R10:=R13          ;Read value via register e.g. 4678.123
2   N1:=N13           ;Read value via register e.g. 967
3   M25:=N1>11        ;M25 is set if content of N1 is greater than 11.
4   S1:=GRIPPER       ;GRIPPER is loaded to string S1
5   STORE.TA2_PE_001  ;Overwrite assignments in the PNC program
                      TA2:=PE_001 with current values of the
                      registers and flags
6   END
```

Program TA2_PE_001 before execution of the STORE command:

```
1   R10:=000000000    ; A1 pick-up position for product type 1
2   N1:=0000000000    ; Number of products to be processed
3   M25:=0            ; Product flag for product 1
4   S1:="LAMP_025"    ; Labelling for subsequent output
5   END
```

Program TA2_PE_001 after execution of the STORE command:

```
1   R10:=4678.123     ; A1 pick-up position
2   N1:=967           ; Number of products to be processed
3   M25:=1            ; Product flag for product 1
4   S1:="LAMP_016"    ; Labelling for subsequent output
5   END
```

NOTE In the example, blank characters are represented by an underscore " _ ".

1.13.1.6 STOREEND / STORESTART - Deactivating / activating assignment with constants

STOREEND, STORESTART from V5.07

Description:

When executing the STORE command, all “assignments with constants” have in the past been processed using the STORE command.

If some “assignments with constants” should not be processed in a program (i.e. the value is to be retained unchanged) the “STOREEND” and “STORESTART” commands can be used to deactivate and reactivate processing.

Example:

In the example below, the constants for assigning the “N2” and “R1” variables are to be retained unchanged.

Example

Program: Start.pnc

```
1    $A
2    G21 I9.1 SPEICHERE_W_TYP1
3    G22 I10.1 LADE_TYP1
4    ;
5    $ SPEICHERE_W_TYP1
6    STRORE LADE_TYP1
7    JMP A
8    ;
9    END
```

Program: LADE_TYP1

```
1    N1:=1
2    STOREEND
3    N2:=2
4    R1:=0.000000
5    STORESTART
6    R2:=0.000000
7    M1:=0
8    M2:=0
9    S2:=”OTTO IST GROSS”
10   END
```

STOREEND deactivates processing before “N2:=...”

STORESTART reactivates processing after “R1:=...”

1.13.1.7 CASE.STORE.N - Storage of values if content of N-register

CASE.STORE.N

CASE.STORE.(Variable)

Description:

The **CASE.STORE** command is a branch which is dependent on the content of the integer register **Ni**. PA-CONTROL checks the content of the integer register and carries out a subroutine call in accordance with the value. If the content of the integer register is less than 1 or greater than the number of elements in the name table (in this example: TA2:=PE_001, TA2:=PE_002), a jump to the label defined after ELSE is carried out. Otherwise the respective subroutine is called and processed and the program run is continued with the program line after the ELSE branch.

The current values of N registers, R registers and flags are stored in a PNC/PAB program by means of the **CASE.STORE** command *(see STORE command).*

Command format:

CASE.STORE.Ni

(Name1)

(Name2)

...

ELSE Label_e

Application:

Branch for storing current register values in various PNC/PAB files, e.g. traversing positions (R register) or status flags can be stored for different product types and called up again as required.

Example

```
1       $ERROR
2       N8:=N13                 ;Read value via register
3       CASE.STORE.N8           ;Check contents of N8 and call subroutine if:
4       (TA2_PE_001)            ;N8=1: check subroutine "TA2:=PE_001" for
                                assignments and update
5       (TA2_PE_002)            N8=2 : check subroutine " TA2:=PE_002" for
                                assignments and update
6       ELSE ERROR              ;Jump to label "ERROR" if value of N8 is out
                                of the specified range (in this case if N8<1 or
                                N8>2)
7       END
```

Program: TA2_PE_001

```
1       R10:=0                  ;A1 pick-up position for product type 1
2       R11:=0                  ;A1 intermediate position for product type 1
3       N10:=0                  ;Number of products to be processed
4       M101:=0                 ;Product flag for product 1
5       END
```

Program: TA2_PE_002

```
1       R10:=0                  ;A1 pick-up position for product type 2
2       R11:=0                  ;A1 intermediate position for product type 2
3       N10:=0                  ;Number of products to be processed
4       M102:=0                 ;Product flag for product 2
5       END
```

1.14 Counter

1.14.1 General principles

The PA-CONTROL can be equipped with up to 32 hardware counters (CNT1 to CNT32) via the CAN bus. The plug X1 of PA-CONTROL smart is equipped with an additional counter (CNT0).

The “Incremental encoder” operating mode has been implemented by the functions (operating modes) of the bus terminals (KL5101, KL5111) as “Quadrature decoder” with 4-way evaluation and the forward/reverse counter. The hardware counter (CNT0) at PA-CONTROL smart is only implemented with the “Quadrature decoder” with 4-way evaluation.

Once the PA-CONTROL has been switched on and re-initialised, the counters are always set to 0 and to “ENCODER” operating mode.

Description	CNT 0 (only PA-CONTROL Smart)	CNT1 to CNT16 (about CAN-Bus, KL5101, KL5111)
Input level	RS422	See KL51xx
Maximum input frequency	200 000 increments per seconds	See KL51xx, the counter reading is updated approx. every 2ms via the CAN bus
Forward / reverse counter	No	Yes
Quadrature decoder	Yes	Yes
Counter value	31 bits plus prefix	31 bits plus prefix
Diagnosis with WinPAC	SN110	SN111 to SN126

1.14.2 Hardware

The CANopen bus terminal from Beckhoff BK5120 is used on the PA-CONTROL with the “KL5101” or “KL511” terminals.

The following definitions have been laid down:

- Maximum of 2 terminals per bus node
- Maximum of 16 bus nodes with counter terminals in the CANopen system

Once switched on, the available counter terminals of the bus nodes are assigned to the PA-CONTROL counters in ascending order.

CAN-ID – bus terminal	Equipped with terminals	Counter (CNT) in the PA-CONTROL
ID17 – BK5120	4 x KL2134 1 x KL5111	 CNT1
ID18 – BK5120	4 x KL1104 1 x KL4032 1 x KL3064	-
ID19 – FESTO CPV	-	-
ID20 – BK5120	1 x KL1104 1 x KL5111 1 x KL5101	- CNT2 CNT3

1.14.3 Commands

1.14.3.1 Initialising counters

CNTn.INIT.i

CNT4.INIT.0
CNT2.INIT.1

Description:

The **CNTn.INIT.i** command is used to initialise the counters via the counters connected to the CANopen bus.

Command format:

CNT4.INIT.0 ; counter 4 operating in “Encoder” operating mode
CNT2.INIT.1 ; counter 2 operating in “Forward/reverse counter” operating mode

Operating mode: “Incremental encoder” as “Quadrature decoder” with 4-way evaluation		
Input	Function of input	
Input A	Encoder-A	
Input B	Encoder-B	
Input C	No function	

Operating mode: Forward / reverse counter		
Input	Function of input	
Input A	Count, the positive flanks of the input pulses are counted	
Input B	Forward / reverse input 0 : counts in forward direction 1 : counts in reverse direction	
Input C	Gate input: 0 : counter is approved 1 : counter is blocked	

Application:

Definition of “operating mode” of the detected counters on the CANopen bus (Encoder or forward/reverse counter operating mode).

1.14.3.2 Setting counters

CNTn:=Ni

CNTn:=<Constant>
CNTn:=Ni

Description:
The **CNTn.INIT.i** command can be used to assign a value to a counter connected via CANopen bus.

Command format:
CNT1:=1234
CNT3:=N67

Application:
Definition of output values for counting processes, e.g. positions.

1.14.3.3 Reading counters

Ni:=CNTn

Ni:=CNTn

Description:
The **Ni:=CNTn** command can be used to read a value from a counter connected via CANopen bus. The read value is stored in the Ni integer register.

Command format:
N34:=CNT2

Application:
Querying counter readings

CNTn.i.NM

CNTn.i.Nm

Description:

The **CNTn.i.Nm** command can be used to compare the content of a counter connected via CANopen bus with a constant.

Command format:

CNT2.1.1234	wait until counter reading of counter 2 is greater than 1234
CNT2.0.-345	wait until counter reading of counter 2 is less than -345
CNT2.1.N4	wait until counter reading of counter 2 is greater than the content of N4
CNT2.0.N4	wait until counter reading of counter 2 is less than content of N4

1.15 Temperature control

1.15.1 General principles

In Version 5.07 and higher, temperature can be controlled using the PA-CONTROL. The modular “KS VARIO” multi-channel control system which is connected via the CAN bus is used for this purpose. The field bus terminal can be equipped with a 4-channel or 30-channel temperature controller (*for details, refer to operating manual of PA-CONTROL, Options chapter*).

The “KS VARIO” temperature controllers are only communicated with via SDOs. The following commands have been set up for this purpose for automatically operating the PA-CONTROL:

- Write parameter value *(see chapter 1.15.2.1, page 215)* and
- Read parameter value (*see chapter 1.15.2.2, page 216)*

The temperature controller is started and the interface data set using the “Blue Control” program from PMA.

1.15.1.1 Parameters

The parameters listed in the table below are important for programming:

Designation in command <Parameter name>	Used for		PMA designation	Description
SP	N,R	R/W	SP	Controller set point, (temperature specification in °C)
SP_LO	N,R,	R/W	SP.LO	Minimum limit value for set point specification
SP_HI	N,R	R/W	SP.HI	Maximum limit value for set point specification
X_EFF	N,R	R	X.Eff	Effective actual value
X_PHYS	N,R,	R	Xphys	Physical actual value directly from the terminal
C_OFF	N	R/W	X.Off	Switch off controller 1 = controller off, 0 = controller on
YPID	N,R	R	Ypid	Actuating variable of controller
DIGOUT1	N	R	DIGOUT1	Signal on digital output 1
DIGOUT2	N	R	DIGOUT2	Signal on digital output 2
C_STA	N	R	C.Sta	Status of controller
C_STA2	N	R	C.Sta2	Status of controller
BOOST	N	R/W	BOOST	The boost function briefly increases the set point

For a detailed description of the individual parameters, please refer to the “KSvario CON.pdf” document from PMA.

1.15.2 Commands for temperature controls

1.15.2.1 Writing parameter value

TCn.Parameter name:=	from V5.07

TC3.Sp:=25.5
TC2.C_OFF:=1

Description:
The **TCn.Parameter name:=value** command is used to transfer a parameter value from the control to the controller

Command format:
TC<channel number>.<parameter name>:=<value>

TC<channel number>.<parameter name>:=Ni
TC<channel number>.<parameter name>:=Rn
TC<channel number>.<parameter name>:=INTEGER_CONSTANT
TC<channel number>.<parameter name>:=REAL_ CONSTANT

Example:
Please see complex example on *page 216.*

1.15.2.2 Reading parameter value

Rn:=TCi.Parameter name from V5.07

R27:=TC1.X_EFF
N35:=TC2.C_STA

Description:

The **Rn:=Tci. Parameter name** command is used to bring a current value from the controller into the PA-CONTROL

Command format:

Register:=TC<channel number>.<parameter name>
Rn:=TC<channel number>.<parameter name>
Nn:=TC<channel number>.<parameter name>

Example

```
1                                   ; Temperature controller
2       N10:=0
3       TC1.C_OFF:=N10              ;Switch on temperature controller
4
5       R10:=100.0                  ; Load set point
6       TC1.SP:=R10                 ; Set set point
7
8       $LOOP
9       R11:=TC1.X_PHYS             ; Read actual value
10      M10:=R11<R10
11      G21 M10.1 LOOP              ;Wait until end of temperature control
12      M10:=0                      ; Reset, end of temperature control
13
14      N10:=1
15      TC1.C_OFF:=N10              ;Switch off temperature controller
16      END
```

1.16 Command groups of the PA-CONTROL

1.16.1 Commands of the G18x group

1.16.1.1 G18x - Acquisition of several values

G18x.

Introduction:

The **G180** command group is used to acquire A-D converter values. A-D converter values are retrieved from the A-D converter in accordance with the acquisition criteria position/path or time and stored in the N registers. The stored values can then be further processed.

1.16.1.2 G180 - A-D values synchronous to the axis movement

G180

G180.(Axis).(Position).(Resolution).(AD No).(1st N Register).(Number)

Command format:

G180.A1.R27.R31.5.100.25 (see the following table)

G180.A3.R23.R2.N7.N2.N3

G180.A1.R!23.R!2.N!7.N!2.N!3

Axis / e.g. A1	:	Axis which is monitored, always as a constant
Position e.g. R Reg. 27	:	The axis position from which the A-D values are acquired is stored in this R register. The information as to whether the acquisition is to be started if the position is larger or smaller is derived from the sign character of the resolution. An R register must be used.
Resolution / e.g. R Reg. 31	:	The resolution, i.e. the path according to which the next A-D value is to be retrieved, is stored in this R register. The sign character of the resolution applies as a criterion as to whether the acquisition is to be started if the position is larger or smaller. An R register must be used.
AD No. e.g. A-D converter 5	:	Specifies the number (from 1 to 8) of the A-D converter, from which the values are retrieved.
1st N Register / e.g. N Reg. 100	:	Specifies the number of the first N register, starting from which the A-D values are stored. The address of the N register is increased by 1 with each new measured value. The specification can ensue as a constant or via an N register.
Number / e.g. max. number = 25	:	Specifies the maximum number of A-D values which are to be acquired. This number also specifies the number of integer registers required for storage of the measured values. The specification can ensue as a constant or via an N register.

NOTE The number of the acquired A-D values is stored in the system N registers SN51 – 56 (SN51 for axis 1, SN52 for axis 2, etc.) and is available for polling, see the example below. If the axis is stopped or deactivated, the G180 command is terminated.

NOTE **Maximum sampling rate:**

PA-CONTROL MP with A-D converter 2x	:	0.2ms
PA-CONTROL MP with A-D converter 8x	:	1 ms
PA-CONTROL SINGLE	:	1 ms
PA-CONTROL COMPACT	:	1 ms
PA-CONTROL STEUER	:	1 ms

Example

