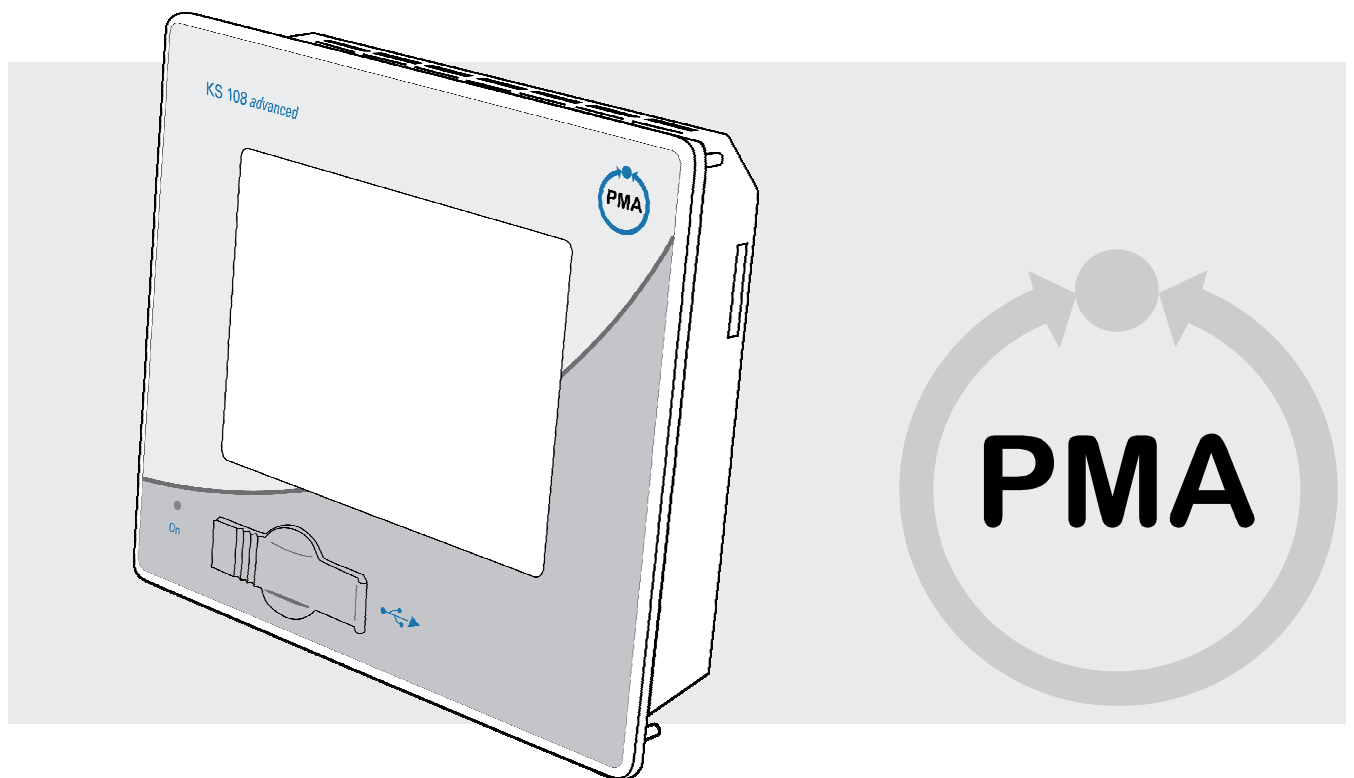


PMA Prozess- und Maschinen-Automation GmbH

User Manual

KS 108 easy



CE

Operator's guide, please read before using product

Bestellnummer:
9499-040-85911

© PMA
Prozeß- und Maschinen-Automation GmbH
Miramstraße 87
34123 Kassel

Tel.: + 49 / 0561 / 505-0
Fax.: + 49 / 0561 / 505-1710

mailbox@pma-online.de
www.pma-online.de

Release:
Revision: 1.3

Contents

| | | |
|------------|---|-----------|
| I | Operator guide | 14 |
| I-1 | General | 14 |
| I-1.1 | Information on the operator's guide | 14 |
| I-1.2 | Manufacturer | 14 |
| I-1.3 | Guarantee conditions | 14 |
| I-1.4 | Customer service | 14 |
| I-1.5 | Explanation of symbols | 15 |
| I-1.6 | Limitation of liability | 16 |
| I-1.7 | Copyright | 16 |
| I-1.8 | Transport, packaging, and storage | 17 |
| I-1.8.1 | Transport | 17 |
| I-1.8.2 | Unpacking | 17 |
| I-1.8.3 | Storage | 17 |
| I-1.9 | Disposal | 17 |
| I-2 | Safety | 19 |
| I-2.1 | General | 19 |
| I-2.2 | Owner's responsibility | 20 |
| I-2.3 | Operating personnel | 21 |
| I-2.3.1 | Requirements | 21 |
| I-2.3.2 | Instruction | 21 |
| I-2.4 | Safety integrated systems | 22 |
| I-2.5 | Special dangers | 22 |
| I-2.5.1 | Device failure, maintain, decommissioning | 22 |
| I-2.5.2 | Explosion protection | 22 |
| I-2.5.3 | Electric components | 23 |
| I-2.5.4 | Batteries | 23 |
| I-2.5.5 | Application development | 23 |
| I-2.6 | Environmental protection | 24 |
| I-2.7 | Intended use | 24 |
| I-3 | Structure and function, technical data | 25 |
| I-3.1 | Device description | 25 |
| I-3.2 | Models | 26 |
| I-3.3 | Accessories | 27 |
| I-3.4 | Performance features - overview | 27 |
| I-3.5 | Device overview | 28 |
| I-3.6 | Technical data | 28 |
| I-3.7 | Type plate | 30 |
| I-3.8 | Block diagram | 31 |
| I-3.9 | Interfaces | 31 |
| I-3.9.1 | Overview | 31 |
| I-3.9.2 | Power supply | 32 |
| I-3.9.3 | 10/100 Base-T network connection (Ethernet) | 32 |
| I-3.9.4 | USB | 33 |
| I-3.9.5 | CAN bus | 33 |
| I-3.9.6 | Serial interfaces | 34 |

| | | |
|------------|---|------------|
| I-3.10 | Declaration of conformity..... | 34 |
| I-4 | Mounting and commissioning | 35 |
| I-4.1 | Scope of delivery | 35 |
| I-4.2 | Mounting..... | 35 |
| I-4.2.1 | Mounting cutout KS 108..... | 35 |
| I-4.2.2 | Mounting execute..... | 36 |
| I-4.2.3 | Connecting the device | 37 |
| I-4.3 | Commissioning | 39 |
| I-4.3.1 | Checking the installation | 39 |
| I-4.3.2 | Switching on the device | 39 |
| I-4.3.3 | Loading the program from USB stick | 40 |
| I-4.3.4 | Configure network settings | 42 |
| I-4.3.5 | Using an SD card | 42 |
| I-4.3.6 | External I/O connection | 43 |
| I-5 | Operation..... | 44 |
| I-5.1 | User operating page | 44 |
| I-5.2 | Main menu..... | 46 |
| I-5.3 | Using the menu..... | 46 |
| I-5.4 | Navigating in the menu | 47 |
| I-5.5 | Changing parameters and configurations | 47 |
| I-5.5.1 | Using the list selection editor | 48 |
| I-5.5.2 | Using the numerical value editor | 48 |
| I-5.5.3 | Using the binary number editor..... | 49 |
| I-5.5.4 | Using the text editor | 49 |
| I-5.6 | Using the "General Data" menu | 50 |
| I-5.6.1 | Configure network | 52 |
| I-5.7 | Operating pages | 54 |
| I-5.7.1 | Operating page V_ALARM | 55 |
| I-5.7.2 | Operating page V_DISPLAY | 58 |
| I-5.7.3 | Operating page V_BAR..... | 60 |
| I-5.7.4 | Operating page V_LOGGING / V_LOGGING2 | 61 |
| I-5.7.5 | Operating page V_TREND | 64 |
| I-5.7.6 | Operating page PASSWORD | 69 |
| I-5.7.7 | Operating page CONTROL | 71 |
| I-5.7.8 | Using controllers in controller cascades..... | 76 |
| I-5.7.9 | Detailed information on self-optimization | 76 |
| I-5.7.10 | Operating page A_PROG | 89 |
| I-5.7.11 | Operating page D_PROG | 95 |
| I-5.7.12 | Operating page PROGRAMMER..... | 97 |
| I-6 | Maintenance and cleaning | 106 |
| I-6.1 | Maintenance..... | 106 |
| I-6.1.1 | Real-time clock | 106 |
| I-6.1.1 | Changing batteries..... | 107 |
| I-6.2 | Cleaning..... | 109 |
| I-6.2.1 | Cleaning heavy fouling | 109 |
| I-6.3 | After maintenance..... | 110 |
| I-7 | Troubleshooting | 111 |

| | | |
|-------------|---|------------|
| I-7.1 | Fault table..... | 112 |
| II | Development environment..... | 114 |
| II-1 | Installation and configuration..... | 115 |
| II-1.1 | Installation..... | 115 |
| II-1.1.1 | Installing BlueDesign..... | 115 |
| II-1.1.2 | Installing Vario-Configurator..... | 116 |
| II-1.2 | Licensing BlueDesign..... | 117 |
| II-1.3 | Configuring BlueDesign..... | 118 |
| II-2 | The components of the development environment..... | 120 |
| II-2.1 | KS 108 easy device with runtime environment..... | 120 |
| II-2.2 | BlueDesign..... | 120 |
| II-2.3 | PMA library..... | 121 |
| II-2.4 | BlueSimulation..... | 122 |
| II-3 | Working with the development environment..... | 123 |
| II-3.1 | Fundamentals..... | 123 |
| II-3.1.1 | Project structure..... | 123 |
| II-3.1.2 | <i>BlueDesign</i> operating modes..... | 124 |
| II-3.2 | Using operating modes..... | 126 |
| II-3.2.1 | Switching from edit mode to run mode..... | 126 |
| II-3.2.2 | Switching from run mode to online observation mode..... | 127 |
| II-3.2.3 | Switching to edit mode..... | 127 |
| II-3.3 | Creating a project..... | 128 |
| II-3.4 | Working with program blocks..... | 129 |
| II-3.4.1 | Creating program blocks..... | 129 |
| II-3.4.2 | Deleting program blocks..... | 130 |
| II-3.4.3 | Renaming program blocks..... | 130 |
| II-3.4.4 | Copying program blocks..... | 130 |
| II-3.4.5 | Exporting program blocks..... | 131 |
| II-3.4.6 | Displaying/changing program block call name..... | 131 |
| II-3.4.7 | Defining program block cycle time..... | 132 |
| II-3.5 | Working with blocks..... | 133 |
| II-3.5.1 | Positioning blocks..... | 133 |
| II-3.5.2 | Connecting blocks..... | 134 |
| II-3.5.3 | Deleting blocks..... | 134 |
| II-3.5.4 | Defining the sequence of blocks..... | 134 |
| II-3.5.5 | Copying blocks..... | 135 |
| II-3.6 | Specifying inputs and outputs..... | 136 |
| II-3.7 | Positioning inputs and outputs..... | 138 |
| II-3.8 | Editing the configuration..... | 138 |
| II-3.9 | Using parameters..... | 139 |
| II-3.10 | Working with macro blocks..... | 142 |
| II-3.10.1 | Creating a new macro block..... | 142 |
| II-3.10.2 | Using the macro block..... | 143 |
| II-3.10.3 | Editing parameters in macro blocks..... | 144 |
| II-3.11 | Using global variables..... | 145 |
| II-3.11.1 | Using backup copies..... | 146 |
| II-3.12 | Using the development environment..... | 148 |

| | | |
|--------------|---|------------|
| II-3.12.1 | Logon to target system | 148 |
| II-3.12.2 | Disconnecting from target system | 150 |
| II-3.12.3 | Download | 151 |
| II-3.12.4 | Changing worksheet size..... | 151 |
| II-3.12.5 | Adapting worksheet view..... | 152 |
| II-3.12.6 | Moving a worksheet | 153 |
| II-3.12.7 | Printing the project | 154 |
| II-3.12.8 | Using symbols..... | 155 |
| II-3.12.9 | Using background pictures | 156 |
| II-3.12.10 | Formatting structure blocks..... | 157 |
| II-3.12.11 | Using keyboard commands..... | 158 |
| II-3.13 | Application visualization | 160 |
| II-3.13.1 | Start Mask-Designer..... | 160 |
| II-3.13.2 | Structure of mask editor | 160 |
| II-3.13.3 | Editing properties..... | 161 |
| II-3.13.4 | Adding resources (bitmap)..... | 164 |
| II-3.13.5 | Editing resources | 166 |
| II-3.13.6 | Creating a new mask..... | 167 |
| II-3.13.7 | Creating a new object..... | 168 |
| II-3.13.8 | Using interface blocks | 169 |
| II-3.13.9 | Properties of the interface blocks | 170 |
| II-3.13.10 | Calling operating pages..... | 171 |
| II-3.13.11 | Switching between masks | 172 |
| II-3.13.12 | Displaying general data..... | 172 |
| II-4 | Working with the Simulator | 174 |
| II-4.1 | Starting the simulator..... | 174 |
| II-4.2 | Adapting the language | 174 |
| II-4.3 | Establishing connection to simulator..... | 174 |
| II-4.4 | Debugging | 176 |
| II-4.4.1 | Value display with popup windows | 176 |
| II-4.4.2 | Value display with debug blocks..... | 176 |
| II-4.4.3 | Using debug blocks..... | 177 |
| II-5 | A practical project | 180 |
| II-5.1 | Step 1: Creating a project | 183 |
| II-5.2 | Step 2: Creating controllers | 185 |
| II-5.3 | Step 3: Creating a simulation..... | 191 |
| II-5.4 | Step 4: Specifying interfaces, connecting blocks..... | 195 |
| II-5.5 | Step 5: Defining parameters | 198 |
| II-5.6 | Step 6: Creating a visualization | 201 |
| II-5.7 | Step 7: An application test..... | 218 |
| II-5.8 | Step 8: Working with macros..... | 219 |
| III | Function blocks..... | 225 |
| III-1 | Scaling and calculating functions..... | 227 |
| III-1.1 | ABSV (Absolute value (No. 01))..... | 227 |
| III-1.2 | ADD (Addition/subtraction (No. 03))..... | 228 |
| III-1.3 | MUDI (Multiplication / division (No. 05)) | 230 |
| III-1.4 | SQRT (Square root function (No. 08))..... | 231 |

| | | |
|--------------|---|------------|
| III-1.5 | SCAL (Scaling (No. 09)) | 233 |
| III-1.6 | 10EXP (10s exponent (No. 10))..... | 235 |
| III-1.7 | EEXP (e-function (No. 11))..... | 235 |
| III-1.8 | LN (Natural logarithm (No. 12)) | 237 |
| III-1.9 | LG10 (10s logarithm (No. 13)) | 238 |
| III-2 | Non-linear functions..... | 240 |
| III-2.1 | LINEAR (Linearization function (No. 07)) | 240 |
| III-2.2 | GAP (Dead band (No. 20)) | 241 |
| III-2.3 | CHAR (Function generator (No. 21))..... | 243 |
| III-3 | Trigonometric functions..... | 245 |
| III-3.1 | SIN (Sinus function (No. 80))..... | 245 |
| III-3.2 | COS (Cosinus function (No. 81))..... | 246 |
| III-3.3 | TAN (Tangent function (No. 82))..... | 247 |
| III-3.4 | COT (Cotangent function (No. 83))..... | 248 |
| III-3.5 | ARCSIN (Arcus sinus function (No. 84)) | 249 |
| III-3.6 | ARCCOS (Arcus cosinus function (No. 85))..... | 250 |
| III-3.7 | ARCTAN (Arcus tangent function (No. 86)) | 252 |
| III-3.8 | ARCCOT (Arcus cotangent function (No. 87))..... | 253 |
| III-4 | Logic functions..... | 255 |
| III-4.1 | AND (AND gate (No. 60)) | 255 |
| III-4.2 | NOT (Inverter (No. 61)) | 256 |
| III-4.3 | OR (OR gate (No. 62)) | 257 |
| III-4.4 | BOUNCE (Debouncer (No. 63))..... | 258 |
| III-4.5 | EXOR (Exclusive OR gate (No. 64))..... | 260 |
| III-4.6 | FLIP (D flipflop (No. 65)) | 261 |
| III-4.7 | FLIPM (Multi D flipflop (No. 128))..... | 262 |
| III-4.8 | MONO (Monoflop (No. 66))..... | 264 |
| III-4.9 | STEP (Step function for sequencing (No. 68)) | 266 |
| III-4.10 | Delay_D (Timer (No. 69))..... | 268 |
| III-5 | Signal converters | 271 |
| III-5.1 | AOCTET (Data type conversion (No. 02)) | 271 |
| III-5.2 | ABIN (Analog <=> binary conversion (No. 71))..... | 273 |
| III-5.3 | TRUNC (Integer portion (No. 72))..... | 276 |
| III-5.4 | PULS (Analog pulse-conversion (No. 73))..... | 277 |
| III-5.5 | COUNT (Up/down counter (No. 74)) | 280 |
| III-5.6 | MEAN (Mean value formation (No. 75))..... | 283 |
| III-6 | Time functions..... | 287 |
| III-6.1 | LEAD (Differentiator (No. 50))..... | 287 |
| III-6.2 | INTEGRATE (Integrator (No. 51))..... | 290 |
| III-6.3 | FILTER (Filter (No. 52))..... | 293 |
| III-6.4 | DELAY (Delay time (No. 54)) | 294 |
| III-6.5 | BAND_FILT (Filter with tolerance band (No. 55))..... | 296 |
| III-6.6 | Timer (Timer (No. 67)) | 298 |
| III-6.7 | TIMER 2 (Timer (No. 70))..... | 300 |
| III-7 | Selecting and storage | 303 |
| III-7.1 | EXTREM (extreme value selection (No. 30))..... | 303 |
| III-7.2 | PEAK (peak value memory (No. 31)) | 305 |

| | | |
|---------------|--|------------|
| III-7.3 | HOLD (hold amplifier (No. 32)) | 307 |
| III-7.4 | SEL_C (Constant selection (No. 33)) | 308 |
| III-7.5 | SEL_D (selection of digital variables (No. 06)) | 309 |
| III-7.6 | SEL_P (parameter selection (No. 34)) | 311 |
| III-7.7 | SEL_OUT (selection of output (No. 36)) | 312 |
| III-7.8 | REZEPT (recipe management (No. 37)) | 313 |
| III-7.9 | ZOF3 (2-out-3-selection with mean value formation (No. 38)) | 316 |
| III-7.10 | SEL_V (cascadable selection of variables (No. 39)) | 318 |
| III-8 | Limit value signalling and limiting | 321 |
| III-8.1 | ALLP (alarm and limiting with fixed limits (No. 40)) | 321 |
| III-8.2 | ALLV (alarm and limiting with variable limits (No. 41)) | 324 |
| III-8.3 | EQUAL (comparison (No. 42)) | 327 |
| III-8.4 | VELO (rate-of-change limiting (No. 43)) | 329 |
| III-8.5 | LIMIT (multiple alarm (No. 44)) | 331 |
| III-8.6 | ALARM (alarm processing (No. 45)) | 333 |
| III-9 | Human-machine interface | 335 |
| III-9.1 | General information | 335 |
| III-9.2 | Bargraf (display of a value as the length of a bar (no. 401)) | 338 |
| III-9.3 | Bitmap (display of a graph (no. 406)) | 341 |
| III-9.4 | Button (button field (no. 402)) | 343 |
| III-9.5 | Rectangle (display or input field for value and text (no. 403)) | 345 |
| III-9.6 | Mask (mask field, grouping several HMI objects) | 349 |
| III-10 | Visualization | 353 |
| III-10.1 | TEXT (text container with language-dependent selection (No. 79)) | 353 |
| III-10.2 | V_BAR (bargraf-display (No. 97)) | 355 |
| III-10.3 | V_Display (display / definition of process values - (No. 96)) | 360 |
| III-10.4 | V_TREND (trend display (No. 99)) | 368 |
| III-10.5 | V_LOGGING (logging-function (No. 140)) | 374 |
| III-10.6 | V_LOGGING2 (scalable logging-function (No. 141)) / V_LOGGING2_D (connector for signals of V_LOGGING2 (No. 142)) | 377 |
| III-10.7 | V_ALARM (display of all alarms on one alarm-operating page (No. 109)) | 381 |
| III-10.8 | V_PARA (parameter operation (No. 98)) | 383 |
| III-11 | Communication | 387 |
| III-11.1 | L1READ_BOOL (Read function data bool (No. 132)) | 388 |
| III-11.2 | L1READ_INT (Read function data integer (No. 130)) | 388 |
| III-11.3 | L1READ_FLOAT (Read function data float (No. 100)) | 390 |
| III-11.4 | L1WRITE_BOOL (Write function data bool (No. 133)) | 391 |
| III-11.5 | L1WRITE_INT (Write function data integer (No. 131)) | 392 |
| III-11.6 | L1WRITE_FLOAT (Write function data float (No. 101)) | 393 |
| III-12 | Supplementary functions | 395 |
| III-12.1 | CALLPG (Function for calling up an operating page (No. 127)) | 395 |
| III-12.2 | SAFE (safety function (No. 94)) | 396 |
| III-12.3 | STATUS (Status function (No. 125)) | 397 |
| III-12.4 | INFO (Information function (No. 124)) | 398 |
| III-12.5 | PASSWORD (password function (Nr. 126)) | 399 |
| III-13 | General device functions | 402 |
| III-13.1 | GENERAL_EASY (General device functions (No. 200)) | 402 |
| III-13.2 | Using the "General Data" menu | 404 |

| | | |
|---------------|---|------------|
| III-13.2.1 | Configure network | 406 |
| III-14 | Programmer | 410 |
| III-14.1 | A_PROG (analog programmer (No. 24))/ A_PROG_D (APROG-Data (No. 26)) | 410 |
| III-14.2 | D_PROG (digital programmer (No. 27))/ D_PROG_D (D_PROG_data (No. 28)) | 429 |
| III-14.3 | PROGRAMMER (Universal programmer (No. 29)) | 436 |
| III-15 | Controller | 460 |
| III-15.1 | CONTROL (Controlfunction with one parameterset (No. 90)) | 461 |
| III-15.2 | CONTROLP (Controlfunction with six parametersets (No. 91)) | 481 |
| III-15.3 | PIDMA (Control function with particular self-tuning behaviour (No. 93)) | 500 |
| III-15.4 | Controller applications: | 520 |
| III-15.5 | Setpoint functions | 521 |
| III-15.6 | Process value calculation | 528 |
| III-15.7 | Correcting variable processing | 531 |
| III-15.8 | Small controller-ABC | 534 |
| III-15.9 | Control behaviour | 539 |
| III-16 | Process output | 552 |
| III-16.1 | OUT (process output (No. 116)) | 552 |
| III-17 | Vario I/O modules | 554 |
| III-17.1 | Short-form instructions for building up a vario I/O system: hardware | 555 |
| III-17.2 | Short-form instructions for building up a VARIO I/O system: HW configuration | 556 |
| III-17.3 | Short-form instructions for building up a VARIO I/O system: Software and debugging | 560 |
| III-17.4 | VARIO_BK_ETH (Vario ethernet buscoupler (No. 150)) | 561 |
| III-17.5 | AI_2_SF (I/O-module with 2 analog inputs (No. 165)) | 562 |
| III-17.6 | AI_8_SF (I/O-module with 8 analog inputs (No. 166)) | 564 |
| III-17.7 | AO_1_SF (I/O-module with 1 analog output (No. 155)) | 566 |
| III-17.8 | AO_2_U_BP (I/O-module with 2 analog outputs (No.156)) | 566 |
| III-17.9 | DI_2 (I/O-module with 2digital inputs (No.181)) | 567 |
| III-17.10 | DI_4 (I/O-module with 4 digital inputs (No.182)) | 568 |
| III-17.11 | DI_8 (I/O-module with 8 digital inputs (No.183)) | 569 |
| III-17.12 | DI_16 (I/O-module with 16 digital inputs (No.184)) | 569 |
| III-17.13 | DO_2 (I/O-module with 2 digital outputs (No.170)) | 570 |
| III-17.14 | DO_4 (I/O-module with 4 digital outputs (No.171)) | 571 |
| III-17.15 | DO_8 (I/O-module with 8 digital outputs (No.172)) | 571 |
| III-17.16 | DO_16 (I/O-module with 16 digital outputs (No.173)) | 572 |
| III-17.17 | DO_1_230 (I/O-module with 1 digital relay output (No.174)) | 573 |
| III-17.18 | DO_4_230 (I/O-module with 4 digital relay outputs (No.175)) | 573 |
| III-17.19 | RTD_2 (I/O-module with 2 analog inputs (No. 163)) | 574 |
| III-17.20 | RTD_6 (I/O-module with 6 analog in- and outputs (No. 169)) | 576 |
| III-17.21 | RTD_6_HC (I/O-module with 6 analog in- and outputs and heating current (No. 162)) | 578 |
| III-17.22 | UTH_2 (I/O-module with 2 analog inputs (No. 161)) | 582 |
| III-17.23 | UTH_4 (I/O-module with 4 analog in- and 8 digital outputs (No. 168)) | 585 |
| III-17.24 | UTH_4_HC (I/O-module with 4 analog in-, 8 digital outputs and heating current (No. 164)) | 588 |
| III-17.25 | UTH_8 (I/O-module with 8 analog in- and 8 digital outputs (No. 167)) | 593 |

| | | |
|---------------|---|------------|
| III-17.26 | UTH_8_HC (I/O-module with 8 analog in-, 8 digital outputs and heating current (No. 160))..... | 596 |
| III-18 | rail line I/O-Module..... | 602 |
| III-18.1 | General..... | 602 |
| III-18.2 | I/O - error coding | 602 |
| III-18.3 | Short-form instructions for building up a rail line RL 400 I/O system: Hardware..... | 603 |
| III-18.4 | Short-form instructions for building up a rail line RL 400 I/O system: Software and debugging | 605 |
| III-18.5 | Railline CI 45 (Universal Transmitter (No. 285))..... | 606 |
| III-18.6 | Railline KS 45 (Universal Controller (No. 286))..... | 608 |
| III-18.7 | Railline TB 45 (Temperaturelimiter/ -monitor (No. 287)) | 610 |
| III-18.8 | Railline SG45 (Universal transmitter (No. 288)) | 612 |
| III-18.9 | Railline BK CAN 10 (Buscoupler CAN (No. 280))..... | 614 |
| III-18.10 | Railline BK CAN 20 (Buscoupler CAN (No. 281))..... | 617 |
| III-18.11 | Railline BK CAN 40 (Buscoupler CAN (No. 282))..... | 620 |
| III-18.12 | Railline BK CAN 62 (Buscoupler CAN (No. 283))..... | 624 |
| III-18.13 | Railline 422 (Analog input-module (No. 260)) | 628 |
| III-18.14 | Railline 423-0 (Input module resistance-thermometer (No. 265)) | 634 |
| III-18.15 | Railline 423-1 (Input module resistance-thermometer (No. 266)) | 637 |
| III-18.16 | Railline 423-2 (Input module resistance-thermometer (No. 267)) | 640 |
| III-18.17 | Railline 424-0 (Input module for thermocouple TC (No.270)) | 643 |
| III-18.18 | Railline 424-1 (Input module with 2 thermocouple / O ₂ inputs (No. 271)) | 647 |
| III-18.19 | Railline 424-2 (Input module for thermocouple TC (No. 272)) | 652 |
| III-18.20 | Railline 431 (Analog output module (No. 262)) | 656 |
| III-18.21 | Railline 442 (Digital input module (No. 250)) | 659 |
| III-18.22 | Railline 443 (Digital input module AC (No. 251))..... | 661 |
| III-18.23 | Railline 451 (Digital output module (No. 255))..... | 663 |
| III-18.24 | Railline 452 (Relay module (No. 256))..... | 665 |
| III-18.25 | Railline 461 (Analog combination module (No. 261))..... | 667 |
| III-19 | Universal programmer..... | 674 |
| III-19.1 | General..... | 674 |
| III-19.2 | Prerequisites | 674 |
| III-20 | Installation and configuration | 675 |
| III-20.1 | BlueDesign programming environment | 675 |
| III-20.1.1 | Installing BlueDesign..... | 675 |
| III-20.1.2 | Licencing BlueDesign..... | 675 |
| III-20.1.3 | Configuring BlueDesign..... | 675 |
| III-20.1.4 | Installing the Vario configurator..... | 675 |
| III-20.2 | BlueEdit program editor | 675 |
| III-20.2.1 | Installing, configuring and licencing BlueEdit | 675 |
| III-21 | Components for program creation | 677 |
| III-21.1 | KS 108 easy with run-time environment | 677 |
| III-21.2 | BlueDesign..... | 677 |
| III-21.3 | PMA library | 678 |
| III-21.4 | BlueSimulation | 678 |
| III-21.5 | BlueEdit..... | 679 |
| III-22 | From the idea to the programm sequence in the device..... | 680 |
| III-22.1 | Preparation: Defining programs | 681 |

| | | |
|---------------|--|------------|
| III-22.1.1 | Basic program structure | 681 |
| III-22.2 | BlueDesign: Creating an engineering using the PROGRAMMER function block..... | 682 |
| III-22.3 | BlueEdit: recipes..... | 697 |
| III-22.4 | KS108 easy : Starting the programmer operation | 698 |
| III-22.4.1 | Loading the engineering..... | 698 |
| III-22.4.2 | Loading a recipe | 699 |
| III-22.4.3 | Starting the programmer | 700 |
| III-23 | Program management: BlueEdit..... | 711 |
| III-24 | Tutorial: A practical example of a "programmer project" | 712 |
| III-24.1 | Step 1: Creating a new project..... | 717 |
| III-24.2 | Step 2: Creating a controller..... | 719 |
| III-24.3 | Step 3: Creating a programmer | 723 |
| III-24.4 | Step 4: Creating the password management..... | 725 |
| III-24.5 | Step 5: Creating the simulation..... | 728 |
| III-24.6 | Step 6: Determining interfaces, connecting program blocks | 730 |
| III-24.7 | Step 7: Determining parameters | 733 |
| III-24.8 | Step 8: Generate a symbol file | 737 |
| III-24.9 | Step 9: Creating a recipe | 738 |
| III-24.10 | Step 10: An application test | 739 |
| IV | Index | 741 |

Preface

The user manual describes the installation, operation, and maintenance of the device and of the associated software.



HINT!

The device is not used "stand-alone" but always applied to control machines or industrial processes. Therefore this user manual is focused on system integrators and constructors of plants etc., rather than end users. For the process implementing the device an individual (user) documentation has to be written.



NOTE!

The device is not used as a standalone device, rather it is used to control or regulate machines or industrial processes. Consequently this engineering manual is not designed for the end user of the system, rather it is designed for the plant engineer etc. Separate (user) documentation must be created for the system.

Content

The engineering manual is divided to the following sections:

- **Operator's guide:** A description of the device, as well as instructions on mounting, commissioning, and operation are provided here. Specifically you will find information on the following:
 - Safety
 - Device structure and function, technical data
 - Information on mounting and commissioning the device
 - Instructions on maintenance and cleaning as well as malfunction resolution
 - The PMA library makes numerous standard operation pages available for the KS 108 easy. Here you learn how to operate controllers, profilers etc. In addition to that you become familiar with the standard menu of the device.
- **Development environment:** This section provides information on the development environment and how to work with it. Specifically you will find information on the following:
 - Installation of software
 - Connection set-up to KS 108 easy
 - Structure and use of the PMA library in BlueDesign
 - Practical example: Based on a practical example you will learn how PMA library, BlueDesign and KS 108 easy cooperate.
- **Function block reference:** Description of the function block library of KS108 easy.
 - Scaling and computing (LIB001)
 - Non-linear functions (LIB004)
 - Trigonometric functions (LIB011)
 - Logic functions (LIB009)
 - Signal processing (LIB002)
 - Timing functions (LIB008)
 - Selection and storage (LIB003)
 - Limits and limit values (LIB007)
 - Human machine interface (LIB401)
 - Display and operation (LIB010)
 - Communication (LIB014)
 - Supplementary functions (LIB013)
 - General device functions (LIB018)
 - Programmers (LIB006)
 - Controllers (LIB012)
 - Process output (LIB016)

- Vario I/O functions (LIB019)
 - rail line I/O functions (LIB020)
- **Programmer's environment:** This section provides information on the environment for the programmer "programmer" (program editor: BlueEdit) and how to work with it. Specifically you will find information on the following:
- Installation and configuration of software
 - Components for program creation
 - From the idea to the program sequence in the device
 - Program management: BlueEdit
 - Practical example: Based on a practical example you will learn how (program editor) BlueEdit, BlueDesign and KS 108 easy cooperate.

I Operator guide

I-1 General

I-1.1 Information on the operator's guide

Compliance with all safety instructions and handling instructions specified in the operator's guide is the prerequisite for safe work and proper handling of the device.

In addition, guidelines, standards, local accident prevention regulations, and safety regulations must be complied with for the implementation area of the device.

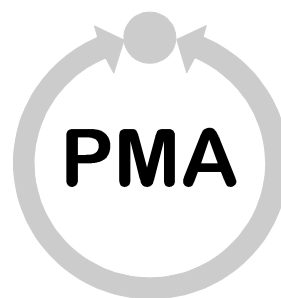
The operator's guide is a component of the product and must be kept accessible in the immediate vicinity of the device at all times for installation, operating, maintenance, and cleaning personal.