```
1   G25.A1 G90.A1
2   R1:=10          ;Acquisition starting from axis position 10mm
3   R2:=0.5         ;An A-D value is to be acquired every 0.5mm
4   G180.A1.R1.R2.1.100.200
                    ;Starting from the 10mm position, acquire at
                    the next positioning of axis 1, a maximum of
                    200 values every 0.5mm from A-D converter 1
                    and store these in the memory starting from N
                    register 100
5   N1000:=SN51     ;Load system N register 51 to N register 1000
6   M55:=N1000>180  ;Compare contents of N Reg. 1000 with 180
7   G22 M55.1 ERROR ;Jump to UP ERROR if <SN51> >180
8
9   A1:=1000        ;Traverse to position 1000
10  END
```

Program: ERROR

```
2   O32:=1          ;Switch on error lamp
3   I32.1           ;Wait for acknowledgement
4   I32.0           ;
5   O32:=0          ;Switch off error lamp
6   END
```

1.16.1.3 G181 - Enter A-D values in a defined time grid

G181

(G181.time grid.A-D-no.Start Register.Number)

Description:

The A-D values are acquired at a specified interval.

Command format:

G181.5.1.100.50

G181.N1.1.N2.N3

Application:

Acquisition of dynamic A-D values, e.g. torque values for screwing or force values for joining and pressing.

Example 1

1	*G210.A1*	;Activates the start positioning mode for axis 1
2	*A1:=100*	*;Approach position 100*
3	*G230.1.A1.50*	*;Wait until position >50.0 has been reached*
4	***G181.5.1.100.50***	*;Values of A-D converter 1 are read at an interval of 5ms and stored from N register 100 to N register 149.*
5	*G213.A1*	*;Go to the normal positioning mode*
6	*END*	

1.16.1.4 G182 – A-D values in synch with counter value (CNT0)

G182	from V5.13

(G182. <Counter no>.<Start value>.<Resolution>.<AD no>.<1.N register>.<Quantity>)

Description:

The A-D values are recorded depending on the counter status of counter 0 and stored in N-registers in the PA-CONTROL.

Command format:

G182.0.N10.N11.1.100.10

G182.0.N10.N11.1.N11.N12

G182.0.N!10.N!11.1.N!11.N!12

Application:

Recording dynamic A-D values when traversing an axis: e.g. torque values when screw-connecting or force values when joining and pressing or envelope of a magnet.

Counter no:	Counter which is monitored, always 0, only available for CNT0,
Initial value:	The counter value from which the A-D values are recorded is saved in this N register. The resolution prefix indicates whether recording is started when the position is greater or less than the indicated value. An N register must be used. The initial value must be between -32768 and 32767 (15 bits).
Resolution:	The resolution, i.e. the difference in counter reading after which the next A-D value is to be obtained, is saved in this N register. The resolution's prefix applies to the start of recording and indicates whether the position is greater or less than the feature. An N register must be used. The resolution must be between -32768 and 32767.
A-D no.:	Indicates the A-D converter number (between 1 and 8) from which the values are obtained.
1st N register:	Indicates the number of the first N register as of which the A-D values are saved. The N register is increased by 1 with each new measurement. It can be stated as a constant or via an N register.
Quantity:	Indicates the maximum number of A-D values which are to be recorded. This number also determines the number of the integer registers needed to store the measurements. It can be stated as a constant or via an N register.

NOTE The G182 command is only available with the PA-CONTROL smart.

The <initial value> and <resolution> parameters are limited to 15 bits (+/-326767).

The setting for the counter input signal should be selected so that the maximum input frequency of counter 0 of 200 000 increments a second is not exceeded.

The number of recorded A-D values is saved in the SN130 system N registers and is available for queries.

Example 1

A movement (rotary or translatory) is undertaken with axis 1. A-D values (one A-D value approx. every 1ms) should be recorded with high repetitive accuracy in synch with this movement

The following hardware setup has been selected to ensure this:

- PA-CONTROL smart with
  - A-D converter 1-way (8-way)
  - CNT0
- LV-servoTEC S2
- Wiring LV-servoTEC S2 output “Encoder emulation“ X11 on counter 0 of PA-CONTROL smart

```
1   N1:=20                  ; Start recording A-D values from counter value
                            20 (greater than or equal to)(see G182
                            command)
2   N2:=10                  ; Obtain the next A-D value when counter
                            reading has increased by 10 (see G182-
                            command)
3   CNT0:=0                 ;Set counter 0 to a defined initial value
4   G182.0.N1.N2.1.10.1000  Initialise G182 command:
                            ; Counter number      : 0
                            ; Initial counter value  : N1:=20
                            ; Counter resolution  : N2:=10
                            ; A-D converter no    : 1
                            ; 1st N register      : 10
                            ; Max. quantity       : 1000
5   A1:=100                 ;Approach position 100 and record A-D values
                            during this movement
6   N3:=SN130               Obtain number of recorded A-D values
7   END
```

1.16.1.5 G183 - Obtain A-D values in synch with counter value (CNT0)

G183	from V5.13

(G183. <Counter no>.<Initial value>.<Resolution>.<A-D no>.<CAN ID>.<Quantity>)

Description:

The AD-values are recorded as a function of the counter value of the counter 0 and immediately sent as a TxPDO via the CAN bus.

Command format:

G182.0.N10.N11.1.33.10

G182.0.N10.N11.1.N23.N12

G182.0.N!10.N!11.1.N!23.N!12

Application:

Recording dynamic A-D values when traversing an axis: e.g. torque values when screw-connecting or force values when joining and pressing or envelope of a magnet.

Counter no: Counter which is monitored, always 0, only available for CNT0,

Initial value: The counter value from which the A-D values are recorded is saved in this N register. The resolution prefix indicates whether recording is started when the position is greater or less than the indicated value. An N register must be used. The initial value must be between -32768 and 32767 (15 bits).

Resolution: The resolution, i.e. the difference in counter reading after which the next A-D value is to be obtained, is saved in this N register. The resolution's prefix applies to the start of recording and indicates whether the position is greater or less than the feature. An N register must be used. The resolution must be between -32768 and 32767.

A-D no.: Indicates the A-D converter number (between 1 of 8) from which the values are obtained.

CAN ID: Indicates the CAN ID over which the recorded A-D measurements are transmitted as TxPDO1.
It can be stated as a constant or via an N-register.

Quantity: Indicates the maximum number of A-D values which are to be recorded. This number also determines the number of integer registers needed to save the measurements. It can be stated as a constant or via an N-register.

NOTE The G183 command is only available with the PA-CONTROL smart.

The <initial value> and <resolution> parameters are limited to 15 bits (+/-326767)

The setting for the counter input signal should be selected so that the maximum input frequency of counter 0 of 200000 increments a second is not exceeded.

At a baud rate of 500KBaud, no more than 2 A-D values should be transferred per millisecond via the CAN bus.

The number of recorded A-D values is saved in the SN130 system N-registers and is available for queries.

Structure of TxPDO1:

The recorded A-D values and the counter status during recording are transferred via the CAN bus in the following form:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Counter reading (signed int, 32 bit)s				A-D value (signed short, 16 bits)	

Example 1

A movement (rotary or translatory) is undertaken with axis 1. A-D values (one A-D value approx. every 1ms) should be recorded with high repetitive accuracy in synch with this movement and transmitted straight away via the CAN bus

The following hardware setup has been selected to ensure this:

- PA-CONTROL Smart with
  - A-D converter 1-way (8-way)
  - CNT0
- LV-servoTEC S2
- Wiring LV-servoTEC S2 output “Encoder emulation“ X11 on counter 0 of PA-CONTROL smart

```
1   N1:=20          ; Start recording A-D values from counter value 20 (greater than or equal to)(see G182 command)
2   N2:=10          ; Obtain the next A-D value when counter reading has increased by 10 (see G182-command)
3   CNT0:=0         ;Set counter 0 to a defined initial value
4   G183.0.N1.N2.1.33.4   Initialise G183 command:
                          ; Counter number       : 0
                          ; Initial counter value : N1:=20
                          ; Counter resolution    : N2:=10
                          ; A-D converter no      : 1
                          ; CAN ID RxPDO1         :33 → 0x221
                          ; Max. quantity         : 10
5   A1:=100         ;Approach position 100 and record A-D values during this movement
6   N3:=SN130       Obtain number of A-D values recorded
7   END
```

1.16.2 Commands of the G19x Group

1.16.2.1 General statements to the commands of the G19x Group

G19x.

Introduction:

The in-/output modules BK5120, BK5150 and LC5100 are equipped with bus terminals depending on desired function.

Complex terminals, which have a processor, are able to exchange bidirectional data with a higher-level control. These terminals are hereafter called ‘intelligent bus terminals’.

Such ‘intelligent bus terminals’ may include:

- analog inputs (0-10V, -10-10V, 0-20mA, 4-20mA)
- analog outputs (0-10V, -10-10V, 0-20mA, 4-20mA)
- serial interface terminals (RS485, RS232, TTY, data exchange terminals)
- counter terminal, encoder-interface, SSI-Interface,

Internally, all intelligent terminals possess a data structure that is identical in terms of their essential characteristics. This data range is organized word-serially and it comprised 64 memory locations. With this structure the main data and parameters of the terminals are adjustable and readable. Additionally, function call-ups with appropriate parameters are possible. Each logical channel of such an intelligent terminal has such a structure (therefore, 4-channel analog terminals have 4 register sets).

With the G19x-commands the following terminals and registers can be accessed on:

- G190: gets the number of inserted bus terminals
- G191: reads the content of the register
- G192: describes a register with a value

NOTE The G19x-command can only be used for CAN-IO-modules of the type:

- BK5120
- BK5150

1.16.2.2 G190 – Get the number of inserted bus terminals

G190

G190.<CAN-Address>.Ni

Description:

This command gets the current number of inserted bus terminals of a CAN-IO-module.

Command form:

G190.17.N8000

Application:

If you want to access to the registers of a bus terminal with the G191 or G192 commands, use these commands to check whether the terminal is plugged in or available.

NOTE The commands can only be applied for CAN-IO-modules of the type:

- BK5120
- BK5150

Example

See examples for G19x-commands on the following pages.

1.16.2.3 G191 – Read the content of a register

G191

G191.<CAN-Address>.<Terminal-No>.<Channel-No>.<Register-No>.Ni

Description:

The G191 command gets the content of a register and stores it into the N-register “Ni”.

Command form:

G191.17.6.0.8.N8001

Application:

Get the terminal type (R8) in order to check, if the required terminal type can be accessed on.

Get the „feature-register” (R32), in which the settings of the operation mode of the terminal are stored, in order to change the settings if necessary.

Parameter:

CAN-Address:	CAN-Address of the CAN-IO-modules [17…32]
Terminal-No.:	Number of the K-Bus-Terminal, which can be find on the left of the bus node [1…64]
Channel-No.:	Number from the data channel of the terminal for example: KL3064 0 → AD1 1 → AD2
Register-No.:	Register of the terminal for example: KL3064 R8 → Terminal type R32 → Feature register

NOTE If the information for “terminal-no.”, “channel-no.” or “register-no.” are outside of the defined range, an error “Exxx” (E313, E330, E314,…) is generated.

More detailed information on the channel- and register number can be found in the technical documentation of the Beckhoff Automation comp.

Example

See examples for G19x-commands on the following pages.

1.16.2.4 G192 – Describes a register with a value

G192

G192.<CAN-Adress>.<Terminal-No>.<Channel-No>.<Register-No>.Ni/Constant

Description:

The G192 command writes the content of an N-register or a constant value into the register of the terminal.

Command form:

G192.17.6.0.8.N8003

G192.17.6.0.8.16#1106

Application:

Activate or deactivate the function „Overrange Protection“ via changing the content of the feature-register on a KL3064 terminal (AD converter).

Parameter:

CAN-Address:	CAN-Address of CAN-IO-Modules [17…32]
Terminal-No:	Number of the K-Bus-Terminal, which can be find on the left of the bus node [1…64]
Channel-No:	Number of the data channel on the terminal e.g. KL3064 0 → AD1 1 → AD2
Register-No:	Register of the terminal e.g. KL3064 R8 → Terminal type R32 → Feature-Register

NOTE If the information for “terminal-no.”, “channel-no.” or “register-no.” are outside of the defined range, an error “Exxx” (E313, E330, E314,…) is generated.

More detailed information on the channel- and register number can be found in the technical documentation of the Beckhoff Automation comp

Example

See examples for G19x-commands on the following pages.

1.16.2.5 Example G19x commands

AD2OverangeNo.PNC

```
; R32 - Feature-Register AD-converter : KL3064, ...
;
; Bit8  : Overrange Protection [1]
;
; ----
; R32 = 16#1106 (default), Overrange Protection --> activated
; R32 = 16#1006 (default), Overrange Protection --> deactivated
;
;--- get number of terminals
G190.17.N8010
M8000:=N8010>6              ; terminal 6 available ?
G21 M8000.0 ERROR           ; No
;
;--- get terminal type of 6th terminal
G191.17.6.1.8.N8011
M8000:=N8011=3064           ; terminal KL3064 ?
G21 M8000.0 ERROR           ; No
;
;--- read R32-Feature-Register, AD1=channel 0
G191.17.6.1.32.N8012
;
N8013:=N8012&16#0100        ; what is its status of Bit8
M8000:=N8013>0              ; Bit8 is set,
                            ; Overrange Protection activated
G21 M8000.0 OK_isnotset
;
; resetting Bit8 "Overrange Protection"
N8013:=N8012&16#FEFF        ;
;
;--- write R32-Feature-Register, AD2=channel 1
G192.17.6.1.32.N8013
;
;--- checking-reading R32-Feature-Register, AD2=channel 1
G191.17.6.1.32.N8014
JMP END
;
$ERROR
I10.1     ; ERROR
$OK_isnotset
$END
END
```

1.16.3 Commands of the G2xx group

1.16.3.1 Basic explanations of G2xx commands group

G2xx

Introduction:

PA-CONTROL has two modes for processing positioning commands.

Standard positioning mode:	The positioning procedure is started and the process waits until the axis has reached its target position. The next command in the program is not processed until then. If several positioning commands are defined consecutively in one line, they are processed in parallel.
Start positioning mode:	The positioning procedure is started in order to continue with the next command in the program. The positioning procedure is controlled in the background.

The standard positioning mode is the default mode.

The switch-over and behaviour are selected according to the axis.

If an axis is in the start positioning mode, it can be accessed through various commands of the G200 group. Before re-starting the axis, a check should be made via one of the commands (G213, G211, G221, etc.) to ensure that it is not traversing.

1.16.3.2 G210 - Activate the start positioning mode

G210

G210.(Axis)

Description:

PA-CONTROL has two operating modes. In the standard positioning mode, *see chapter 1.16.3.5, page 234,* the controller waits for the movement to end after an axis start, before starting the next command. In the start positioning mode, the traverse of a second and further axes can be initiated immediately after the start of an axis. The axes traverse in the background.

The G210 command activates the start positioning mode and deactivates the standard positioning mode. (see this section)

When programming, always remember that the movement of one axis must be terminated before a further start. An error results if this pre-requisite is disregarded. When the start positioning mode is used, a check should be made to determine whether the axis can be started at this time.

The G213 command is used to change over to the standard positioning mode.

Command format:

G210.A0

G210.A1

Application:

The positioning of an axis is started. The controller continues to process the programme without waiting for the axis to arrive at the stop position.

Example

See the following commands

1.16.3.3 G211 - Position-conditional jump

G211

G211.(Axis).(Condition.) (Mask)

Please note:

The command only evaluates whether the axis/axes are in position or not in position.

Command format:

G211.A0.j Label

G211.An.j Label

Application:

Execution of a jump, depending on whether the selected axis or all axes are traversing or are at standstill. A stands for the axis number, j for the condition.

The following applies:

- A0 → All axes
- A1 → Axis 1
- A2 → Axis 2
- An → Axis n

- j = 0 → jump if the position of the selected axes has not yet been reached
- j = 1 → jump if the position of the selected axes has been reached

Example 1

```
1   G25.A1 G25.A2          ;Approach to reference points of the A1 and A2 axis
2   G90.A0                 ;Absolute dimension system
3   G210.A0                ;Activate the start positioning mode for all axes
4   A1:=10000 A2:=6000     ;Start axes
5   $SET
6   O1:=1 T50 O1:=0        ;Set output 1 for 500ms
7   T50                    ;Wait 500ms
8   G211.A0.0 SET          ;Jump to the label "SET" if all axes have not yet reached their position
9   G213.A0                ;Wait until all axes are in position and activate the standard positioning mode (deactivate the start positioning mode)
10  O2:=1
11  END
```

Example 2

```
1     G25.A1 G25.A2          ;Approach to reference points of the A1 and A2 axis
2     G90.A0                 ;Absolute dimension system
3     G210.A0                ;Activate the start positioning mode for all axes
4     A1:=10000 A2:=6000     ;Start axes
5     $LOOP
6     O1:=1 T50 O1:=0        ;Set output 1 for 500ms
7     T50                    ;Wait 500ms
8     G211.A2.1 READY        ;Jump to "READY" if the A2 axis has reached its position
9     JMP LOOP               ;Jump to "LOOP"
10    $READY
11    G213.A0                ;Wait until all axes are in position and activate the standard positioning mode (deactivate the start positioning mode)
12    O2:=1
13    END
```

Example 3

```
1     G25.A1 G25.A2          ;Approach to reference points of the A1 and A2 axis
2     G90.A0                 ;Absolute dimension system
3     G210.A0                ;Activate the start positioning mode for all axes
4     A1:=10000 A2:=6000     ;Start axes
5       $LOOP
6       O1:=1 T50 O1:=0      ;Set output 1 for 500ms
7       T50                  ;Wait 500ms
8       G211.A1.0 LOOP       ;Jump to the label "LOOP" if the A1 axis has not yet reached its position
9     G213.A0                ;Wait until all axes are in position and activate the standard positioning mode (deactivate the start positioning mode)
10    O2:=1
11    END
```

1.16.3.4 G212 - Position-conditioned subroutine call

G212

G212.(Axis).(Condition) (Program)

NOTE The command only evaluates whether the axis/axes are in position or not in position.

Command format:

G212.A0.j Name

G212.An.j Name

Application:

Execution of a subroutine call, depending on whether the selected axis or all axes are traversing or are at standstill. **A** stands for the axis name, **j** for the condition.

The following applies:

- A0 → All axes
- A1 → Axis 1
- A2 → Axis 2
- An → Axis n

- j = 0 → jump to subroutine if the position of the selected axes has not yet been reached.
- j = 1 → jump to subroutine if the position of the selected axes has been reached.

Example

```
1   G25.A1 G25.A2          ;Approach to reference points of the A1 and A2 axis
2   G210.A0                ;Activate the start positioning mode for all axes
3   A1:=100 A2:=500        ;Start axes
4   $LOOP                  ;Jump label
5   G212.A0.0 BLINKING     ;Jump to subroutine BLINKING if all axes have not yet reached their position
6   G211.A1.0 LOOP         ;Jump to the label "LOOP" if the A1 axis has not yet reached its position
7   G213.A0                ;Wait until all axes are in position and activate the standard positioning mode (deactivate the start positioning mode)
8   END
```

Program: BLINKING.PNC

```
1   O9:=1                  ;Set output 1
2   T50                    ;Wait 500ms
3   O9:=0                  ;Reset output 1
4   T20                    ;Wait 200ms
5   END
```

1.16.3.5 G213 - Activate the standard positioning mode

G213

G213.(Axis)

Description:

The G213 command activates the standard positioning mode and deactivates the start positioning mode. The controller waits until all current positioning processes for all axes (G213.A0) have been concluded or until the current positioning process for a defined axis (G213.Ai) has been concluded before starting the next command.

Command format:

G213.A0

G213.An

Example

See other G200 commands

1.16.3.6 G221 - Position-conditional jump (current position)

G221

G221.(Condition).(Axis).(Position) (Label)

Command format:

G221.j.An.Pos Label

G221.j.ARn.Pos Label

Application:

A conditional jump depending on a current axis position can be executed by means of the command **G221**. **j** stands for the jump condition, **An** for the axis and the position and **label** for the target to be jumped to.

The following applies:

j = 0	→	jump if the current axis position is less than the position (An)
j = 1	→	jump if the current axis position is greater than the position (An)

NOTE This command evaluates the absolute position of the stated axis irrespective of the selected measuring system (G90.An / G91.An).

Example 1

```
1    G25.A1 G25.A2          ;Approach to reference points of the A1 and A2 axis
2    G90.A0                 ;Absolute dimension system
3    G210.A0                ;Activate the start positioning mode for all axes
4    A1:=10000 A2:=6000     ;Start axes
5    $LOOP
6    O1:=1 T50 O1:=0 T50
7    G221.1.A1.8000 RED     ;Jump to "RED" if the A1 position is greater than 8000
8    O2:=1 T10 O2:=0
9    $ROT
10   G211.A0.0 LOOP         ;Jump to "LOOP" if all axes are not yet in position
11   G213.A0                ;Wait until all axes are in position and activate the standard positioning mode (deactivate the start positioning mode)
12   END
```

Example 2

```
1    G25.A1 G25.A2          ;Approach to reference points of the A1 and A2 axis
2    G90.A0                 ;Absolute dimension system
3    G210.A0                ;Activate the start positioning mode for all axes
4    A1:=10000 A2:=6000     ;Start axes
5      $LOOP
6      O1:=1 T50 O1:=0 T50
7      G221.1.A2.2000 RED   ;Jump to "RED" if the A2 position is greater than 2000
8      O2:=1 T10 O2:=0
9      $RED
10     G211.A0.0 LOOP       ;Jump to "LOOP" if all axes are not yet in position
11   G213.A0                ;Wait until all axes are in position and activate the standard positioning mode (deactivate the start positioning mode)
12   END
```

Example 3

```
1    G25.A1 G25.A2          ;Approach to reference points of the A1 and A2 axis
2    G90.A0                 ;Absolute dimension system
3    G210.A0                ;Activate the start positioning mode for all axes
4    A1:=10000 A2:=6000     ;Start axes
5    $LOOP
6    O1:=1 T50 O1:=0 T50
7    G221.0.A1.8000 RED     ;Jump to "RED" if the A1 position is greater than 8000
8    O2:=1 T10 O2:=0
9    $RED
10   G211.A0.0 LOOP         ;Jump to "LOOP" if all axes are not yet in position
11   G213.A0                ;Wait until all axes are in position and activate the standard positioning mode (deactivate the start positioning mode)
12   END
```

1.16.3.7 G222 - Position-conditional subroutine call (current pos.)

G222

G222.(Condition) (Axis).(Position) (Program)

NOTE This command evaluates the absolute position of the stated axis irrespective of the selected measuring system (G90.An / G91.An).

Command format:

G222.j.An.Pos Name

Application:

A conditional subroutine call depending on a current axis position can be executed by means of the command **G222**.

j stands for the jump condition, **An** for the position of the axis and **Name** for the subroutine which is to be called.

The following applies:

j = 0	→	jump if the current axis position is less than the position (**An**)
j = 1	→	jump if the current axis position is greater than the position (**An**)

Example 1

```
1    G25.A1 G25.A2          ;Approach to reference points of the A1 and A2 axis
2    G90.A0                 ;Absolute dimension system
3    G210.A0                ;Activates the start positioning mode for all axes
4    A1:=10000 A2:=6000     ;Start axes
5    $LOOP
6    O1:=1 T50 O1:=0 T50
7    G222.1.A1.8000 RED     ;Call of the subroutine "RED" if the A1 position is greater than 8000
8    G211.A0.0 LOOP         ;Jump to "LOOP" if all axes are not in position yet
9    G213.A0                ;Wait until all axes are in position and activate the standard positioning mode (deactivate the start positioning mode)
10   END
```

Program: RED

```
1    O2:=1 T10 O2:=0
2    END
```

Example 2

```
1    G25.A1 G25.A2          ;Approach to reference points of the A1 and A2 axis
2    G90.A0                 ;Absolute dimension system
3    G210.A0                ;Activates the start positioning mode for all axes
4    A1:=10000 A2:=6000     ;Start axes
5    $LOOP
6    O1:=1 T50 O1:=0 T50
7    G222.1.A2.1000 RED     ;Call of the subroutine "RED" if the A2:= position is greater than 1000
8    G211.A0.0 $LOOP        ;Jump to "LOOP" if all axes are not yet in position
9    G213.A0                ;Wait until all axes are in position and activate the standard positioning mode (deactivate the start positioning mode)
10   END
```

Program: RED

```
1    O2:=1 T10 O2:=0
2    END
```

Example 3

```
1    G25.A1 G25.A2          ;Approach to reference points of the A1 and A2 axis
2    G90.A0                 ;Absolute dimension system
3    G210.A0                ;Activates the start positioning mode for all axes
4    A1:=10000 A2:=6000     ;Start axes
5    $LOOP
6    O1:=1 T50 O1:=0 T50
7    G222.0.A1.4000 RED     ;Call of the subroutine "RED" if the A1 position is less than 4000
8    G211.A0.0 $LOOP        ;Jump to "LOOP" if all axes are not yet in position
9    G213.A0                ;Wait until all axes are in position and activate the standard positioning mode (deactivate the start positioning mode)
10   END
```

Program: RED

```
1    O2:=1 T10 O2:=0
2    END
```

1.16.3.8 G230 - Wait until current position </> than value

G230

G230.(Condition).(Axis).(Position)

NOTE This command evaluates the absolute position of the stated axis irrespective of the selected measuring system (G90.An / G91.An).

Command format:

G230.j.An.Pos

G230.j.An.R!n
G230.j.An.Konstante

G230.j.An.Rn

G230.j.An.R!

Application:

The further processing of the program can be interrupted until a certain position has been reached by means of the **G230** command.

j stands for the condition, **An** for the axis and the position.

The following applies:

j = 0	→	wait until the position is less than the current position
j = 1	→	wait until the position is greater than the current position

Example