The graphic illustrations in this guide are used to present the explained subject matter and consequently are not necessarily shown to scale and can deviate slightly from the actual version of the device.

I-1.2 Manufacturer

PMA Prozeß- und Maschinen-Automation GmbH
Miramstr. 87
D-34123 Kassel

Tel. +49 0561 505-1307
Fax +49 0561 505-1710
e-mail: mailbox@pma-online.de



I-1.3 Guarantee conditions

The current guarantee provisions and information concerning the guarantee are in our general terms and conditions, for example on the Internet at (<http://www.pma-online.de/de/pdf/agbs.pdf>).

I-1.4 Customer service

Our customer service organization is available to provide technical information, see chapter "Manufacturer's address".

Moreover our employees are always interested in new information and experiences associated with the applications and which can be of value in improving our products.

I-1.5 Explanation of symbols

Warning instructions

Warning instructions are indicated in this operator guide by symbols. The instructions are introduced by signal words that express the scope of the hazard.

Strictly comply with the instructions and act with prudence to avoid accidents, personal injury and property damage.



DANGER!

...indicates an immediately dangerous situation that can cause death or serious injuries, if not avoided.



WARNING!

...indicates a possible dangerous situation that can cause death or serious injuries, if not avoided.



CAUTION!

...indicates a possible dangerous situation that can cause insignificant or minor injuries, if not avoided.



CAUTION!

...indicates a possible dangerous situation that can cause property damage, if not avoided.



CAUTION!

ESD-sensitive components!

... indicates a situation that can damage or destroy components through electrostatic discharge.

Tips and recommendations



NOTE!

... indicates tips and recommendations as well as information for efficient and malfunction-free operation.

I-1.6 Limitation of liability

All information and instructions in this guide have been compiled under due consideration of applicable standards and guidelines, the current state of the technology, as well as our extensive knowledge gained in years of experience.

Manufacturer accepts no liability for damages due to:

- Failure to heed the instructions in the guide
- Non-intended use
- Deployment of untrained personnel
- Unauthorized conversions
- Technical changes
- The use of non-approved replacement parts

The actual scope of delivery can deviate from the explanations and presentations provided in this guide in the case of special versions, if additional options are ordered, or due to the latest technical changes.

In all other cases the obligations set forth in the supply contract, manufacturer's terms and conditions, as well as the valid legal regulations at the time the contract was concluded, apply.



NOTE!

*This operator guide must be read carefully prior to starting any work on and with the device!
The manufacturer accepts no liability for damage and malfunctions that arise from failure to heed the instructions in the operator guide.*

I-1.7 Copyright

Treat the operator guide with confidentiality. It has been prepared exclusively for personnel working with the device. Supplying the operator guide to third parties without the written approval of the manufacturer is prohibited.



NOTE!

Content information, text, drawings, graphics, and other presentations are protected by copyright, and are subject to intellectual property rights. Any abusive utilization is punishable.

Duplication in any manner - including excerpts - as well as utilization and/or communication of the content are prohibited without manufacturer's written declaration. Offenders are liable for damages. The right to enforce additional claims remains reserved.

I-1.8 Transport, packaging, and storage

I-1.8.1 Transport

Keep the original packaging in order to ship the device in the original packaging if there is a guarantee case. Protect the device from severe mechanical stress in transport. Always transport the device in the original packaging. The installed components are sensitive to impact and strong vibration.

I-1.8.2 Unpacking

Proceed as follows to unpack the device:

- We recommend inspecting the device for transport damage immediately after it is delivered. Only acknowledge receipt with reservation (for example on the freight document), if there is reason to suspect damage. Note the presumed damage on the freight document and inform the manufacturer.

- Carefully remove the packaging.

It is a good idea to keep the packaging. It can be used for later necessary transport of the device, (for example if there is a device defect).

If there is no appropriate agreement concerning take-back of packaging material, then the packaging remains with the customer.



NOTE!

Cardboard and plastics (foil and foam) have been used as packaging material. If the packaging will be disposed of, then environmentally responsible disposal must be ensured in accordance with applicable disposal guidelines.

I-1.8.3 Storage



CAUTION! Condensation of water!

Condensation of water is possible due to temperature fluctuations. This can destroy the device or subsequent malfunctions can result.

Therefore:

- After storage or transport in cold weather, or if there are extreme temperature fluctuations, the device must slowly adjust to the ambient temperature at the implementation site before it is started up.
- If condensation has formed, then the device should only be placed in service after a waiting period of 12 hours.

The following guidelines apply for storage:

- Relative humidity: Max. 85 %
- Ensure that the packages are not stored outside
- Store in a dust-free environment
- Storage temperature -20 to +70 °C
- Avoid mechanical impact and damage

I-1.9 Disposal

If no return or disposal agreements have been made, then the professionally dismantled components must be recycled:

- Scrap metallic material remnants.
- Take plastic elements to plastic recycling facilities.
- Sort the other components for disposal, based on material condition.

**NOTE!**

Electrical scrap, electronic components, lubricants and other auxiliary substances are subject to guidelines for special waste and should only be disposed of by approved specialized companies.

I-2 Safety

This section provides an overview of all important safety aspects for optimal protection of personnel, as well as for safe and malfunction-free operation.

Significant dangers can arise if the handling instructions and safety instructions listed in this guide are not heeded.

I-2.1 General

The device and the associated software have been developed in accordance with the acknowledged rules of standard engineering practice, and they are operationally safe.

This device has been manufactured and tested in accordance with VDE 0411-1 / EN 61010-1, and it left the plant in perfect safety-related condition.

However this device can cause hazards if it is not used by properly trained personnel, or if it is used improperly, or if it is not used as intended.

- Every person who is assigned to perform work on or with the device must have read and understood the operator guide prior to starting work on the device. This also applies if the person concerned has already worked with such a device or with a similar device, or if he has been trained by the manufacturer.
- Knowledge of the content of the operator guide is one of the prerequisites to protect personnel from hazards, as well as for avoiding errors, and thus is a prerequisite for safe and malfunction-free device operation.
- To avoid danger and to assure optimal performance, neither changes nor conversions should be made to the device unless they have been expressly approved by the manufacturer.
- All safety warning signs and operating warning signs on the device must be maintained in legible condition. Damaged or signs or signs that have become illegible must be replaced immediately.
- The setting values or value ranges that are specified in the operator guide must be complied with.
- The owner is recommended to have personnel verifiably confirm that they have understood the content of the operator guide (see the section "Instruction" in this regard).

I-2.2 Owner's responsibility

The device is implemented commercially. Thus the owner of the device is subject to legal industrial safety obligations.

In addition to the industrial safety instructions in this operator guide, the safety, accident prevention, and environmental protection regulations, applicable at the site of implementation must be complied with. In particular:

- Owner must inform himself of applicable industrial safety regulations and determine additional hazards that arise due to the special work conditions prevailing at the site where the device is implemented, in a risk analysis. The risk analysis must be implemented in the form of work instructions for device operation.
- Owner must check throughout the entire implementation period of the device, whether the work instructions that owner has created satisfy current legislation, and must adapt them if necessary.
- Owner must clearly regulate and specify the responsibilities for installation, operation, maintenance, and cleaning.
- Owner must ensure that all employees who work with the device have read and understood the operator guide.

Moreover owner must train personnel and inform them of dangers at regular intervals.

In addition owner is responsible to ensure that the device is always in a technically perfect condition; therefore the following applies:

- Owner must ensure that the maintenance intervals described in this operator guide are complied with.
- Owner must have all safety devices inspected regularly for function and completeness.
- Owner must provide personnel with the required protective gear.

I-2.3 Operating personnel

I-2.3.1 Requirements



WARNING!

Danger of injury if personnel are not qualified!

Improper handling of the device can cause serious personal injury and property damage.

Therefore:

- Only have those persons who have been designated in the respective chapters of this operator guide perform special activities.
- If in doubt obtain the assistance of specialists.

In the operator's guide the following qualifications are cited for various activity areas:

■ **Instructed person**

has been instructed by owner in a training session concerning the assigned tasks and possible hazards in the event of improper behavior.

■ **Skilled personnel**

are personnel who, due to their specialized training, knowledge, and experience, as well as knowledge of applicable regulations, are capable of executing the tasks assigned to them and of recognizing possible hazards on their own.

■ **Qualified electrician**

is a person who due to his specialized training, knowledge, and experiences, as well as knowledge of applicable standards and regulations, is capable of performing work on electrical equipment, and who can recognize possible hazards on his own.

The certified electrician has been especially trained for the work environment where he is active and knows the relevant standards and regulations.

In Germany the qualified electrician must satisfy the provisions of the accident prevention regulation BGV A3 (for example certified master craftsman for electrical fittings). Similar regulations apply in other countries.

Only persons, from whom can be expected that they reliably perform their work, are approved as personnel. Persons whose reaction capability is influenced by drugs, alcohol, or medication are not approved.

Comply with the age-specific and job-specific regulations applicable at the implementation site.

I-2.3.2 Instruction

Personnel must be instructed regularly by operator. For better tracking training execution must be logged. Such a log can look like this:

| Date | Name | Type of instruction | Instruction performed by | Signature |
|------------|----------------|---------------------------------------|--------------------------|-----------|
| 04.04.2006 | Heinz Lehrling | First safety instruction for XY | Willi Meister | |
| 17.05.2006 | Horst Werker | Annual safety instruction 2006 for XY | Willi Meister | |
| ... | ... | ... | ... | |

Table 1: Instruction log

I-2.4 Safety integrated systems

Use of the device (as with all other PLC's) in safety integrated systems requires special measures. If safety integrated implementation is planned, then the user must take the applicable standards (e.g. DIN EN 61508) into consideration, and in addition should consult with PMA.

I-2.5 Special dangers

The following section lists the residual risks that arise due to the risk analysis.

Heed the safety instructions listed here, and the warning instructions in subsequent chapters of this guide, to reduce health hazards and dangerous situations.

I-2.5.1 Device failure, maintain, decommissioning



DANGER!

Danger of injury due to unforeseeable system function sequences and movement sequences!

System components can be placed in movement during maintenance work, configuration work, or function checks, if they are not disconnected from the device.

Therefore:

If the device is taken out of service, if new or changed applications are loaded on the device, or if maintenance or a function check is performed, the following must be heeded:

- All system components must be disconnected from the device!
- All switched off system components must be safeguarded from being inadvertently switched on again!
- In general the effects of switching off the system must be taken into consideration and appropriate measures must be taken.



DANGER!

Danger of injury due to uncontrolled/unforeseeable operating sequences!

As with any electronic controller system, device failure can result in an uncontrolled and/or unforeseeable operating sequence. Death, serious injury, or significant property damage can be the result.

Therefore:

- Ensure that appropriate measures are in place every time the device is used.

I-2.5.2 Explosion protection



Warning!

Explosion hazard due to live components!

In hazardous environments live devices can trigger explosions. The device does not have explosion protection.

Therefore:

- The device should not be used in hazardous atmospheres.

I-2.5.3 Electric components

**CAUTION!****Electrical hazard for components!**

Device components can be damaged or destroyed by electric voltages.

Consequently when working inside the device ensure the following:

- Disconnect the device from the power supply.
- Ensure that no objects or tools fall into the device.

I-2.5.4 Batteries

**WARNING!****Danger of injury if batteries are handled improperly!**

Batteries must be handled with particular care.

Therefore:

- Do not throw batteries into a fire or expose them to high temperatures. There is an explosion hazard.
- Do not charge batteries. There is an explosion hazard.
- Liquid that escapes from batteries, if they are used improperly, can cause skin irritations. Avoid contact with the liquid. If there is contact with the liquid rinse thoroughly. If the liquid gets into the eyes immediately rinse with water for 10 minutes and seek medical attention without delay.

I-2.5.5 Application development

**WARNING!!****Danger of injury or danger of property damage through unforeseeable program sequence!**

As with any electronic controller system, software errors can result in an uncontrolled and/or unforeseeable operating sequence. Death, serious injury, or significant property damage can be the result.

Therefore:

- Ensure that the system is only used after an extensive test (see EN 61131 in this regard).
- Virtually all function blocks should only be called once during a sampling step. If they are called multiple times then a new instance of the function block will not be created, rather there is only an additional reference to an existing function block (the existing instance). Under behavior is the result.

I-2.6 Environmental protection



CAUTION!

Incorrect handling causes environmental hazards!

Incorrect handling of environmentally harmful substances, particularly improper disposal, can cause significant damage for the environment.

Therefore:

- Always comply with the instructions listed below.
- Initiate suitable measures immediately if environmentally harmful substances inadvertently get into the environment. If in doubt inform the responsible municipal authorities about the damage

I-2.7 Intended use

The operational safety of the device is only ensured if the device is used as intended in accordance with the instructions in the operator's guide.

- The device is designed for use as a multi-function controller or for controlling and/or regulating machines and industrial processes. It should only be used within overvoltage category I (IEC 364-4-443), in low voltage systems where the rated supply voltage does not exceed 1000 V AC voltage (50/60 Hz), or 1500 V DC.
- Intended use includes correct compliance with the mounting, operating, maintenance, and cleaning instructions.
- Any use extending beyond intended use, or any other type of device utilization is prohibited and is considered as non-intended use! Claims of any type against the manufacturer and/or manufacturer's authorized agents for damages due to non-intended use of the device are excluded. The owner is solely liable for all damages due to non-intended use.

I-3 Structure and function, technical data

I-3.1 Device description

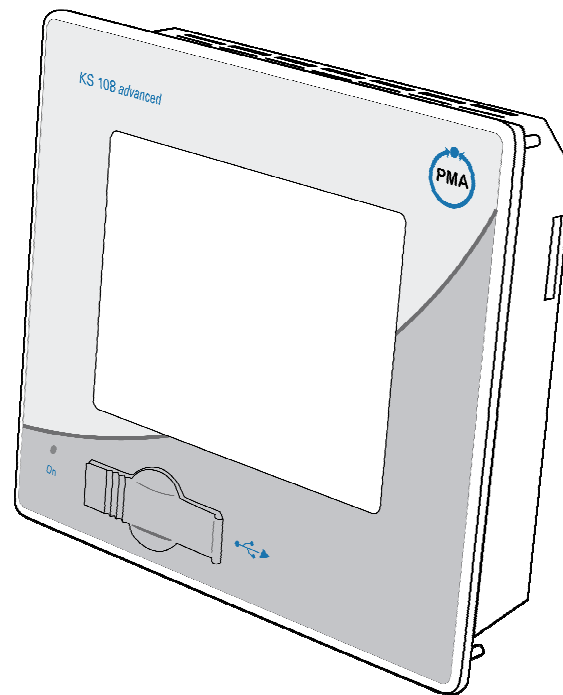


Fig. 1: Device description

The KS 108 is a control module with real-time capability with a touch-sensitive screen (touchscreen) and a broad spectrum of data interfaces. The device is programmed in accordance with the IEC 61131-3 standard with the BlueDesign development environment.

The KS 108 is designed for front panel mounting or control cabinet mounting in rough industrial environments. Maintenance requirements are minimal due to the fan-less design and flash memory.

I-3.2 Models

The device is available in the following models (future modules are indicated by gray font).

| Automation unit KS 108 | KS 108 - x x x - x x x x x - x xx | | | | | | |
|--|--|---|---|---|---|---|----|
| BASIC UNIT | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ |
| KS 108 easy | 1 | | | | | | |
| KS 108 flexible | 2 | | | | | | |
| KS 108 PLC | 3 | | | | | | |
| DISPLAY | | | | | | | |
| 5.7" CSTN, QVGA (320 x 240) | 0 | | | | | | |
| 5.7" TFT, QVGA (320 x 240) | 3 | | | | | | |
| Not assigned | 0 | | | | | | |
| FIELDBUS OPTION ¹⁾ | | | | | | | |
| On request | | | | 0 | | | |
| Additional interfaces (on request) | | | | 0 | | | |
| INSTALLED I/O | | | | | | | |
| None | | | | 0 | | | |
| ADDITIONAL INTERFACES ²⁾ | | | | | | | |
| On request | | | | | 0 | | |
| PMA FUNCTION LIBRARY | | | | | | | |
| only KS 108 Flexible | | | | | 0 | | |
| DEFAULT SETTING | | | | | | | |
| Standard setting | | | | | | 0 | |
| Setting according to specification | | | | | | 9 | |
| APPROVALS | | | | | | | |
| CE | | | | | | | 0 |
| CE, UL/cUL certified (applied for) | | | | | | | U |
| FRONT MEMBRANE | | | | | | | |
| PMA standard | | | | | | | 00 |
| Customer-specific front membrane (from 100 ST) | | | | | | | xx |

1) Communication protocol is configurable: (e.g. **MODBUS/TCP**, Ethernet/IP, ProfiNet, CANopen, DeviceNet, etc.)

2) Interfaces included in the basic unit: 1 x CAN (galv. separated), 1 x RS485 (galv. separated), 1 x RS232, 1 x Ethernet (galv. separated), 1 x USB (front), 1 x SD card. Additional interfaces could be: second CAN bus, etc.

I-3.3 Accessories

The following accessories can be ordered:

| Accessories | KS 108 easy | KS 108 Flexible | KS 108 PLC |
|--|-------------|-----------------|------------|
| BlueDeign (software development environment) | x | - | - |
| CoDeSys (software development environment) | - | x | x |
| Ethernet switch (3 to 5x, for top-hat rail) | x | x | x |
| Engineering manual | x | x | x |
| Remote I/O components | x | x | x |

I-3.4 Performance features - overview

- Freescale POWERPC™ CPU / 266 MHz
- 5.7" TFT display, with touchscreen, 320 X 240 pixels or
5.7" CSTN display, with touchscreen, 320 X 240 pixels
- User program memory and data memory (RAM): 64 MB / 32 MB for application
- User program memory (flash): 16 MB / 8 MB for application
- Retain memory 16 KB
- 1 Ethernet 10/100 interface
- 1 USB host interface
- 1 CAN interface
- 1 serial interface - RS 232 for programming tools
- 1 serial interface - RS 485
- Real-time clock
- MMC-/SD card slot
- optional:
 - Three extension slots for I/Os
 - One communication module (Profibus)
 - Second CAN interface
 - Second serial interface

I-3.5 Device overview

- 1 Display (touch sensitive)
- 2 Mode display
- 3 Cover USB interface
- 4 MMC-/SD card slot
- 5 Stud bolts

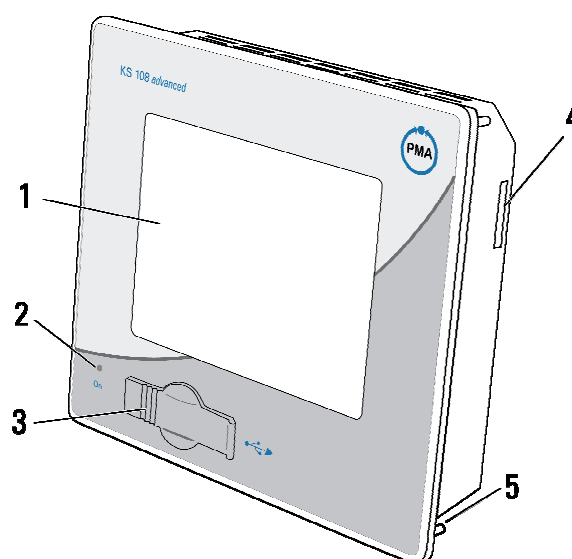


Fig. 2: Device overview (front)

I-3.6 Technical data

| Feature | Value | |
|--------------------------------------|---|------------|
| Display | | |
| Type | CSTN, color | TFT, color |
| Service life CCFL (backlit display) | 40,000 h | 60 000 h |
| Diagonal | 5,7" | |
| Resolution | 320 x 240 pixels (1/4 VGA) | |
| Colors | 256 (8-bit per pixel) | |
| Touchscreen | resistive | |
| CPU, application memory | | |
| CPU | Freescale PowerPC 266 Mhz | |
| Program memory (flash) | 16 MB / 8 MB for applications | |
| Program memory and data memory (RAM) | 64 MB / 32 MB for applications | |
| Retain memory | 16 KB | |
| Real-time clock | Yes, battery buffered (SONY CR1620) | |
| Size and weight | | |
| Dimensions (W x H x T) | 194 mm x 172 mm x 50 mm (incl. Front panel) Mounting depth 70.0 mm | |
| Weight | Approx. 1.5 kg | |

| Feature | Value |
|---|--|
| Operating conditions | |
| Ambient temperature | 0° C to 50° C |
| Relative humidity | Max. 85% non-condensing |
| Resistance to vibration | |
| Vibration | Sinusoidal (EN 60068-2-6) Test: Fc 10 ... 150 Hz, 1 G |
| Shock resistance | 15 G (approximately 150 m/s ²), 10 ms duration, half sine (EN 60068-2-27) test: Ea |
| EMC, protection class | |
| Noise radiation | EN 61000-6-4, EN 61326-1, industrial |
| Interference immunity | EN 61000-6-2, EN 61326-1, industrial |
| General | |
| Safety | EN 61010-1 (VDE 0411-1), EN 61131-2: Overvoltage category: II Contamination level: 2 Protection class: III (safety extra-low voltage) |
| Dielectric strength | EN 61131-2, DC 500 V test voltage |
| Protection class | Front IP 65, rear IP 20 |
| CE marking | Satisfies the EMC and low voltage directive |
| Energy supply (24 V power pack unit) | |
| Supply voltage | +24 V DC (18 V ... 32 V) SELV Maximum residual ripple 4 V _{ss} |
| Power consumption | typ. 1.0 A, max. 2.0 A at +24 VDC Fusing depending on load of the possibly present I/O max. 12A |
| Reverse polarity protection | Yes |
| Galvanic isolation | Yes |
| Ethernet interface | |
| Number/ type of interface | 1 x 10/100 Base T |
| Connection | RJ45 |
| Potential isolation | Yes |
| USB interface | |
| Number/ type of interface | 1 x host USB 1.1, () |
| CAN bus interface | |
| Number/ type of interface | 1 x standard CAN ISO 11898 |
| Galvanic isolation | Yes |
| Transmission rate | Max. 1Mbit/s |
| Terminating resistor | Reversible |

| Feature | Value |
|----------------------------|--|
| Serial interface | |
| Number/ type of interfaces | 1 x RS232 1 x RS485 |
| Galvanic isolation | For RS 485 |
| Terminating resistor | Reversible for RS485 |
| SD card slot | |
| SD cards recommended | 256 MB (Kingston/SanDisk Extreme) 512 MB (Kingston) |
| Software | |
| Operating system | LINUX |
| Runtime system | CoDeSys ¹ or BlueDesign ² |
| Development environment | CoDeSys ¹ or BlueDesign ² |

1) KS 108 flexible and plc

2) KS 108 easy

I-3.7 Type plate

- 1 Product line
- 2 Model / order no.
- 3 Identification number
- 4 Hardware version
- 5 Supply voltage
- 6 **Serial number:** ID no.: 00041, date: 0626
- 7 Barcode
- 8 CE marking
- 9 **Production data:** Year: 06, CW: 26
- 10 Corporate logo



Fig. 3: Type plate KS 108 flexible

I-3.8 Block diagram

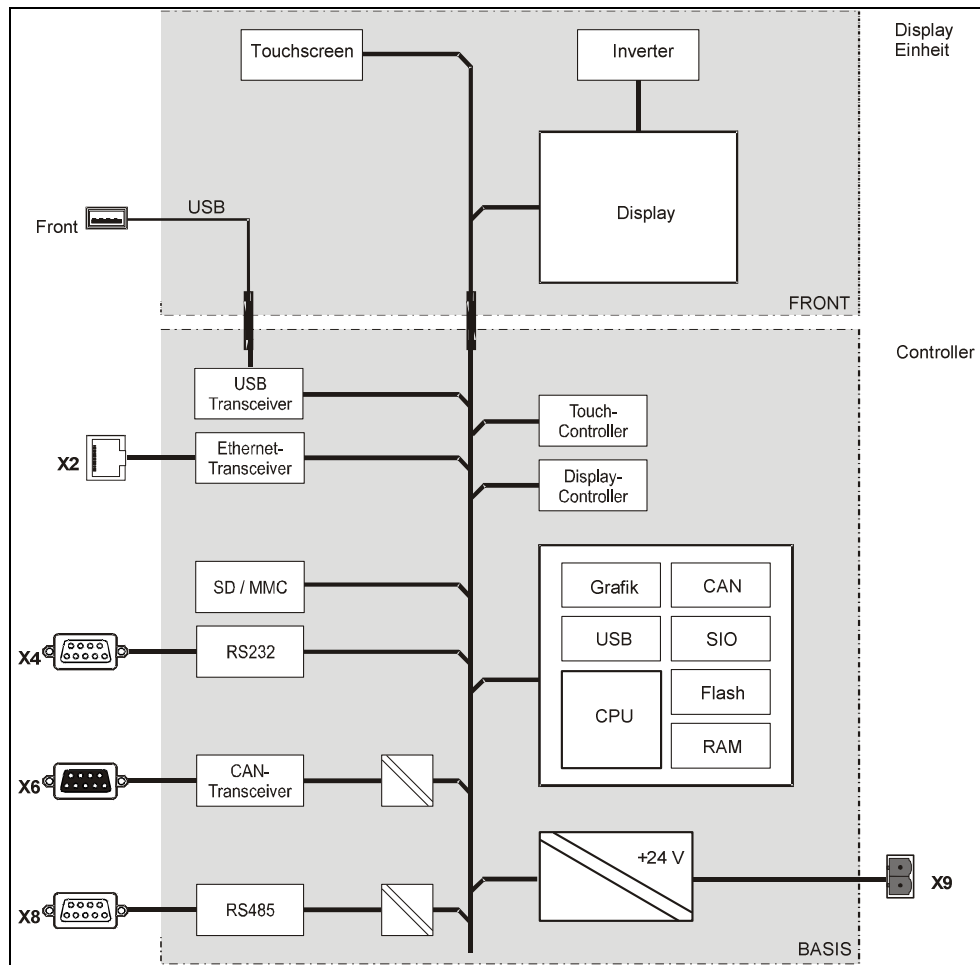


Fig. 4: Block diagram

I-3.9 Interfaces

I-3.9.1 Overview

- 1 Status LED - supply voltage
- 2 Mode selector switch
- 3 Status LED - program status
- 4 Network connection (Ethernet)
- 5 Serial interface (RS 232)
- 6 Switch - CAN BUS terminating resistor
- 7 CAN interface
- 8 Switch - serial interface terminating resistor
- 9 Serial interface (RS 485)
- 10 Network connection

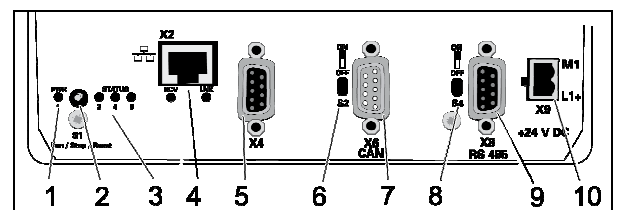


Fig. 5: Interface overview

I-3.9.2 Power supply

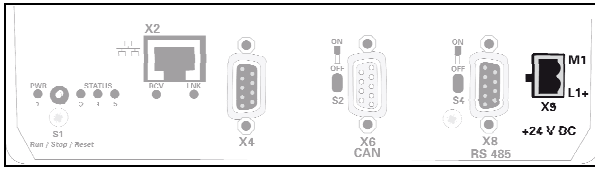


Fig. 6: Power supply

The device will provide with 24 V DC (18 V to 32 V). In addition it has integrated reverse polarity protection and inrush current limitation.

Pinning:

| Overview | Feature | Description |
|----------|---------|---|
| | L1+ | External power supply 24 V DC (18 V ... (32 V). |
| | M1 | External power supply GND |

I-3.9.3 10/100 Base-T network connection (Ethernet)

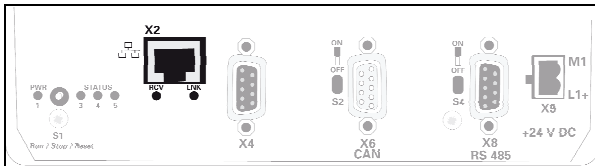


Fig. 7: Ethernet interface (RJ45)

Pinning:

| Overview | Feature | Description |
|------------|---------|--------------------------|
| | 1 | TX+ |
| | 2 | TX- |
| | 3 | RX+ |
| | 4 | 75 Ohm |
| | 5 | 75 Ohm |
| | 6 | RX- |
| | 7 | 75 Ohm |
| | 8 | 75 Ohm |
| Connection | | RJ-45 |
| LED "LNK" | Green | ON – ready for operation |
| LED "RCV" | Green | FLASHING - data receive |

I-3.9.4 USB

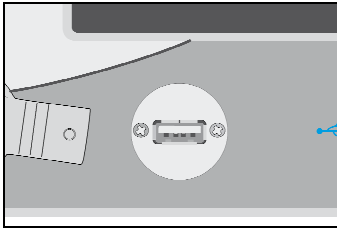


Fig. 8: USB interface

The USB connection is on the front of the device.

Pinning:

| Feature | Description |
|---------|-------------|
| B1 | VCC |
| B2 | D- |
| B3 | D+ |
| B4 | GND |



CAUTION!

Damage to USB devices is possible!

A max. amperage of 0.5 A is available on the USB port. Devices that require more current will not function under some circumstances and can be damaged.

Therefore:

- Only use devices that do not require more than 0.5 A.

I-3.9.5 CAN bus

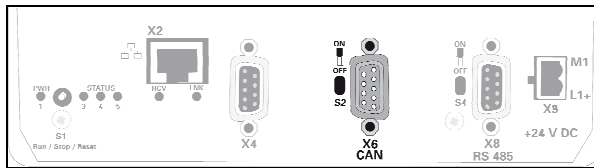


Fig. 9: CAN interface

The CAN interface corresponds to the ISO 11898 standard and can be operated to a maximum baud rate of 1Mbit/s. The interface has an additional isolating element.



NOTE!

A terminating resistor must be located at the beginning and end of a CAN bus topology. Therefore: If the device is at the beginning or end of the CAN bus topology then the terminating resistor must be activated with switch (Fig. 642/6) for the CAN bus. To do this move the switch up to the "ON" position.

Pinning:

| PIN | Description |
|-----|---------------------|
| 1 | NC (do not connect) |
| 2 | CAN_L |
| 3 | CAN_GND |
| 4 | NC (do not connect) |
| 7 | CAN_H |

I-3.9.6 Serial interfaces

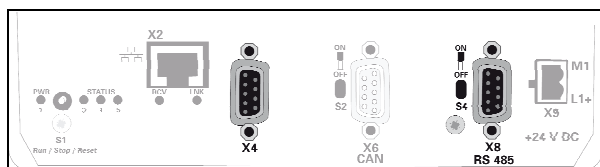


Fig. 10: Serial interface

The device has two serial interfaces:

- RS 232 [X4 / X5: COM1 / COM2] (Fig. 10/5)
- RS 485 [X8 : COM3] (Fig. 10/9): This interface is optically isolated.

RS232 pin assignment:

| PIN | Description |
|------|---------------------|
| 1 | NC (do not connect) |
| 2 | RXD |
| 3 | TXD |
| 4 | NC (do not connect) |
| 5 | GND |
| 6 -9 | NC (do not connect) |

RS485 pin assignment:

| PIN | Description |
|------|---------------------|
| 1 | RTXD- |
| 2-3 | NC (do not connect) |
| 4 | RTXD+ |
| 5 | GND |
| 6 -9 | NC (do not connect) |

I-3.10 Declaration of conformity



NOTE!

Currently the KS 108 is still in the verification process. A CE marking is sought for device operation in industrial as well as in residential applications.

The KS 108 is oriented to DIN 61131, conformity is sought but currently is not present.

I-4 Mounting and commissioning



DANGER!

Danger due to improper installation and commissioning!

Installation and commissioning require trained, specialized personnel with adequate experience. Installation errors can result in life threatening situations or cause significant property damage.

Therefore:

- Only have installation and commissioning performed exclusively by expert employees.
- An installation test must always be performed before startup.
- All switched off plant components must be safeguarded from being inadvertently activated!

I-4.1 Scope of delivery

Before mounting check the scope of supply for completeness:

- Device
- Connecting plug for power supply
- Operator's guide

I-4.2 Mounting

I-4.2.1 Mounting cutout KS 108

The device is designed for front mounting.



NOTE!

The material thickness of the bearing material should not exceed 6 mm. Otherwise the device cannot be (reliably) fastened with the stud bolts.

The mounting cutout must have the following format (all specifications are in mm):

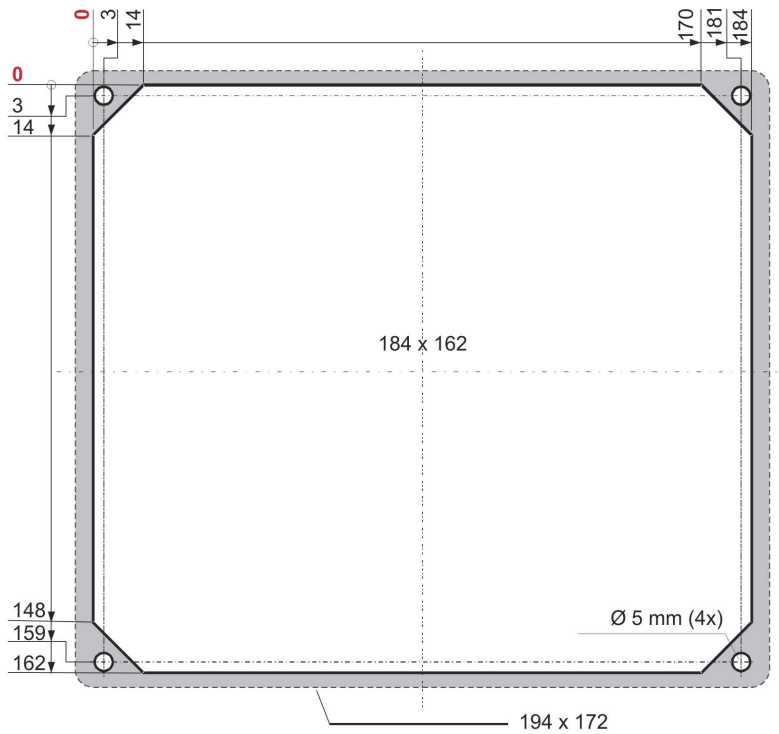


Fig. 11: Mounting cutout

- The device is fastened with four stud bolts (Fig. 11). Four bores with a diameter of 5 mm are provided for the stud bolts.
- The front frame (shown in the illustration above in gray) has dimensions of 194 x 172 mm, thus it will have an overhang relative to the mounting cutout.

I-4.2.2 Mounting execute



NOTE!

When mounting ensure that

- There is at least 20 mm clearance to the nearest device, or to the nearest wall.
- There is at least 50 mm clearance between the rear of the device and the wall for power and interface cables. The entire mounting depth is 100 mm.

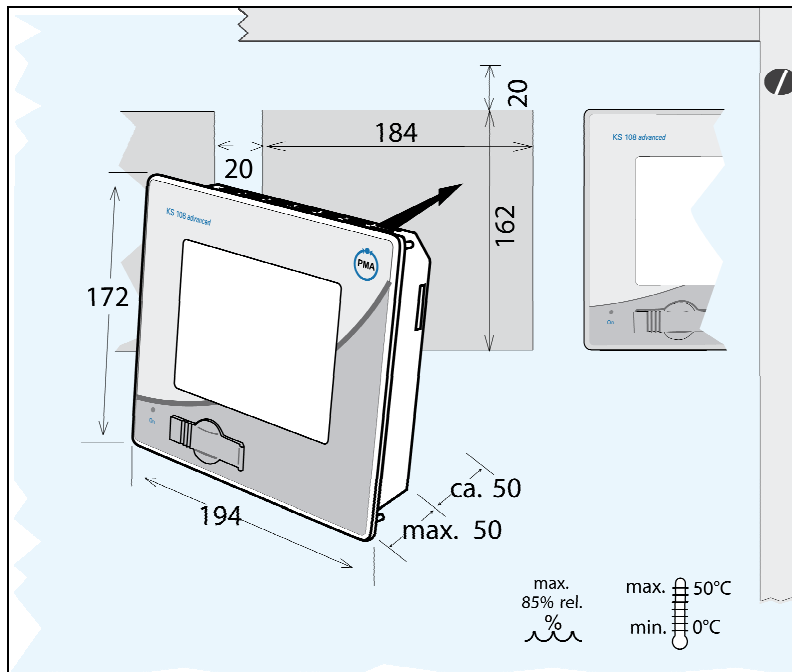


Fig. 12: Mounting

Mount the device as follows:

1. **Remove the fastening nuts and washers:** Remove the fastening nuts and washers from the four stud bolts.
2. **Slide the device into the mounding cutout:** Carefully slide the device into the mounting cutout from the front.
3. **Fix the device in place:** Fasten the device in the mounting cutout. Place a washer on each of the stud bolts. Now mount the nuts on the stud bolts and carefully tighten them.

I-4.2.3 Connecting the device



DANGER!

Injury hazard due to unforeseeable system function sequences and movement sequences

System parts can be placed in motion when mounting, if they are not appropriately secured.

Therefore:

If the device is mounted the following must be heeded:

- All system parts that are switched off must be safeguarded from being switched on inadvertently!
- In general, the effects of switching off the system must be considered, and appropriate measures must be taken.

Connecting the PE

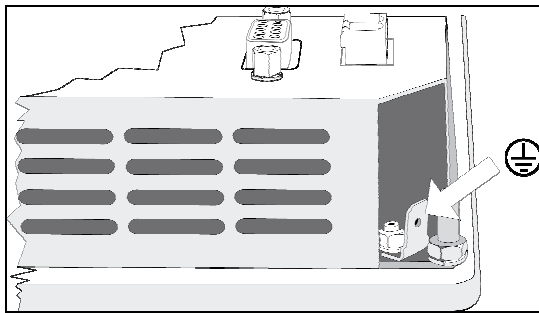


Fig. 13: PE

1. **Connect PE:** Connect the tab of the PE connection (see Fig. 13) to the protective earth. To do this use a line with a core cross-section of at least Cu 1.5 mm².

Connecting the power supply

The KS 108 requires an external power supply with the following specifications:

| External power supply specifications for the KS 108 | |
|---|---|
| Output voltage | +24 VDC SELV (18 ... 32 VDC) |
| Residual ripple | Max. 4 Vss The DC voltage level should not exceed 18 V |
| Power output | Max. 2.0 A at +24 V DC at 25 °C |



CAUTION!

Danger of damaging the device!

Connecting to the wrong power supply can damage the device.

Therefore:

- Ensure that the specifications for the power supply are complied with.
- All cables and connections must be executed in such a manner that malfunctions cannot be caused by inductive and capacitive interference.
- The feed lines must have sufficient current-carrying capacity and withstand voltage.
- Ensure that polarity of the power supply is not reversed.

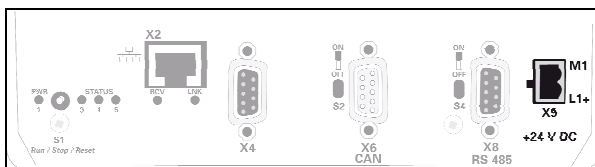


Fig. 14: Power supply

1. **Connecting the power supply:** Connect the power line using the supplied connector with the X9 interface on the rear of the device (see Fig. 14/10).

Connecting the interfaces



CAUTION!

Danger of malfunction or damage to the device!

Incorrect configuration/use of the interfaces can result in malfunctions or damage to devices!

Therefore:

- Comply with the instructions concerning the interfaces provided in the chapter "Interfaces"!
- For USB devices: Ensure that only those types of USB devices are used that require a current of no more than 0.5 A.
- For CAN bus topologies: Ensure that a terminating resistor is installed at the end of a CAN bus topology.

1. **Connect the interfaces:** You can connect the interfaces (Ethernet, CAN, etc.) with off-the-shelf cable, ensure that there is adequate shielding and grounding.

I-4.3 Commissioning



DANGER!

Danger of injury if operating personnel are not qualified to work on the device!

Improper handling of the device can cause serious personal injury and property damage.

Therefore:

- The device should only be placed in service by specialized personnel.
Specialized personnel are considered to be persons who have adequate knowledge of the device, the automated processes, and the equipment.

I-4.3.1 Checking the installation

The installation must be checked before the device is placed in service. This is done based on an installation log that must be created for the respective application case.

Here all installation tasks must be checked, however at least the following:

- Protective grounding
- Fuses for the power supply cable and for the master switch
- Agreement of device specification and PLC programming with the actual operating conditions.
- Agreement of the application with local and national guidelines

I-4.3.2 Switching on the device

Proceed as follows to switch on the device.

1. **Switch on:** Switch on the device using the power switch.
After switching on the start screen will be displayed after a short while.
Then you will either see the service menu or you will see the application that will be executed.



NOTE!

Additional information on the operating status is provided in the following section.



DANGER!

Danger of injury due to unforeseeable plant function sequences and movement sequences!

If the device is used with other devices/equipment then consequential actions can be induced on these devices/actuators, etc. by switching on the device.

Therefore:

- Prior to switching on the device at any time consider the effects of switching on the device and ensure that appropriate measures have been taken!



DANGER!

Danger of injury due to an incorrect or defective program!

The device is freely programmable. Signal processing and signal output, and thus the behavior of connected devices/equipment are determined by the loaded program

Therefore:

- Prior to starting up the device it is strictly necessary to ensure that the correct program is loaded.
- Only place programs in service if their error-free status has been ensured.

I-4.3.3 Loading the program from USB stick



DANGER!

Danger of injury due to unforeseeable system function sequences and movement sequences!

System components can be placed in movement during maintenance work, configuration work, or function checks, if they are not disconnected from the device.

Therefore:

If the device is taken out of service, if new or changed applications are loaded on the device, or if maintenance or a function check is performed, the following must be heeded:

- All system components must be disconnected from the device!
- All switched off system components must be safeguarded from being inadvertently switched on again!
- In general the effects of switching off the system must be taken into consideration and appropriate measures must be taken.

New application programs (firmware updates) can be transferred to the KS 108 via USB stick. To do this, proceed as follows:

1. **Copy to USB stick:** Normally you will receive updates via e-mail. Save the attachment to the USB stick. Ensure in this process that a directory with the name "autoinst" has been created on the topmost directory level. Ensure that the files "update.tgz" and "autoinst.ini" are located in this directory.

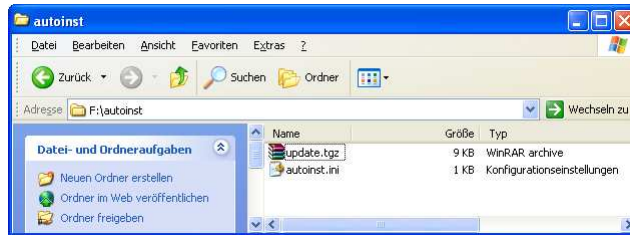


Fig. 15: Update to USB stick

**NOTE!**

An update can only be executed if the USB stick contains the directory structure and files described above!

2. **Switch off the KS 108:** Switch off the KS 108 using the power switch.
3. **Insert the USB stick:** Carefully lift the cover of the USB connection (Fig. 16/1) on the front of the device and turn it to the side. Now insert the USB stick (Fig. 16/2) in the connection (Fig. 16/3).

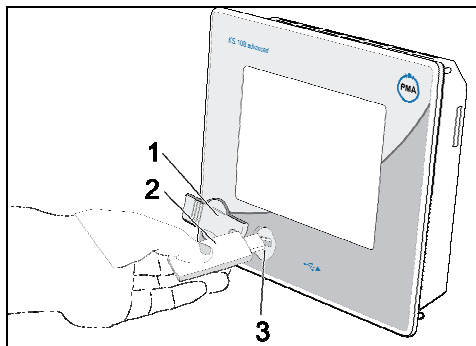


Fig. 16: Using the USB stick

4. **Switch on the device:** Now switch on the device again, The update starts automatically. First you will see the PMA logo and then a terminal window with status messages about the update. After the update the message "USB stick can be unplugged" will be displayed. The device will now restart automatically.

**NOTE!**

The update is executed based on a script. Depending on script execution the messages can be different. It is also possible that a different message will be displayed when the update is concluded.

5. **Remove the USB stick:** Remove the USB stick.

**NOTE!**

If the device does not restart automatically after the update, then it must be restarted manually. To do this switch the device off using the power switch and then switch it on again.

**DANGER!****Danger of injury due to unforeseeable plant function sequences and movement sequences!**

If the device is used with other devices/equipment then consequential actions can be induced on these devices/actuators, etc. by switching on the device.

Therefore:

- Prior to switching on the device at any time consider the effects of switching on the device and ensure that appropriate measures have been taken!

I-4.3.4 Configure network settings

Prior to commissioning you must configure the network settings of the device.

**NOTE!**

Additional information on configuring the network settings is available in the chapter "Menu - using 'General Data'".

I-4.3.5 Using an SD card

Applications for data logger, error memory or historic alarms require an SD card to store their data. Proceed as follows to install the SD card:

**CAUTION!****Danger of data loss!**

Removing the SD card while the device is in operation can cause data loss. The file structure of the SD card, as well as device data structure can be damaged.

Therefore:

- Always switch the device off first, and then remove the card!

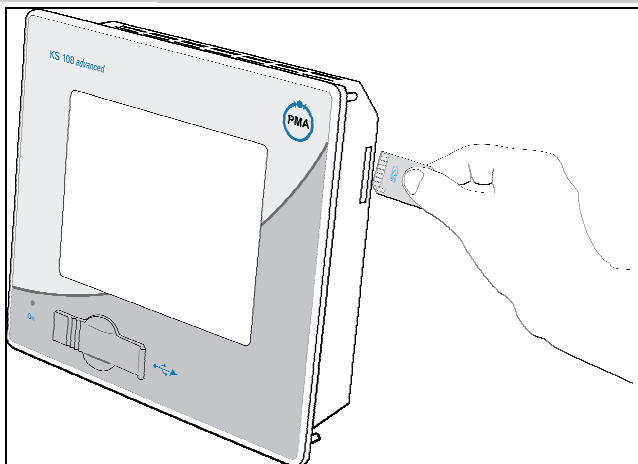


Fig. 17: Using the SD card

Insert card

1. **Switch off device:** Switch off the device.
2. **Insert card:** Carefully insert the SD card into the card slot. When inserting ensure that the beveled side is facing down and that the contacts are pointing to the front.

Removing the card

1. **Switch off device:** Switch off the device.

- 2. Remove card:** First push the card that you want to remove carefully into the device until the card is unlocked. Now let go of the card. If the card has been unlocked it will be pushed several millimeters out of the card slot.
You may now remove the card.

I-4.3.6 External I/O connection

The device offers a CANopen conformant interface and an Ethernet interface for connecting IO systems, sensors, and actuators.



NOTE!

The configuration of the external peripherals differs depending on the device used. The description is in the engineering manual.

I-5 Operation

The *KS 108 easy* is a powerful and flexible multi-functional unit. The device is adapted to the respective application via the *BlueDesign* development environment.

Structure and number of user pages depend exclusively on the programming - in other words they depend on the application. However normally most control elements of the applications are made available through predefined operating pages of the PMA library (e.g. the operating pages for controllers or program generators).

Below you will learn how you can work with these predefined operating pages. In this regard you should note the following: The precise structure of the operating pages depends on the respective application, thus for you the number and type of operating pages, parameters, etc. will be different.

Information on the following topics is provided in this chapter:

- Menu
 - Using the menu
 - Changing parameters and configurations
 - Using the "General Data" menu
- Operating pages
 - Operating pages overview
 - Bargraph "V_BAR"
 - Datalogger "V_LOGGING"
 - Trend "V_TREND"
 - Controllers
 - Digital program generator "D_PROG"
 - Analog program generator "D_PROG"
 - Alarm page "V_ALARM"
 - User display "V_DISPLAY"



DANGER!

Danger of injury or danger of property damage due to a defective program sequence!

Depending on the application numerous configurations or parameters can be changed via operating pages, or direct intervention in regulating or controlling processes are possible. Faulty intervention in applications can result in an uncontrolled and/or unforeseeable operating sequence (as with any electronic controller system). Death, serious injury, or significant property damage can be the result.

Therefore:

- Prior to any action consider the effects of the action and ensure that appropriate measures have been taken.

I-5.1 User operating page

The user operating page is the start page from which the user can branch to other operating pages. The page is application-specific and thus must be completely designed by application developers. The user operating page can include graphics (for instance an overview graphic of the process).

An example of a simple user operating page is provided in the chapter, "A practical example"; for (simplified) control of a three-chamber oven the following control elements are provided here:

- 1 Display of the process value (PV) for each of the three oven chambers
- 2 Symbolic presentation of the control process
- 3 Display of the setpoint (SP) for each of the three oven chambers
- 4 Button: Call main menu
- 5 Button: Call operating pages for controller

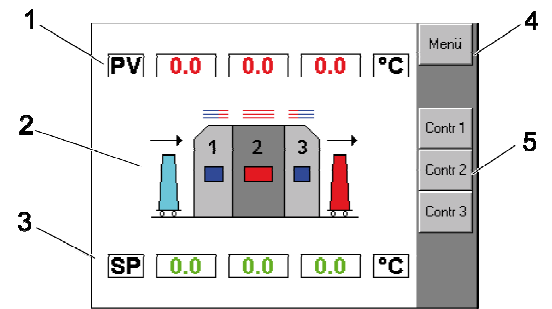


Fig. 18: Example project - three-chamber oven, main operating page

I-5.2 Main menu

Application programs that are created using the PMA library normally have a main menu with the following structure:

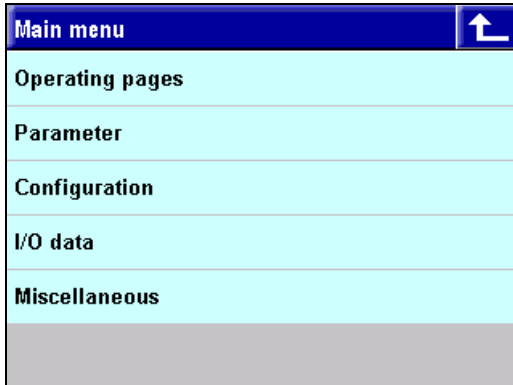


Fig. 19: Main menu

- **Operating Pages:** Calls the PMA library operating pages
- **Parameters:** Calls the PMA library parameter pages.
- **Configuration:** Calls the PMA library configuration pages
- **I/O data:** Calls the I/O pages from the PMA library.
- **General Data:** Calls the general device settings (this menu option is deactivated as default)

I-5.3 Using the menu

With the exception of the menu command *General Data* all menu commands first call a list of function blocks for which the respective option (for instance operating pages) is available.

Consequently proceed as follows to select a page (example: You would like to select the I/O data of a function block):

1. **Call the main menu:** Call the main menu. How this occurs depends on the application program.

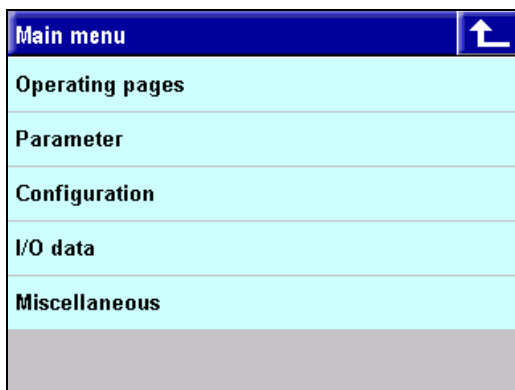


Fig. 20: Example using the menu (main menu)

2. **Call menu *I/O data*:** In the main menu call the submenu *I/O data*.

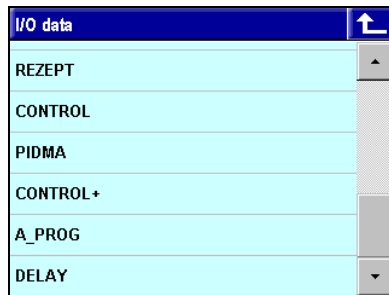


Fig. 21: Example using the menu (I/O data)

- Select function block:** Select the function block for which you want to see the I/O data (in the example: *Delay*).

Now you see the I/O display of the selected function block.

| DELAY - I/O data | |
|------------------|-------|
| ai_X_1 | 0 |
| ai_Preset | 0 |
| di_reset | FALSE |
| di_preset | FALSE |
| ao_Y_1 | 0 |

Fig. 22: Example of menu use (display I/O data)

- Return to main menu:** Tap on the "Back" button (↶) to return to the higher level menu.

I-5.4 Navigating in the menu

With the exception of the main menu you will find buttons and a scroll bar on the right side of the screen. Use these buttons to navigate in the menu or to return to the higher level menu.

The page is structured as follows:

- Back:** Return to the higher level menu.
- Up:** Browse one entry up.
- Browse:** Browse down or up.
- Slider:** Slide the controller to browse up or down.
- Down:** Browse down.

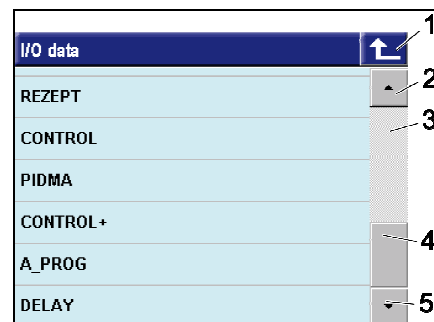


Fig. 23: Using the menu (navigation)

I-5.5 Changing parameters and configurations

You can easily change parameters and configurations via the menu.

To do this, proceed as follows:

- Select submenu:** In the main menu select either the submenu *Parameters* or *Configuration*.
- Select function block:** Now select the function block for which you want to change parameters or the configuration.
- Select value:** Click on the value you want to change (for example "pi_p behavior").

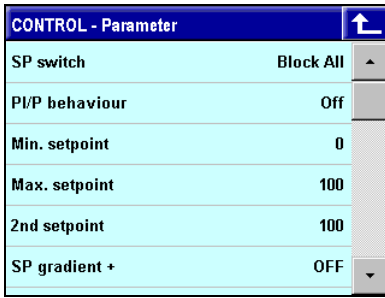


Fig. 24: Changing parameters/configuration

Depending on the selected value you will now see either a list selection field or an editor for entering numerical values.

Under some circumstance entries will be displayed in gray in the selection list. These are entries that are generally possible as an option but have been blocked as part of the current application.

I-5.5.1 Using the list selection editor

Many parameters are input via selection lists. For example the parameter "pi_p behavior" is specified via the following selection list:



Fig. 25: List selection editor

- **Making the selection:** Click on the desired value in the selection list to select it. The selected value will be transferred and the selection list will close.
- **Leaving the list without making a selection:** Click on the "⬆" button to leave the list without making a selection.

Structure of the selection list

Each list entry is comprised of a numerical index value (for instance "0") and a descriptive text (such as, "Switch off"). Background: Non-numeric parameters are managed numerically in the PMA library via index values that represent the desired parameter.

I-5.5.2 Using the numerical value editor

Numerical values (temperature specifications, times, etc.) are entered via a numerical value editor. The current value is displayed in the title line of the numerical value editor (in this case "Grw+ = OFF").

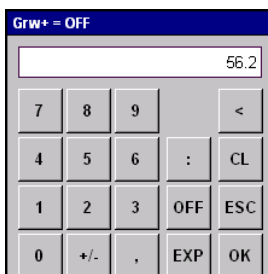


Fig. 26: Numerical value editor

| Button | Use |
|--------|----------------------------|
| 0 - 9 | Entering numerical values. |

| | |
|--|--|
| | Change leading signs. |
| | Enter decimal or comma. |
| | Enter colon. Colons are necessary for time specifications for example. |
| | Select switch-off value. "OFF" appears in the display. This value can only be selected if it is possible in principle for the current setting. |
| | Select exponents. Tap on the "Exponent" button, and then enter the exponents. An "e" will appear in front of the entry. |
| | Delete last character "Backspace" key). |
| | Delete entire entry. |
| | Leaving the text editor without updating the entry. |
| | Accept input and leave the editor. |

The possibility of using the buttons is context dependent. Thus the colon can only be used for time specifications, e.g. it is deactivated for numerical specifications.



NOTE!

If an invalid value is entered in the numerical value editor then it will be corrected without further feedback when the entry is accepted (when you click on the "OK" button)! In this case the nearest valid (limit) value will be used.

I-5.5.3 Using the binary number editor

Binary numbers (e.g. forced values for the digital program generator) are entered with an editor. Binary numbers are changed by bit. Consequently tap on the respective bit (or field) to change a value from 0 to 1 (or vice versa).

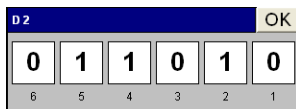


Fig. 27: Numerical value editor

I-5.5.4 Using the text editor

Text (file names, comments, etc.) are entered via a text editor. As is the case with the numerical value editor, the characters that will be entered are entered via buttons.

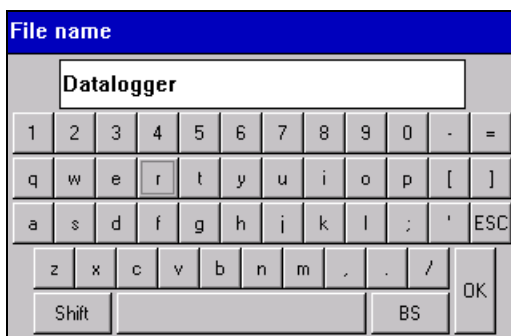


Fig. 28: Text editor

| Button | Use |
|--------|-----|
|--------|-----|

| | |
|--|---|
| | Alphanumeric value input |
| | Shift upper case/lower case |
| | Delete last character. |
| | Leaving the text editor without updating the entry. |
| | Accept input and leave the editor. |

NOTE!
The maximum length of a text field is limited and is usually 16 characters. If an entry is longer than the prescribed length, the text will be corrected without further feedback when the entry is accepted (when you click on the "OK" button)!

I-5.6 Using the "General Data" menu

In the "General Data" menu you can make fundamental system settings. For example you can set the date, the time, or the IP address.

NOTE!
The structure of the "General Data" menu and the possibility to activate the menu depend on the application development. You can only access the "General Data" menu or access the sub-menu commands, if these possibilities have been provided by the application technician. This means: Only if function block GENERAL_EASY is in the engineering the menu "Miscellaneous" is provided.

The following submenu commands are available as maximum:

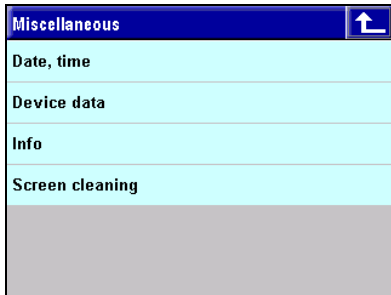


Fig. 29: Menu "General Data"

- **Date, Time** Set the system time
- **Device Data:** Select the language used by the PMA library. German and English can be selected.
- **Info:** Information on the software release version. This information can be useful when requesting service.
- **Clean screen:** An empty screen is displayed so that commands are not executed when the screen is being cleaned.

Date, Time

Set the system date in the *Date, Time* menu.

| Date, time | |
|------------|---------|
| Year | 0 |
| Month | January |
| Day | 0 |
| Hour | 0 |
| Minute | 0 |

Fig. 30: Menu "Date, Time"

- Tap on a menu option to change a setting. Use the numerical value editor or the list selection editor (for month) to make the selection.

| Date, Year = 2007 | |
|-------------------|----------------|
| Year | 2007 |
| Month | 7 8 9 < |
| Day | 4 5 6 : CL |
| Hour | 1 2 3 OFF ESC |
| Minute | 0 +/- , EXP OK |

| Date, Month | |
|-------------|--------------|
| Year 1: | January 2007 |
| Month 2: | February May |
| Day 3: | March 4 |
| Hour 4: | April 13 |
| Minute 5: | May 49 |
| 6: | June |

Fig. 31: Year and month selection

Device Data

In the *Device Data* menu configure the network and change device settings (language and screen settings)

| Device data | |
|-------------|--------------|
| Language | English |
| IP address | 85.86.52.18 |
| NetMask | 120.86.52.18 |
| Gateway | 120.86.52.18 |
| LinkMode | 100BaseTx-FD |
| DHCP Mode | DHCP Enabled |

Fig. 32: Menu "Device Data"

- **Language:** Tap on the menu option *Language* to change the device language. Input is executed with the list selection editor.
- **Brightness/contrast:** Tap on the menu options "Brightness" or "Contrast" to change the screen settings. Input is executed with the numerical value editor.



NOTE!

Information on changing the network settings is provided in section "**Error! Reference source not found.** Configure network".

Info

In the *Info* menu you will find information about the software release version that can be useful when requesting service. A software release differing in error corrections and / or added functions from the prior release is marked with a different version number. Mind these version numbers when transferring an engineering and therefor also when doing software updates.




Fig. 33: Menu "Info"

- **Op-Version:** Version of the PMA library on the *KS 108*.
- **FW-Version:** Version of the firmware.
- **HW-Code:** Hardware code of the device. The hardware code is a unique device designation.

I-5.6.1 Configure network

If you are using an Ethernet connection for communication with the *KS 108* then you must configure network settings. To do this, proceed as follows:

 **NOTE!**
 Contact your system administrator to determine the type of your network or the type of network connection of the *KS 108*.
 Only make changes to the network configuration if you are aware of all necessary network parameters. An incorrect device network setting suppresses device communication and can cause general network malfunctions (e.g. double assignment of IP-Addresses).

Enter IP-Address

With the device you can either assign a permanent IP-Address, or you can have an address assigned dynamically via a DHCP server. Additional information on the "DHCP" option is available below. A fixed IP-Address is assigned as follows:

1. **Call the "General Data" menu:** Call the submenu "General Data".
2. **Call the "General Data" menu:** Call the "Device Data" submenu.
3. **Select the "IP-Address menu command:** Tap on the menu command "IP-Address". The current IP-Address is displayed on the right side of the button.
4. **Enter the IP-Address:** Enter the new IP-Address with the numerical value editor. Tap on the "OK " button to save your entry.

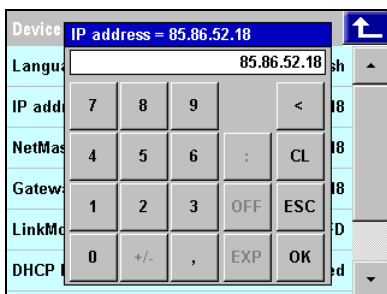


Fig. 34: Enter IP-Address

Enter the network mask

1. **Call the "General Data" menu:** Call the submenu "General Data".
2. **Call the "General Data" menu:** Call the "Device Data" submenu.

3. **Select the "NetMask" menu command:** Tap on the menu command "NetMask". The current IP-Address is displayed on the right side of the button.
4. **Enter the netmask:** Enter the new netmask with the numerical value editor. Tap on the "OK " button to save your entry.

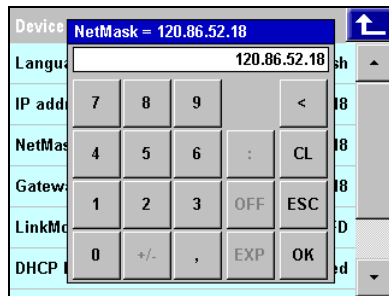


Fig. 35: Enter NetMask

Enter the gateway IP-Address:

1. **Call the "General Data" menu:** Call the submenu "General Data".
2. **Call the "General Data" menu:** Call the "Device Data" submenu.
3. **Select the "Gateway" menu command:** Tap on the "Gateway" menu command. The current gateway is displayed on the right side of the button.
4. **NetMask input:** Enter the new gateway with the numerical value editor. Tap on the "OK " button to save your entry.

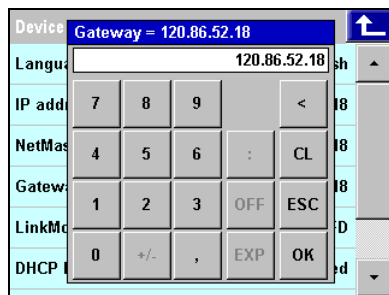


Fig. 36: Input gateway

Configuring the network connection type

Use the "LinkMode" setting to specify the Ethernet standard that your network works with. Normally you can use the "Auto" setting. However if problems should occur you can also explicitly specify the type of network connection.

The following options are available:

| Option | Explanation |
|---------------|--|
| Auto | The communication parameters will be determined automatically. |
| 100base-Tx FD | 100 Mbit/s, full duplex |
| 100base-Tx HD | 100 Mbit/s, half duplex |
| 10base-T FD | 10 Mbit/s, full duplex |
| 10base-T HD | 10 Mbit/s, half duplex |

Proceed as follows to configure the network connection type:

1. **Call the "General Data" menu:** Call the submenu "General Data".
2. **Call the "General Data" menu:** Call the "Device Data" submenu.

3. **Select the "LinkMode" menu command:** Tap on the "LinkMode" menu command. The current network connection type is displayed on the right side of the button.
4. **Select LinkMode:** Select the new network connection type in the list selection editor.

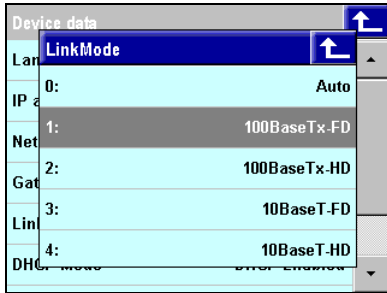


Fig. 37: Specify LinkMode

Configure DHCP mode

If the *KS 108* will not be working with a fixed IP-Address, but rather with an IP-Address that is dynamically assigned by a DHCP server, then the steps below are necessary:

1. **Call the "General Data" menu:** Call the submenu "General Data".
2. **Call the "General Data" menu:** Call the "Device Data" submenu.
3. **Select the "DhcpMode" menu command:** Tap on the "DhcpMode" menu command. The current status is displayed on the right side of the button.
4. **Select DhcpMode:** Select the desired mode in the list selection editor.

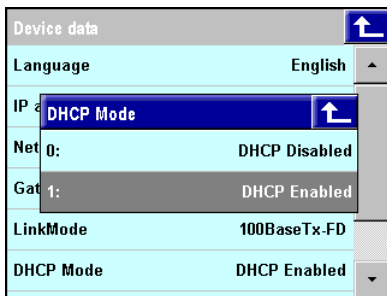



Fig. 38: Specify DhcpMode

I-5.7 Operating pages

Operating pages are made available through the PMA library. They provide information about the current status of the program and enable (as the name indicates) operation of the application program. Application development determines exclusively which operating pages can be found in an application program.