```
1   G25.A1                  ;Approach to reference point of the A1 axis
2   G90.A0                  ;Absolute dimension system
3   R10:=3000
4   G210.A0                 ;Activate the start positioning mode for all axes
5   A1:=10000               ;Start the axis
6   O1:=1 T10 O1:=0
7   G230.1.A1.3000 O2:=1    ;Output 2 is set from A1:=3000 to A1:=5000
8   G230.1.A1.5000 O2:=0
9   O1:=1 T10 O1:=0
10  G213.A0
11  G210.A0                 ;Process commands beyond line limit; beginning
12  A1:=0
13  O1:=1
14  R5:=10
15  G230.0.A1.R!5 O1:=0     ;Output 1 is set from A1:=10000 to A1:=3000
                            (value 3000 is in R10)
16  G213.A0                 ;Wait until all axes are in position and activate the
                            standard positioning mode (deactivate the start
                            positioning mode)
17  END
```

1.16.3.9 G231 - Wait until the residual distance of current positioning is reached or undershot

G231

G231.(Axis).(Residual distance)

Command format:

G231.An.Ri

G231.An.R!i

G231.An.Constant

Description:

With this command, the system waits to complete the process until the residual axis distance still to be traversed (positioning active) is less than the comparison residual distance.

NOTE This command evaluates the absolute position regardless of the measurement system selected (G90 or G91).

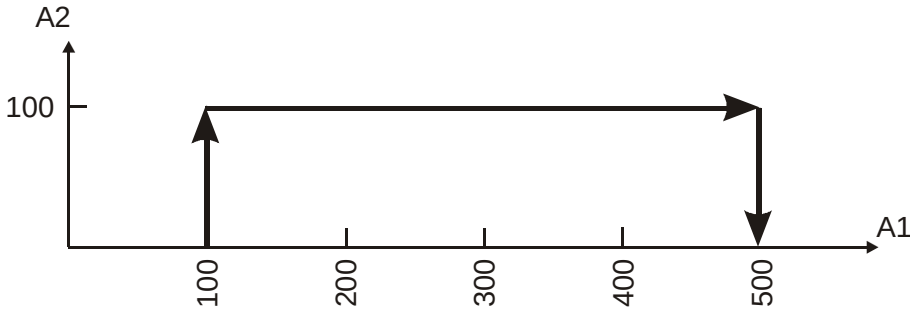
Application:

Command **G231** can be used to process the rest of the program while the residual distance is being traversed.

NOTE If the axis has already reached its target position, i.e. is in position, or if positioning is not active (not yet active), the system does not wait. The residual distance is stated as an amount, i.e. without prefix. Negative comparison values are meaningless. The control stops in such cases (E532 – value outside range).

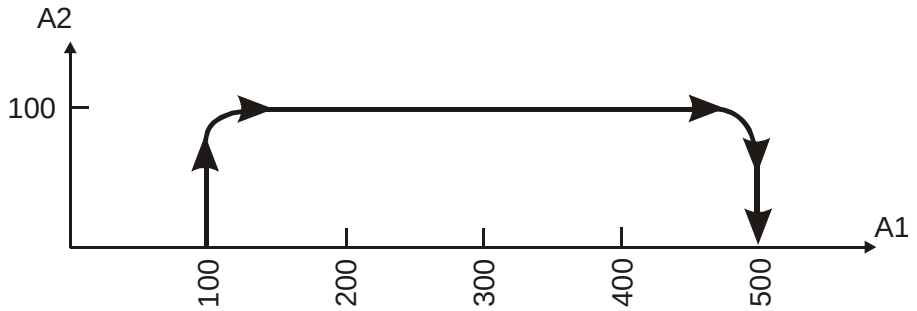
Examples

without G231



1	*G25.A1 G25.A2*	***G231.(Axis).(Residual distance)***
2	*G90.A0*	
3	*A2:=100*	**Command format:**
4	*A1:=500*	G231.An.Ri
5	*A2:=0*	G231.An.R!i
6	*END*	G231.An.Constant

with G231



1	*G25.A1 G25.A2*	*;A1 / A2 axis approaching reference point*
2	*G90.A0*	*;Absolute measuring system*
3	*G210.A0*	*;Activate start positioning mode for all axes,*
4	*A2:=100*	*;Start axis 2 and move to absolute position 100 mm*
5	*G231.A2.20*	*;Wait until the residual distance of axis 2 is less than 20 mm*
6	*A1:=500*	*;Start axis 1 while axis 2 is still moving*
7	*G231.A1.20*	*;Check residual distance of axis 1 until it is less than 20 mm*
8	*A2:=0*	*;Start axis 2 when residual distance of axis 1 is less than 20 mm*
9	*G213.A0*	*;Wait until all axes are in position and activate normal positioning mode (deactivate start positioning mode)*
10	*END*	

1.16.4 Commands of the G4xx-Group

1.16.4.1 G4xx - Time monitoring commands

G4xx

Introduction

Time monitoring commands of the PA-CONTROL Family

The commands described below are used if the defined end of process has to be monitored and utility programs for troubleshooting or error registration have to be started in the case of irregularities. Such situations can occur when processes which run in parallel to and independently of PA-CONTROL programs, e.g. the activation of pneumatic cylinders, are started by the controller.

After the time has elapsed, a conditional jump or subroutine call can be executed depending on the specified command. The set time conditions can be reset or cancelled. The time condition is checked and also reacted to before PA-CONTROL executes a command.

G421.1.

G421.1.(Time) (Label)

Description

The monitoring time is activated by the command **G421.1.200** label. The monitoring is reset if the reset command G401.1 is detected before the monitoring time has elapsed. Otherwise the program is continued at the specified label.

A **G421** command running in the program is cancelled by another **G421** or **G422**, and a new time monitoring command is started at the same time. The remaining time of the first monitoring job is not processed any further.

NOTE After a monitoring time has been started, a subroutine call should not be executed, since the branch target for the G421 command is no longer available.

Command format:

G421.1.nnn Label

G421.1.Ni Label

NOTE The maximum transfer time is:
with specification via constant 65535 x 10ms
with specification via N-register 2^{31} x 10ms

Example

```
1   $BEGINNING
2   O1:=1                    ;Set output 1
3   G421.1.200 ERROR_1       ;Jump to label "ERROR_1" if 2 seconds have
                             elapsed and the time condition has not been
                             reset
4   I1.1 O1:=0               ;Wait for input 1 = "1" and reset output 1
5   G401.1                   ;Reset the time condition
6   END
7   $ERROR_1
8   O2:=1                    ;Set malfunction signal
9   I2.1                     ;Wait for error acknowledgement signal
10  O1:=0 O2:=0              ;Reset output 1 and malfunction signal
11  JMP  BEGINNING
```

1.16.4.3 G422 - Time monitoring with conditional subroutine call

G422.1.

G422.1.(Time) (File)

Description

The monitoring time is started by the command **G422.1.200** Name. The monitoring is reset if the reset command G401.1 is detected before the monitoring time of 2 seconds (200*10ms) has elapsed.

Otherwise a jump to the specified subroutine is executed.

NOTE The return from the time monitoring subroutine (in the example ERROR) follows the PAC command which was interrupted by the time monitoring.

Command format:

G422.1.nnn Name

G422.1.Ni Label

NOTE The maximum transfer time is:

with specification via constant	65535	x 10ms
with specification via N register	2^{31}	x 10ms

Example

```
1   O1:=1                ;Set output 1
2   G422.1.200 ERROR     ;Subroutine call if 2 seconds have elapsed
                         and the time condition has not been reset
3   I1.1 O1:=0           ;Wait for input 1 = "1" and reset output 1
4   G401.1
5   END
```

Program: ERROR

```
1   O3:=1                ;Malfunction at ...
2   I5.1                 ;Acknowledgement by the operator
3   I5.0                 ;
4   END
```

1.16.4.4 G423 - with conditional subroutine call (return in same line)

G423.1.

G423.1.(Time) (File)

Description

The monitoring time is started by the command **G423.1.200** Name. The monitoring is reset if the reset command G401.1 is detected before the monitoring time of 2 seconds (200*10ms) has elapsed.

Otherwise a jump to the specified subroutine is executed.

NOTE After returning from the subroutine, the program is continued in the same line in which the G423 command is located. The program in which the G423 command was activated must still be active, i.e. termination may not be executed at another point by the CANCEL command.

Command format

G423.1.nnn Name

G423.1.Ni Label

NOTE The maximum transfer time is:
with specification via constant 65535 x 10ms
with specification via N register 2^{31} x 10ms

Example

```
1   O1:=1                 ;Set output 1
2   G423.1.200 ERROR      ;Subroutine call if 2 seconds have elapsed
                          and the time condition has not been reset.
                          After returning from the subroutine, the
                          program is continued in the same line in which
                          the G423 command is located. The time
                          monitoring is run through once again, thereby
                          ensuring that input 1 really has the logical
                          state 1. The program in which the G423
                          command was activated must still be active
3   I1.1 O1:=0            ;Wait for input 1 = "1" and reset output 1
4   G401.1
5   END
```

Program: ERROR

```
1   O3:=1                 ;Malfunction at ...
2   I5.1                  ;Acknowledgement by the operator
3   END
```

G401.1

Description

The time monitoring which was started by the commands G421, G422 or G423 is stopped by means of this command.

Time monitoring is therefore no longer active.

Command format

G401.1

Example

See examples of the command G421 on *chapter 1.16.4.2, page 243* or command G422 on *chapter 1.16.4.3, page 244* or command G423 on *chapter 1.16.4.4, page 245.*

1.16.5 Commands of the G5xx-Group

1.16.5.1 G5xx - Text and value output via the current data channel

G5xx

Introduction:

Texts, register contents, statuses of inputs, outputs and flags can be output via the current data channel during the program run (automatic mode) by means of this command group.

The following data channels can be used:

Data channel 0	:	LC display on the IEF console of PA-CONTROL
Data channel 1	:	1st serial port of PA-CONTROL (COM 1)
Data channel 2	:	2nd serial port of PA-CONTROL (COM 2)
Data channel 3	:	3rd serial port of PA-CONTROL (COM 3)
Data channel 4	:	4th serial port of PA-CONTROL (COM 4)

NOTE The information about the current data channel is assigned to each task (to each parallel run) and must be initialised before using the commands.

If a serial port has already been initialised and communication is active, switch-over is only possible on specification of the data channel number (e.g. G500.1). Receiving and transmitting buffers remain unchanged.

1.16.5.2 G500 - Selection of the data channel / Initialisation of the ports

G500.

G500.(Channel number).(Baud rate).(Data format).(Handshake)

Description:

The current data channel can be selected by means of the "G500." command. Serial ports can also be initialised (baud rate, number of data bits, parity,...).

If the serial port is selected with initialisation (e.g. G500.1.6.1.0), the receive buffer of the serial port is also deleted. If, on the other hand, the serial port is selected without initialisation (e.g. G500.1), the receive buffer of the serial port remains unchanged.

See the list for the meaning of the transfer parameters.

NOTE Outputs to the LCD display of PA-CONTROL are only practical without CR & LF.

If the G500 command is executed after a G533.n command, the end criterion is deleted!

Command format:

G500.No.Baud rate.Data format.Handshake

G500.No

G500.0

Example

See examples of the following commands.

Meaning of the transfer parameters:

No.:		
0	→	LC display on the PAC front plate is active
1	→	Port 1 is activated (COM 1)
2	→	Port 2 is activated (COM 2)
3	→	...

Baud rate:		
1	→	110 baud
2	→	300 baud
3	→	1200 baud
4	→	2400 baud
5	→	4800 baud
6	→	9600 baud
7	→	19200 baud
8	→	38400 baud (currently not available)

Data format:		
1	→	8 data bits, 1 stop bit, no parity
2	→	7 data bits, 1 stop bit, no parity
3	→	7 data bits, 1 stop bit, even parity
4	→	7 data bits, 1 stop bit, odd parity
5	→	8 data bits, 1 stop bit, even parity (from V4.49)
6	→	8 data bits, 1 stop bit, odd parity (from V4.49)
7	→	8 data bits, 2 stop bits, no parity (from V4.55 K)
8	→	7 data bits, 2 stop bits, no parity (from V4.55 K)
9	→	7 data bits, 2 stop bits, even parity (from V4.55 K)
10	→	7 data bits, 2 stop bits, odd parity (from V4.55 K)
11	→	8 data bits, 2 stop bits, even parity (from V4.55 K)
12	→	8 data bits, 2 stop bits, odd parity (from V4.55 K)
Handshake:		
0	→	Hardware handshake (CTS) inactive
1	→	Hardware handshake (CTS) active

G501

Description:

The G501 command clears the display of the current display medium and places the cursor in the top left corner (1st column, 1st line).

NOTE Before using the G501 command, switch over to the current display medium with the "G500." command.

This command is only meaningful for outputs to the display!

Command format:

G501

Example

```
1   G11.0          ;Switch off run displays
2   G500.0         ;Switches the G500 commands over to the
                    current display medium (LC display or serial
                    port in the case of "Simulation of the front
                    plate")
3   G501           ;Clears the display and places the cursor in
                    the top left corner (1st column, 1st line)
4   G510.Hello     ;Outputs the text "Hello" to the current display
                    medium
5   END
```

1.16.5.4 G502 - Clear up to end of line

G502

Description:

The G502 command clears the display on the current display medium from the current cursor position to the end of the line. The cursor position remains unchanged.

The command can be used to delete the value previously displayed during the output of register values, in order to output the new register value.

NOTE Before using the G502 command, switch over to the current display medium with the "G500." command.

This command is only appropriate for outputs to the display!

Command format:

G502

Example

```
1   G11.0          ;Switch off run displays
2   G500.0         ;Switches the G500 commands over to the
                    current display medium (LC display or serial
                    port in the case of "Simulation of the front
                    plate")
3   G501           ;Clears the display and places the cursor in
                    the top left corner (1st  column, 1st line)
4   N1:=234
5   $LOOP
6   G503.1.2       ;Place cursor in the 1st column of the 2nd line
7   G510.VALUE N1
8   G503.11.2      ;Place cursor in the 11th  column of the 2nd
                    line
9   G502           ;Clear from the cursor position to the end of
                    the line
10  G520.N1        ;Output the value of N1 to the display (left-
                    justified)
11  T100
12  DEC.N1 LOOP
13  END
```

G503 / 504

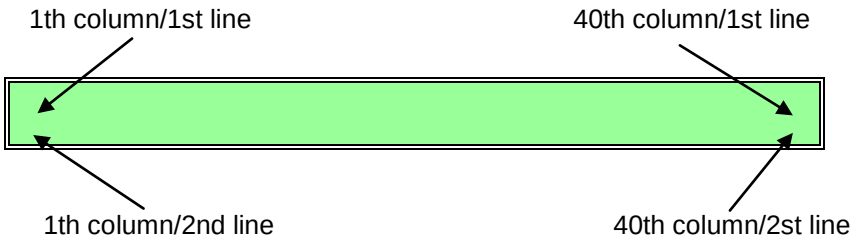
G50X.(Column).(Line)

Description:

The G503 command is used to position the cursor in the display, so that characters are put out to the correct position in the display.

The difference from the G503 command is that the blinking cursor is fixed blinking at the specified position in the display for a keyboard input. The cursor is deactivated by input of the first character or by the G504 command with the parameters 0 (G504.0.0).

The LC display is divided into 40 columns and two lines.



NOTE Before using the G503 command, switch over to the current display medium with the "G500." command.

This command is only appropriate for outputs to the display!

Command format:

G503.n.m
G503.Nn.m
G503.n.Nm
G503.Nn.Nm

G504.n.m
G504.Nn.m
G504.n.Nm
G504.Nn.Nm

Example

```
1    G11.0                ;Switch off running displays
2    G500.0               ;Switches the G500 commands over to the
                          current display medium (LC display or serial
                          port in the case of "Simulation of the front
                          plate")
3    N23:=5
4    N35:=2
5    G501                 ;Clears the display and places the cursor in
                          the top left corner (1st column, 1st line)
6    G503.4.1             ;Places the cursor in the 4th column of the 1st
                          line
7    G510.HELLO           ; puts out the text "HELLO" to the current
                          cursor position on the current display medium
8    G503.N23.N35         ;Places the cursor in the 5th column of the
                          2nd line, see values of N23 and N35
9    G504.N23.N35
10   G510.OPERATOR        ;puts out the text "OPERATOR" to the current
                          cursor position on the current display medium
11   END
```

1.16.5.6 G510 - Text output

G510.

G510.(Text)

Description:

G510. sends the text via the selected serial port or to the display.

NOTE No CR or LF is sent at the end of the text. No further command may be in this line after G510.

The text begins after "G510" and ends with the end of line
(Conclude entry with ENTER in the program editor).

Command format:

G510.Text

Example