The following section offers an overview of the types of operating pages that are available.

 **NOTE!**
 General information only is provided here for understanding and using the operating pages. Fundamental information on the PMA library is available in the chapter "Function library".

I-5.7.1 Operating page V_ALARM

The alarm page provides an overview of alarm messages. There are two different types of alarm messages:

- **Alarms with acknowledgement:** These are alarms that must be confirmed by the user.
- **Alarms without acknowledgement:** These alarms must be confirmed by the user by clicking on a button.

Alarms are displayed as long as they are active. All alarms and all status changes of an alarm are stored in a history file.

Properties

- **Alarm overview:** The dialog offers a tabular overview of all alarm messages.
- **Alarms acknowledge:** Alarm messages can be acknowledged.
- **Save the alarm history:** The alarm history can be copied to a USB stick.
- **Delete alarm history:** The alarm history can be deleted.



NOTE!

Additional information on the "V_alarm" operating page is available in the function block reference.

Overview

- 1 Title
- 2 Button "Leave Operating Page"
- 3 Button "Service"
- 4 Non-acknowledged active alarm (red) that must be acknowledged.
- 5 Active alarm (red) that does not need to be acknowledged, or that has already been acknowledged.
- 6 Alarm (black) that is no longer active which must be acknowledged.

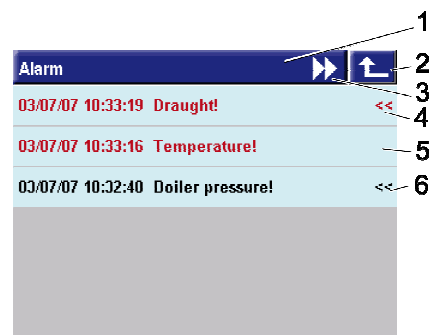


Fig. 39: Alarm page



NOTE!

The characters "<<" indicate alarms that must be acknowledged.

Operation

Call alarm page

The alarm page can be called from all operating pages via the "Alarm" button. If an alarm is not active then this triangle is gray, an active alarm is signaled by a flashing red warning triangle. The alarms are displayed with a plant-specific text, in the sequence of their occurrence.

- **Call:** Click on the "Alarm" button (Fig. 40/1) to call the alarm page.

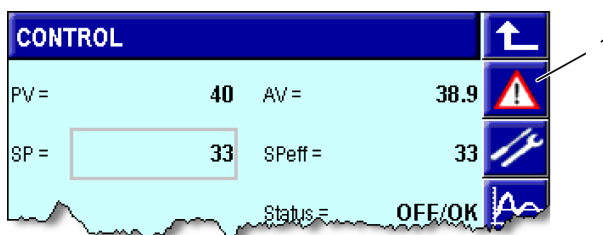




Fig. 40: Active alarm

 **NOTE!**
 If alarm messages are present then you will see a red flashing warning triangle () on the "Alarm" button. Otherwise the triangle is gray.

Confirm alarm

- **Acknowledge an alarm:** If the alarm must be acknowledged then the message "Alarm Quit" is displayed. Tap on the "OK" button to acknowledge the alarm.

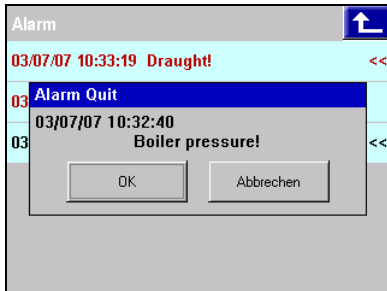


Fig. 41: Confirm alarm

Delete alarm history

Alarms and alarm status changes are stored on the SD card of the KS 108. On the operating page "Service Alarm Files" you will see information on the storage media (total memory and free memory) and you can copy or delete the alarm data.

1. **Call "Service-alarm" operating page:** Tap on the button "Service" (Fig. "Alarm page"/3). Now you will see the operating page "Service - alarm files".
2. **Delete alarms:** Tap on the "Delete" button if you want to delete the alarm messages on the SD card.

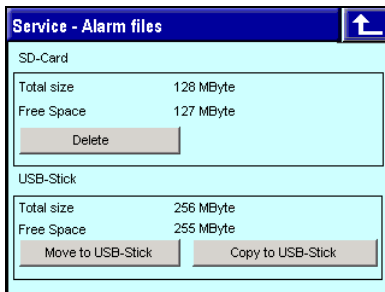



Fig. 42: Alarm page (dialog "Service – Alarm Files")

Alarm history - copy/move alarms

On the alarm page the current alarms and the alarms that must be acknowledged are displayed. If you want to see the alarm messages in chronological sequence, you must copy the messages to a USB stick and open the file in *Microsoft Excel* for example.

To do this, proceed as follows:

 **CAUTION!**
Danger of data loss!
 Removal of the USB stick during the copy process can result in data loss. The file structure of the USB stick, as well as device data structure can be damaged.
 Therefore:
 – Only pull the USB stick out if it has been unmounted (when the display of the remaining storage space goes out).

1. **"Call "Service-alarm" operating page:** Tap on the button "Service". Now you will see the operating page "Service - alarm files".
2. **Insert the USB stick:** Carefully lift the cover of the USB connection (Fig. 43/1) on the front of the device and turn it to the side. Now insert the USB stick (Fig. 43/2) in the connection (Fig. 43/3).

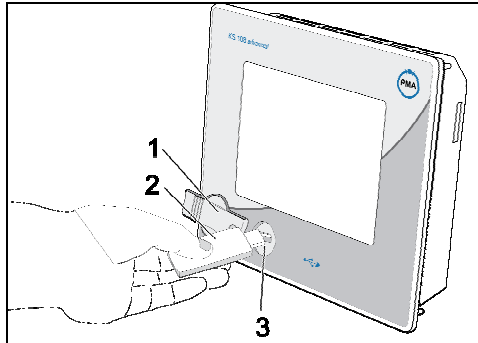


Fig. 43: Using the USB stick

The USB stick should be automatically detected, the remaining storage space on the stick will be displayed after a few seconds. The buttons for saving to the USB stick are displayed after the storage medium has been integrated.

3. **Start copy/move:** Tap on the "Move to USB stick" button if you want to move data to the USB stick. The data on the SD card will be deleted in this process.
Tap on the "Copy to USB stick" button if you want to copy data to the USB stick. The data are stored on the SD card.



NOTE!

After copying the USB stick is unmounted. Consequently the display of the remaining storage space on the USB stick goes out. The buttons for saving to the USB stick are hidden. If you want to copy additional data to the stick, you must remove the stick and then re-insert it in the USB connection.

Alarm history files are CSV files (CSV: "Comma Separated Values") and have the extension "*.alm". The file on the USB stick contains a time stamp so that files that may already be on the USB stick are not overwritten. The copied file can be edited for example with *Microsoft Excel*. The maximum file size of the alarm file on the SD card is determined exclusively by the available storage space on the SD card.

I-5.7.2 Operating page V_DISPLAY

The "V_DISPLAY" operating page can display up to six analog or digital values in six lines. Optionally the values can also be edited.



NOTE!

The precise design of the operating page depends exclusively on the application environment. Thus below you will find only an overview of operating elements that can be used.

Properties

The operating page can contain the following elements:

- Display fields (analog, digital, time, text)
- Input fields (analog, digital, time, text)
- Buttons (buttons, switches, menu)
- Radio buttons
- Any number of screen pages

Inputs can only be made for elements for which this function has been enabled by the application developer.



NOTE!

Additional information for the user display is available in the function block reference in the chapter "V_Display".

Overview and operation

- 1 Title
- 2 Switch "Toggle": Switch over one logical value with one click.
- 3 Display "Toggle":
- 4 Value display
- 5 Input field: One click on the field opens the numerical value editor.
- 6 Button "Leave Operating Page"
- 7 Button "Alarm"
- 8 Button "Call Operating Page"
- 9 Button "Previous page": The previous screen page will be called.
- 10 Button "Next page": The next screen page will be called.
- 11 Radio buttons: Select a value with a single click on the desired value. Only one element can be selected.

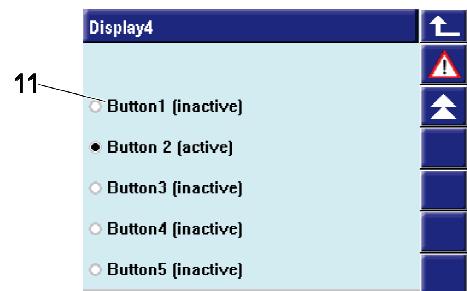
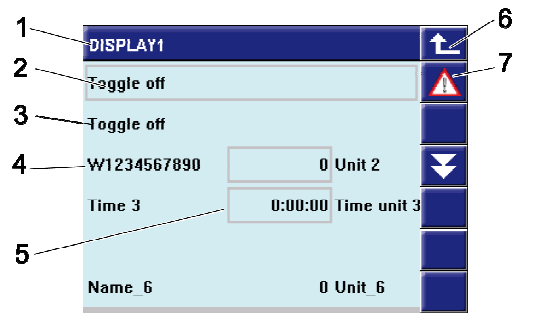


Fig. 44: User display

I-5.7.3 Operating page V_BAR

Use the "V_BAR" operating page to visualize two analog values as a bargraph.

Properties

- **Bar graph display:** Two analog values can be displayed as a vertical or horizontal bargraph. If a range is underranged then an arrow appears at the beginning or end of the bargraph (◀ or ▶).
- **Numerical value display:** Two additional analog values can be displayed as numerical value and changed if necessary. These numerical values can be identical to the values visualized in the bargraphs, however this is not mandatory.
- **Marker:** With four additional values (analog inputs) it is possible to define two markers for each bargraph. These markers are displayed in red on the bargraph for emphasis.

 **NOTE!**
Additional information on the "Bargraph" operating page is available in the function block reference.

Overview

- 1 Title
- 2 Name of the value displayed in the bargraph
- 3 Scale value of the bargraph
- 4 Origin of the bargraph
- 5 Unit of the displayed value
- 6 Bargraph
- 7 Button "Leave Operating Page"
- 8 Button "Alarm"
- 9 Display and input field for the value
- 10 Limit value marks for the bargraph

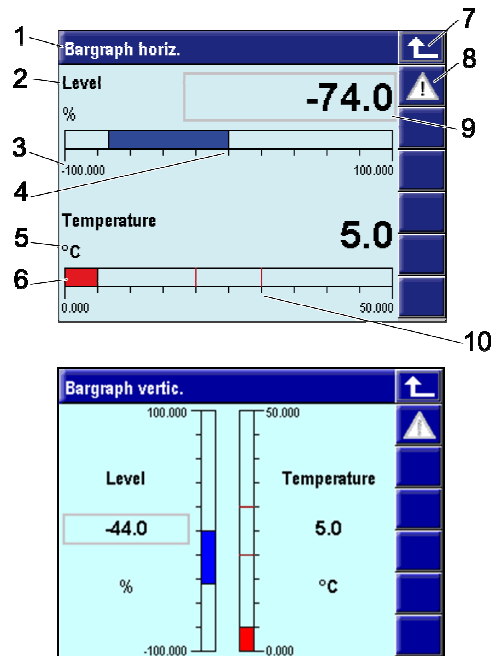


Fig. 45: Bargraph (horizontal and vertical)

Operation

Entering values

If this has been provided by the developer of the application program then values can be entered in the display and input field (see Fig. "Bargraph"/9). Note: If value input is possible then the field will have a frame. Tap the field to enter a value.

I-5.7.4 Operating page V_LOGGING / V_LOGGING2

With the "V_LOGGING" or "V_LOGGING2" operating page the trend of analog and digital values can be recorded.

Properties

- **Logging:** With function block "V_LOGGING" 12 digital tracks and 6 analog tracks can be recorded. With "V_LOGGING2" there is any extend of tracks available. The scan time is specified by the application developer.
- **Data storage on SD card:** The data are stored on an SD card. The maximum file size as well as the number (100 maximum) of files used for data storage are specified here by the application developer. After the last file has been described up to the maximum file size, the system will start again with the first file. The files are numbered consecutively. The number is appended to the file name.
- **Copy:** The logging files can be copied to a USB stick.



NOTE!

Additional information on the "V_LOGGING" or "V_LOGGING2" operating page is available in the function block reference.

Overview

- 1 Title
- 2 Name of the output file
- 3 Current size of the output file
- 4 Remaining free storage space on the SD card
- 5 Remaining free storage space on the USB stick (this is only displayed if a USB stick is connected to the device)
- 6 Button "Leave Operating Page"
- 7 Button "Alarm"
- 8 Call button "Start/stop recording". Depending on the application development this button may not be visible.
- 9 Button "Copy"
- 10 Status display ("run": Datalogger is running, "off": Data logger stopped)
- 11 Input field "File name"
- 12 Input field "Header"
- 13 Status message

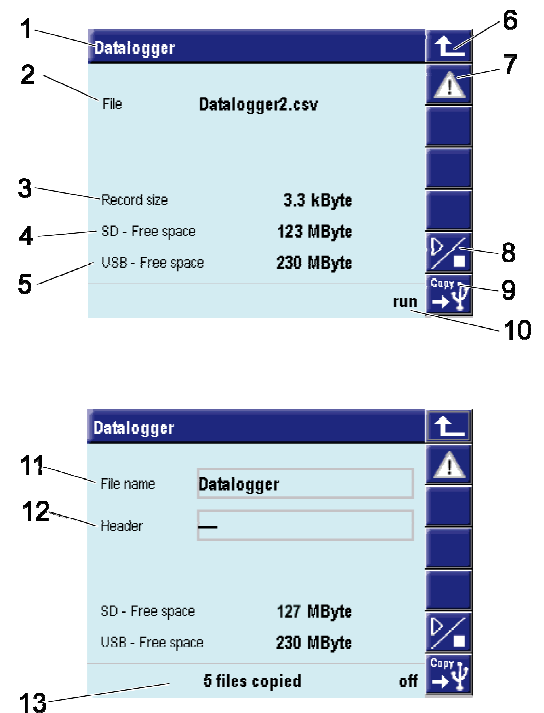


Fig. 46: Data logger (overview, input file name, copy)

Operation

Start recording

If the recording is not yet running (the status display shows the value "off") you can start the recording as follows:

1. **Start recording:** Tap on the button "Start/stop recording" (Fig. "Data logger"/8), to start the recording.
2. **Specify file name/comment:** Tap on the fields "File name" or "Header" to enter the file name of the header/comment. The comment appears as the first line of the output file. Use the "Text input" dialog to make the input.

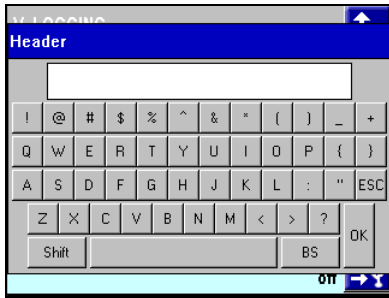


Fig. 47: Entering a header

Ending the recording

If the recording is running (the status display shows the value "run") you can stop the recording as follows:

1. **Ending the recording:** Click on the button "Start/stop recording" (Fig. "Data logger"/8).

Exporting a file

You can copy the files to a USB stick. The file name of the copied file has a time stamp e.g. "V_LOGGING_20070420_123439.csv". This ensures that an old file is not overwritten by a new one on the USB stick.

The log file is a CSV file (CSV: "Comma Separated Values"). The file can be edited, for example with *Microsoft Excel*.

Proceed as follows to copy a file:



CAUTION!
Danger of data loss!

Removal of the USB stick during the copy process can result in data loss. The file structure of the USB stick, as well as device data structure can be damaged.

Therefore:

- Only pull the USB stick out if it has been unmounted (when the display of the remaining storage space goes out).

1. **Inserting the USB stick:** Carefully lift the cover of the USB connection (Fig. 48/1) on the front of the device and turn it to the side. Now insert the USB stick (Fig. 48/2) in the connection (Fig. 48/3).

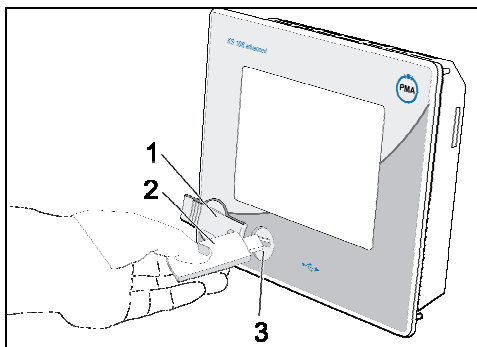


Fig. 48: Using the USB stick

The USB stick should be automatically detected, the remaining storage space on the stick will be displayed after a few seconds.

2. **Start copy:** Tap on the "Copy" (Fig. "Data logger"/9), button to start copying. After concluding the copy process the number of copied files will be displayed (Fig. "Data logger"/13). Please note that approximately 20 MB per minute can be copied from the USB stick (transmission of 1 GB takes approximately 50 minutes).

**NOTE!**

After copying the USB stick is unmounted. Consequently the display of the remaining storage space on the USB stick goes out. If you want to copy additional data to the stick, you must remove the stick and then re-insert it in the USB connection.

I-5.7.5 Operating page V_TREND

Use the operating page "V_TREND" to visualize the trend of analog and digital values as a function graph.

Properties

- **Graph display:** Analog or digital values are displayed in their temporal trend.
- **Measured value selection:** The associated axis scaling can be displayed for a trend. Selection is executed via the "Measured value selection" button.
- **Tracks:** A maximum of 12 digital tracks and 6 analog tracks can be recorded.
- **Memory:** A maximum of 10 000 values can be stored (the precise number depends on the configuration of the block).
- **Time grid:** Values are recorded in a fixed time grid. The minimum time grid is 0.2 seconds, the maximum is 3600 seconds (the precise value depends on the configuration of the block).

 **NOTE!**
Additional information on the "V_Trend" operating page is available in the function block reference.

Overview

- 1 Title
- 2 Analog trend curves (the curve is read from right to left)
- 3 Cursor
- 4 Time specification (beginning, at cursor position, end). The specification is made in hours/minutes/seconds.
- 5 Value specification for the selected measured value at the cursor position
- 6 Button "Leave Operating Page"
- 7 Button "Alarm"
- 8 Value range of the selected measured value on the Y-axis
- 9 Button "Call parameter page"
- 10 Button "Cursor to the left"
- 11 Button "Cursor to the right"
- 12 Button xxx
- 13 Button xxx
- 14 Digital tracks

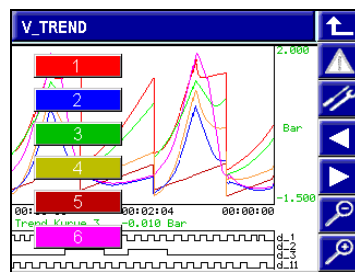
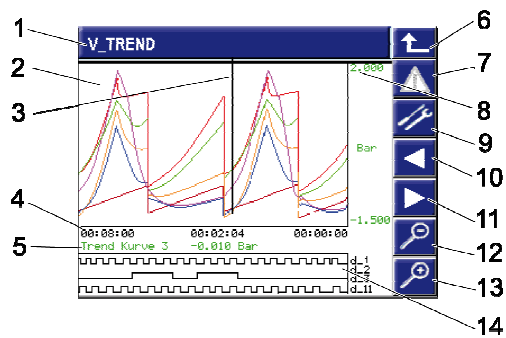


Fig. 49: Trend (Trend curves and measured value selection)

Operation

Measured value selection:

1. **Call "Measured value selection":** Tap right on the free area next to the Y-axis.
Now you will see an overview of the available measured values, displayed as a colored rectangle 1-6.

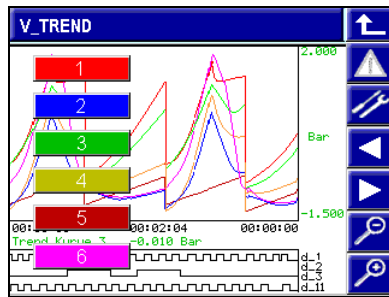


Fig. 50: Measured value selection

2. **Select a measured value:** Tap on one of the rectangles to select the desired measured value. The "Measured value selection" dialog will close. Value specification and axis scaling are adapted to the selected measured value.

**NOTE!**

The value specifications for the current cursor position always have the color of the corresponding measured value.

Selecting the value specification

Move the cursor to the X axis with the cursor buttons (Fig. "Trend"/7 and 8). The value specification will be adjusted at the selected point in time.

Browse - cursor to the end

Move the cursor on the X-axis to the beginning or end of the display with the cursor buttons (Fig. "Trend"/7 and 8). If more values are stored than are displayed then the system will advance or go back a half page.

Hide / show values

You can show or hide analog and digital values.

To do this, proceed as follows:

1. **Call the "Parameters" operating page:** Tap on the button "Parameters" (Fig. "Trend"/10) to call the "Parameters" operating page.
2. **Call the parameter <Value>:** In the parameter list call the name of the value that you would like to show or hide (e.g. "Display value type1" in the following illustration). You will see a selection list.

| V_TREND - Parameter | | |
|---------------------|--------|---|
| Step size | 1 | ▲ |
| Start delay | 0 | |
| Display Mode 1 | Single | |
| Display Mode 2 | Single | |
| Display Mode 3 | Single | |
| Display Mode 4 | Single | ▼ |

Fig. 51: "V_Trend" parameter

3. **Show / hide value:**
For digital values: In the selection list select the option "off" if the value should not be displayed, or "on" if the value will be displayed.

**NOTE!**

Hide the digital values to increase space for the analog values.

For analog values: Select the option "off" in the selection list, if you do not want to display the value.

Select one of the options "individual", "average", or "min/max" to display the value (additional information on these options is available in the section "Adjusting the display of analog values").

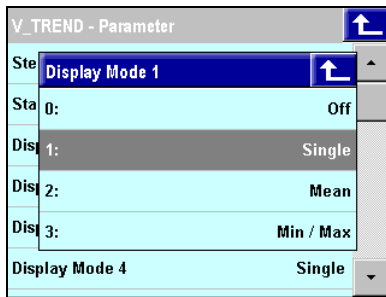


Fig. 52: Hide / show values

Combining analog values

If you would like to view a larger time segment you can combine multiple points into one point. To do this, proceed as follows:

1. **Call the "Parameters" operating page:** Tap on the button "Parameter page" (Fig. "Trend"/10) to call the "Parameters" operating page.

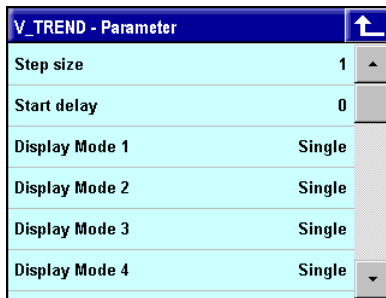


Fig. 53: "V_Trend" parameter

2. **Enter the parameter "Increment":** Tap on the parameter "Increment". Now you will see the numerical value editor. Here enter the number of values that will be combined into one value.

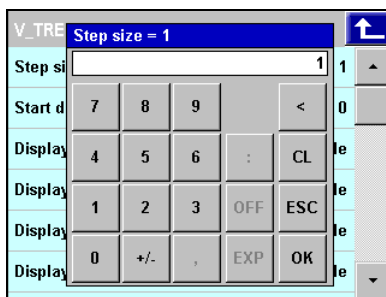


Fig. 54: Enter increment

Adjusting the display of analog values

The display of analog values can be adjusted. The following options are available in this regard:

- **Off:** The value is not displayed.
- **Single:** Multiple points are combined into one point.
- **Average:** The average value of the points that will be combined is calculated and displayed.
- **Min/max:** A line is drawn between the minimum and maximum value of the points that will be combined.

Proceed as follows to select these options:

1. **Call the "Parameters" operating page:** Tap on the button "Parameters" to call the "Parameters" operating page.

| V_TREND - Parameter | | ↑ |
|---------------------|--------|---|
| Step size | 1 | ▲ |
| Start delay | 0 | |
| Display Mode 1 | Single | |
| Display Mode 2 | Single | |
| Display Mode 3 | Single | |
| Display Mode 4 | Single | ▼ |

Fig. 55: "V_Trend" parameter

2. **Call the parameter <Value>:** In the parameter list (see the Fig. above) call the display type of the value that you want to scale to.
3. **Select option:** Select the desired option from the selection list.

| V_TREND - Parameter | | ↑ |
|---------------------|--------|---|
| Step size | 1 | ▲ |
| Start delay | 0 | |
| Display Mode 1 | Single | |
| Display Mode 2 | Single | |
| Display Mode 3 | Single | |
| Display Mode 4 | Single | ▼ |

Fig. 56: Select parameter

Specifying the Y-axis value range

You can specify the value range of the Y-axis (for analog values) as follows:

1. **Call the "Parameters" operating page:** Tap on the button "Parameter page" (Fig. "Trend"/10) to call the "Parameters" operating page.

| V_TREND - Parameter | | ↑ |
|---------------------|-----|---|
| Display Mode 12 | Off | ▲ |
| Scale start X1 | 0 | |
| Scale end X1 | 100 | |
| Scale start X2 | 0 | |
| Scale end X2 | 100 | |
| Scale start X3 | 0 | ▼ |

Fig. 57: "V_Trend" parameter

2. **Enter the parameter "Value range":** You must first select the measured value to which the display will be scaled. The measured values are consecutively numbered from X1 to X6). For each measured value you can select the lower limit (e.g. "Scale beginning X1") and the upper limit ("Full scale X1"). Select the appropriate menu entry and enter the value range with the numerical value editor (see the following Fig.).

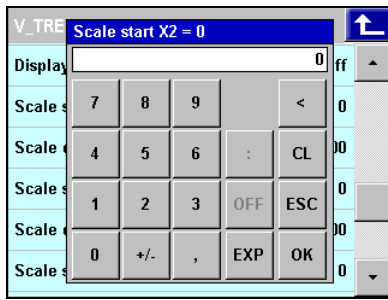


Fig. 58: Entering the value range

I-5.7.6 Operating page PASSWORD

Use the "PASSWORD" operating page to enable different access levels of operation for four users (or user groups).

Properties

- **Access levels:** The access levels are displayed in five rows. The topmost row displays the most restricted access level. More operation is available the lower the row is.
- **Marker:** The active access level is marked with a grey box.
- **Names of access levels:** Define names of access levels in the engineering.



NOTE!

Additional information on the "Bargraph" operating page is available in the function block reference.

In the levels the following functions are enabled for operation according to the access level, the higher levels including the access rights of the lower levels:

| Access level | User (example) | Operation |
|--------------|----------------|---|
| 0 | (locked) | Information only (no changes) |
| 1 | operator | Operating pages enabled: change setpoints. For programmers: select program recipe , start/stop/reset/preset/search. For function block PROGRAMMER: Change of running program (without saving) |
| 2 | engineer | (Permanent) changes of program recipes and parameters |
| 3 | commissioner | Changes of basic functions, configuration |
| 4 | chief engineer | Changes of basic functions, configuration |

Overview

- 1 Title
- 2 Lowest "access level": only display, no operation.
- 3 Access level 1 ("operator"): access to operating pages
- 4 Access level 2 ("engineer"): access level 1 and additionally parameters and (permanent) changes of programs of programmers
- 5 Access level 3/4 ("commissioner", "chief engineer"): access level 2 and additionally configurations
- 6 Button "Leave Operating Page"
- 7 Button "Alarm"
- 8 Marker of active access level (here: access level 2 "engineer")

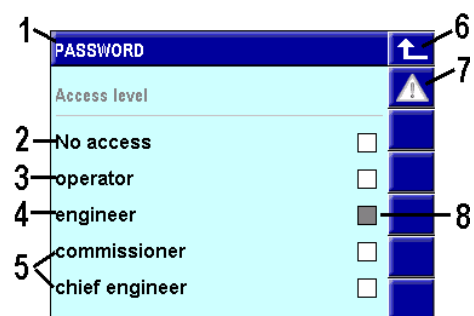


Fig. 59: PASSWORD (with all 4 access levels)

Operation

- **Change of access levels:** If a user selects a lower level, the device switches to this level. For a higher level the user selects his level and enters his password, afterwards he can change to any lower level.

Example

The engineer wants to enable the locked operation. The user "engineer" is set for level 2. So he selects level 2 ("engineer") and enters his password for level 2. Accordingly level 2 is enabled and marked. Now the engineer can activate level 1 (without password check).



NOTE!

For access level 2 (engineer) there is set a time out of 10 minutes, which means if in level 2 the touch display is not operated for 10 minutes the level is reduced to level 1 (= user).



NOTE!

The two highest levels (3 and 4) have the same access rights, but enable to define different users (or user groups).

I-5.7.7 Operating page CONTROL

The operating page "CONTROL provides information about controllers and process control loops and permits intervention in these control loops (e.g. switch over to manual mode).

The PMA library contains three controllers:

- **Controller "Control"**: The "Control" controller includes a PID controller with numerous functions (e.g. setpoint ramp, setpoint switch over, setpoint/process value tracking, self-optimization, override control, feed-forward control, control variable guidance, ratio control and three-component control in 12 different controller type variants (e.g. continuous, 2-point, 3-point, and motor-step).
- **Controller "Control+"**: The controller basically includes the same functions as the "Control" controller. In addition the "Control+" controller enables guided adaptation. Six parameter sets can be activated depending on process criteria (process value, setpoint, controller output, system deviation) and plant or batch properties. The parameter sets can be determined by self-optimization independent of each other.
- **Controller "PIDMA"**: The PIDMA controller basically corresponds to the "Control" controller. However differences occur due to a different implementation of the PID controller kernel. In this case a different control algorithm and different processes are used for self-optimization.



NOTE!

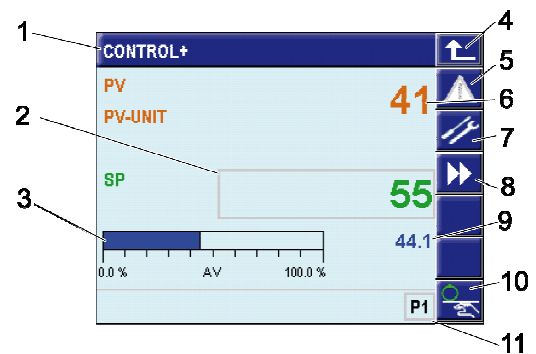
Additional information on the controller is available in the function block reference.

Properties

- **Target value display/process value display**: Display of target value and process value.
- **Controller output**: The controller output is displayed numerically and as bargraph.
- **Manual mode**: Switch over between automatic mode and manual mode is possible.
- **Self-optimization**: A separate operating page enables the starting of a self-optimization process and provides information about its trend and results.

Overview

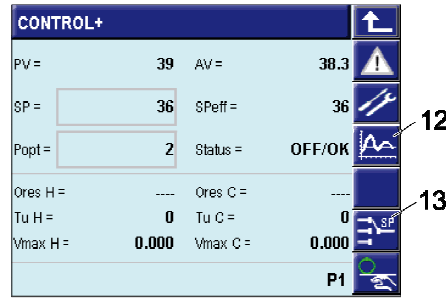
- 1 Title
- 2 Display and input field for the setpoint ("SP" = Setpoint)
- 3 Bargraph: Controller output ("AV" = "Actuating Variable"), deviation variable ("DV" = "Deviation Variable") or process value ("PV" = "Process Variable")
- 4 Button "Leave Operating Page"
- 5 Button "Alarm"
- 6 Display process value ("PV" = "Process Variable")
- 7 Button "Call parameter page"
- 8 Button "Call self-optimization"
- 9 Display controller output
- 10 Button "Switch over automatic mode/manual mode"
- 11 Controller parameter set selection (only for Control+)



Page Self-optimization

The "Self-optimization" page shows a new button compared with the main page of the controllers:

- 12 Button "Start/stop self-optimization"
- 13 Button "Setpoint switch over" (this button is also available on the main page, if switch over via parameter assignment is permitted).



| Field | Description |
|---------------|--|
| PV | "Process Variable"/process value |
| SP | "Setpoint"/ internal setpoint |
| Popt | Parameter set that will be optimized # |
| T | Optimization time* |
| AV | "Actuating Variable"/controller output |
| SPeff | ("Setpoint effective")/effective setpoint |
| Status | Current status of self-optimization Note: More detailed information on the status display is available in the function block reference. |
| Ores H | Optimization result - heating |
| Tu/ Vmax H | Process properties - heating |
| Ores C | Optimization result - cooling |
| Tu/ Vmax C | Process properties - cooling |
| Ores | Optimization result - heating/cooling* |
| Kp/Tn/Tv | Control parameters* |
| Orun | Display of "ORun", if the optimization is running, and display of "OErr" is the optimization is defective. |

* Only for controller PIDMA

Only for controller CONTROL+

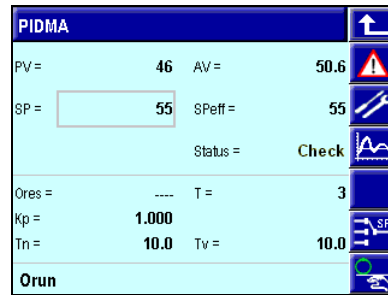


Fig. 60: Controllers "Control"/"Control+" and controller "PIDMA" (controller overview and self-optimization)

Operation

Entering values

If provided by the developer of the application program the operating page has input fields (bordered in gray). Tap on the field to enter a value.

Enter the values either via the numerical value editor (see the section "Using the numeric value editor") or via the following dialog:

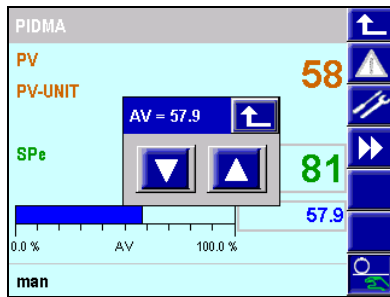


Fig. 61: Entering numerical values

Here tap on the button "▲" or "▼" to increase or decrease the value. The actual value will be displayed in the header.

Working in manual mode

1. **Activate manual mode:** Tap on the button "Switch over Automatic/manual mode" to start manual mode. "man" will be displayed in the status line.

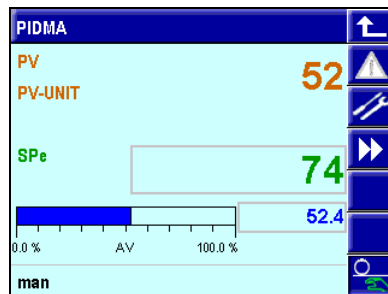


Fig. 62: Working in manual mode

2. **Entering values:** If manual mode is activated then you can enter the desired values manually. To enter values tap on the appropriate field (e.g. "AV").
3. **Ending manual mode:** Tap a second time on the button "Switch over automatic mode/manual mode" to end manual mode.

Starting self-optimization

1. **Call the "Self-optimization" operating page:** Tap on the button "Call self-optimization" (Fig. "Controllers"/8). The "Self-optimization" operating page will be displayed.
2. **Starting manual mode:** Tap on the button (Fig. "Controllers"/10), to change to manual mode.
3. **Setting the setpoint reserve:** Tap on the setpoint and enter a setpoint that is at least 10% above the current actual value.
4. **Starting self-optimization:** If "OFF/OK" is shown in the "Status" field, (i.e. self-optimization is *not* yet running): Tap on the "Call self-optimization" button (Fig. "Controllers"/12), to start self-optimization.




NOTE!

Detailed information on self-optimization of the PIDMA controller is available in the section "PIDMA Self-optimization".

Ending self-optimization

1. **Call the "Self-optimization" operating page:** Tap on the button "Call self-optimization" (Fig. "Controllers"/8). The "Self-optimization" operating page will be displayed.

- End self-optimization:** If "OFF/OK" is NOT shown in the "Status" field, (i.e. self-optimization is running): Tap on the "Start/end self-optimization" button (Fig. "Controllers"/12), to end self-optimization.

 **NOTE!**
 If this has been provided by the application developer, alternatively you can end self-optimization with the button "Switch over automatic mode/manual mode" (Fig. "Controllers"/10).

Understanding optimization messages

Messages are displayed during the optimization process, these have the following meaning:

Ores1/Ores2 for CONTROL and CONTROL+

| Ores1/2 | Meaning or cause of the problem | Solution possibility |
|---------|--|--|
| 0 | No attempt executed or attempt interrupted by <i>Stop</i> or by switching over to manual mode. | |
| 1 | Cancel: Wrong direction of information flow of the actuating variable, X does not change in direction W. | Change the controller's direction of information flow. |
| 2 | Terminated: Self-optimization has been successfully executed (point of inflexion found; estimation sure). | |
| 3 | Cancel: The controlled variable does not react or is too slow (change of ΔX is less than 1% in 1 hour). | Close control loop. |
| 4 | Terminated without <i>AdaErr</i> : Successful attempt, segment has a deep-set point of inflexion. | Best possible result at deep-set point of inflexion. |
| | Canceled, with <i>AdaErr</i> : Unsuccessful attempt inadequate process stimulation (point of inflexion found, however the estimation is unsafe). | Increase actuating value step <i>dYopt</i> |
| 5 | Cancel: Optimization cancelled due to the risk of exceeding the setpoint. | Increase the distance between process value (X) and setpoint (W) at start or reduce <i>YOptm</i> . |
| 6 | Terminated: Attempt successful however optimization cancelled due to the risk of exceeding the setpoint (point of inflexion not yet reached; estimation safe). | |
| 7 | Cancel: Actuating value step too small, $\Delta Y < 5\%$. | Increase <i>Ymax</i> or set <i>YOptm</i> to a lesser value. |
| 8 | Cancel: Sollwertreserve zu klein oder Sollwertüberschreitung während PiR-Überwachung läuft. | Change steadying actuating value <i>YOptm</i> . |

Ores for PIDMA

| Ores | Meaning or cause of the problem | Solution possibility |
|------|---------------------------------|----------------------|
| 0 | No attempt executed | |

| | | |
|-----------|------------------|--|
| 1 | Xlimit too small | step threshold too small: Compared to the process noise the step threshold is too small. Start a new attempt with a greater actuating pulse. |
| 2 | DYopt large | Actuating pulse too large: The actuating pulse would exceed the selected actuating limits at output of the selected pulse height. A new attempt with a lesser actuating pulse height should be stated, or the actuating variable should be reduced in manual mode beforehand. |
| 3 | Restart | No idle state: The auto tuner has detected that the process is probably not in idle state. Please wait until the idle state has been reached. Optionally the drift compensation can be activated or the actuating pulse can be increased. Comment: For pulse width modulated (PWM) control outputs (2-point and 3-point controllers) even in manual mode oscillations of the process value PV can occur if the corresponding cycle time t_1 (t_2) is too long. In this case you must set the shortest possible switch cycle times on the controller. |
| 4 | DYopt small | Actuating pulse too small: The step response declines in the process noise. A new attempt with a greater actuating pulse height should be started, or the overlaid noise should be reduced via suitable measures (e.g. filters). |
| 5 | No extremum | Max detection failed: After output of the actuating pulse no maximum/minimum was detected in the process value trend. The settings for the zone type (with/without compensation) should be checked. |
| 6 | Actuating limit | Actuating limits exceeded during optimization. The actuating variable MV exceeded the actuating limit during the attempt. The attempt should be repeated in manual mode with a reduced actuating pulse or reduced actuating value. |
| 7 | Controller type | No optimization result can be found for the specified combination P/I/D. |
| 9 | Monotony | Process not monotone: The process shows pronounced all pass behavior (temporary counter behavior of the process value) or a significant fault occurred. |
| 10 | Estimation error | Extrapolation failed: After end of the actuating pulse a drop in the process value was not detected, possibly due to excessive process noise. Increase actuating pulse or attenuate noise. |
| 11 | No result | Unusable result: Excessive process noise or the control parameters determined do not agree with the description of a zone with dead time. Start a new attempt with greater actuating pulse or attenuate the existing noise. |
| 12 | Man. Cancel: | The optimization attempt has been cancelled by the operator through "STOP". |

Direction

Wrong direction of information flow: The expected direction of information flow of the step response runs counter to the actuating value. This can be due to wrong setting of the direction of information flow, or for example it could be due to inverting final control elements. Change the controller's direction of information flow.

I-5.7.8 Using controllers in controller cascades

Control loops can be broken down into partial control loops (cascade control). Cascade control consists of at least two controllers, a guide controller (master) and a following controller (slave). The master controller specifies the setpoint to the slave controller and thus influences the main control value via the slave controller. If a controller is used as slave controller, then the status display "Cascade or "Casc-open" is shown on the controller page. The PMA library enables convenient operation of a controller cascade on the operating page of the slave controller.

NOTE!
Basic information on cascade control is available in the function block reference.

Overview

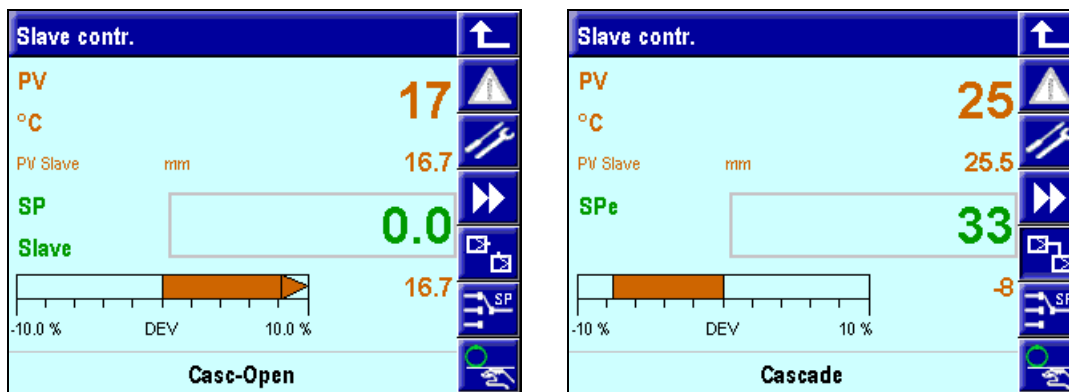


Fig. 63: Controller cascade

If provided by the developer you can interrupt the cascade, i.e. "open" the cascade. which means use the slave controller(s) independently from the master controller.

- **Open cascade:** If the cascade is closed (in the footer the word "Cascade" is displayed) then tap on the "📄" button to open the controller cascade. The display "Cask open" will be displayed.
- **Close cascade:** If the cascade is open (display in the input field "Casc-open") then tap on the "📄" button to close the controller cascade. "Cascade" is displayed.

I-5.7.9 Detailed information on self-optimization

Detailed information on self-optimization of the controllers CONTROL, CONTROL+ and PIDMA is provided in the following section.

I.5.7.9.1 Controller characteristic values (CONTROL and CONTROLP)

To determine the control parameters you must first be aware of the zone data. The zone data are detected automatically by the controller in the self-optimization process and are converted into control parameters. Nevertheless in exceptional cases it can be necessary to manually determine the zone data. The temporal trend of the measured value PV can be referenced after a step-like change of the actuating value AV.

In practical terms, often it is not possible to completely record the step response (0 to 100%) as the measured value cannot exceed certain values. With the values T_g and PV_{max} (step from 0 to 100%) or Δt and ΔPV (part of the step response) the maximum rate of rise V_{max} can be calculated.

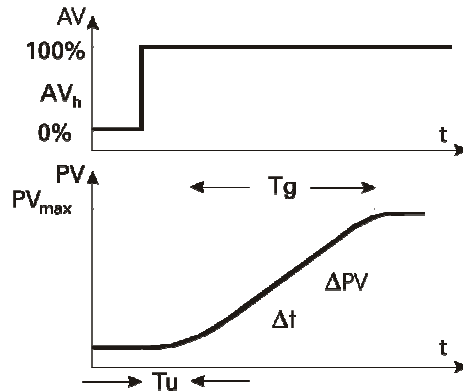


Fig. 64: Characteristic values of the controller zone

$$K = \frac{V_{max}}{PVh} \cdot Tu \cdot 100\%$$

y = Controller output

AVh = Correcting range

Tu = Delay time (s)

Tg = Compensation time (s)

$$V_{max} = \frac{PV_{max}}{Tg} = \frac{\Delta PV}{\Delta t} \triangleq \text{max. rate of rise of the}$$

control variable

$PV_{max} \triangleq$ Maximum value of the controlled system

$PVh =$ Control range = $PV_{hi} - PV_{lo}$

Characteristic values of the controllers

Normally a fast and overshoot-free smoothing to the setpoint is desired. Depending on the controlled system the following control behaviors are useful:

- **PD controller:** Easily controllable zones ($K < 10\%$) should be regulated with PD controllers.
- **PID controllers:** Moderately controllable zones ($K = .10...22\%$) should be regulated with PID controllers.
- **PI controllers:** Zones that are difficult to control ($K < 22\%$) should be regulated with PI controllers.

From the determined values (delay time Tu , the maximum rate of rise V_{max} , the control control range PV_h and characteristic value K) the required control parameters can be determined according to the following "rules of thumb". A more precise setting must be made in accordance with the setting aids. If there is an oscillating run-up to the setpoint then Xp must be increased.

Rule of thumb

| Behavior | $Xp[\%]$ | $Tv[s]$ | $Tn[s]$ |
|-----------------------------|----------|---------|-----------------|
| (D)PID | 1.7 K | 2 Tu | 2 Tu |
| PD | 0.5 K | Tu | $\infty = 0000$ |
| PI | 2.6 K | 0 | 6 Tu |
| P | K | 0 | $\infty = 0000$ |
| 3-point step controller PID | | | |
| | 1.7 K | Tu | 2 Tu |

Setting aids

| Characteristic value | Control process | Fault | Actuating process |
|----------------------|-----------------|------------------------|-------------------------|
| Xp | Greater | More highly attenuated | Slower smoothing |
| | Less than | Weaker attenuation | Faster smoothing |
| Tv | Greater | Weaker attenuation | Stronger reaction |
| | Less than | More highly attenuated | Weaker reaction |
| Tn | Greater | More highly attenuated | Slower smoothing |
| | | | Slower energy reduction |

| | | | |
|-----------|--------------------|------------------|-------------------------|
| Less than | Weaker attenuation | Faster smoothing | Faster energy reduction |
|-----------|--------------------|------------------|-------------------------|

Direct inverse switch over is possible generally, it is executed with the configuration parameter CMode (direction of information flow). The following illustration shows the principle:

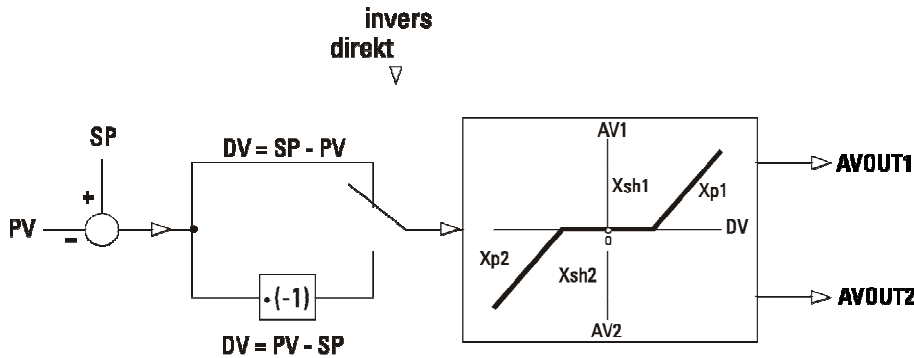


Fig. 65: Principle direct/inverse switch over

1.5.7.9.2 Self-optimization: Controller adaptation to the control zone

Self-optimization can be executed to determine the optimal parameter for a process. This is applicable for control zones with compensation and non-dominating dead time as well as $K \leq 30\%$.

After the user starts the self-optimization the controller executes an adaptation attempt to determine the zone characteristic values T_u and V_{max} . It calculates from this the control parameters for a fast, overshoot-free smoothing to the setpoint.

Detailed information on controller self-optimization is provided in the following section.



NOTE!

The following sections assume that you are familiar with the BlueDesign development environment. Additional information in this regard is available in the chapter "Development environment/working with the development environment". More detailed information on the parameters is available in the function block reference.



DANGER!

Danger of injury due to unforeseeable plant function sequences and movement sequences!

The controller is not functional during self-optimization. Plant parts can be subject to unforeseeable reactions if they are not uncoupled from the device.

Therefore:

The following must be heeded during self optimization:

- All system parts that are switched off must be safeguarded from being restarted inadvertently!
- In general, the effects of switching off the system must be considered, and appropriate measures must be taken.



NOTE!

The graphics in this chapter (e.g. "Fig. 68: "process at rest" monitoring") use deviating variable designations. X is used instead of "PV", and "Y" is used instead of "AV".

Preparation for self optimization

To determine the optimal parameters for a process, you can start self-optimization of the controller. To do this, proceed as follows:

1. **Set control behavior:** Set the following control behavior on the "Self-optimization" page:

| Controllers | Behavior |
|-----------------|---------------------------|
| P-controllers | $T_n = 0.0$, $T_v = 0.0$ |
| PD controllers | $T_n = 0.0$, $T_v > 0.0$ |
| PI controllers | $T_n > 0.0$, $T_v = 0.0$ |
| PID controllers | $T_n > 0.0$, $T_v > 0.0$ |



NOTE!

The parameters "Tn" or "Tv" can be switched off by setting the value 0.0. Thus they do not participate in the self-optimization.

2. **Select parameter set (CONTROL+):** If you use the CONTR+ controller you must now select the parameter set that will be optimized. Select the parameter set in *BlueDesign* with the parameter *POpt* (see the following illustration).

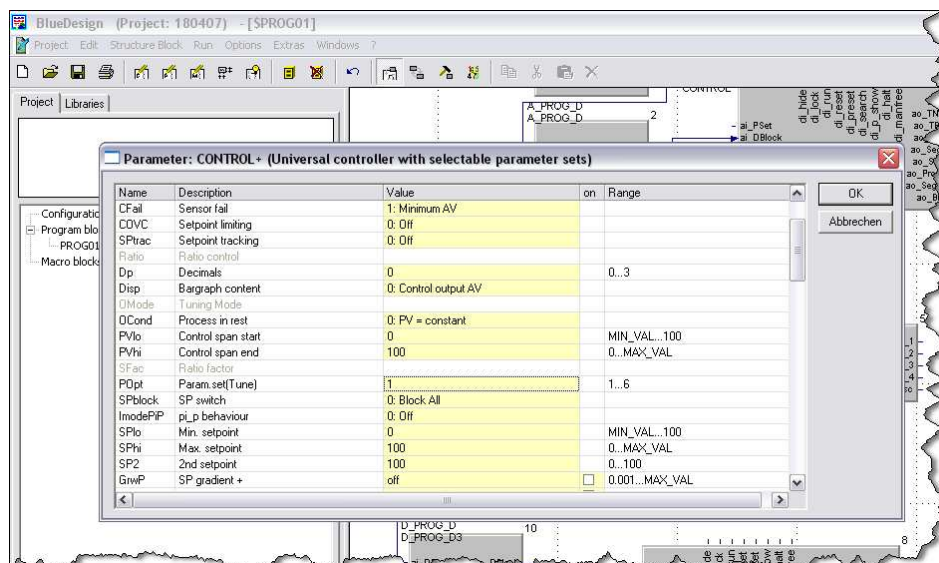


Fig. 66: Select parameter "POpt"

3. **Configuring the conditions for process at rest:** Determine the mode for which the state "process at rest" (*PAR_H*) will be detected in the *BlueDesign* parameter dialog using the *OCond* parameter.

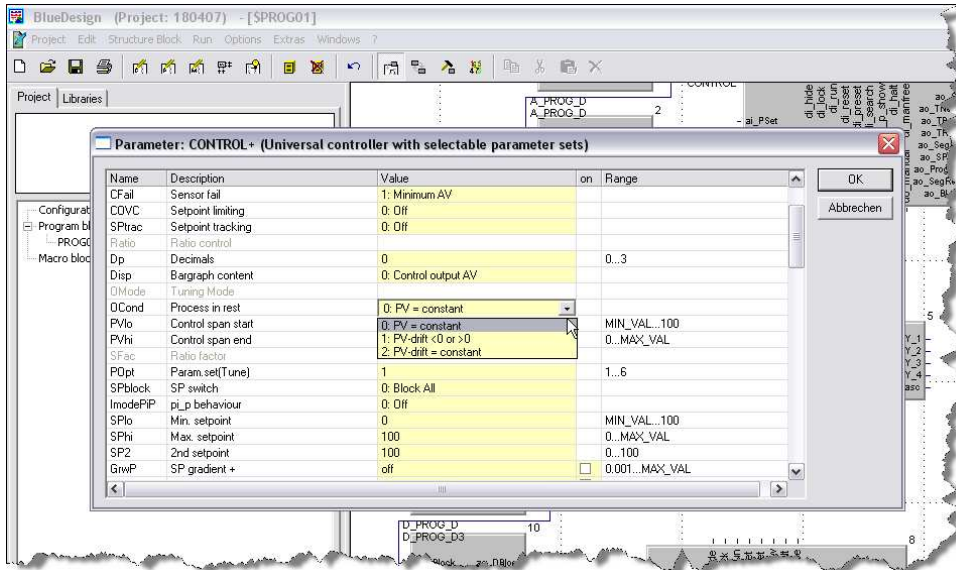


Fig. 67: Select parameter "OCond"

"Process at rest" monitoring is executed consecutively. The process is in rest if the control variable remains within a tolerance band of $\pm\Delta PV = 0.5\%$ for more than 60 seconds:

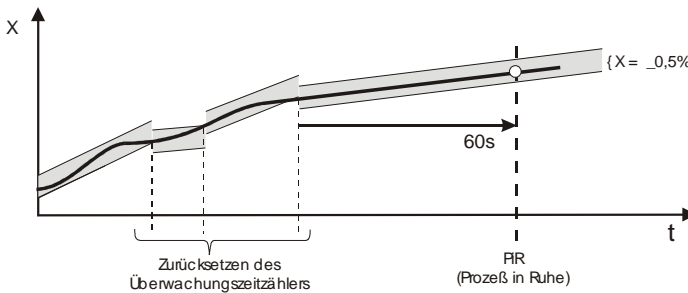


Fig. 68: "process at rest" monitoring

If the process value leaves this tolerance range then the monitoring time counter is reset to zero. If for example in *PaR* is detected in control mode and then when starting self optimization a markedly deviating steady manipulated variable *AVOptm* is output then you must wait until the full *PaR* time interval has elapsed.

Use the configuration word *OCond* to specify the "process at rest" mode - detection, the following options are available:

| Parameters | Meaning |
|---------------------|--|
| PV = constant | The status "process at rest" is detected if the value PV is constant. |
| PV drift <0 or >0 | The status "process at rest" is detected if <ul style="list-style-type: none"> ■ The value PV uniformly decreases with the inverse direction of information flow. ■ The value PV uniformly increases with the direct (non-inverse) direction of information. |
| PV drift = constant | The status "process at rest" is detected if the value <i>PV</i> changes uniformly. |

4. **Specify control variable *AVOptm*:** Specify the control variable *AVOptm* in the *BlueDesign* parameter dialog. This is the start value that is used in automatic mode when starting self-optimization.

5. **Specify the control variable $dAVopt$:** Specify the control variable $dAVopt$ in the *BlueDesign* parameter dialog. This parameter determines the value by which the controller output steps. The point of departure in automatic mode is the start value $AVOptm$ or in manual mode it is the original controller output.

**NOTE!**

Pay attention to the necessary setpoint reserve. Self-optimization can only be executed if the interval between setpoint and process value is greater than 10% of the value of "SPlo" and "SPhi", prior to the setpoint step.

The setpoint reserve is either achieved automatically by reduction of the setpoint during the PaR phase (in automatic mode) or via manual change of the setpoint/process value (in manual mode).

Because the setpoint should not be exceeded in the optimization, the setpoint must be greater than the process value for inverse operating controllers, by the amount of the setpoint reserve. For direct working controllers on the other hand the setpoint must be less than the process value by the amount of the setpoint reserve.



Starting self-optimization

Information on starting self-optimization is available in the section "Operating page CONTROL".

**NOTE!**

Self-optimization can be started from automatic mode or from manual mode.

Cancel the adaptation

- **Click on the button "Switch over automatic mode/manual mode"** If the  button has not been locked (1 signal on the input "di_oplock"), then self-optimization can be ended at any time. Click on the  button to end self-optimization.

**NOTE!**

Alternatively self-optimization can be stopped in any case via the button "Start/stop self optimization" Additional information in this regard is available in the section "Operating page control".

Self-optimization sequence

Sequence of self-optimization when starting from automatic mode

After starting self optimization the steady state controlled variable $AVOptm$ is output. If "process at rest" (*PaR*) is detected and a sufficient setpoint reserve (r) is available then the controlled variable can be changed by the controlled variable step $dAVOpt$ (raised for inverse working controllers, lowered for direct working controllers lowered). The process for determining the characteristic value is executed based on the changing process value.

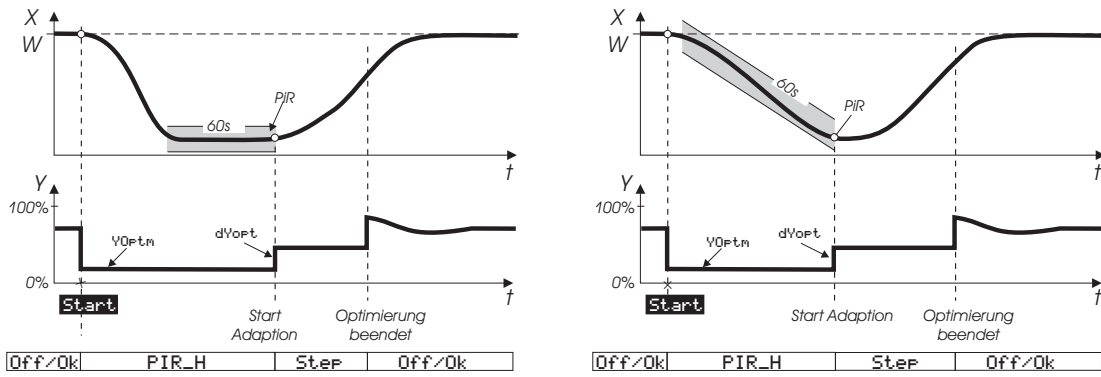


Fig. 69: Self-optimization process in automatic mode

After a successful adaptation attempt the controller goes into automatic mode and controls the setpoint with the newly determined parameters. The parameter *Ores* indicates the result with which self-optimization has been concluded.



NOTE!

If self-optimization concludes with an error (display: "Ada_Err"), then you must manually end self-optimization mode (see the section "Cancel self-optimization"). As long as self-optimization is not concluded the steady state controlled value will be output.

Sequence of self-optimization when starting from automatic mode



NOTE!

A description of how the controller goes into manual mode is provided in the section "Operating page/control".

If the controller is in manual mode then the last used manipulated will be transferred as temporary steady state manipulated value when self-optimization starts. As is the case in automatic mode the setpoint can be adjusted at anytime.

If "process at rest" (*PaR*) is detected and a sufficient setpoint reserve (*r*) is available then the controlled variable can be changed by the controlled variable step *dAVOpt* (raised for inverse working controllers, lowered for direct working controllers lowered). The process for determining the characteristic value is executed based on the changing process value.

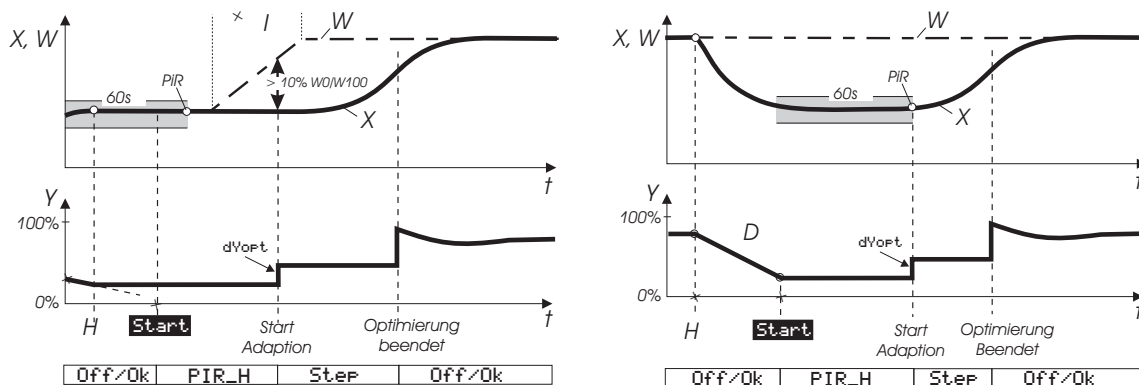


Fig. 70: Self-optimization in manual mode

After a successful adaptation attempt the controller goes into automatic mode and controls the setpoint with the newly determined parameters. The parameter *Ores* indicates the result with which the self-optimization has been concluded.

**NOTE!**

If self-optimization concludes with an error (display: "Ada_Err"), then you must manually end self-optimization mode (see the section "Cancel self-optimization"). As long as self-optimization is not concluded the steady state controlled value will be output.

Sequence of self optimization for "heating" type processes.

For 2-point controllers, motor step controllers, and continuous controllers, after achieving the status "process at rest" the control zone is stimulated with a controller output step, and from the process reaction if possible the step response T_{u1} and V_{max1} are determined at the point of inflexion.

Sequence of self optimization for "heating" and cooling type processes.

For 3-point controllers and split-range controllers self-optimization basically runs as described above for the application case "heating". However after controller self-optimization is concluded the system will smooth to the specified setpoint until PaR is reached again. Then to determine the "cooling" zone a step will be initiated in order to then determine T_{u2} and V_{max2} based on the step response. On the basis of these characteristic values the controller will then be set for the "cooling" process.

If the attempt is cancelled the parameters of the "heating" zone will also be transferred for the "cooling" zone; no error will be reported.

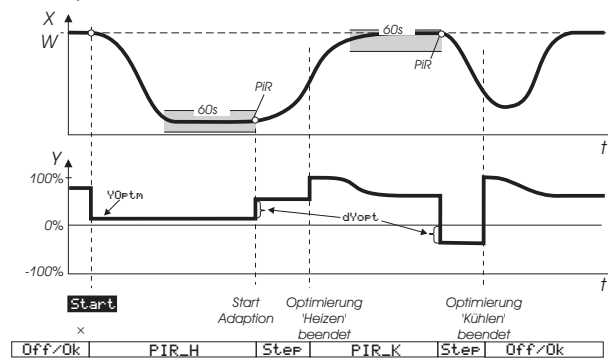


Fig. 71: Self-optimization for "heating" and cooling type processes.

1.5.7.9.3 Controller characteristic values - PIDMA

As opposed to CONTR and CONTR+, PIDMA contains a modified controller kernel in a parallel structure. This results in the following additional parameters.

Additional parameters for PIDMA

| Parameters | Description | Value range |
|---------------|---|--|
| PType | Process type (a priori information) | 1: with compensation 2: without compensation (integral) |
| Drift | Drift compensation of the process value at the beginning of self optimization | 0: Off 1: On |
| CSpeed | Desired control loop dynamics | 1: Slow 2: Normal 3: Fast |
| Tpause | Minimum actuating pause time (step controller) | 0,1...999999 [s] |
| thron | Switch on threshold for OPEN and CLOSE (step controller) is not effective | 0,2...100% |

| | | |
|---------------|--|-------------------|
| throff | Switch off threshold for OPEN and CLOSE (step controller) is not effective | 0,2...100% |
| Xlimit | Switch off point for controller output step (process value change) | 0,5...999999 |
| Tdrift | Time window for drift determination of the process value | 0...999999 [s] |
| Tnoise | Time window for noise determination of the process value | 0...999999 |
| Kp | Control gain (replaces Xp1/Xp2 of the CONTR) | 0,001...999,9 [%] |
| VD | Derivative action gain (Td/T1) | 1...999999 |
| bW_p | Setpoint weighting in the P-component | 0...1 |
| cW_d | Setpoint weighting in the D-component | 0...1 |
| Tsat | Time constant for I-component in AV limitation (anti-reset wind-up) | 1...999999 [s] |
| xsh | Neutral zone in which the I-component is specified | 0 ... 999999 |

Parameter for stepper motor control

Tpause, *thron* and *throff* supplement the parameters for stepper motor activation. *Tpause* in addition to limiting the minimum pulse (*Tpuls*) also allows adjustment of the minimum pause.

With *Xsh* the switch frequency and the fine adjustment of the final control element can be influenced. *Xsh* determines the dead zone of the deviation variable in the main controller. The I-component of the controller is stopped within this zone.

Integrated position controller:

The PIDMA function block includes two controllers for the setting *3-point step PF* (motor step with position feedback):

- **Main controller:** The main controller controls the process value and supplies the desired setting of the final control element to an integrated position controller.
- **Position controller:** This ensures the desired position of the final control element via position feedback.

1.5.7.9.4 Self-optimization PIDMA

Detailed information on self-optimization of the PIDMA controller is provided in the following section.



NOTE!

The following sections assume that you are familiar with the BlueDesign development environment. Additional information in this regard is available in the chapter "Development environment/working with the development environment". More detailed information on the parameters is available in the function block reference.

**DANGER!****Danger of injury due to unforeseeable plant function sequences and movement sequences!**

The controller is not functional during self-optimization. Plant parts can be subject to unforeseeable reactions if they are not uncoupled from the device.

Therefore:

The following must be heeded during self optimization:

- All system parts that are switched off must be safeguarded from being restarted inadvertently!
- In general, the effects of switching off the system must be considered, and appropriate measures must be taken.

Parameters - PIDMA

PType, *Drift*, *Cspeed*, *Xlimit*, *Tdrift* and *Tnoise* supplement the parameter *dAVopt* that is also available for CONTR. These parameters specify the conditions for self-optimization.

Start the *BlueDesign* development environment and edit the parameters of the respective PIDMA block. Ensure that you are not in edit mode.

- **PType** specifies whether for the plant a process without compensation (after an output pulse, a new process value is set on a higher level, e.g. in case of fill level of a container without discharge or of a very well insulated oven). Uniform drop or increase of the process value prior to optimization can be detected via drift monitoring that can be switched on and taken into account for subsequent optimization.
- **CSpeed**: The *Cspeed* specifies whether the setpoint will be achieved with slight overshoot or more slowly with a more gentle approach. With *Cspeed* the parameters can also be switched over after optimization, as long as the control parameters have not been manually changed.
- **Tdrift**: After start of self optimization within the time frame *Tdrift* the system attempts to detect a drift.
- **Tnoise**: After the time frame *Tdrift* in the time frame *Tnoise* the system attempts to detect the noise (the output value dependent fluctuations) on the process value.