```
1    G500.0               ;Selects the LC display of PA-CONTROL
2    G501                 ;Clears the display and places the cursor in the 1st
                          column, 1st line
3    G510.Machine Error!  ;Puts out the text "Machine Error" to the current
                          display medium, the LC display
3    END
```

G511.

G511.(Text)

Description:

G511. Puts out texts via the selected serial port and sends a CR and LF after the text and is therefore not appropriate for output to the PA-CONTROL display.

NOTE The transfer is ended with CR and LF. No further command may be in this line after G511.

The text begins after "G511" and ends with the end of line (Conclude entry with ENTER in the program editor).

Command format:

G511.Text

Example

```
1   G500.1.6.1.0          ;Initialisation of the current port (COM1)
2   G511.Machine Error!   ;puts out the text "Machine Error!" via the serial
                          port COM1 and concludes the transfer with CR and
                          LF
3   END
```

1.16.5.8 G512 - Output of control characters

G512.

G512.(0-255)

Description:

G512. interprets n as the nth character (ordinal number) in the ASCII table and puts out the corresponding character via the serial port. The corresponding character can also be selected by the content of an integer register or by the indirect addressing of an integer register.

All numbers from 0 to 255 are permissible.

NOTE No CR or LF is sent after the character.

Command format:

G512.n

G512.Ni

G512.N!i

Application:

Output of control characters to a printer. Please refer to the ASCII table of your printer for the meaning of the individual characters.

Example

```
1   G500.2.6.1.0   ;Initialisation of the current port (COM2)
2   G512.7F        ;Puts out the command „DEL" via the serial port
3   G512.36        ;Puts out the symbol "$" via the serial port
4   G512.4         ;Puts out 04Hex (EOT) via the serial port
5   END
```

G515.

G515.(Line).(PTX File)

Description:

G515. Puts out a text, the contents of the line **i** of a program file (*.PTX) to the current data channel. This command is suitable for using stored messages or user tips in different program sections.

NOTE The file must be a *.PTX type file and available during the runtime of the program. If the respective text line cannot be found (number is greater than the number of lines in the file), the last program line is specified.

Command format:

G515.i.Name

G515.Ni.Name

Example

Program: MESSAGE.PTX

```
1   Machine ready
2   Malfunction part feed
3   Part magazine empty
4   Text not defined
5   END
```

Program: EXAMPLE.PNC

```
1   G500.0          ;Initialization, LC display is current data channel
2   G501            ;Clears the display and places the cursor in the 1st
                     position of the 1st line
3   G515.2.MESSAGE  ;puts out the 2nd line of the program
                     "MESSAGE.PTX" to the current data channel
4   N5:=3
5   G503.1.2        ;Places the cursor in the 1st position of the 2nd line
6   G515.N5.MESSAGE ;puts out the 3rd line of the program "MESSAGE.PTX"
                     to the current data channel
6   END
```

G520

G520.(Operand).(Field width).(Decimal places)

Description:

G520. determines the status of the operand (register, input, output, flag) and puts out the corresponding value via the current serial port or to the display.

NOTE The output can be formatted or unformatted.

Command format unformatted:

G520.Rn
G520.Nn
G520.In
G520.On
G520.Mn
G520.Sn

Command format formatted (with leading blank characters)**:**

G520.Nn.m
G520.Rn.m.i
G520.Rn.m.0

Limits for the formatting

Formatted : (G520.Nn.m)

Field width max. = 20

Formatted: (G520.Rn.m.i)

Decimal places min. = 0 max. = 6

Field width max. = 20

NOTE No CR or LF is sent in the output.

Output values for inputs, outputs and flags

Operand		Status		Output (ASCII characters)
Input		hot		1
Input		cold		0
Output		set		1
Output		reset		0
Flag		set		1
Flag		reset		0

Example

```
1   O10:=1              Set output 10
2   O20:=0              Reset output 20
3   M25:=1              Set flag 25
4   M35:=0              Reset flag 35
5   R2:=546.067         Load the number 546.06 to real number
                        register R2
6   N3:= 34             Load the number 34 to integer register N3
7   G500.2.6.1.0        ;Initialisation of the current port (COM2)
8   G520.I7             ;Puts out the status of input 7 via the current
                        data channel (0 or 1, depending on the status)
9   G520.O10            ;Puts out the status of output 10 via the
                        current data channel (output 1)
10  G520.O20            ;Puts out the status of output 20 via the
                        current data channel (output 0)
11  G520.M25            ;Puts out the status of flag 25 via the current
                        data channel (output 1)
12  G520.M35            ;Puts out the status of flag 35 via the current
                        data channel (output 0)
13  G520.N3.10          ;Puts out the content of N3, formatted, right-
                        justified, (10 places with eight leading
                        characters) via the current data channel
14  G520.R2             ;Puts out the content of R2, left-justified
                        unformatted, via the current data channel
15  G520.R2.10.2        ;Puts out the content of R2, right-justified and
                        formatted, via the current data channel (field
                        10 characters wide with two decimal places)
16  G520.R2.10.0        ;Puts out the content of R2 right-justified and
                        formatted via the current data channel (field
                        10 characters wide with no decimal places)
17  END
```

G521.

G521.(Operand).(Field width).(Decimal places)

Description:

G521. determines the status of the operand (register, input, output, flag) and puts out the corresponding value via the current serial port or to the display.

NOTE The output can be formatted or unformatted.

Command format unformatted:

G521.Rn
G521.Nn
G521.In
G521.On
G521.Mn
G521.Sn

Command format formatted (with leading blank characters)**:**

G521.Nn.m
G521.Rn.m.i
G521.Rn.m.0

Limits for the formatting

Formatted : (G521.Nn.m)

Field width max. = 20

Formatted: (G521.Rn.m.i)

Decimal places min. = 0 max. = 6

Field width max. = 20

NOTE CR and LF is sent in addition to the actual value.

Output values for inputs, outputs and flags

Operand		Status		Output (ASCII characters)
Input		hot		1
Input		cold		0
Output		set		1
Output		reset		0
Flag		set		1
Flag		reset		0

Example

```
1   O10:=1                  Set output 10
2   O20:=0                  Reset output 20
3   M25:=1                  Set flag 25
4   M35:=0                  Reset flag 35
5   R2:=546.067             Load the number 546.06 to real number
                            register R2
6   N3:= 34                 Load the number 34 to integer register N3
7   G500.2.6.1.0            ;Initialisation of the current port (COM2)
8   G521.I7                 ;Puts out the status of input 7 via the current
                            data channel (0 or 1, depending on the status
                            as well as CR and LF)
9   G521.O10                ;Puts out the status of output 10 via the
                            current data channel (output 1 as well as CR
                            and LF)
10  G521.O20                ;Puts out the status of output 20 via the
                            current data channel (output 0 as well as CR
                            and LF)
11  G521.M25                ;Puts out the status of flag 25 via the current
                            data channel (output 1 as well as CR and LF)
12  G521.M35                ;Puts out the status of flag 35 via the current
                            data channel (output as well as CR and LF)
13  G521.N3.10              ;Puts out the content of N3 right-justified and
                            formatted, (10 places with eight leading
                            characters as well as CR und LF ) via the
                            current data channel
14  G521.R2                 ;Puts out the content of R2 as well as CR and
                            LF left-justified and unformatted via the
                            current data channel
15  G521.R2.10.2            ;Puts out the content of R2 right-justified and
                            formatted via the current data channel (field
                            10 characters wide with two decimal places as
                            well as CR and LF)
16  G521.R2.10.0            ;Puts out the content of R2 right-justified and
                            formatted via the current data channel (field
                            10 characters wide with no decimal places as
                            well as CR and LF)
17  END
```

G531.

G531.(Operand) (Label)

Description:

G531. waits for characters to be received at the current serial port. The reception is considered to be terminated with the transmission of CR. The received characters are further processed according to the operand.

Time monitoring is not implemented in the G531 command. If required, the commands G401, G421, G422 and G423 can be used for this purpose.

If a transmission error (parity error,...) or an illegal character string is detected during reception, the program is continued at the label.

Command format:

G531.Rn Label

G531.Nn Label

G531.On Label

G531.Mn Label

G531.Sn Label

Example

```
1   G500.2.6.1.0        ;Initialisation of the current port
2   G531.R2 ERROR       ;Waits for the reception of a message at the
                        current serial port and assigns the information
                        to register R2. If there is a transmission error,
                        the program is continued at the label
                        "ERROR"
3   G531.M3 ERROR       ;Waits for the reception of a message at the
                        current serial port and sets flag 3 according to
                        the received information
4   G531.S2 ERROR       ;Waits for the reception of a message at the
                        current serial port and assigns the information
                        to character buffer S2 (in this case the global
                        character buffer). If there is a transmission
                        error, the program is continued at the label
                        "ERROR"
5   JMP END
6   $ERROR              ;Error message program section
7   .....
8   .....
9   $END
10  END
```

Operand		Status		Output (ASCII characters)
Output		Set output		1
Output		Reset output		0
Flag		Set		1
Flag		Set		1
Flag		Reset		0
Real number register		546.09		546.09
Integer register		34		34

1.16.5.13 G532 - Character transfer to the local character buffer S0

G532.

G532.(End criterion).(Label)

Description:

The **G532.**n command receives characters (max. 80) from the current serial port until the end criterion "n" is received or an error occurred. The characters are stored in local character buffer S0. The end criterion "n" is a number greater than 0 and less than 255 and represents the corresponding ASCII character. If an error occurs during the transmission, the transfer is cancelled and the program continues at the label.

Possible errors:

- Parity error, overrun error, framing error during the character transmission
- Character buffer full

NOTE The program stays in the program line (in the following example line 2) until the end criterion has been detected or a transmission error occurs.

Command format:

G532.n label

Example

```
1   G500.2.6.1.0        ;Initialisation of serial port COM2
2   G532.13 U_ERROR     ;Receives characters at the serial port
3                       and stores them in local character buffer S0 until
                        the end criterion 13 = 0Dhex = CR has been
                        received
4   N3:=COPY.2.5 C_ERROR ;Convert the next 5 characters starting from the
                        2nd character in the character buffer to a number
                        and store it in integer register 3
5   JMP END
6   $U_ERROR
7   SUB Error_1
8   JMP END
9   $C_ERROR
10  SUB Error_2
11  $END
12  END
```

G533.

G533.(End criterion)

Description:

The **G533.**n command receives characters in the background (max. 80) from the current serial port until the end criterion "n" has been received or an error occurred. The characters remain in the receive buffer of the serial port.

The end criterion "n" is a number greater than 0 and less than 255 and represents the corresponding ASCII character.

The addition "in the background" means that PA-CONTROL starts a system routine which accepts the characters. The next command of the program is subsequently executed.

The command Ni:=CHN can be used to check whether the character string was received in full.

Possible errors *(see also chapter 1.11.4.2 page 185):*

- Parity error, overrun error, framing error during the character transmission
- Character buffer full

The characters can be copied from the receive buffer of the serial port to the local character buffer by means of the command "S0:=CHN".

NOTE

If communication is set-up via the serial port, data loss may occur.

```
G510.Please send
...                          ; Warning: If the communication partner
                             ;transmits immediately, data may be
                             ;lost !
G533.13                      ;PAC is ready for reception from this point
```

A better solution is:

```
G533.13                      ; PAC is ready for data reception

G510.Please send!            ; PAC is ready to receive
```

Command format:

G533.n

Example

```
1   G500.2.6.1.0          ;Initialisation of serial port COM2
2   G533.13               Receives characters in the background at the
                          serial port and stores them in the receive
                          buffer until the end criterion 13 = 0Dhex = CR
                          is received
3   $LOOP
4   SUB WORK
5   N1:=CHN               ;Character string received completely?
6   CASE.JMP.N1           ;Evaluation of the transfer
7   (CHAR_RCD)            ;End criterion was received
8   (U_ERROR)             ;Error during the transfer of a character
9   (P_FULL)              ;Character buffer is full and the end criterion
                          has not yet been received
10  ELSE LOOP             ;End criterion has not yet been received
11  $CHAR_RCD
12  S0:=CHN               ;Copy content of the receive buffer of the
                          current data channel to the local character
                          buffer S0
13  N3:=COPY.2.5 C_ERROR  ;Convert the next 5 characters starting from
                          the 2nd character in the character buffer to a
                          number and store it in integer register 3
14  JMP END
15  $U_ERROR
16  O16:=1                ;Error display on
17  I16.1                 ;Error acknowledged
18  O16:=0                ;Error display off
19  JMP END
20  $P_FULL
21  G510.BUFFER FULL      ;Output of the error message
22  $END
23  END
```

G534.

G534.(No. of characters)

Description:

The **G534.**n command receives characters in the background (max. 80) from the current serial port. The characters remain in the receive buffer of the serial port.

The end criterion "n" is a number greater than 0 and less than 81 and represents the number of characters to be received.

The addition "in the background" means that PA-CONTROL starts a system routine which accepts the characters. The next command of the program is subsequently executed.

The command Ni:=CHN can be used to check whether the character string was received fully.

Possible errors (*see also chapter 1.11.4.2 page 185):*

- Parity error, overrun error, framing error during the character transmission
- Character buffer full

The characters can be copied from the receive buffer of the serial port to the local character buffer by means of the command "S0:=CHN".

Command format:

G534.n

Example

```
1   G500.2.6.1.0          ;Initialisation of serial port 2
2   G534.4                ;Receives 4 characters in background at the
                          serial port and stores them in the receive
                          buffer
3   $LOOP
4   SUB WORK
5   N1:=CHN               ;Character string received completely?
6   CASE.JMP.N1           ;Evaluation of the transfer
7   (CHAR_RCD)            ;4 characters were received
8   (U_ERROR)             ;Error during the transfer of a character
9   (P_VOLL)              ;Character buffer is full
10  ELSE LOOP             ;End criterion has not yet been received
11  $ZEI_DA
12  S0:=CHN               ;Copy content of the receive buffer of the
                          current data channel to local character buffer
                          S0
13  N3:=COPY.1.4 C_ERROR  ;Convert the next 5 characters starting from
                          the 2nd character in the character buffer to a
                          number and store it in integer register 3
14  JMP END
15  $U_ERROR
16  SUB Error_1
17  JMP END
18  $P_FULL
19  SUB Error_2
20  $END
21  END
```

G540.

G540.(N Register)

Description:

The G540. command checks if a key on the front plate of PA-CONTROL has been actuated.

If this is the case, the key code is stored in the N register. If this is not the case the N register is set to 0.

The keyboard can be polled while a program section is being processed by means of this command, in order to react to operator requests.

NOTE The command only functions with the front plate of PA-CONTROL (LC display, membrane keyboard).

Please refer to the "PAC key code" table in the "Technical Appendix" chapter for the key codes.

Command format:

G540.Nn

Example