**NOTE!**

Depending on the plant, the times "Tdrift" and Tnoise" must be selected large enough that fault-dependent drift and multiple fluctuation of the interference influences can be detected.

- **dAVopt, PVlimit**: After these times the actual output value will be increased by **dAVopt**. If the actual value has then increased by more than **PVlimit** with consideration of drift and noise, then the output value will be reset to the original value. The self-optimization process is only concluded if the actual value after exceeding the maximum, has decayed to virtually half of the initial value. During the decay process (after the output value pulse) the estimated remaining time until the end of optimization is continuously displayed. After conclusion of the process the determined parameters *K*, *Ti* and *Td* are displayed on the optimization page, and together with the parameters *VD*, *BW_p* and *CW_d* are automatically transferred into the function block and activated for the running process.

Control parameters of the PIDMA

As opposed to CONTR the PIDMA does not have any special parameters for the application cases "heating" and "cooling". The parameter *K*, which is valid for both areas determines the control gain of a parallel controller structure.

Additional parameters permit independent weighting of individual controller components:

- **VD**: Derivative action gain ($Rd/T1$) in addition to the control gain permits superelevation or weakening of the D-component.

- **BW_p**: Setpoint weighting in the P-component
- **CW_d**: Setpoint weighting in the D-component

The parameters *BW_p* and *CW_d* can weaken the influence of a setpoint change on the controller reaction. Thus it is possible to set different behavior of the controller in response to setpoint changes (guide behavior) or to actual value changes (interference behavior). The setpoint influence can be loaded with a factor between 0 and 1.



NOTE!

In cases where self-optimization of the controller seems to be necessary use the PMA software PMATune. Contact the manufacturer for more information about PMATune.



NOTE!

*In the dynamic trend of a controller the control algorithm can also temporarily determine values less than 0 or greater than 100 for the output value. These can be restored to the limit values (0 and 100) as needed with an accelerated integral behavior ("*Tsat*" = time constant for the I-component in an AV limitation/anti-wind-up)*

Preparation for self optimization

To determine the optimal parameters for a process you can start the self optimization of the controller. Proceed as follows for this:

1. **Set the control behavior:** Set the following control behavior on the "Self-optimization" page:

| Controllers | Behavior |
|-----------------|------------------------|
| P-controllers | $T_n = 0.0, T_v = 0.0$ |
| PD controllers | $T_n = 0.0, T_v > 0.0$ |
| PI controllers | $T_n > 0.0, T_v = 0.0$ |
| PID controllers | $T_n > 0.0, T_v > 0.0$ |



NOTE!

*The parameters "*Tn*" or "*Tv*" can be switched off by setting the value 0.0. Thus they do not participate in the self-optimization.*

2. **Specify output value step:** In the *BlueDesign* parameter dialog (see the following illustration) specify the output value step *dAvopt*. The output variable will step by this value starting from the actual value.
3. **PVlimit determination:** In the *BlueDesign* parameter dialog (see the following illustration) specify the *PVlimit* parameter. This parameter must be approximately set to half of the expected change in the process value.

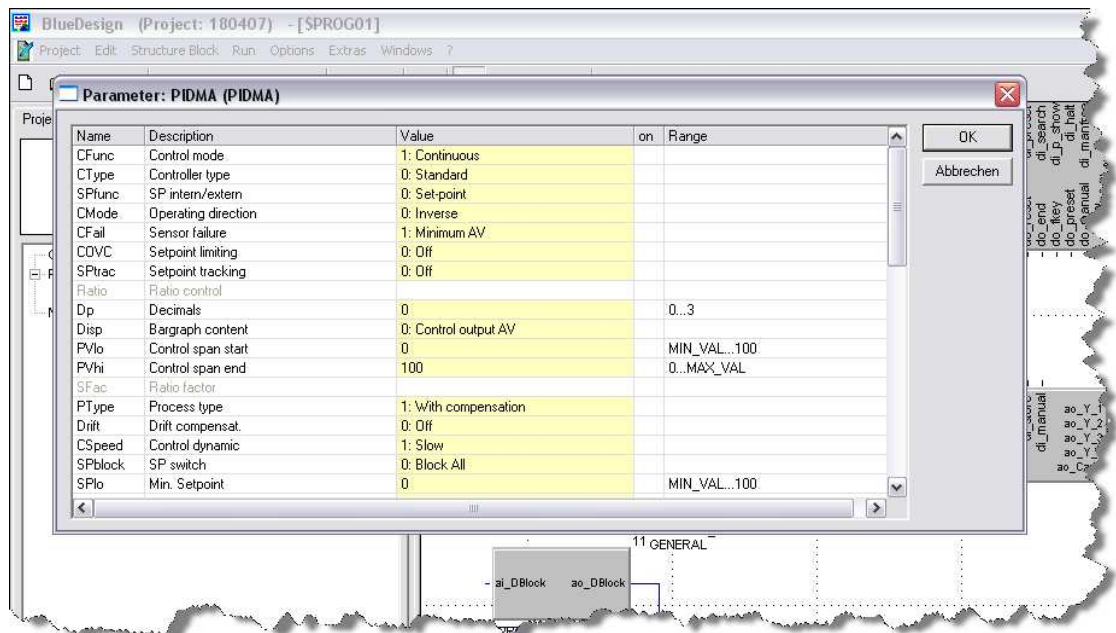


Fig. 72: BlueDesign parameter dialog

"Process at rest" monitoring

The PIDMA does not monitor the rest condition. It is up to the user's discretion to select the suitable start time. You will only obtain optimal results if the process is smoothed, i.e. all dynamic process have decayed. Only in rare cases, where parameter determination is impossible due to decaying dynamics, will the algorithm return the error message "restart".

Starting self-optimization



Information on starting self-optimization is available in the section "Operating page CONTROL".



NOTE!

An adjustment of the setpoint for a start from automatic mode results in a wrong evaluation of the process. However the setpoint can be changed at any time during the self optimization. However as opposed to the CONTROL controller this is not necessary.

Cancellation of self optimization

- **Click on the button "Switch over automatic mode/manual mode"** If the  button has not been locked (1 signal on the PIDMA input "di_oplock"), then the self-optimization can be ended at any time. Click on the  button to end self-optimization.



NOTE!

Alternatively self-optimization can be stopped in any case via the button "Start/stop self optimization" Additional information in this regard is available in the section "Operating page control".

Start in manual mode or in automatic mode

The PIDMA optimization algorithm does not distinguish between a start in manual mode or a start in automatic mode. In either case you must ensure stable conditions in the plant,

Nevertheless in most cases better optimization results are obtained in manual mode. The reason is that in manual mode usually conditions are achieved in the plant that are more stable, because in automatic mode the PIDMA controller regulates with parameters that have not yet been optimized until the beginning of the controller output pulse.

For transition to manual mode the last controller output is transferred as manual controller output and maintained during the estimation times.

PIDMA optimization sequence

1. After starting the self-optimization process first drift detection and then noise signal detection is executed.
2. In the second phase the controller output will be changed by the step-wise change $dAVOpt$. If the process value has changed by more than $PVlimit$ then the controller output will be reset to the original value.
3. In the third phase the PIDMA waits for the maximum value of the rising process value.
4. Thereafter it observes the process value in the fourth phase of decay. During this period an assessment of the remaining time until conclusion of the optimization attempt is output.
5. After a successful adaptation attempt the controller goes into automatic mode and controls the setpoint with the newly determined parameters. The parameter $Ores$ indicates the result with which self-optimization has been concluded.



NOTE!

If self-optimization concludes with an error (display: "Ada_Err"), then you must manually end self-optimization mode (see the section "Cancel self-optimization"). As long as self-optimization is not concluded the steady state controlled value will be output.



NOTE!

After successful self-optimization the parameter "CSpeed" can be used to achieve stronger or weaker attenuation if the system was optimized with the setting for "CSpeed = Normal". In addition merely an increase or reduction of "kP" should be considered. After manual change of the control parameters switch over of "CSpeed" no longer has an effect. The parameter "CSpeed" is changed in the BlueDesign parameter dialog.

Sequence of self-optimization for heating and cooling processes

For 3-point controllers, split-range controllers, and their mixed forms different control gains cannot be specified with PIDMA for heating and cooling. Consequently there is no two-phase optimization attempt in this case.

I-5.7.10 Operating page A_PROG

The operating pages of the analog program generator ("A_PROG") provide information about the running programs and permit interventions in the program.

Properties

- **Process value display:** The process value of the current track will be displayed.
- **Runtime display:** The remaining runtime for recipe and segment, as well as total runtime of the recipe are displayed.
- **Manual mode:** If provided by the application developer switch over between automatic mode and manual mode is possible.
- **Function switch:** Freely configurable button.
- **Change programs/recipes:** The user can intervene directly in the program execution and can also change recipes.



NOTE!

Additional information on the program generator is available in the function block reference.

Overview

- 1 Title
- 2 Recipe name
- 3 Setpoint (SP)
- 4 Physical unit
- 5 Segment number and segment value range (beginning value and end value)
- 6 Net program time
- 7 Remaining program time
- 8 Remaining segment time
- 9 Button "Leave Operating Page"
- 10 Display process value (PV)
- 11 Button "Alarm"
- 12 Button "Call parameter page"
- 13 Button "Track change"
- 14 Button "Supplemental commands"
- 15 Button "Function key"
- 16 Button "Change operating mode"
- 17 Status line program generator (operating states: "end", "run", "reset" and "stop")



| A_PROG - Parameter | |
|--------------------|----------|
| Recipe | Rec 1 |
| Reset setpoint | 0 |
| Segment type 1 | Time |
| Time/Gradient 1 | 10:00:00 |
| Setpoint 1 | 50 |
| Segment type 2 | Time |

Fig. 73: Program generator (overview and parameters)



NOTE!

The button "Function key" (Fig. 73/15) depends on the application configuration. A typical use of the function key is program start and program stop. Additional (general) information is available in the function block reference, more detailed information on using the function key in its application is provided in the application documentation.

Operation



NOTE!

Operation of the program generator depends on the application development. Therefore the following descriptions should only be understood as examples. More detailed information on operating the application is provided in the documentation for the respective application.

Select status

The program generator can take on four states:

- **stop:** The program generator has been stopped.
- **reset:** The program generator has been stopped and has been reset to segment 0 of the recipe.
- **run:** The program generator has been started.
- **error:** A program error has occurred.
- **end:** The program generator has reached the end of the program.

Proceed as follows to change the status of the program generator:

1. **Call the "Commands" selection list:** Tap on the button "Supplemental commands" (Fig. "Program generator"/14).
You will see a selection dialog; the commands displayed here depend on the actual operating status of the application and the program configuration.

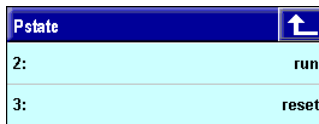


Fig. 74: Selection list "Pstate"

2. **Select command:** Select the desired command here.

Select recipe



NOTE!

The possibility of selecting a recipe in accordance with the following description depends on the application environment.

The recipe can be changed via the program generator. Note that recipes can only be changed if the program generator is in the "reset" state. To do this, proceed as follows:

1. **Call the "Command" selection list:** Tap on the button "Supplemental commands" (Fig. "Program generator"/14).
2. **Select "Stop" status:** Select the status "stop".

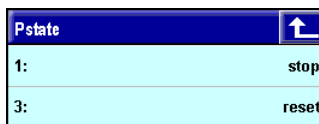


Fig. 75: Selection list "Pstate"

3. **Select "Reset" status:** Select the "reset" status (see Fig. 74).
4. **Calling the "Recipe" selection list:** Tap on the current recipe name (Fig. "Program generator"/2).
The recipe selection list will be displayed. Note: If the application technician has not given the recipes special names, then the system will give these recipes an automatically generated name that consists of the characters "Rec" and a consecutive number (e.g. "Rec 2").



Fig. 76: Select recipe

- Selecting a recipe:** Tap on the desired recipe.

Directly changing programs



DANGER!

Program changes pose an injury hazard!

Faulty intervention in applications can result in an uncontrolled and/or unforeseeable operating sequence (as with any electronic controller system). Death, serious injury, or significant property damage can be the result.

Therefore:

- Prior to making any program change consider the effects of the change and ensure that appropriate measures have been taken.

If provided by the application developer you can directly intervene in the program execution on the main page. This affects the setpoint, the segment selection and the segment times.

Proceed as follows to select a value:

- Select value:** Click on the appropriate value to select it. The numerical value editor will start (see the following fig.).

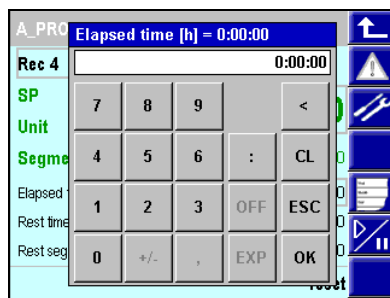


Fig. 77: Change program

- Enter new value:** Enter the new value and close the editor by clicking on "OK".

Change recipe



DANGER!

Recipe changes pose an injury hazard!

Faulty intervention in applications can result in an uncontrolled and/or unforeseeable operating sequence (as with any electronic controller system). Death, serious injury, or significant property damage can be the result.

Therefore:

- Prior to making any recipe change consider the effects of the change and ensure that appropriate measures have been taken.

Recipes are comprised of segments that contain value pairs (time/setpoint). If provided by the programmer you can change the time as well as the setpoint.

To do this, proceed as follows:

1. **Call the "Parameters" operating page:** Tap on the button "Call parameter page" (Fig. "Program generator"/12) to call the "Parameters" operating page.

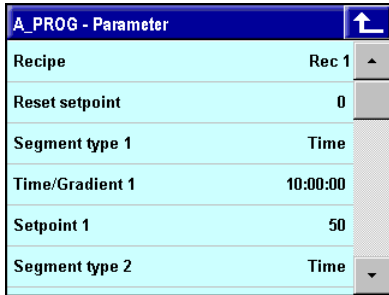


Fig. 78: Program generator ("Parameters" operating page)

2. **Select recipe:** Now you must specify which recipe you want to change. The currently selected recipe will be displayed on the first line (In the above fig.: "Rec 1").

Tap on this entry, if you *do not* want this recipe but rather want to edit a different recipe.

Now you see a list of recipes (see Fig. below), Click on the desired recipe to select it.



Fig. 79: Select recipe

3. **Change recipe:** Now you see the parameter dialog for the selected recipe. Tap on the value (time, setpoint) that will be changed and enter the new value. The appropriate editor will be displayed to edit the value (numerical value editor or binary number editor).

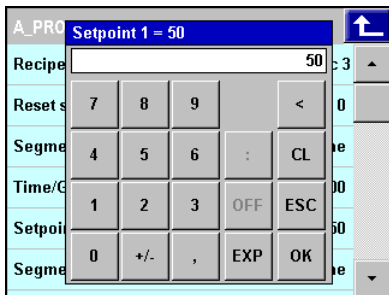


Fig. 80: Change recipe



NOTE!

Recipe changes during program execution only have an effect if they have been made in the current recipe or in a future recipe.

Start/restart recipe



NOTE!

The possibility of selecting a recipe in accordance with the following description depends on the application environment.

Proceed as follows to start the program:

1. **Call the "Commands" selection list:** Tap on the button "Supplemental commands" (Fig. "Program generator"/14).
2. **Select "run" status:** Select "run" status to start the program.

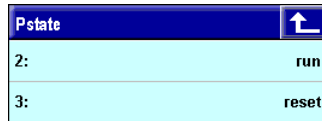


Fig. 81: Select "run" status

If the program generator has been configured so that after execution of the recipe the recipe will stop (the message "end" (Fig. "Program generator"/7) will be displayed), then the recipe can be restarted manually. To do this the program generator must be set to the beginning with the command "Reset".

Proceed as follows to restart the recipe:

1. **Call the "Commands" selection list:** Tap on the button "Supplemental commands" (Fig. "Program generator"/14).
2. **Select "reset" status:** Select "reset" status (see Fig. 81).
3. **Call the "Commands" selection list:** Tap again on the field "Status program generator" to open the selection list again.
4. **Select "run" status:** Select "run" status to restart the program.

Stop program

Proceed as follows to stop a program:

1. **Call the "Commands" selection list:** Tap on the button "Supplemental commands" (Fig. "Program generator"/14).
2. **Select "stop" status:** Select the status "stop".

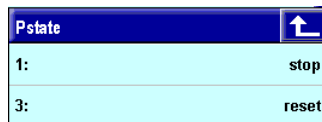


Fig. 82: Select "stop" status

Continue program

Proceed as follows to continue a program (if the recipe has been completely executed, you must use the process that is described above in the section "Restart recipe").

1. **Call the "Commands" selection list:** Tap on the button "Supplemental commands" (Fig. "Program generator"/14).
2. **Select "run" status:** Select the status "run".

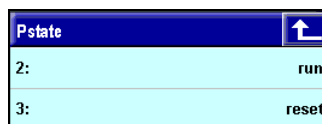


Fig. 83: Select "run" status

Starting/stopping a program with function key

Depending on the application configuration you can start or stop a program with the function key (Fig. "Program generator"/15).

Click on the function key to start or stop the program.

Working with multiple tracks

If at least one additional track is available then you can change the track by clicking on the button "▶▶" (Fig. "Program generator"/13).

Additional clicks take you to the available additional tracks. To return to the master track: Tap on the "▶▶" button until the master track is again displayed (the pages are arranged in a circular series).

I-5.7.11 Operating page D_PROG

The operating pages of the digital program generator ("D_PROG") provide information about program execution and permit intervention in the program.

Properties

- **Process value display:** The process value of the current track will be displayed.
- **Runtime display:** The remaining runtime for recipe and segment, as well as total runtime of the recipe are displayed.
- **Manual mode:** If provided by the application developer switch over between automatic mode and manual mode is possible.
- **Function switch:** Configurable button.
- **Change programs/recipes:** The user can intervene directly in the program execution and can also change recipes.

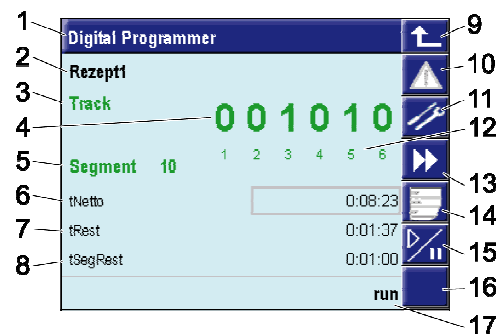


NOTE!

Additional information on the program generator is available in the function block reference.

Overview

- 1 Title
- 2 Recipe name
- 3 Name - forced value
- 4 Forced value
- 5 Segment number
- 6 Net program time
- 7 Remaining program time
- 8 Remaining segment time
- 9 Button "Leave Operating Page"
- 10 Button "Alarm"
- 11 Button "Call parameter page"
- 12 Bit position
- 13 Button "Track change"
- 14 Button "Change operating status" (selection menu is displayed depending on the application)
- 15 Button "Function key" (the meaning of the button is specified in the configuration)
- 16 Button "Switch over automatic mode/manual mode"
- 17 Status line program generator
(Operating states: "man", "end", "reset" and "stop")



| D_PROG - Parameter | |
|--------------------|---------|
| Recipe | Rec 1 |
| Reset control bits | 000000 |
| Segment time 1 | 0:00:00 |
| control bits 1 | 010100 |
| Segment time 2 | 0:00:00 |
| control bits 2 | 001100 |

Fig. 84: Program generator (overview and parameters)

Operation

Use the button "Call parameter page" (Fig. "Program generator"/11) to call the parameter page. Here the recipe name, the six control tracks, and the segment parameters of the currently active recipe are displayed.

Recipe selection

Tap on the "Recipe" button to display a different recipe.



NOTE!

This process is possible at anytime, as it does not cause switch over of the active recipe.

General operation



NOTE!

Basic explanations for operating the program generator are available in the section "Operating page A_PROG".

I-5.7.12 Operating page PROGRAMMER

The operating pages of the universell program generator ("PROGRAMMER") provide information about the running programs and permit interventions in the program.

Properties

- **Display program:** The names of program and segment will be displayed.
- **Display set-points:** The effective set-point of all tracks will be displayed. For analog tracks a trend plot is displayed.
- **Runtime display:** The remaining runtime for segment, as well as total runtime and estimated end time of the recipe are displayed.
- **Manual mode:** If provided by the application developer switch over between automatic mode and manual mode is possible.
- **Change programs/recipes:** The user can intervene directly in the program execution and can also change recipes.
- **Additional information:** There are more operating pages available, e.g. individual pages with details for each track or an overview of the total program.



NOTE!

Additional information on the program generator is available in the function block reference.

Programmer operation

Operating structure

The PROGRAMMER has several operating pages selectable on a main page in the menu of operating pages. If the **hide** input is soft-wired, no PROGRAMMER operating page is displayed.

As shown in the following diagram, there is a main page for the programmer on which a branch to the detailed pages of the tracks can be made by tapping the relevant key. Changing to the parameter page is possible from the detailed pages of the tracks. The editing page is selectable only, when digital input **p_show** is soft-wired and set.

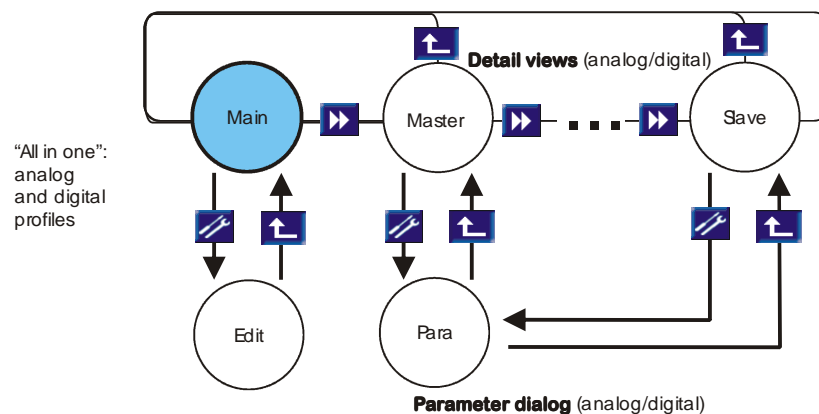
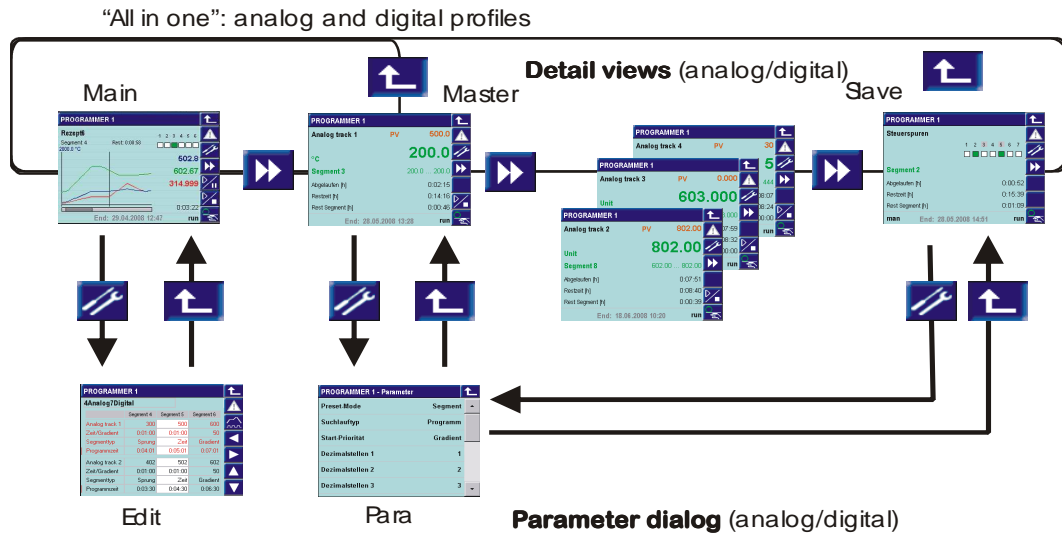


Abb. 85 Operating structure



86 Operating structure

Main operating page



Abb. 87

Keys on the right section of the operating page:

- Key (line) 1: return
- Key 2: Access to the alarm page
- Key 3: Access to the parameter page
- Key 4: Change to the detail page
- Key 5: Programmer control
- Key 6: Programmer control selection
- Key 7: Auto / manual switch-over

Display on left section of main page:

0: Programmer name

1: Active recipe

2: Active segment

3: Remaining time of active segment

4: Current status of control tracks 1 to 6 (status)**5 - 8: Current setpoint of analog track (up to 4)**

9: Elapsed program time (since start)

10: Maximum setpoint (SPhi1) and unit (Unit1) of master

11: Time stamp in the program curve

12: Setpoint curve in the program

13: Bargraph of program time (bar length to overall length as displayed time section to overall program time)

14: Display of manual operation "man"

15: Estimated program end time

16: Program status: run / stop / reset / search / halt / end / error

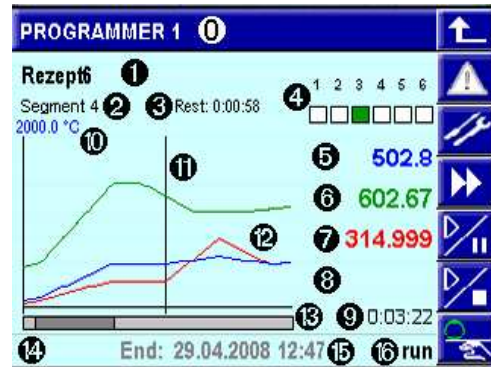


Abb. 88

The items marked in bold letters may include controls with variable values

Display on the main page

- Up to 4 setpoints with their curves are displayed; a vertical line marks the current time in the program. The visible time range is defined in configuration **TChart**. It is always updated, and the current position with the vertical time stamp remains always in the medium range.
- The Y-axis is determined by scaling of the master track with its maximum and minimum setpoint (**SPhi1**, **SPl01**). Depending on configuration, the slave tracks are scaled like the master track, or provided with their own scaling. The scaling of individual slave tracks is not shown on the display.
- The setpoints belonging to the current time marked on the trend curve are displayed in the same colour as the setpoint curves on the right. The current program time is shown as a value below the current setpoints.
- The bar of the bargraph indicates the visible time range in relation to the overall program time.
- Items shown with a frame can be changed by tipping with your finger:
 - Recipe (only in status `reset`)
 - Setpoints (only during manual mode `man`)
 - Control tracks (only in manual mode `man`)
- An overview of the status of the first 6 control tracks is given by the "LEDs" (on / off, "LED" = Square at beginning of the line). Details and further control tracks are found on a list which can be called up by tipping on the LED field with your finger. All control tracks are listed with status and name.

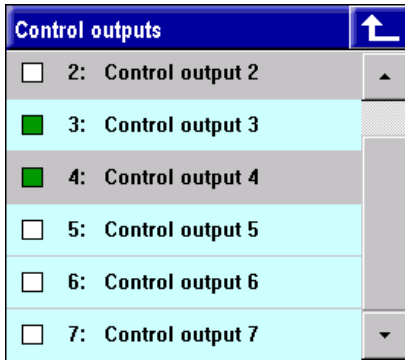



Abb. 89

Control track operation:

1. Symbol: white = off, green = on
2. Number of control track (1 ... max. 16)
3. Name of control track. The individual control tracks can be switched over: gray = manual mode, blue = automatic mode, if at least 1 control track is in manual mode.

- Tipping on a setpoint opens a window with display of track name, setpoint and unit.
- Switch-over to manual mode from the main page, marked by display "man" on the status line, puts all analog and digital tracks into manual mode. The variable items are marked in gray colour: setpoints are shown with a gray frame, control tracks are shown with a gray background.
 - Tipping on a setpoint opens a numeric pad for changing the setpoint.
 - Tipping on the LED field opens the list of control tracks. Change the status (off/on) of an individual control track by tipping on the corresponding icon. Tip on the name of the control track to switch it over between automatic and manual operation (gray background).

In status line "man" is displayed as long as at least 1 track is in manual operation. Tipping on  with "man" displayed all tracks are switched to automatic operation.

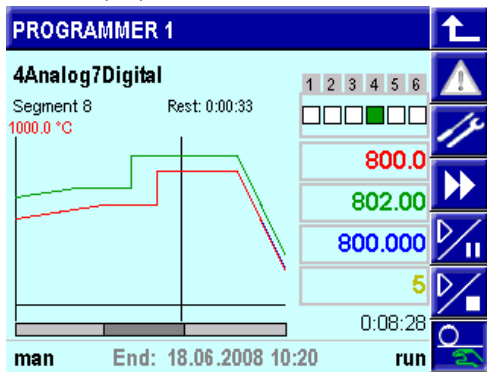






Abb. 90

Tapping  on the main page switches all tracks to manual or to automatic operation in common.

- The program can be set to a preset value. Preset to a segment or to a time in the program is possible. Preset can be activated via the profile overview page ( displays the list: "reset / preset / search", **preset** changes to the overview page). Tap the navigation keys to select the required time or segment. When leaving the page, the preset is or isn't activated ("OK" / "No"). If the preset is activated, the elapsed time is adapted and switch-over to the main operating page occurs (see operating page "Review profiles").
- Program setting on the operating page: Direct access to the recipe parameter setting (= editing page) is enabled, when control input **p_show** = „1" is set at the programmer function block. Access is using the parameter key .
- The **run / stop / reset / preset** states are always valid for the overall PROGRAMMER, i.e. for all tracks. They can be switched over centrally on the main operating page, unless they are determined via control inputs.

Detailed pages of the analog tracks

When manual mode is activated on the operating page of a track (key ) , only this track is switched to the manual mode. All other tracks remain in the automatic mode.

**NOTE!**

On the status line of the main operating page, "man" is displayed, if at least 1 analog or digital track is in the manual mode.

The setpoint of a track can be adjusted only, if the track is in manual mode.

A search run can be started in the current segment. Additionally, a program search run can be started on the operating page of the master track (track 1).

| | | | |
|-----------------------|-----------|------------------------------|-----------|
| PROGRAMMER 1 | 1 | | 12 |
| Analog track 1 | 2 | PV | 3 |
| | | 500.0 | 13 |
| °C | 4 | 200.0 | 14 |
| Segment 5 | 5 | 200.0 ... 200.0 | 15 |
| Elapsed time [h] | 6 | 0:02:15 | 16 |
| Rest time [h] | 7 | 0:14:16 | 17 |
| Rest segment [h] | 8 | 0:00:46 | 17 |
| 9 | 10 | End: 28.05.2008 13:28 | 11 |
| | | run | 17 |

Abb. 91

- 1: PROGRAMMER name
- 2: Track name
- 3: Process value (if any) as a PV with value
- 4: Setpoint unit and **current setpoint**
- 5: Segment name/segment start and end value
- 6: Elapsed program time (master track)
- 7: Remaining program time (master track)
- 8: Remaining segment time (master track)
- 9: Display of manual mode: "**man**"
- 10: Estimated time for program end (master track)
- 11: Program status: stop, run, reset, search, error, halt (master track)
- 12: Return key
- 13: Key for opening the alarm page
- 14: Key for opening the parameter page
- 15: Track change key
- 16: Search run key (master track: selection dialogue)
- 17: automatic/manual switch-over key

Detailed pages of control tracks (digital tracks)

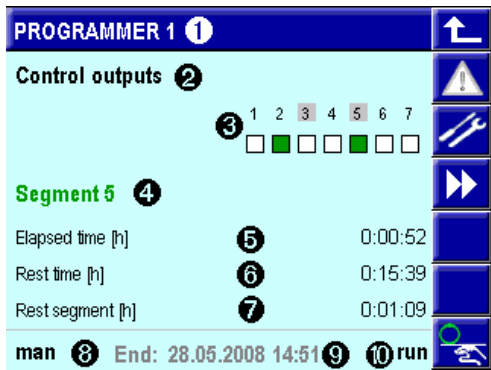





Fig. 92

- 11: PROGRAMMER name
- 12: Display: Control track operating page
- 13: Display of control tracks:
gray marking = control track is in manual mode
green = control track on, white = control track off
- 14: segment name
- 15: Elapsed program time
- 16: Remaining program time
- 17: Remaining segment time
- 18: Status line (only display) with
- 19: Estimated time for program end
- 20: Program status: stop, run, reset, search, error
- 21: Return key
- 22: Key for opening the alarm page
- 23: Key for opening the parameter page
- 24: Key for track changing
- 25:  auto/manual switch-over key

- The control tracks are displayed as LEDs.
- The name of the segment in which the master track works instead of the current segment number is displayed.
- The time estimated for program end is displayed on the status line.
- When tipping on the LED field, the control track list is opened. Each defined control track is displayed with number, name and a symbol for the status (off/on).
- Tipping on key , the **automatic/manual** key, also opens the list of control tracks. Tip on the name of the control track to switch it over between automatic and manual operation. On the list, the line is shown with a gray background during manual mode.
- When activating the manual mode on the operating page of control tracks () , only the control track selected on the list is switched to manual operation. All other tracks remain in the automatic mode.
- The status of a track can be changed only when it is in manual mode.
- Manual mode of the control tracks is shown by a gray field around the number of the control track. During manual operation, change the off/on status of the individual control track by tipping on the corresponding icon. Additionally, "man" is displayed on the status line, when at least 1 control track is in the manual mode.

Programmer parameters

The parameter page can be opened from the detailed pages of the tracks. On these pages, basic functions are listed, for example, the **search run type**, and can be changed. Some parameters are valid for the overall PROGRAMMER (e.g. **Preset-Mode**), others are valid for single tracks (e.g. setpoint ranges).

| PROGRAMMER 1 - Parameter | | ↑ |
|--------------------------|----------|---|
| Preset mode | Time | ▲ |
| Search mode | Segment | |
| Start priority | Gradient | |
| Decimals 1 | 1 | |
| Decimals 2 | 2 | |
| Decimals 3 | 3 | ▼ |

See table: Parameters

Abb. 93

Editing page

All recipe parameters (analog and digital tracks) are edited on a **single** scrollable page. When opening the page, the active recipe is displayed. Tap on the recipe name to switch over to other recipes.

The upper part always shows the master track in the colour configured with **Color1**. In the lower part, another track is displayed. For track paging, tap keys ▼ and ▲.

The following items can be changed:

- Recipe name (i.e. file selection)
- Setpoint/control track
- Time(s) or gradient
- Bandwidth



WARNING!

Changes on the editing page in the active recipe are effective immediately.

Changes in the program sequence may cause potentially hazardous conditions in the process.

For this reason:

- Consider the effects of the changes and take appropriate measures.

When leaving the editing page, a (memory) dialogue is opened. In this dialogue, the operator decides, if the changes are or aren't stored permanently. Changes in the current program sequence are taken into account in the further program sequence and deleted subsequently, unless they are saved.

Changes in another programm are not taken into account and are deleted, unless they are saved.

If the program is reset via the digital **reset** input or via the interface during editing on the editing page while the memory dialogue is still active, all changes are cancelled and the dialog is closed.

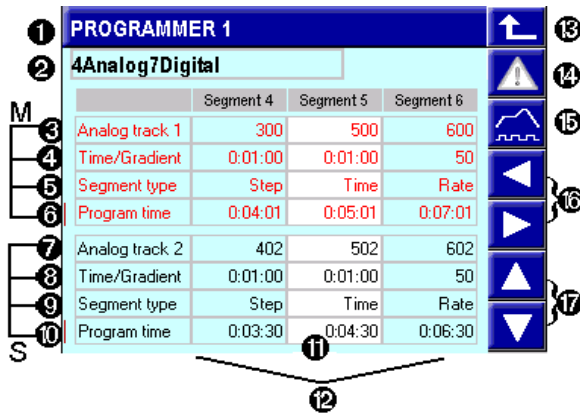


Abb. 94

For description of **segment types**:
see the Segment table.

Bandwidth:

If the separation between setpoint **SP** and process value **PV** exceeds the bandwidth parameter (**BW**), **halt** is activated (for description, the section Bandwidth monitoring).

- 1: PROGRAMMER name
- 2: Recipe (file name)
- 3 to 6: **Master track** (M=Master, always visible):
 - 3: Track name
 - 4: Depending on segment type: time/gradient
 - 5: Segment type (cannot be changed via operation)
 - 6: Change field, marked by a line. Switch-over by tipping:
 - a) Elapsed program time or
 - b) Bandwidth: Adjust the symmetric bandwidth per segment on the line.
- 7 to 10: one of the slave tracks (S=slave, 4 lines, paging with up/down keys), lines as 3 to 6:
 - 7: Track name
 - 8: depending on segment type: time/gradient
 - 9: Segment type (cannot be changed via operation)
 - 10: Program time/band width change field
- 11: Current segment (white background)
- 12: View and operation of 3 segments, scrolling by the up/down key
- 13: Return key
- 14: Key for opening the alarm page
- 15: Key for opening the overview page
- 16: Key for "Segment paging" right/left
- 17: Key for paging up/down slave tracks

Digital tracks on the editing page:

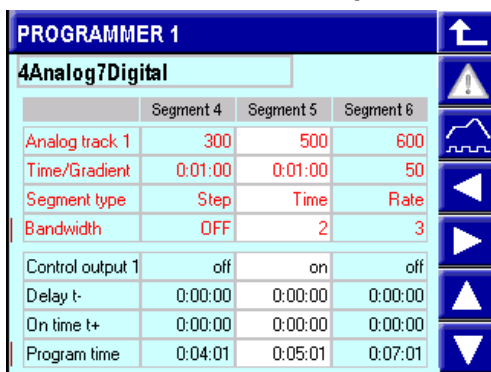


Abb. 95

Same picture as above, but

- 1) **Bandwidth** on the master display
- 2) **Control track** on the slave display.

Switch-on delay t- : delay after which the control track goes to its adjusted value.

Switch-on time t+ : The control track remains on its adjusted value during switch-on time **t+** .

For a description of parameters, see section Segment time under Control tracks.

on = control track is on, **off** = control track is off.

Programmer page Review profiles

To provide a program overview, the "Review profiles" programmer page is opened. Tap to select it on the editing page (tap to open the editing page in the main menu). On this page, you can check the

program, for example, after creation or after changes. The page shows the curves of the analog tracks with setpoints in a graph. Tap the 4 arrow keys to scroll within the segments (◀▶) or per segment (◀▶▶) throughout the program time. Apart from the program time as a reference, the segments with segment names, the analog tracks with the setpoints and the first 6 digital tracks with states are displayed.

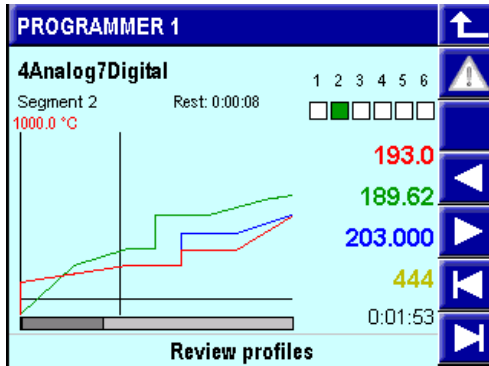




Abb. 96

The overview page is also used to make a preset for jumping to a defined program point. When tapping key  to select **preset** on the main page of the PROGRAMMER, the overview page is opened automatically. Tip on the arrow keys to navigate to the required position in the program and on the return key to leave the page. With the following prompt if a jump to the position is required, the preset can be activated or cancelled.

Note: paging with  is done by 1 pixel at a time (the time for each pixel is a function of the 80-pixel width for the configured visible time range).

I-6 Maintenance and cleaning



WARNING!

Danger of injury if personnel are not qualified to work on the device!

Improper handling of the device can cause serious injury and property damage.

Therefore:

- Only qualified personnel should perform maintenance work on the device.



DANGER!

Danger of injury due to unforeseeable system function sequences and movement sequences!

System components can be placed in movement during maintenance work, configuration work, or function checks, if they are not disconnected from the device.

Therefore:

If the device is taken out of service, if new or changed applications are loaded on the device, or if maintenance or a function check is performed, the following must be heeded:

- All system components must be disconnected from the device!
- All switched off system components must be safeguarded from being inadvertently switched on again!
- In general the effects of switching off the system must be taken into consideration and appropriate measures must be taken.



DANGER!

Danger of injury due to uncontrolled/unforeseeable operating sequences!

As with any electronic controller system, device failure can result in an uncontrolled and/or unforeseeable operating sequence. Death, serious injury, or significant property damage can be the result.

Therefore:

- Ensure that appropriate measures are in place every time the device is used.

I-6.1 Maintenance

I-6.1.1 Real-time clock

The KS 108 is equipped with a battery-buffered real-time clock; its charged status can be monitored.



NOTE!

The real-time clock can be checked using the PMA library XXXX block.

**CAUTION!**

Function restrictions for the real-time clock are possible if the wrong voltage is applied!

If the voltage of the real-time clock underranges 2.0V or exceeds 3.2V then the function of the real-time clock is endangered or the real-time clock can be damaged.

Therefore:

- Ensure that only the battery type specified below is used.
- Regularly check the voltage of the buffer battery.

I-6.1.1 Changing batteries

The installed battery must be replaced every 5 years, regardless of the charge status. This is the only maintenance task required.

**NOTE!**

Application data are lost when the battery is replaced. Therefore after replacing the battery you must reimport the application on the device.

The following charge states of the battery must be heeded:

| Voltage | |
|---------|---|
| 3.2 V | Typical voltage for a new battery. Do not exceed this value! |
| 3.0 V | Battery rated voltage |
| 2.5 V | Battery replacement necessary |
| 2.0 V | Battery must be replaced immediately to ensure the function of the real-time clock. |

**DANGER!**

Explosion hazard!

Therefore:

- Do not throw new or discharged batteries into a fire, do not recharge them, and do not dismantle them.
- Only use type CR1620 (3V lithium battery) batteries!
- When inserting pay attention to correct polarity of the battery.

**NOTE!**

The battery voltage can be checked with a block of the PMA library. When developing applications verification of battery voltage must be integrated that warns the user of a drop in voltage in good time.

**CAUTION!**

The battery holder can be destroyed by bending!

The battery is held by a spring bracket. This spring bracket can be destroyed by lifting.

Therefore:

- When changing the battery never lift the spring bracket.

1. **Unplug the power cable:** Unplug the power cable from the device (see chapter: "Connecting the device").
2. **Removing batteries:** Use insulated tweezers to remove the battery.
3. **Slide the batteries into the device:** Slide the batteries into the battery compartment. In this process pay attention to the correct polarity of the batteries!
4. **Connecting the power supply cable to the device:** Reconnect the power cable to the device (see chapter: "Connecting the device").
5. **Load the application program:** Load the application program (see the section "Loading the application program" in the manual "II Development Environment"),

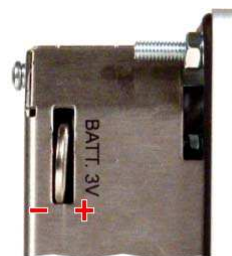


Fig. 97: Changing batteries

I-6.2 Cleaning



DANGER!

Danger of injury due to unforeseeable plant function sequences and movement sequences!

Unintentional or uncoordinated contact of touch sensitive displays can cause malfunctions or unforeseeable plant reactions.

Therefore:

- Switch off the device, the plant, or all plant components, prior to cleaning the device!
- All switched off plant components must be safeguarded from being inadvertently switched on again!
- In general the effects of switching off the plant must be taken into consideration and appropriate measures must be taken.



CAUTION!

Device damage due to improper cleaning!

The use of improper cleaning agents or cleaning improperly can cause significant device damage. The display in particular is extremely sensitive.

Therefore:

- Never use a high-pressure cleaner or steam jet to clean the device.
- Do not use aggressive cleaning agents, solvents, cleanser, or hard objects to clean the device.
- Always use extremely light contact pressure to clean the front panel and the display.

Procedure

Clean the front panel of the device with a lint-free, soft, slightly damp cloth.

I-6.2.1 Cleaning heavy fouling



NOTE!

If the device is heavily fouled, ethyl alcohol or isopropyl alcohol can be used to clean the screen in accordance with DIN 42115 Part 2.



CAUTION!

Danger of non-intended program reactions due to cleaning!

The screen of the *KS 108 easy* is touch sensitive. The cleaning process can trigger program reactions.

Therefore:

- Always have the menu "General Data" available in your application: Call the operating page "General Data/Clean Screen" before cleaning the screen.
- Prior to cleaning always call a screen page that cannot effect any undesired program reactions.

I-6.3 After maintenance

After maintenance work, before placing the device in service, you must ensure that proper operation is possible. Ensure that:

- No foreign objects are in the device.
- All connections have been established correctly.
- The enclosure cover is mounted.
- The PE is correctly connected.

I-7 Troubleshooting

**WARNING!**

Danger of injury if personnel are not qualified to work on the device!

Improper handling of the device can cause serious injury and property damage.

Therefore:

- Only qualified personnel should perform maintenance work on the device.

**DANGER!**

Danger of injury due to unforeseeable system function sequences and movement sequences!

System components can be placed in movement during maintenance work, configuration work, or function checks, if they are not disconnected from the device.

Therefore:

If the device is taken out of service, if new or changed applications are loaded on the device, or if maintenance or a function check is performed, the following must be heeded:

- All system components must be disconnected from the device!
- All switched off system components must be safeguarded from being inadvertently switched on again!
- In general the effects of switching off the system must be taken into consideration and appropriate measures must be taken.

**DANGER!**

Danger of injury due to uncontrolled/unforeseeable operating sequences!

As with any electronic controller system, device failure can result in an uncontrolled and/or unforeseeable operating sequence. Death, serious injury, or significant property damage can be the result.

Therefore:

- Ensure that appropriate measures are in place every time the device is used.

I-7.1 Fault table

| Fault | Presumable cause | Recommended measure |
|---------------------------------|------------------------|--|
| Device does not start | Power supply disrupted | <ul style="list-style-type: none"> ■ Check whether the power supply cable is connected, and whether the power supply is working. ■ Check the pin assignment of the power supply for correct polarity. ■ Check whether the supply voltage corresponds to the device's connection values. |
| Device does not boot completely | Device is defective | <ul style="list-style-type: none"> ■ First try to reboot the device. ■ If that does not help: Contact PMA |

If the measures described above are not successful, please contact PMA.

II Development environment

General

The following section provides an overview of the development environment for application development with the *KS 108 easy*. In addition you will learn how to execute typical tasks (such as changing parameters).

The development environment is comprised of the following components:

- KS 108 device with the PMA and BlueDesign runtime environment
- BlueDesign/PMA library
- BlueSimulation KS108
- IO/system

The development environment components are described later in the chapter and explanations are provided of how they interact.

Requirements

The following skills are prerequisite for working with the development environment:

- Basic knowledge of the operating system *Microsoft Windows™*
- Knowledge of the programming languages standardized in DIN EN 61131-3 (particularly the function block language)
- Basic knowledge of control technology

II-1 Installation and configuration

II-1.1 Installation

Use the CD that was supplied with the *KS 108 easy* to install the development environment. It contains the following installation programs.

- BlueDesign
- BlueSimulation
- Vario-Configurator
- Sample programs

II-1.1.1 Installing BlueDesign

Proceed as follows to install the "BlueDesign" development environment:

1. **Start installation program:** On the installation CD go to the directory "**<anywhere>**" and start the installation program "iBlueDesignE.exe".

You will see the "Welcome" dialog.

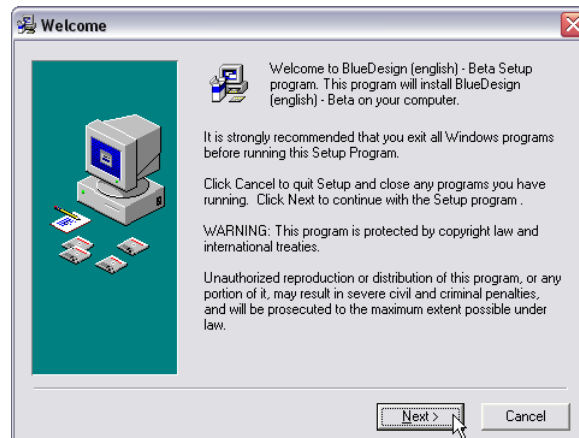


Fig. 98: Installation of BlueDesign, "Welcome" dialog

2. **Start installation:** Click the "Next" button to start the installation.

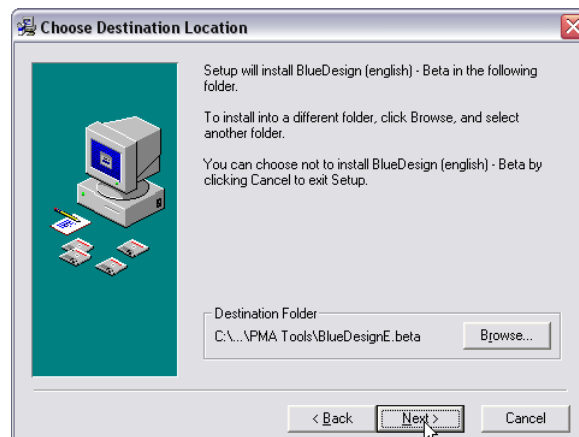


Fig. 99: Installation of BlueDesign, Select target directory

3. **Select target directory:** Select the installation directory. Click the "Browse ..." button if you want to change the default value. Use the "Next>" button to apply the input.

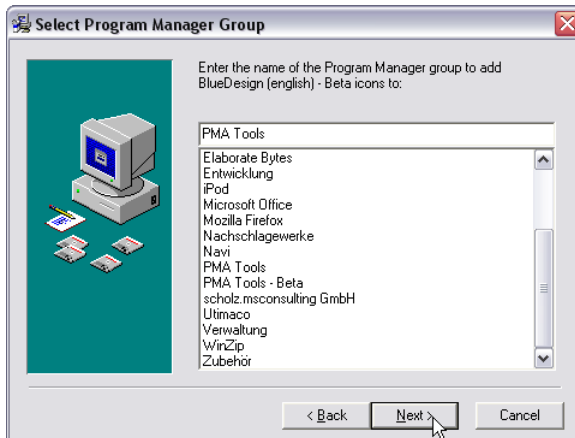


Fig. 100: Installation of BlueDesign, Select program group

4. **Select program manager group:** Select the program manager group from which the application should be started. Then click the "Next>" button.

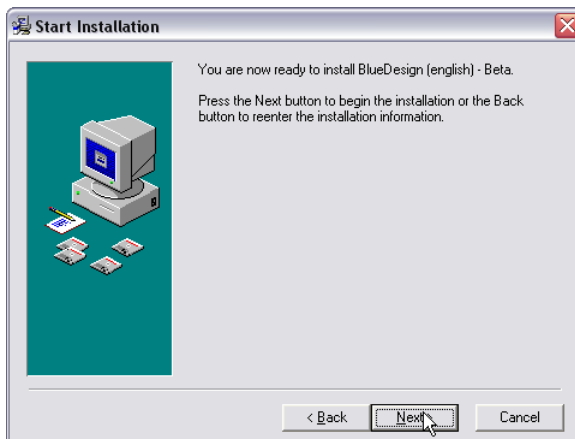


Fig. 101: Installation of BlueDesign, Begin installation

5. **Install program:** Click the "Next>" button to start the installation.

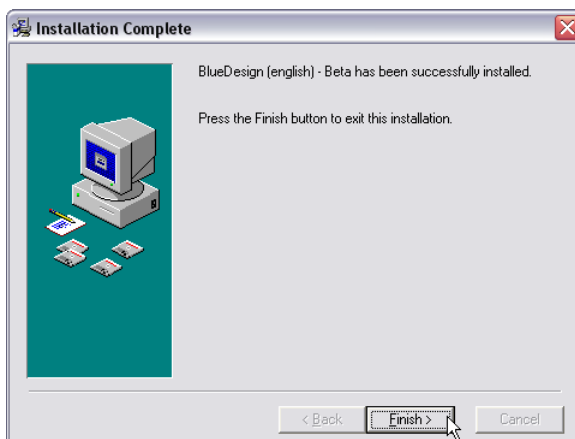


Fig. 102: Installation of BlueDesign, Finish installation

6. **Finish installation:** Click the "Finish>" button to complete the installation.

II-1.1.2 Installing Vario-Configurator

Proceed as follows to install the test program for the "Vario" IO system:

1. **Start installation program:** On the installation CD go to the directory "**<anywhere>**" and start the installation program "Setup.exe".
You will see the dialog for selecting the target path.
2. **Select target directory:** Select the installation directory. Click the "Browse" button if you want to change the default value. Use the "Next>" button to apply the input.

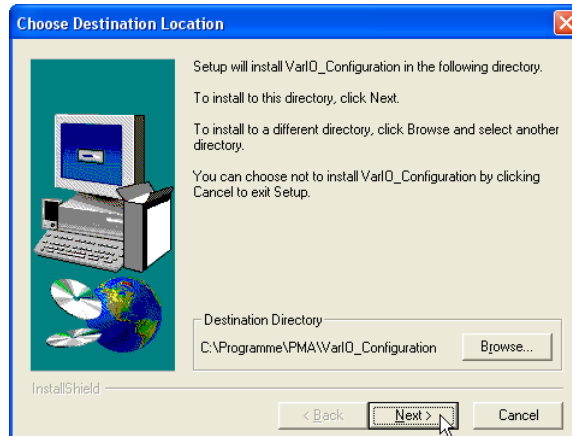


Fig. 103: VarioConfigurator dialog "Choose Destination Location"

3. **Select program manager group:** Specify the name of the program manager group and click the "Next>" button.
The installation will now begin.

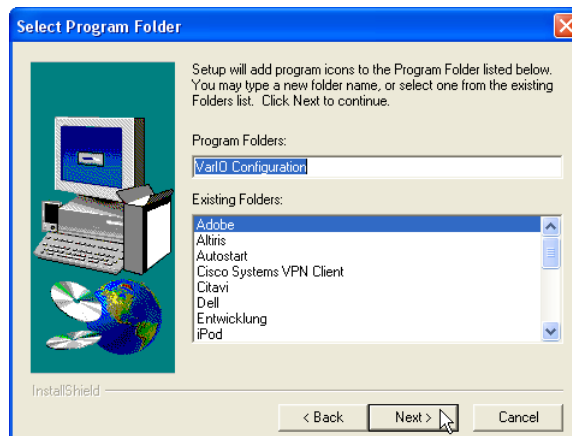


Fig. 104: VarioConfigurator dialog "Select program folder"

II-1.2 Licensing BlueDesign

Without valid license projects can be opened and edited in BlueDesign for a short time only. You fill in the license in the main menu selecting "?" and then menu item "Licence...". A dialog opens to show the registered license.

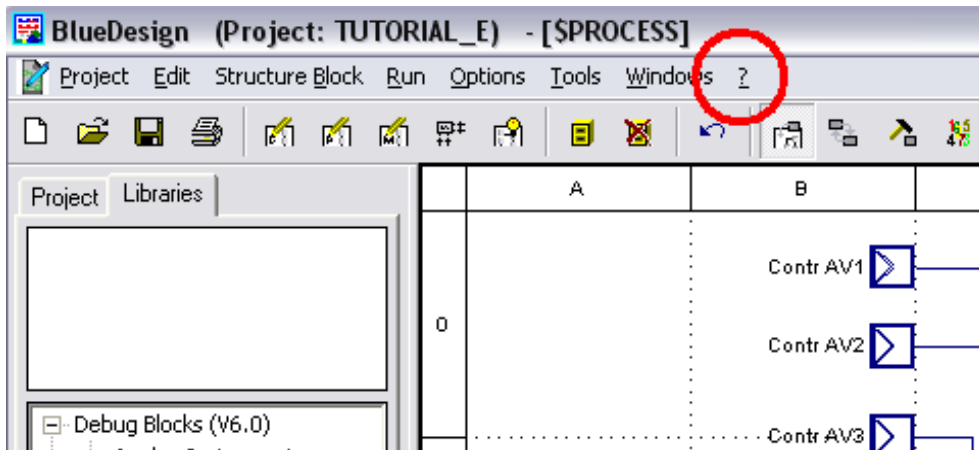


Fig. 105: BlueDesign menu "?" for calling license dialog

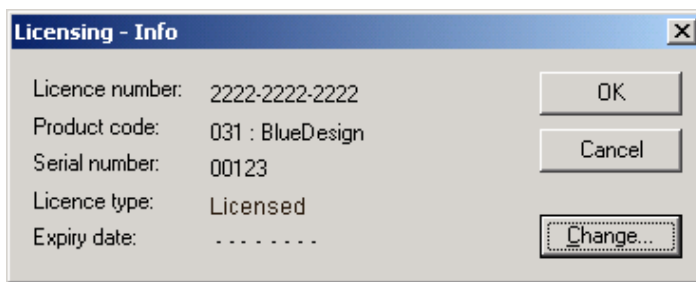


Fig. 106: BlueDesign dialog "License..."

The button "Change..." opens a dialog, in which you enter the license number.

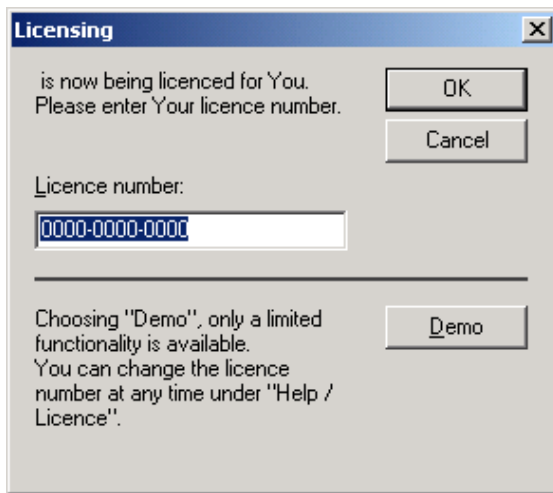


Fig. 107: BlueDesign dialog "(License) change..."

After quitting the dialog via "OK" BlueDesign shows a message affirming the new license.

II-1.3 Configuring BlueDesign

i *NOTE!*
 Information on the configuration of BlueDesign can be found in the section "Using the development environment".

II-2 The components of the development environment

The following section provides an overview of the components of the development environment.

If you have not yet worked with the PMA library or the development environment, it will be worthwhile to first read the chapter "A practical example". Here you will learn how the components interact based on a concrete example.

II-2.1 KS 108 easy device with runtime environment

The *KS 108 easy* is basically a computer that runs using the LINUX operating system. In principle the device could also be used as a "normal" LINUX computer. The KS 108 becomes a high-purpose multi-function controller primarily due to two software components:

- **BlueDesign runtime environment:** This enables execution of BlueDesign applications on the device. Application programs are created in the BlueDesign development environment and transferred to the KS 108. In the KS 108 they are executed in the runtime environment of the BlueDesign system.
- **PMA library:** This makes high-performance function blocks available for operating plants that are used by the BlueDesign applications.

II-2.2 BlueDesign

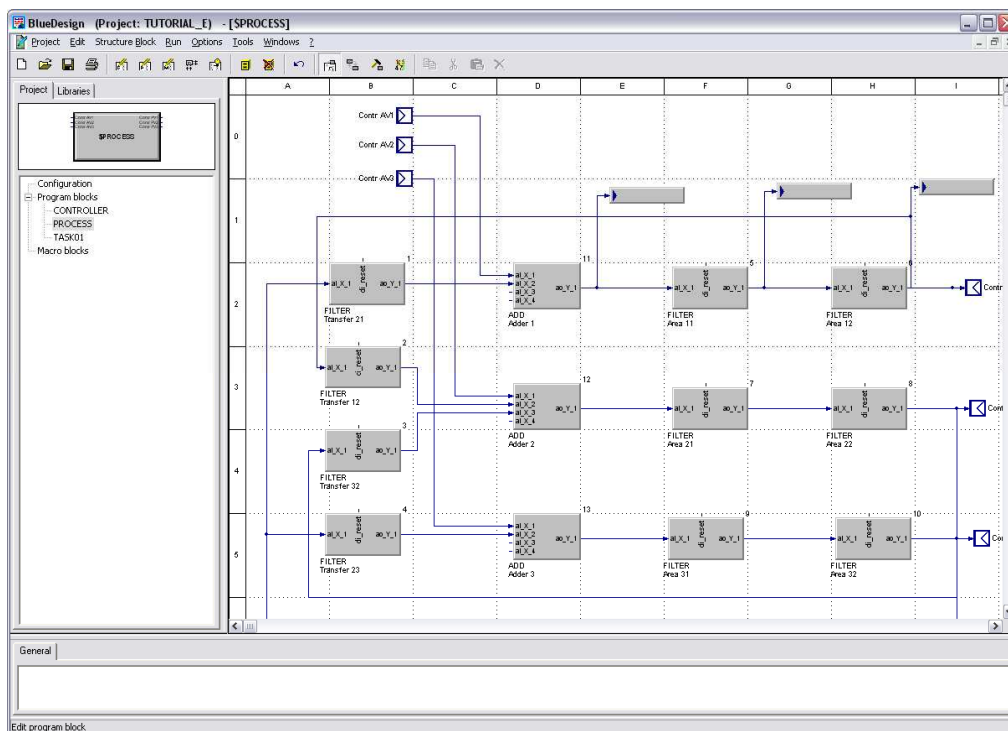


Fig. 108: BlueDesign

BlueDesign is a development environment for control technology applications. Applications are created in BlueDesign by selecting and connecting the pre-fabricated components (e.g. controllers or filters) to be used. Programming knowledge is not necessary.

BlueDesign supports the entire development cycle of a project:

- **Executing the application development:** Applications are composed of application blocks. An application can be structured according to various criteria. In addition, copy templates can be created, so that frequently needed application elements can easily be re-used.

- **Creating the user interface:** The user interface for the application is created on the basis of the PMA library using the graphical editor. Operating elements, displays and images are positioned on the workspace and configured here. The presentation is very similar to the later appearance on the user interface of the *KS 108*.
- **Setting parameters:** The PMA library functions can be used "out of the box" - as is. However in order to adapt them to your concrete requirements, you have to set parameters for the functions. This is also done using the BlueDesign software.
Value lists that offer possible options are displayed for numerous parameters. In addition, inputs are checked. If a value is not permissible, it is corrected to the next permissible value.
- **Testing the application:** Errors in applications can be debugged in two different ways: A connecting line between the blocks can be clicked with the mouse. The current value of the connecting line then appears at the mouse position.
Alternatively, specialized debugging modules from the PMA library can be used. Since the search for errors in applications is referred to as debugging, these blocks are called "debug blocks". These blocks can be used to display various information about the program during runtime. The display is shown only in the development environment (not on the display of the unit).

II-2.3 PMA library

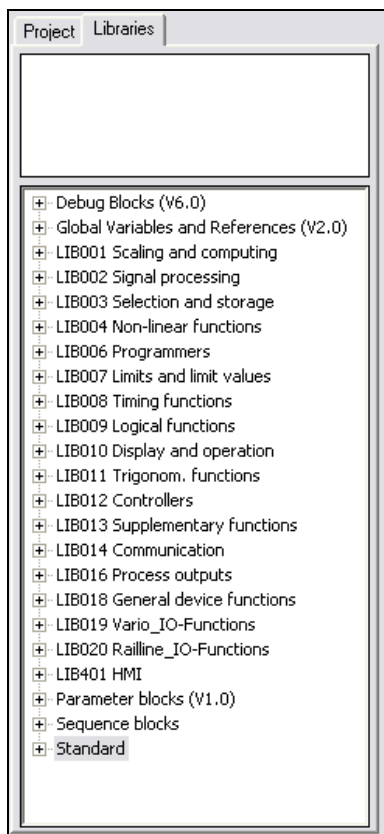


Fig. 109: PMA library

Applications are developed for the *KS 108 easy* with the BlueDesign development environment, on the basis of the PMA library.

The PMA library consists of a large number of function blocks. It covers all functions that are normally required for operating a plant. These include:

- Mathematical functions
- Logical functions

- Alarm and limit value functions
- Controllers

The PMA library helps you to develop programs in three ways:

- **Speed:** Applications can be more quickly developed, since they can be setup almost completely with functions from the library.
- **Quality:** In the PMA library you will find optimized functions that you can consider as "black boxes". Thus possible programming errors are avoided.
- **Visualization:** In addition, the library provides numerous functions with simple objects for visualization.

II-2.4 BlueSimulation

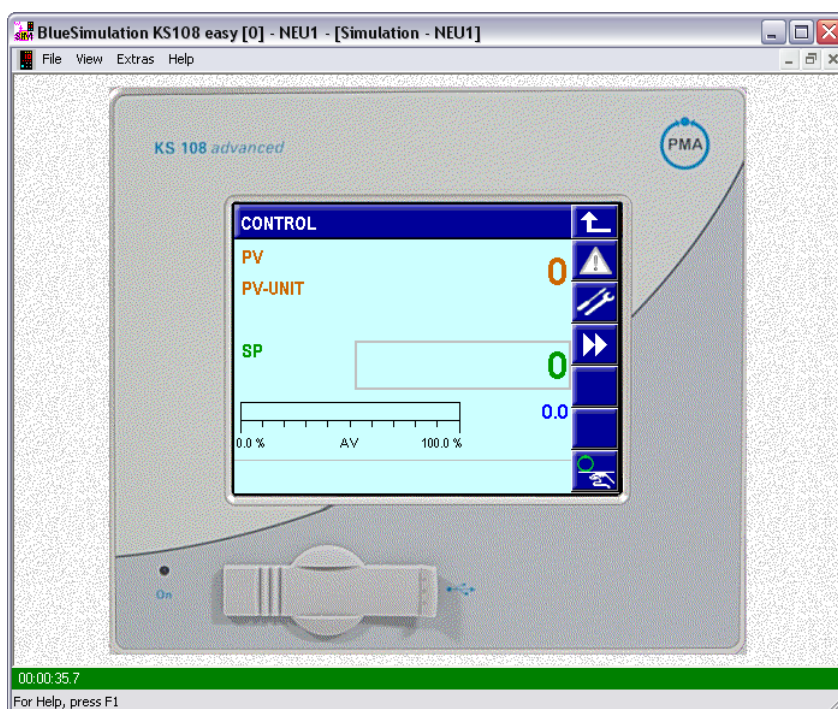


Fig. 110: BlueSimulation

The tool BlueSimulation simulates the *KS 108 easy* device. The behavior and displays correspond exactly to the KS 108. The simulator enables the development of applications and program testing without a device.

II-3 Working with the development environment

The following section provides instructions for typical work with the development environment.

If you have not yet worked with *BlueDesign*, it will be worthwhile to first read the chapter "A practical example". There you will learn to work with the development environment based on a concrete example. On the other hand, the following section describes specific tasks that come up when working with the development environment, and thus is provided more as a reference work rather than a systematic introduction.