```
1   G11.0                ;Switch off running displays
2   G500.0               ;Switches over the G500 commands to the
                         current display medium (LC display or serial
                         port in case of "Simulation of front plate")
3   G501                 ;Clear display
4   N1:=0
5   $LOOP
6   O1:=1 T10
7   O1:=0 T10
8   G540.N1              ;Checks if a key has been actuated and stores
                         the key code in N1; otherwise N1 is set to 0
9   M1:=N1>0             ;Has a key been actuated?
10  G21 M1.0 LOOP        ;No !
11  G503.4.1             ;Place cursor in the 4th column of the 1st line
12  G502                 ;Delete up to end of line
13  G510.N1
14  G512.32
15  G520.N1
16  JMP LOOP
17  END
```

G541.

G541.(N Register)

Description:

The G541. command waits until a key on the front panel of PA-CONTROL has been actuated and stores the key code in the N-register.

The G541. command can be used to create individual menus in PA-CONTROL.

NOTE The command only functions with the front plate of PA-CONTROL (LC display, membrane keyboard).

Please refer to the "PAC key code" table in the "Technical Appendix" chapter for the key codes.

Command format:

G541.Nn

Example

```
1   G11.0          ;Switch off run displays
2   G500.0         ;Switches over the G500 commands to the
                    current display medium (LC display or serial
                    port in case of "Simulation of front plate")
3   G501           ;Clear display
4   N1:=0
5   $LOOP
6   G541.N1        ;Waits until a key has been actuated and
                    stores the key code in N1
7   G503.4.1       ;Place cursor in the 4th column of the 1st line
8   G502           ;Delete up to end of line
9   G510.N1
10  G512.32
11  G520.N1
12  JMP LOOP
13  END
```

1.16.5.18 G542 - Entry of a value via the keyboard

G542

G542.(Column).(Line).(Length).(Target)

Description:

The G542.s.z.l.Ni command enables the operator to enter a value via the keyboard while the program is active.

The command opens an input field (see the chapter on "User interface“). The length and position of the input field are determined by the operators (parameters) of the command.

The system flag **SM5** is polled to check whether the G542 command has been exited by means of the Enter key (i.e. transfer of the value) or by means of the ESC key (entry is cancelled and the previous value is reset).

The following applies:

s	→	Stands for the column
z	→	Stands for the line
l	→	Stands for the length of the entry field
Target	→	Register or character buffer (storage)

Command format:

G542.s.z.l.Ni

G542.s.z.l.Ri

G542.s.z.l.Si *

G542.Ns.Nz.Nl.Ni

G542.Ns.Nz.Nl.Ri

G542.Ns.Nz.Nl.Si *

* Si can be S0, S1 ...S16

Example

```
1    G11.0                          ;Switch off running displays
2    G500.0                         ;Switches over the G500 commands to the
                                    current display medium (LC display or serial
                                    port in case of "Simulation of front plate")
3    G501                           ;Clear display
4    $LOOP
5    G503.1.1
6    G510.Entry Target
7    G542.30.1.3.N1                 ;Entry of the N register via the front plate. The
                                    input field is displayed in the 30th column of
                                    line 1 with a length of 3 characters
8    G501
9    G503.1.2
10   G510.Time for output 1 is
11   G512.32
12   G520.N1
13   G512.32
14   G510. * 10ms
15   O1:=1 TN1 O1:=0
16   JMP LOOP
17   END
```

1.16.6 Commands of the G6xx-Group

1.16.6.1 G6xx - Image to register / Image from register

G6xx.

Introduction:

An image of the contents of registers can be transferred to outputs or flags by means of this command group during the automatic mode. An image of inputs or flags can also be loaded to a register.

General command structure:

G60?.Rn.m.i

G60?.Nn.m.i

Rn / Nn:	Source / Target
m:	First element of target / source address (corresponds to the least significant bit)
i:	Number of elements to / from which an image is to be created. (1 <= i <= 32)

NOTE m + i must be less than or equal to the maximum number of elements used.

The content of the register is used as follows:

- Decimal places are truncated
- An image of a max. of 32 bits is created

Various formats are available:

- Binary image
- BCD image

1.16.6.2 G600 - Binary notation to outputs

G600.

G600.(Register).(1st output).(No. of outputs)

Command format:

G600.Rn.m.i

G600.Nn.m.i

Example 1

```
1    R3:=10.7
2    G600.R3.5.8      ;Content of R3 is represented in binary code
                      with 8 bits starting from output 5
3    END
```

Status of outputs after execution of the above commands:

Output 5	→	reset (0)
Output 6	→	set (1)
Output 7	→	reset (0)
Output 8	→	set (1)
Output 9	→	reset (0)
Output 10	→	reset (0)
Output 11	→	reset (0)
Output 12	→	reset (0)

Example 2

```
1    N3:=18
2    G600.N3.2.4      ;Content of N3 is represented in binary code
                      with 4 bits starting from output 2
3    END
```

Status of outputs after execution of the above commands:

Output 2	→	reset (0)
Output 3	→	set (1)
Output 4	→	reset (0)
Output 5	→	reset (0)

1.16.6.3 G601 - BCD notation to outputs

G601.

G601.(Register).(1st output).(No. of outputs)

Command format:

G601.Rn.m.i

G601.Nn.m.i

Example 1

```
1    R3:=10.7
2    G601.R3.5.8        ;Content of R3 is represented in BCD code
                        with 8 bits starting from output 5
3    END
```

Status of outputs after execution of the above commands:

Output 5	→	reset (0)
Output 6	→	reset (0)
Output 7	→	reset (0)
Output 8	→	reset (0)
Output 9	→	set (1)
Output 10	→	reset (0)
Output 11	→	reset (0)
Output 12	→	reset (0)

Example 2

```
1    N3:=12
2    G601.N3.2.4        ;Content of N3 is represented in BCD code
                        with 4 bits starting from output 2
3    END
```

Status of outputs after execution of the above commands:

Output 2	→	reset (0)
Output 3	→	set (1)
Output 4	→	reset (0)
Output 5	→	reset (0)

G602.

G602.(Register).(1st flag).(No. of flags)

Command format:

G602.Rn.m.i

G602.Nn.m.i

Example 1

```
1    R3:=11.3
2    G602.R3.4.8        ;Content of R3 is represented in binary code
                        with 8 bits starting from flag 4
3    END
```

Status of the flags after execution of the above commands:

Flag 4	→	set (1)
Flag 5	→	set (1)
Flag 6	→	reset (0)
Flag 7	→	set (1)
Flag 8	→	reset (0)
Flag 9	→	reset (0)
Flag 10	→	reset (0)
Flag 11	→	reset (0)

Example 2

```
1    R3:=65535
2    G602.R3.17.32      ;Content of R3 is represented in binary code
                        with 32 bits starting from flag 17
3    END
```

Status of flags 17 to 49 after execution of the above commands:

Flags 17 – 32	→	are set
Flags 33 – 49	→	are reset

Example 3

```
1    N3:=11
2    G602.N3.1.4        ;Content of N3 is represented in binary code
                        with 4 bits starting from flag 1
3    END
```

Status of the flags after execution of the above commands:

Flag 1	→	set (1)
Flag 2	→	set (1)
Flag 3	→	reset (0)
Flag 4	→	set (1)

1.16.6.5 G603 - Inputs in binary format to register

G603.

G603.(Register). (1st input).(No. of inputs)

Command format:

G603.Rn.m.i
G603.Nn.m.i

Example 1

```
1    G603.R3.2.8        ;Inputs 2 to 9 (2=least significant bit) are
                        interpreted as a binary number and the value
                        transferred to register 3
2    END
```

Status of inputs after execution of the above commands:

Input 2	→	cold (0)	→	Value = 1
Input 3	→	hot (1)	→	Value = 2
Input 4	→	cold (0)	→	Value = 4
Input 5	→	hot (1)	→	Value = 8
Input 6	→	cold (0)	→	Value = 16
Input 7	→	cold (0)	→	Value = 32
Input 8	→	cold (0)	→	Value = 64
Input 9	→	hot (1)	→	Value = 128

This is equivalent to the number "10001010" in binary code and the value 138 in register 3.

1.16.6.6 G604 - Flags in binary format to register

G604.

G604.(Register).(1st flag).(No. of flags)

Command format:

G604.Rn.m.i

G604.Nn.m.i

Example 1

```
1    G604.R3.2.9        ;Flags 2 to 9 (2=least significant bit) are
                        interpreted as a binary number and the value
                        transferred to register 3
2    END
```

Status of flags after execution of the above commands:

Flag 2	→	1	→	Value = 1
Flag 3	→	1	→	Value = 2
Flag 4	→	1	→	Value = 4
Flag 5	→	1	→	Value = 8
Flag 6	→	0	→	Value = 16
Flag 7	→	0	→	Value = 32
Flag 8	→	0	→	Value = 64
Flag 9	→	1	→	Value = 128

This is equivalent to the number "10001111" in binary code and the value 143 in register 3.

1.16.7 Commands of the G7xx-Group

1.16.7.1 General

All PA-CONTROL devices from version 4.60 are equipped with the CANopen interface. Various devices, which are divided into groups, can be connected to the CANopen bus. The devices are assigned with station addresses, i.e. their ID numbers, for identification.

ID number	Device	
ID 1 … ID16	Driving axis 1 to driving axis 16	LV-servoTEC, PA-CONTROL MP
ID17 ... ID48	Digital input / output modules Analog input / output modules	Murr Beckhoff FESTO
ID49 … ID52	Any desired CANopen device	Examples
ID52 ... ID59	Reserve	Reserve
ID60 ... ID63	4th control console to 1st control console	SÜTRON

Only devices as shown the table above can be connected as devices with the ID numbers 1 to 48. These devices are automatically detected by PA-CONTROL during booting and are handled by the operating system.

Any desired CANopen device can be connected as device 49 to 52. Commands are made available for operating these devices in the automatic mode. These commands enable the implementation of communication in the respective application by means of PNC/PTX or PAB programs.

Facilities for checking and simple operation are available in the WINPAC program, diagnosis menu, for the initial start-up and service.

NOTE A control console can be connected as device 60 to 63.
Communication ensues via PDOs directly by means of the operating system.
Control consoles of SÜTRON & Co. are used.

NOTE Always follow the information and specifications of the manufacturers of devices which are used on the CANopen bus in your particular case.
This documentation also provides you with essential additional information on the commands described in the sections below.

The command set for CANopen devices consists of the following command groups:

- Network management commands
- Emergency commands
- SDO commands
- PDO commands

The command set is implemented by the command group G7xx.

1.16.7.2 Network management commands

The various operating statuses (*see Figure 13*) of a device on the CANopen bus can be achieved by means of the 5 commands of this command group.

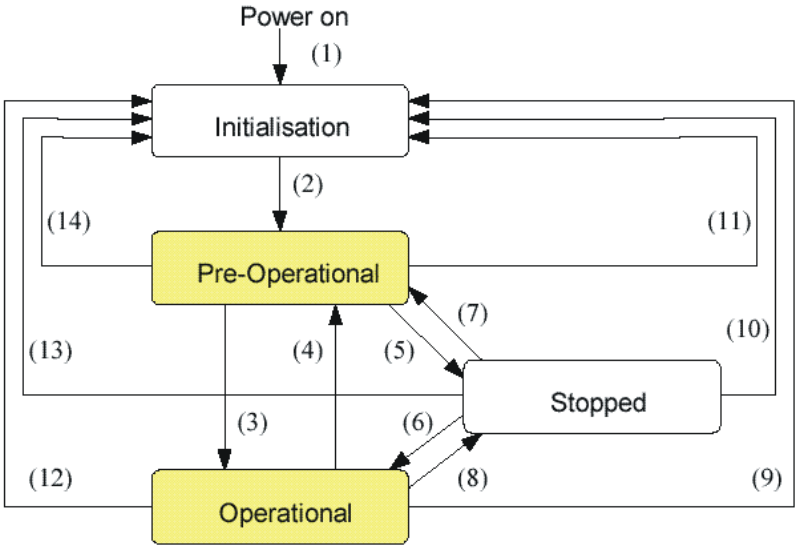


Figure 13: Operating statuses of a CANopen device

1.16.7.3 G701 - Start Remote Node

G701

G701.n

Command format:

G701.49

G701.50

Description:

- The changeover from the operating state "Pre-Operational" to the "Operational" state is implemented by means of this command (*see Figure 13, path 3*).
- The master sends the NMT frame "Start Remote Node" via the CANopen bus to execute this command.

Example

You will find an application for this command at the end of the section as part of a complex example.

1.16.7.4 G702 - Stop Remote Node

G702

G702.n

Command format:

G702.49

G702.50

Description:

- The changeover from the operating state "Operational" to the "Stopped" state (*see Figure 13, path 8*) is implemented by means of this command.
- The master sends the NMT frame "Stop Remote Node" via the CANopen bus to execute this command.

Example

You will find an application for this command at the end of the section as part of a complex example.

1.16.7.5 G703 - Reset Remote Node

G703

G703.n

Command format:

G703.49

G703.50

Description:

- The changeover from the operating state " Operational" to the "Initialisation" state (*see Figure 13, path* 9) is implemented by means of this command. After a successful initialisation, the device independently switches to the "Pre-Operational" state.
- The master sends the NMT frame "Reset Remote Node" via the CANopen bus to execute this command.

Example

You will find an application for this command at the end of the section as part of a complex example.

1.16.7.6 G704 - Enter Pre-Operational State

G704

G704.n

Command format:

G704.49

G704.50

Description:

- The changeover from the operating state " Operational" to the "Pre-Operational" state *(see Figure 13, path 4*) is implemented by means of this command.
- The master sends the NMT frame "Enter-Pre-Operational-State" via the CANopen bus to execute this command.

Example

You will find an application for this command at the end of the section as part of a complex example.

1.16.7.7 G705 - Reset Communication

G705

G705.n

Command format:

G705.49

G705.50

Description:

- The changeover from the operating state "Operational" to the "Initialisation" state (*see Figure 13, path 9*) is implemented by means of this command. After a successful initialisation, the device independently switches to the "Pre-Operational" state.
- The master sends the NMT frame "Reset Communication" via the CANopen bus to execute this command.

Example

You will find an application for this command at the end of the section as part of a complex example.

1.16.7.8 G711 - Functional monitoring of the devices

G711

G711.(ID).Nn (Label)

Command format:

G711.49.12 ERROR

Application:

The Node Guarding Protocol is used for the functional monitoring of the devices. In this connection, the device is to be addressed by PA-CONTROL, the CANopen master, at the same intervals by means of a "NODE GUARDING frame" and a message requested from the device.

This type of device monitoring is provided when the communication is processed only via PDOs after initialization. The maximum time interval between two requests to the device, also referred to as a Node Guard frame, results from the product of the monitoring time (Guard Time) and a factor (Life Time Factor). If one of these values is 0, the response monitoring is deactivated.

If the device is not addressed within the monitoring time, refer to the description of device for its reaction.

Description:

The G711 command requests a monitoring message from a selected device by means of a data request frame (remote frame). The answer is stored in the N register specified in the command. The content of this encoded message can be found in the manual of the device.

If no monitoring message is received from the device within 200ms, the PA-CONTROL program branches to the programmed label.

Example

You will find an application for this command at the end of the section as part of a complex example.

1.16.7.9 G72x - Commands for checking device errors

Device error messages ("Emergency Messages") are initiated by internal device errors. They have a high priority, in order to ensure prompt processing.

A device error message has a prescribed structure:

- Error field with pre-defined error number (2 bytes)
- Error register (1 byte)
- Error category (1 byte)
- Additional information

The higher order byte of the error number specifies the error class, the lower order byte specifies the error number in the class. Error numbers from xx00h to xx7Fh are defined in the communication profile.

The error numbers from xx80h to xxFFh result from specifications of the manufacturers. The error categories allow the occurring errors to be classified according to their importance.

1.16.7.10 G721 / G722 - Check whether a device error message is present

G721 / G722

G721.(ID) (Label)
G722.(ID) (Name)

The bit "New device error message received" is checked. If it is set, a new device error message has been received.

In this case, the message is copied to the S0 character buffer and the bit reset. Afterwards, either the "label" (G721) specified in the command is branched to or the subroutine "Name" (G722) invoked.

If there is no new device error message, the program is continued with the next command.

Representation of the S0 string after a device error (Definition CiA Draft 301)

Byte	1	2	3	4	5	6	7	8
Content	LowByte	HighByte	Error register (Object 1001)	Error field, specified by the manufacturer				
	Device error code							

Refer to the manual of the defective CANopen device for the possible content of the device error message.

Command format:

G721.49 EMERGENCY
G722.50 GRIPPER

Example for the evaluation of errors:

```
...
N1:=GETI.1.2          ; Error code
...
N2:=GETI.3.1          ; Error register (Object 1001hex)
```

1.16.7.11 Commands for processing service data objects

In the same way as in other familiar field bus systems, CANopen makes a distinction between two basic data transfer mechanisms, i.e. the high-speed exchange of short process data via so-called process data objects (PDOs) and the access to entries of the object data directory via service data objects (SDOs). The service data objects are primarily used for the transfer of parameters during the device configuration and for the transfer of lengthy data areas.

The service data objects are accessed with a variable length.

NOTE In principle, a request frame and an answer frame are required for the reading or writing of a service data object. As a result, a device should be accessed in only one parallel run.

If this is disregarded, data errors may occur if the request frame comes from run 1 and the answer frame from run 2.

1.16.7.12 G730 - Read the content of a service data object (Initiate Domain Upload Rq)

G730

Command format:

G730.<ID>.<Index(hex)>.<Subindex>.< No. Data Bytes >.<Error>.<Read Data> <Label>

If an error occurs during the read operation, e.g. an incorrect index or incorrect subindex, the error code is stored in the specified integer register. The program branches to the <Label>. If the command proceeds normally, the data is stored in the register for the read data and the program run continued with the next command.

If the request to read is not acknowledged within 1 second, PA-CONTROL branches after the label with error 9, "IEF time-out".

NOTE The index is transferred as a hex number.

Example

G730.49.1000.0.4.N1.N2 ERROR_SDO

In this example command, the device definition (device type) of CANopen device 49 is read and stored in integer register N2. In the case of an error, the error would be stored in register N1 and the program continued at the label "ERROR_SDO".

1.16.7.13 G731 - Write the content of a service data object (Initiate Domain Upload Rq)

G731

Command format:

G731.<ID>.<Index(hex)>.<Subindex>.<No. data bytes>.<Error>.<Write Data> <Label>

If an error occurs during the write operation, the error code is stored in the specified integer register. The program branches to the <Label>. If the command proceeds normally, the program run is continued with the next command.

If the request is not acknowledged within 1 second, PA-CONTROL branches after the label with error 9, "IEF time-out".

NOTE The index is transferred as a hex number.

Command format:

G731.49.100C.0.2.N1.N2 FEHLER_SDO

In this example, the monitoring time (Guard Time) of CANopen device 49 is overwritten with the content of register N2. In the case of an error, the error is stored in register N1 and the program continued at the label "ERROR_SDO".

1.16.7.14 Commands for the processing of process data objects

As described in the previous section, CANopen makes a distinction between two basic data transfer mechanism, i.e. the high-speed exchange of short process data via so-called process data objects (PDOs) and the access to entries of the object data directory via service data objects (SDOs).

Process data objects can be optionally transferred event-driven or synchronized. The event-driven mode enables the constant transfer of the complete process image to be dispensed. Only the changes need to be transferred. In this way, the bus loading and the reaction time can be reduced to a minimum. A high communication performance is obtained at a comparatively low baud rate.

Process data objects can be sent and received. Sending and receiving is always regarded from the point of view of the slave, e.g. an I/O module.

Four process data objects are assigned to each device on the CANopen bus. There is a command and one storage location for each of them in PA-CONTROL.

The CANopen protocol is based on the CAN Application Layer (CAL) and makes standardised device profiles available. The communication profile DS-301 forms the basis for communication in the CANopen bus. It defines the permissible communication mechanisms between the devices on the bus. The communication profile is the basis for diverse device profiles which enable standardised access to all parameters of a device.

The profiles listed below have been implemented so far:

Profile	Function
DS-301	CANopen communication profile
DS-401	Device profile for digital and analog I/O subassemblies
DS-402	Device profile for drive
DS-403	Device profile for control stations
DS-404	Device profile for sensors and controllers
DS-405	Interface to programmable systems (IEC1131)
DS-406	Device profile for rotary encoders and encoders
DS-407	Application profile for local public transport

According to the definition of DEVICE-PROFILE 401 "I/O modules", the process data objects are assigned as follows. The order can be changed by "variable mapping".

Receive PDO	
PDO no.	**Data**
1	Maximum of 64 digital outputs (O1 – O64)
2	Maximum of 4 analog outputs (DA1 – DA4)
3	Free
4	Free

Transmit PDO	
PDO no.	**Data**
1	Maximum of 64 digital inputs (I1 – I64)
2	Maximum of 4 analog inputs (AD1 – AD4)
3	Free
4	Free

The third and fourth process data object is not permanently defined in the standard. The manufacturers of I/O modules apply the next digital and analogue inputs and outputs to these PDOs depending on the hardware configuration of the I/O module. Refer to the technical documentation of the manufacturer for further information about the application case.

Example: Murr I/O module with
- 16 digital inputs
- 8 digital outputs
- 4 analog inputs (12 bit, +/-10V)
- 4 analog outputs (12 bit, +/-10V)

Receive PDO	
PDO no	**Data**
1	Maximum of 64 digital outputs (O1 – O8)
2	Maximum of 4 analog outputs (DA1 – DA4)
3	Free
4	Free

Transmit PDO	
PDO no	**Data**
1	Maximum of 64 digital inputs (I1 – I16)
2	Maximum of 4 analog inputs (AD1 – AD4)
3	Free
4	Free

Commands for all four objects are available for accessing the process data objects (PDO).

- PDO1
- PDO2
- PDO3
- PDO4

The number of data bytes of a PDO can vary between 1 and 8.

G74x

G74x.(ID). (No. data bytes)

The CANopen device sends its processes data to PA-CONTROL cyclically or after synchronisation, depending on the configuration, i.e. if an input is changed. The received data is stored in the assigned PDO receive buffer (PDO1, PDO2...).

The command "Read process data" causes the data to be copied from the receive buffer to the local S0 character buffer.

NOTE The process data must always be transferred via the CANopen bus in the INTEL format for evaluation of the data. The data in the local S0 character buffer is read out using the "GETI command" (GET-INTEL format), (*see chapter 1.11.5.13, page 196*) and is then available for evaluation.

Command format:

G741.(ID).(No. data bytes)	PDO1
G742.(ID).(No. data bytes)	PDO2
G743.(ID).(No. data bytes)	PDO3
G744.(ID).(No. data bytes)	PDO4

Examples

G741.49.4	The first four bytes of the data received from the first process data object (PDO1) are copied from CANopen device 49 to the S0 string.
G742.50.2	The first two bytes of the data received from the second process data object (PDO1) are copied from CANopen device 50 to the S0 string.

1.16.7.16 Send process data

G75x

G75x.(ID). (No. data bytes)

Two steps are required to send process data. The data to be sent is entered in the S0 string in the first step. This is followed by the actual transmission to the CANopen device.

NOTE The process data is transferred via the CANopen bus in the INTEL format. This must always be considered when entering the data in the local S0 character buffer.

The PUTI command (PUT-INTEL format), (*see chapter 1.11.5.13, page 196*) writes the content of an N register to the S0 string.

Command format:

G751.(ID).(No. data bytes)	PDO1
G752.(ID).(No. data bytes)	PDO2
G753.(ID).(No. data bytes)	PDO3
G754.(ID).(No. data bytes)	PDO4

Example

G751.49.4	Sends the first four bytes from the S0 string to the first process first data object (PDO1) of CANopen device 49.

1.16.7.17 Typical applications of command group G7xx

Example 1 : for MURR I/O module

The CANopen module 55900 of Murr & Co. is used in the example.

55920:	DI8	⇨ 8 inputs	⇨ from flag M1 to	M8
55922:	DO8/0,5A	⇨ 8 outputs	⇨ from flag M9 to	M16

Example 1: Start-SDO.pnc

```
1   G701.49                                        ; START REMOTE NODE
2   T50                                            ; wait until CAN modules are in
                                                   OPERATION mode
3   ;
4   $A
5   G721.49 EMERGENCY                              ; Error message there?
6   ;
7   G730.49.6200.2.1.N1.N2 READ_SDO_ERROR          ; Read SD= ⇨ inputs
8   G602.N2.1.8                                    ; Image to flag M1 – M8
9   ;
10  G604.N3.9.8                                    ; Image from flag M9 – M16
11  G731.49.6200.1.1.N1.N3 WRITE_SDO_ERROR         ; Write SDO ⇨ outputs
12  JMP A
13  ;
14  $READ_SDO_ERROR
15  $WRITE_SDO_ERROR
16  O1:=1                                          ; Report error
17  I1.1                                           ; SDO error
18  BREAK                                          ;
19  ;
20  $EMERGENCY
21  N5:=GETI.1.2                                   ; Error code
22  N6:=GETI.3.1                                   ; Error register object 0x1001
23  M100:=N5=0                                     ; Error <>0?
24  G21 M100.1 A                                   ;NO, no error!
25  I1.1                                           ; Emergency error
26  Break                                          ;
27  ;
28  END
```

NOTE The NODE GUARDING, by means of which the device is addressed via SDOs, can be dispensed with in the above example.

Example 2 for MURR I/O module

The CANopen module 55900 of company Murr & Co. is used in the example.

55920:	DI8	⇨ 8 inputs	⇨ from flag M1 to	M8
55922:	DO8/0,5A	⇨ 8 8 outputs	⇨ from flag M9 to	M16

Example 2: Start-pdo.pnc

```
1   G701.49                 ; START REMOTE NODE
2   T50                     ; wait until CAN modules are in OPERATION
                              mode
3   ;
4   $A
5   G721.49 EMERGENCY       ; Error message there?
6   G711.49 NODE_GUARD      ;
7   G741.49.2               ; Read PDO1 →S0
8   N2:=GETI.2.1            ; Get 2 bytes from S0
9   G602.N2.1.8             ; Image to flag M1 – M8
10  ;
11  G604.N3.9.8             ; Get image from flag M9 – M16
12  PUTI.1.1.N3             ; Insert in S0
13  G751.49.1               ; S0 → write PDO1
14  JMP A
15  ;
16  $EMERGENCY
17  N5:=GETI.1.2            ; Error code
18  N6:=GETI.3.1            ; Error register Object 0x1001
19  M100:=N5=0              ; Error <>0 ?
20  G21 M100.1 A            ; NO, no error!
21  I1.1                    ; Emergency error
22  BREAK                   ;
23  ;
24  $NODE_GUARD             , Device has not reported to NODE GUARD
25  I1.1
24  BREAK
23  ;
24  END
```

2 PA-CONTROL key code

Dec	Hex	Character	PAC Keyboard
0	0	NUL	
1	1	SOH	
2	2	STX	
3	3	ETX	
4	4	EOT	
5	5	ENQ	
6	6	ACK	
7	7	BEL	
8	8	BS	Shift+DEL
9	9	HT	
10	A	LF	
11	B	VT	
12	C	FF	
13	D	CR	ENTER
14	E	SO	
15	F	SI	
16	10	DLE	
17	11	DC1	
18	12	DC2	
19	13	DC3	
20	14	DC4	
21	15	NAK	
22	16	SYN	
23	17	ETB	
24	18	CAN	
25	19	EM	
26	1A	SUB	
27	1B	ESC	ESC/Shift+ESC
28	1C	FS	
29	1D	GS	
30	1E	RS	
31	1F	US	
32	20	SP	SPACE BAR
33	21	!	Shift+1
34	22	„	Shift+2
35	23	#	
36	24	$	Shift+4
37	25	%	Shift+5
38	26	&	Shift+6
39	27	‘	
40	28	(	Shift+8
41	29	)	Shift+9
42	2A	*	Shift++
43	2B	+	+
44	2C	,	,
45	2D	-	-
46	2E	.	.

Dec	Hex	Character	PAC Keyboard
47	2F	/	Shift+7
48	30	0	0
49	31	1	1
50	32	2	2
51	33	3	3
52	34	4	4
53	35	5	5
54	36	6	6
55	37	7	7
56	38	8	8
57	39	9	9
58	3A	:	Shift+ .
59	3B	;	Shift+ ,
60	3C	<	<
61	3D	=	Shift+0
62	3E	>	Shift+<
63	3F	?	Shift+3
64	40	@	Ctrl+Q
65	41	A	Shift+A
66	42	B	Shift+B
67	43	C	Shift+C
68	44	D	Shift+D
69	45	E	Shift+E
70	46	F	Shift+F
71	47	G	Shift+G
72	48	H	Shift+H
73	49	I	Shift+I
74	4A	J	Shift+J
75	4B	K	Shift+K
76	4C	L	Shift+L
77	4D	M	Shift+M
78	4E	N	Shift+N
79	4F	O	Shift+O
80	50	P	Shift+P
81	51	Q	Shift+Q
82	52	R	Shift+R
83	53	S	Shift+S
84	54	T	Shift+T
85	55	U	Shift+U
86	56	V	Shift+V
87	57	W	Shift+W
88	58	X	Shift+X
89	59	Y	Shift+Y
90	5A	Z	Shift+Z
91	5B	[	Ctrl+8
92	5C	\	
93	5D	]	Ctrl+9

Dec	Hex	Character	PAC Keyboard
94	5E	^	
95	5F	_	Shift+ -
96	60	´	
97	61	a	A
98	62	b	B
99	63	c	C
100	64	d	D
101	65	e	E
102	66	f	F
103	67	g	G
104	68	h	H
105	69	i	I
106	6A	j	J
107	6B	k	K
108	6C	l	L
109	6D	m	M
110	6E	n	N
111	6F	o	O
112	70	p	P
113	71	q	Q
114	72	r	R
115	73	s	S
116	74	t	T
117	75	u	U
118	76	v	V
119	77	w	W
120	78	x	X
121	79	y	Y
122	7A	z	Z
123	7B	{	Ctrl+7
124	7C	|	
125	7D	}	Ctrl+0
126	7E	~	
127	7F	DEL	
128	80	PAD	
129	81	HOP	
130	82	BPH	
131	83	NBH	
132	84	IND	
133	85	NEL	
134	86	SSA	
135	87	ESA	
136	88	HTS	
137	89	HTJ	
138	8A	VTS	
139	8B	PLD	
140	8C	PLU	

Dec	Hex	Character	PAC Keyboard
141	8D	RI	
142	8E	SS2	
143	8F	SS3	
144	90	DCs	
145	91	PU1	
146	92	PU2	
147	93	STS	
148	94	CCH	
149	95	MW	
150	96	SPA	
151	97	EPA	
152	98	SOS	
153	99	SGCI	
154	9A	SCI	
155	9B	CSI	
156	9C	ST	
157	9D	OSC	
158	9E	PM	
159	9F	APC	
160	A0	NBSP	
161	A1	¡	
162	A2	¢	
163	A3	£	
164	A4	€	
165	A5	¥	
166	A6	Š	
167	A7	§	
168	A8	š	
169	A9	©	
170	AA	ª	
171	AB	«	
172	AC	¬	
173	AD	SHY	
174	AE	®	
175	AF	¯	
176	B0	°	
177	B1	±	
178	B2	²	
179	B3	³	
180	B4	Ž	
181	B5	µ	
182	B6	¶	
183	B7	·	
184	B8	ž	
185	B9	¹	
186	BA	º	
187	BB	»	

Dec	Hex	Character	PAC Keyboard
188	BC	Œ	
189	BD	œ	
190	BE	Ÿ	
191	BF	¿	
192	C0	À	
193	C1	Á	
194	C2	Â	
195	C3	Ã	
196	C4	Ä	Alt+A
197	C5	Å	
198	C6	Æ	
199	C7	Ç	
200	C8	È	
201	C9	É	
202	CA	Ê	
203	CB	Ë	
204	CC	Ì	
205	CD	Í	
206	CE	Î	
207	CF	Ï	
208	D0	Ð	
209	D1	Ñ	
210	D2	Ò	
211	D3	Ó	
212	D4	Ô	
213	D5	Õ	
214	D6	Ö	Alt+O
215	D7	×	
216	D8	Ø	
217	D9	Ù	
218	DA	Ú	
219	DB	Û	
220	DC	Ü	Alt+U
221	DD	Ý	
222	DE	Þ	START
223	DF	ß	
224	E0	à	
225	E1	á	
226	E2	â	
227	E3	ã	
228	E4	ä	Ctrl+A
229	E5	å	
230	E6	æ	
231	E7	ç	
232	E8	è	
233	E9	é	

Dec	Hex	Character	PAC Keyboard
234	EA	ê	
235	EB	ë	
236	EC	ì	
237	ED	í	
238	EE	î	
239	EF	ï	
240	F0	ð	
241	F1	ñ	
242	F2	ò	
243	F3	ó	
244	F4	ô	
245	F5	õ	
246	F6	ö	Ctrl+O
247	F7	÷	
248	F8	ø	
249	F9	ù	
250	FA	ú	
251	FB	û	
252	FC	ü	Ctrl+U
253	FD	ý	
254	FE	þ	
255	FF	ÿ	

Additions

Dec	Hex	PAC Keyboard
222	DE	Start button
294	126	Alt+L
327	147	Shift+Arrow left
328	148	Arrow up
329	149	Shift+Arrow up
331	14B	Arrow left
333	14D	Arrow right
335	14F	Shift+Arrow right
336	150	Arrow down
337	151	Shift+Arrow down
338	152	INS Shift+INS
339	153	DEL

3 INDEX

A

A1 93
A1 (Axis) 93
ABS 161
Absolute dimension system 94
Absolute value
of a real number 161
of an integer 161
Activate
Standard positioning mode 234
Activate the start positioning mode 230
Addition 153
ADi 197
Analog-to-digital converter
Acquisition of A-D values in a defined time base 219
Acquisition of several A-D values 217
Synchronous acquisition of A-D values 217
AND-LD 173, 174
Approach to reference point 92
Arithmetic operations
Addition 153
Division 156
Multiplication 155
Subtraction 154
Automatic mode
Automatic mode with STOP 46
Sequence of malfunction 47
Automatic run
start 52
stop 53
Axis
positioning 93
Axis error number 23, 27

B

BCD notation to outputs 271
Binary notation to flags 272
Binary notation to outputs 270
BREAK 51

C

Cancel 44
Cancel automatic mode 51
CANopen bus
Checking device errors 279
Device error message is present 280
Enter Pre-Operational State 278

Functional monitoring of the devices ... 279
Network management commands ... 276
Processing of process data objects ... 282
Processing of service data objects ... 280
Read process data ... 284
Read service data object ... 281
Reset Communication ... 278
Send process data ... 285
Start-Remote Node ... 276
Stop-Remote Node ... 277
Typical applications ... 286
Write service data object ... 281
CANopen-Bus
Reset Remote Node ... 277
CASE.STORE ... 208
Changing the motor current ... 134
Character transfer
in the background ... 262, 264
to the local character buffer ... 261
Check
Character transfer in the background completed ... 185
Check if a key has been actuated ... 265
CHN ... 185
Clear the display ... 249
Clear up to end of line ... 250
CNTn.i.Nm ... 213
Command code ... 50
Command groups ... 39
Compare
complex examples ... 163
Comparison operations ... 162
Comparisons ... 144
Conditional jump ... 56
Conditional Subroutine call ... 65
Convert characters ... 188
COPY ... 188
Current task number ... 23, 27

D

DAi ... 200
Digital-to-analog converter ... 199
Brief description ... 199
Direct addressing ... 150
N register ... 150
R register ... 150
Division ... 156

E

Entry of a value via the keyboard ... 267
Error management

activate ... 82
ERROROFF ... 82
ERRORON ... 82
Error management
deactivate ... 82
ERROROFF ... 82
ERRORON ... 82
Explanations ... 50

F

F 97
FB
= n ... 123
FB.i
=n / FB.i
=Rn 123
Flow diagram
PA-CONTROL in the initial position ... 48
Start automatic mode ... 45
Following error
PA-CONTROL MP ... 111

G

G01 ... 124
G01 A1
=(POS) A2
=(POS)..... 126
G100 ... 96
G100.B.n ... 122
G101 ... 134, 135, 137
G123 ... 99
G123Q ... 101
G123Q ... 101
G140 ... 86
G141 ... 87
G142 ... 109
G143 ... 110
G144 ... 120
G145 ... 121
G150 ... 102
G170 ... 201
G171 ... 202
G172 ... 203
G173 ... 204
G180 ... 217
G181 ... 219
G210 ... 230
G211 ... 231
G212 ... 233

G213 234
G221 234
G222 237
G230 239
G231 240
G25.An 92
G26 105
G29 106
G401.1 246
G422.1 244
G423.1 245
G500 247
G501 249
G502 250
G503/G504 251
G510 252
G511 253
G512 254
G515 255
G520 256
G521 258
G531 260
G532 261
G533 262
G534 264
G540 265
G541 266
G542 267
G600 270
G601 271
G602 272
G603 273
G701 276
G702 277
G703 277
G704 278
G705 278
G711 279
G721 280
G722 280
G730 281
G731 281
G74x 284
G75x 285
G90 94
G91 95
Get a character from the keyboard 266
Get A-D values 197

Get axis position 108
Get program status 81
GETM 145

H

Hold-up time 146

I

Image from register 269
Image to register 269
In.m 138
Incremental dimension system 95
Indirect addressing 150
N register 150
R register 150
Initialisation of the ports 247
Inputs in binary format to register 273
INT 160
Interpolation
Abort 128
Activate outputs before starting 131
Activation of outputs during interpolation 132
IPOEND 128

J

JMP 55
JMP-LINE.Ni 57
JMP-LINE-IPO.Ni 133
Jump distributor 58

L

Limit switch monitoring
switch on 110
LINE command 54
Linear interpolation 124
List of system flags 14
List of system N registers 23, 27
Load register 54, 151
Load register with axis parameters 118
Logic operations 164
Command AND 165
Command AND-LD 166
Command LD 165
Command OR 165
command OR-LD 166
Command OUT 167
Complex logic operations 176
Logic AND operations 168
logic OR operations 169

Multi-level logic AND operations 174
Multi-level OR operations 175
Supplemental instructions 170
Loop with conditional jump 68, 69
LVservoTEC
Get following error 111

M

MALFUNCTION 44
Mn.M 143
Mn:=m 141
Multiplication 155

N

Nesting 62
Ni
=CNTn 212
=XOR.Sn 192
Ni.Value 178
Ni/Ri
=MODBUS 183
Ni:= PEAn 111
Ni:=ENCn 112
Nn
=16#yyy 152
=2#xxxxxxx 152
=SNn 179
Nn:=An 108

O

OFF
An 88
ON.An 89
On.m 139
On:=m 140
Operating hours counter 23, 27
Operating tenths of a second counter 23, 27
Output of control characters 254
Output of D-A values 200
Output of value 256, 258
Overview of the command set 29

P

PA-CONTROL key code 288
Parallel run
starting with CASE 75
POS 187
Position the cursor 251

Position-conditional
jump ... 231
subroutine call ... 233, 237
Position-conditional jump ... 234
Program status
Standard request ... 81
Program structure ... 49
Programming elements ... 12
Programming tips ... 42
Cancel ... 44
MALFUNCTION ... 44
START after STOP ... 44
Programs in parallel run
Ending ... 74
Ending with Case ... 79
Start ... 71
Stopping ... 73
Stopping with Case ... 77

R

RAn.m
=Ri ... 119
Read character from PTX file ... 201
Real number register
Decimal component ... 161
integer component ... 160
Real-time clock
Read out ... 149
Set date ... 147
Set time ... 148
Request
current path speed ... 129
Current speed of an axis during interpolation ... 130
Request for the current line number ... 54
Ri:= PEAn ... 111
Rn
=SRn ... 180
=TCi.Parameter name ... 216
Rn/Nn:= ... 151
Rn:=An ... 108
Rn:=ENCn ... 112
Rn:=FB ... 129
Rn:=Ran ... 118
Root function ... 160
Run error number ... 23, 27
Run interpreter ... 42

S

S0 - Copying character strings ... 186
S0:= CHN ... 186

Search for character position 187
Selection of the data channel 247
Sequence number 23, 27
ServoTEC Parameters
Read 117
ServoTEC-Parameters
Write 115
SET / RES 170
Set / reset
Flag 141
Set position to dimension 106
Set position to zero 105
Set/Reset
Output 140
Sn - Copying character strings 186
Sn:= Si 186
Specifying the acceleration 96
SQRT 160
Standard Subroutine call 61
START after STOP 44
Start of an interpolation at a specified program line 57, 133
STOP.An 90
Storage of values 208
Store string (Sn) in a line of a PTX file 204
Subroutine call
extended form 63
Program name from regsister 64
Standard 60
Subroutine distributor 66
Subtraction 154
Switch display on / off 184
System error number 23, 27
System flags 14
System functions 43
Cancel 44
MALFUNCTION 44
Program at MALFUNCTION 43
Program at START 43
Program at START after STOP 43
Program at STOP 43
START after STOP 44
System integer registers 23, 27

T

T 146
TCn.Parameter name
= 215
Text and value output via the current data channel 247
Text output 252, 253, 255

Time monitoring
conditional subroutine call ... 244
Time monitoring commands ... 242
Transfer encoder position ... 112
Transfer parameters ... 248
Transfer the content of an N register
in the INTEL format to S0 ... 196
to the local character buffer ... 195
Transfer the content of the local character buffer
to an integer register ... 193
Transfer the content of the local character buffer S0
to an integer register in Intel format ... 194
Traverse as long as condition fulfilled ... 99
Traverse segment with Start-Stop ... 102
Traversing speed ... 97
Trigonometrical functions ... 157
Cosine ... 158
Sine ... 157
Tangent ... 159

U

Unconditional jump ... 55

V

Value transfer ... 260

W

Wait for position
from the SSI interface ... 114
Wait until current position </> than value ... 239
Waiting for logical status ... 138
Flag ... 143
Input ... 138
N register ... 178
Output ... 139
Write character to PTX file ... 202
Write line from PTX file to string (Sn) ... 203