Most of the commands used here can be initiated in different ways (e.g. via the menu, the button bar or a function key). The following chapter uses menu commands where this is possible. Information on other options can be found in the section *II-3.12.11* Using keyboard commands and in the online help.

II-3.1 Fundamentals

II-3.1.1 Project structure

As can be seen in the graphic, you will find three main categories on the "Project" tab that subdivide your project.



NOTE!

The "Project" tab is available only when the program is in "edit mode". Further information on the program modes can be found in the following section "BlueDesign operating modes".

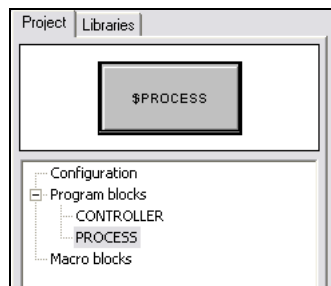


Fig. 111: Project structure in BlueDesign

- **Program blocks:** Program blocks are the basic units which make up your project. Program blocks contain blocks (e.g. controllers, filters, adders), connections between blocks and inputs and outputs. Your application can contain a maximum of 15 program blocks.
- **Configuration:** In the "Configuration" area you can make settings for the interaction of your program blocks. Each program block is designed as a separate sub-application or task, which means that it receives its own computing time from the processor. The basic procedure used by the runtime environment is to start the program blocks one after the other. However, you can also specify via the "Cycle time" that (some) program blocks are called at greater intervals (i.e. they are sometimes "skipped").

You can also specify the sequence ("Priority") in which the program blocks are started. These settings are made in the "Configuration" area.

- **Macro blocks:** Frequently you will use identical combinations of blocks in your program blocks. In such cases, macro blocks facilitate your work. You can use macro blocks to create a copy template. You can use the copy template just like any other block in your projects. Macro blocks themselves can also use other macro blocks. In addition, you can use macro blocks to structure your application.

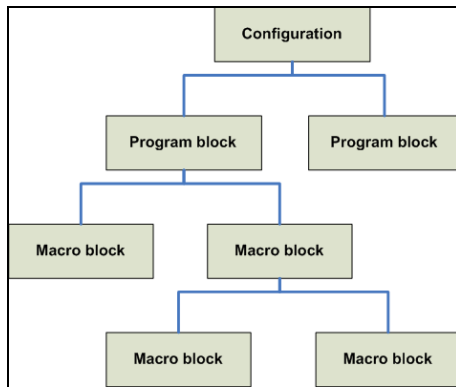


Fig. 112: Sample project structure

II-3.1.2 BlueDesign operating modes

Three operating modes are used for working in *BlueDesign*:

- Edit mode:** Edit mode is the standard mode for *BlueDesign*. The essential editing of a project is done in this mode. You use edit mode to edit and connect blocks, delete blocks, change their order or work with macro blocks. The blocks and their connections are edited on the workspace (Fig. 113/3). A tree structure of the project elements is displayed on the "Project" tab (Fig. 113/1). The "Libraries" tab (Fig. 113/2) contains a hierarchical list of the blocks that can be used in the project.

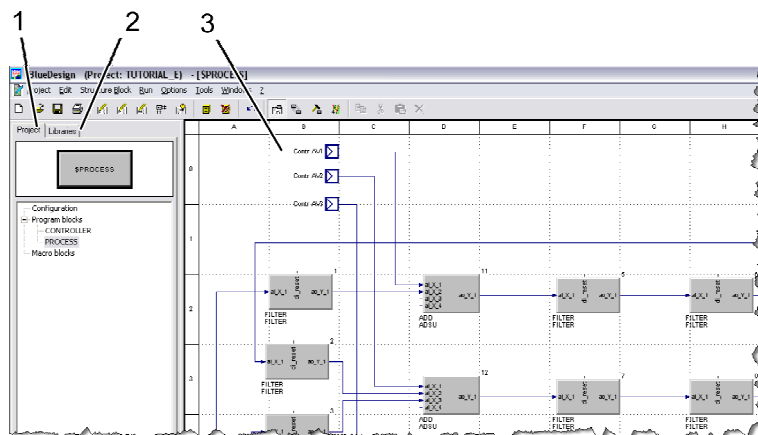


Fig. 113: Example: Edit mode

- Run mode:** In this mode, parameters can be assigned for each copy ("Instance") of a block (Example: Fig. 114/ 3). In addition, the computing time can that is allocated to each sub-application ("Task") can be specified (Example: Fig. 114/2). Basic editing of the projects is not possible in this mode. Instead of the project overview of the edit mode, in which the project is displayed according to categories, the logical structure of the application is displayed in run mode (Fig. 114/1). Components that are called (program blocks, macros, etc.) are displayed *below* the calling component.

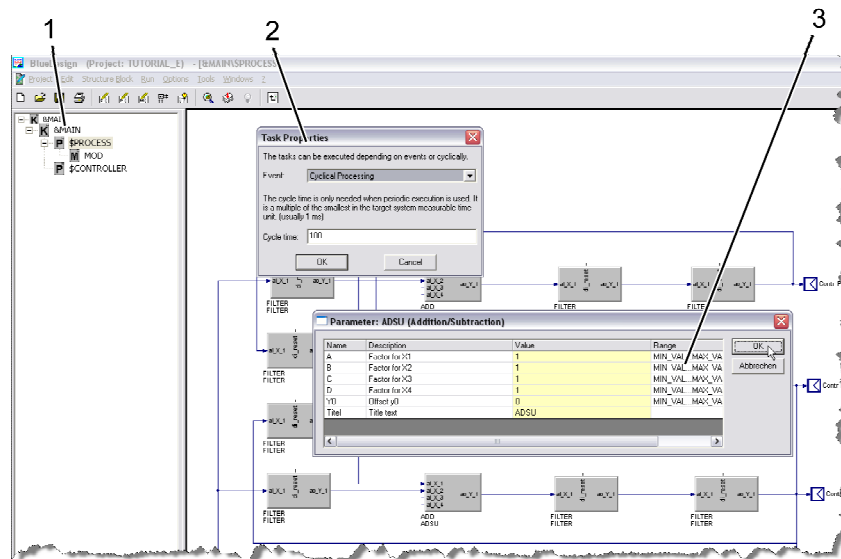


Fig. 114: Example: Run mode

- Online observation mode:** After loading and starting the program on a target system, the online observation mode is started automatically. In this mode it is possible to monitor and change the program flow. Debug blocks can be used to monitor the program flow, e.g. in order to display the current value of a signal (Fig. 115/2). In addition, clicking a connection displays its current value (Fig. 115/1). The program can be changed by writing values to the system online.

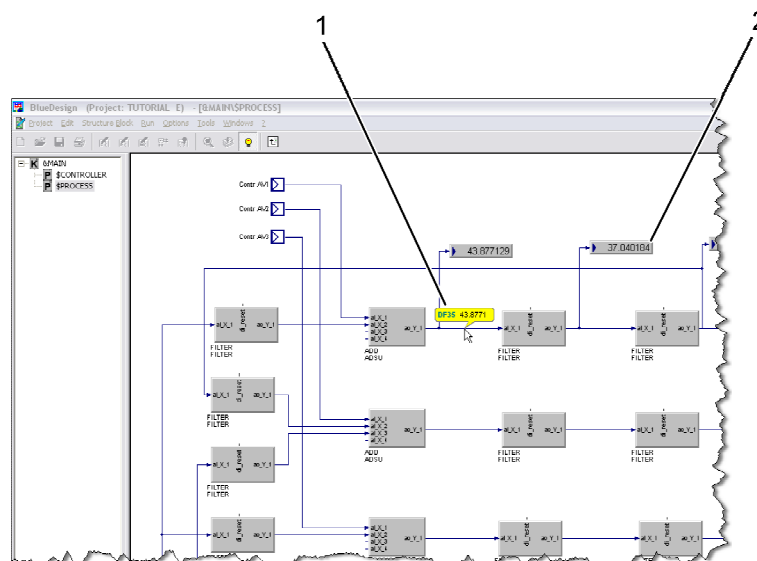


Fig. 115: Example: Online observation mode

What is run mode used for?

It was mentioned above that edit mode differs from run mode in that instance parameters can be assigned here. However, parameters can also be assigned in edit mode.

This is due to the following reason: program blocks can be used more than once in a project. In the example below, each program block is used twice.

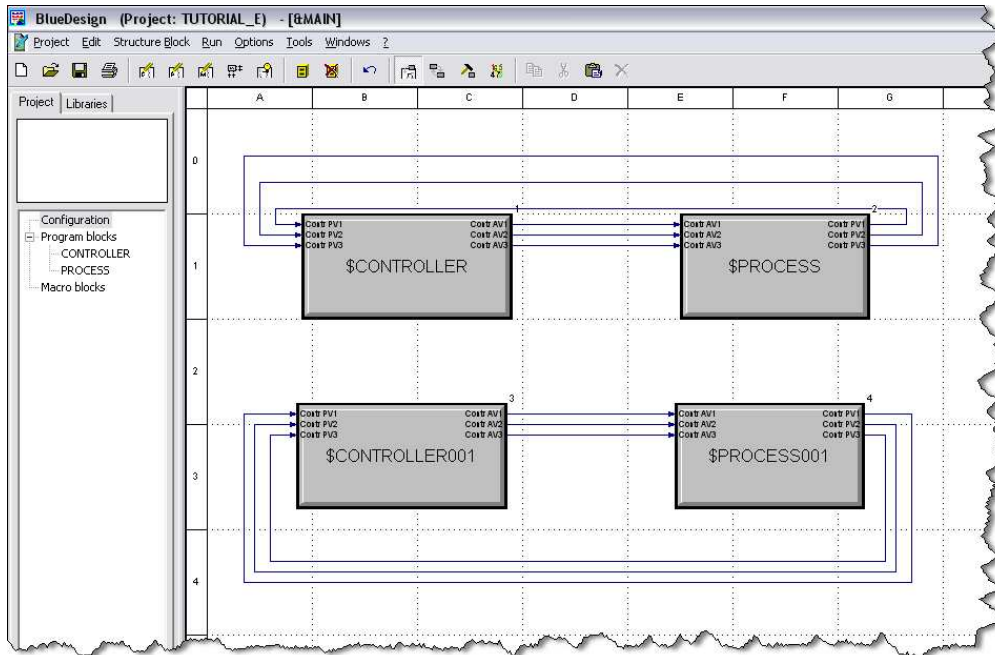


Fig. 116: Double use of program blocks

If parameters were defined only in edit mode, each copy of the program block would have the same parameters. This is obviously not desirable. Let us assume that there are the two oven simulations from the practical example (see the chapter "A practical example"). It is possible for the two ovens to have different properties (e.g. maximum temperature). Nevertheless, both ovens should be able to be implemented using the same program blocks.

II-3.2 Using operating modes

NOTE!
 General information on the operating modes can be found in the section "II-3.1.2 BlueDesign operating modes".

Projects can be edited with the following operating modes:

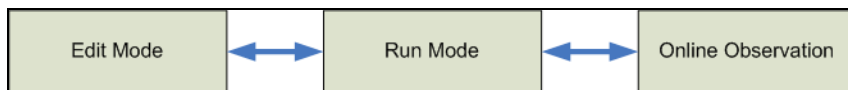


Fig. 117: System statuses

You can switch from edit mode to run mode and from run mode to online observation mode (and vice versa). Proceed as follows to switch modes:

II-3.2.1 Switching from edit mode to run mode

Start the menu command "Run/Enter" in order to switch to run mode.

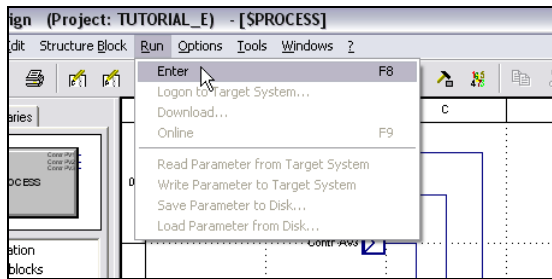


Fig. 118: Start run mode

II-3.2.2 Switching from run mode to online observation mode



DANGER!

Danger of injury or property damage due to unforeseeable plant function sequences and movement sequences!

If the *KS 108 easy* device is used with other devices/equipment then consequential actions can be induced on these devices/actuators, etc. by transferring the program.

Therefore:

- Consider the effects of a program update and ensure that appropriate measures are in place prior to establishing a connection!
- Prior to establishing a connection it is strictly necessary to ensure that the correct program is loaded.
- Prior to establishing a connection the error-free status of the program must be ensured.

1. **Logon to target system:** If you have not already done so, you first have to select and logon to the target system for which your application is intended.
For further information, see section II-3.12.1 Logon to target system.
2. **Switch to online mode:** Execute the menu command "Run/Online".
If the current application is not loaded on the KS 108 (or on the Simulator), the following dialog window will appear:

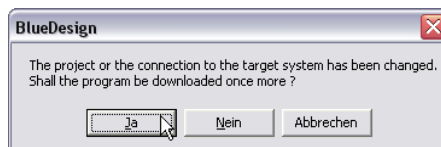


Fig. 119: Switch to online mode

Click the "Yes" button if you wish to transfer the modified program.

II-3.2.3 Switching to edit mode

Open the "Edit" menu. Select what you want to edit (configuration, program block or macro block).

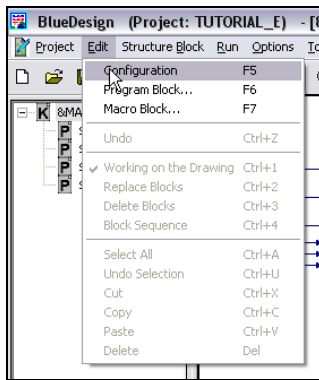


Fig. 120: Switch to edit mode

II-3.3 Creating a project

1. **Open dialog:** Open the "New Project" dialog with the menu command "Project/New".

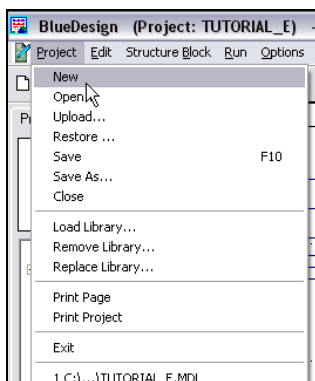


Fig. 121: "Project/New" command

2. **Enter project information:** Select the desired target system in the left area of the "New Project" dialog. Enter the project name in the "Project" field and specify the location for saving the project in the "Directory" field. Click the button in order to select the project directory from within a dialog.



NOTE!

Save each project in a separate directory! Although it is possible to save several projects in one directory, this is very disorganized. Also, there is a danger that projects may accidentally be overwritten.

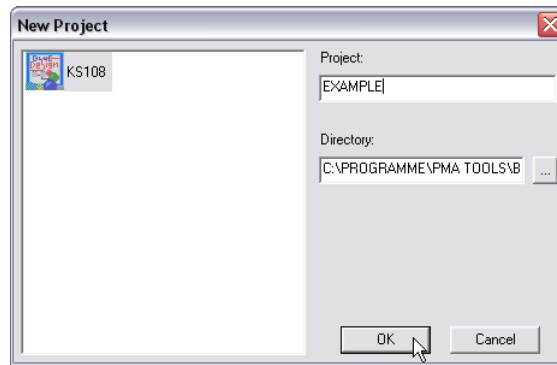


Fig. 122: "New Project" dialog

3. **Enter project information:** Apply your input by clicking "OK".
By default, *BlueDesign* creates a project with an empty program block with the name "PROG01".

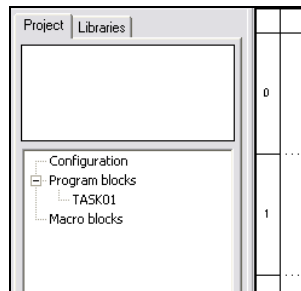


Fig. 123: Start view of a new project

II-3.4 Working with program blocks

Program blocks are independent sub-programs ("tasks"). They are used to implement the desired functions by means of blocks, macros and connections. The character "\$" is prefixed to program blocks in run mode (e.g. "\$PROG1").

It is a very good idea to establish a fundamental structure of the desired solution before implementing a project. The goal of structuring the project should be to determine whether and how the solution can be distributed among several program blocks and to what extent macro blocks can be used.

II-3.4.1 Creating program blocks

1. **Start edit mode:** Make sure that you are in edit mode.
2. **Select program blocks node:** Select the "Program blocks" node with a mouse click.
3. **Execute context menu command "New Program Block":** Open the context menu with the right mouse button and select the command "New Program Block".

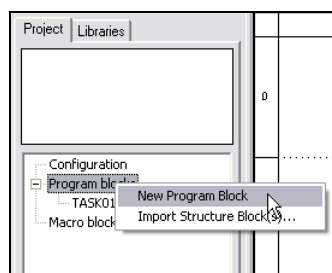


Fig. 124: Context menu command "New Program Block"

4. **Enter program block names:** The new program block is given an automatically generated name. Enter a new name here if desired.

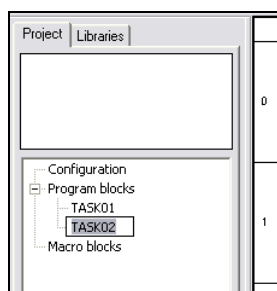


Fig. 125: Assign name for program block

II-3.4.2 Deleting program blocks

1. **Start edit mode:** Make sure that you are in edit mode.
2. **Select program blocks node:** Select the respective program block with a mouse click.
3. **Execute context menu command "Delete":** Execute the context menu command "Delete".



NOTE!

The program block will be deleted without further prompting!

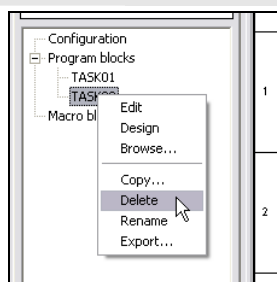


Fig. 126: Delete program block

II-3.4.3 Renaming program blocks

1. **Start edit mode:** Make sure that you are in edit mode.
2. **Select program blocks node:** Select the respective program block with the right mouse button.
3. **Execute context menu command "Rename":** Execute the context menu command "Rename".
4. **Enter program block names:** Click the program block name in order to change it.

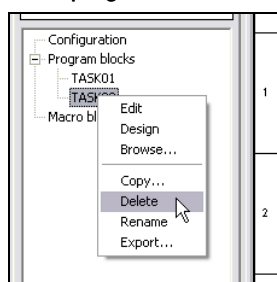


Fig. 127: Rename program block

II-3.4.4 Copying program blocks

1. **Start edit mode:** Make sure that you are in edit mode.
2. **Select program blocks node:** Select the respective program block with the right mouse button.
3. **Execute context menu command "Copy":** Execute the context menu command "Copy".
Now you will see the dialog "Name of Copied Structure Block".

Enter the name of the new block in the "Name" field. In the selection field "Type", specify whether the new block should be a program block or a macro block.

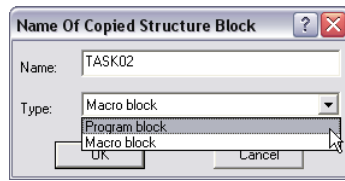


Fig. 128: "Name Of Copied Structure Block" dialog

4. **Apply input:** Apply the changes by clicking "OK".

II-3.4.5 Exporting program blocks

Program blocks can be exported - for example, in order to make them available to other employees. Proceed as follows to export blocks:

1. **Start edit mode:** Make sure that you are in edit mode.
2. **Select program blocks node:** Select the respective program block with the right mouse button.
3. **Execute context menu command "Export":** Execute the context menu command "Export".
You now see the dialog "Select Structure Block Export File".

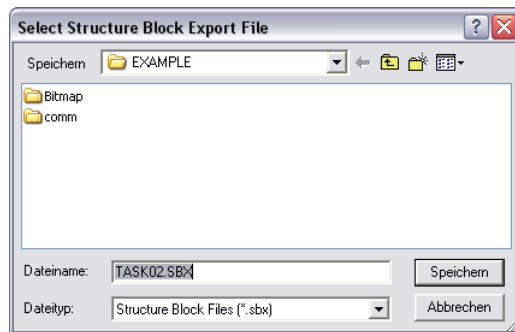


Fig. 129: "Select Structure Block Export File" dialog

4. **Save export file:** Select the directory and file name and save the file by clicking the "Save" button.

II-3.4.6 Displaying/changing program block call name

In run mode, blocks are automatically assigned a unique name. This is referred to as the "call name". You can view and change the name as follows:

1. **Start run mode:** Make sure that you are in run mode.
2. **Select program blocks node:** Select the respective program block with the right mouse button.
3. **Execute context menu command "Call Name":** Execute the context menu command "Call Name".
Now you see the dialog of the same name.

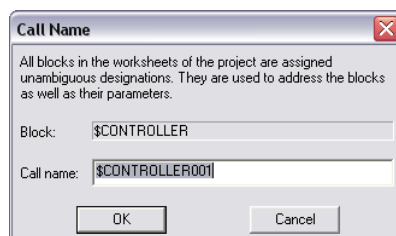


Fig. 130: "Call Name" dialog

4. **Change call name:** Change the call name if desired. Take into account that the name must be unique.
5. **Apply changes:** Click the "OK" button to apply your changes. Note: If you (accidentally) enter an existing name, an error message will appear. In this case, your input will be rejected.

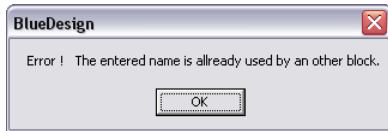


Fig. 131: Error message: "Call name already in use"

II-3.4.7 Defining program block cycle time

Each program block is executed as a separate sub-application or "task". It is allocated its own computing time from the runtime environment. The basic procedure used by the runtime environment is to start the program blocks one after the other (see following graphic). You can also specify via the cycle time that (some) program blocks are called at greater intervals (i.e. they are sometimes "skipped").

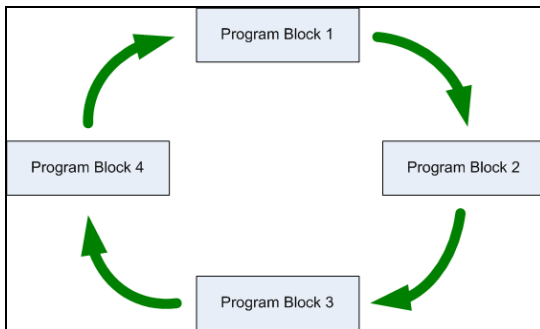


Fig. 132: Starting program components

Proceed as follows to change the cycle time:

1. **Start run mode:** Make sure that you are in run mode.
2. **Select program blocks node:** Select the respective program block with the right mouse button.
3. **Execute context menu command "Task Properties":** Execute the context menu command "Task Properties ...".

Now you see the "Task Properties" dialog.

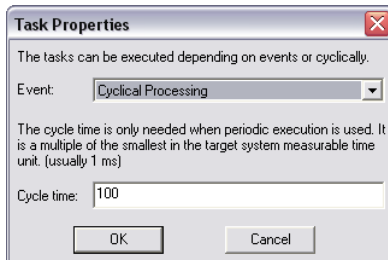


Fig. 133: "Task Properties" dialog

4. **Specify cycle time:** Enter the desired cycle time in the field with the same name.



NOTE!

*There is NO guarantee that the program block will actually be called in this sequence. Reason: Program blocks are **always** completed, regardless of whether another block is supposed to be called according to the cycle. Therefore, it is probable that program blocks are called with a "time delay".*

The system is therefore not suitable for realtime applications!

II-3.5 Working with blocks

Blocks are the basis of every application in *BlueDesign*. Applications are created by implementing the desired functions by the use of blocks and their connections. Examples of blocks are programmers, controllers, timers or filters.

II-3.5.1 Positioning blocks

1. **Start edit mode:** Make sure that you are in edit mode.
2. **Select "Libraries" tab:** Select the "Libraries" tab.
The "Libraries" tab shows you the available blocks in a tree structure. The "Libraries" are in the uppermost level of the tree structure. Related blocks are grouped in a library.
3. **Select block:** Click the \oplus symbol in order to open the library. Then click the block that you wish to use.
The selected block is displayed at the top of the tab:

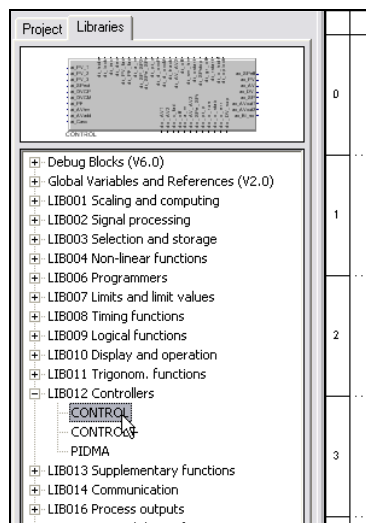


Fig. 134: Using a library

4. **Use block:** With the mouse, click the block at the top of the tab. Hold down the left mouse button and drag the block to the right onto the workspace. Then release the left mouse button.

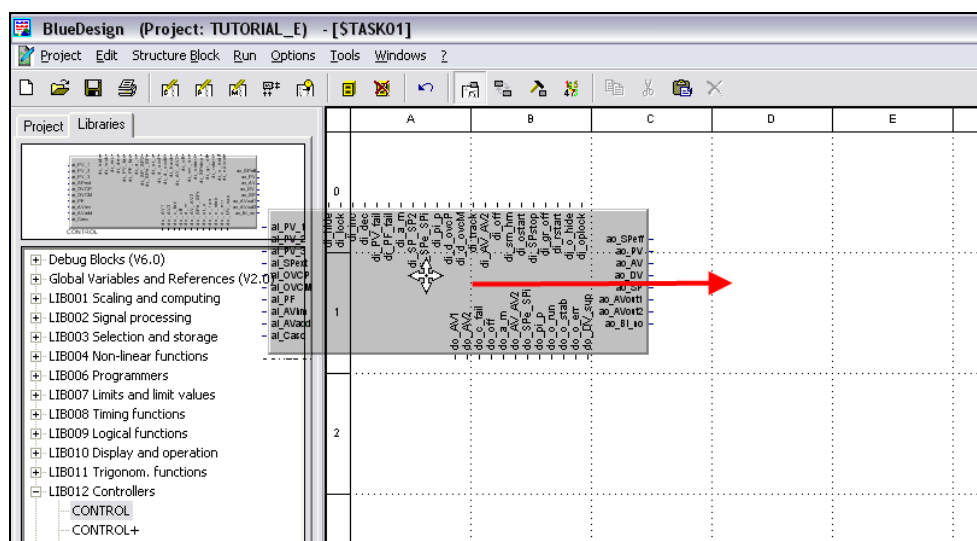


Fig. 135: Use block from library

II-3.5.2 Connecting blocks

The signal flow from block to block is achieved by connections. These are created as follows:

1. **Start edit mode:** Make sure that you are in edit mode.
2. **Position mouse over output:** Position the mouse over the output of a block. The mouse pointer appears as a pencil (see following graphic).
3. **Drag connection:** Click with the left mouse button and move the mouse to the desired input. Click again with the left mouse button.
The connection is now drawn.

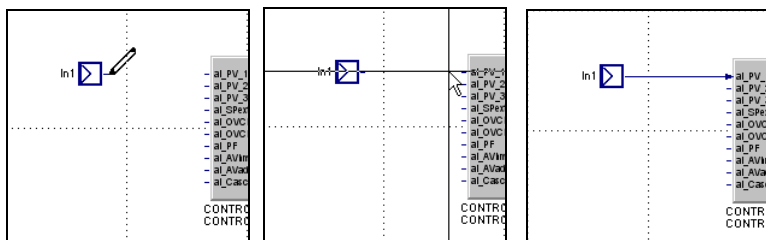


Fig. 136: Draw connecting line.



NOTE!

Make sure to connect only inputs and outputs that use the same data type. Otherwise, a connection cannot be created and the following error message will appear: "Error! Invalid connection".

II-3.5.3 Deleting blocks

1. **Start edit mode:** Make sure that you are in edit mode.
2. **Position mouse over block:** Position the mouse on a block.
3. **Delete block:** Start the context menu command "Delete" with the right mouse button in order to delete the block (see following graphic).



NOTE!

The block will be deleted without prompting!

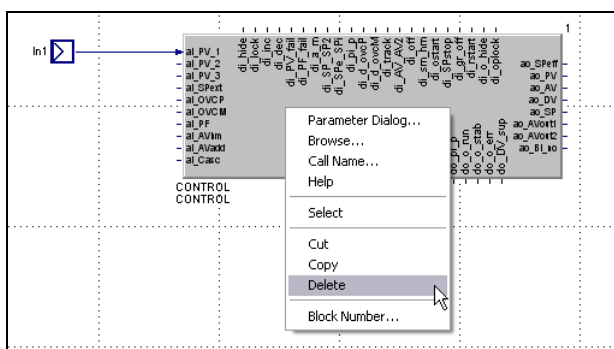


Fig. 137: Deleting a block

II-3.5.4 Defining the sequence of blocks

You can change the sequence in which blocks are executed. By default, the blocks are executed in the sequence in which you inserted them.

In many cases, however, the sequence in which the blocks are called is important.

Proceed as follows to change the sequence:

1. **Start edit mode:** Make sure that you are in edit mode.
2. **Start "Edit sequence" mode:** Start the "Edit sequence" mode. To do this, click the "Block sequence" button in the button bar:

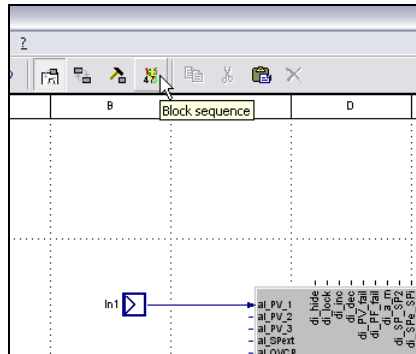


Fig. 138: Start "Edit sequence" mode

3. **Change sequence:** The position of a block in the sequence of execution is displayed by the number above the block at the right. Change the sequence as follows: Start with the block that should have the lowest number. Right-click the block for which you wish to change the position. The "Block sequence" dialog opens.

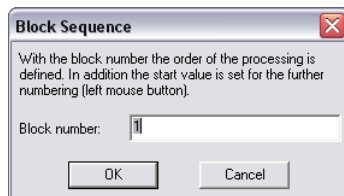


Fig. 139: "Block sequence" dialog

4. **Enter new position:** Enter the new position of the block in the dialog and save the input by clicking "OK". The remaining blocks in the program block will be re-numbered accordingly.
5. **Adapt position of the remaining blocks:** As already mentioned in the previous project: When a block is assigned a new position, all other blocks are automatically numbered. However, you may require different numbering. You can open the "Block sequence" dialog for every module/function block. The following procedure is simpler, however. Left-click the block that should have the subsequent number in the block sequence. A click with the left mouse button causes the block to be assigned the next number.
6. **End "Edit sequence" mode:** Click the "Edit drawing" button in the button bar to end the "Edit sequence" mode.

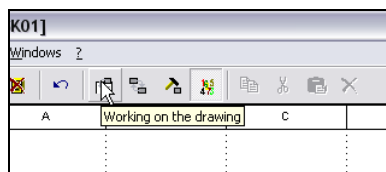


Fig. 140: Start "Edit drawing" mode

II-3.5.5 Copying blocks

1. **Start edit mode:** Make sure that you are in edit mode.
2. **Select elements:** Select the elements to be copied by holding down the left mouse button. A black rectangle shows the selected area.



NOTE!

Only elements that are entirely within the black rectangle will be selected. Selected elements are enclosed by a dotted line.

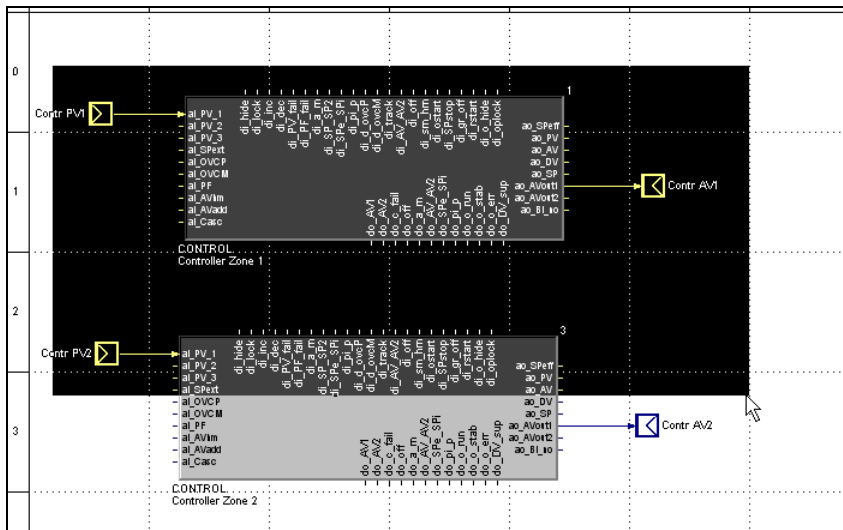


Fig. 141: Select elements

3. **Copy and insert elements:** Copy and insert the elements using the standard Windows commands.

II-3.6 Specifying inputs and outputs

If program blocks (or macro blocks) should pass values to other blocks or receive values from them, you need inputs and outputs.

1. **Start edit mode:** Make sure that you are in edit mode.
2. **Select "Libraries" tab:** Select the "Libraries" tab.
The "Libraries" tab shows you the available blocks in a tree structure. The "Libraries" are in the uppermost level of the tree structure. Related blocks are grouped in a library.
3. **Select block:** Click the \boxplus symbol in front of the "Standard" library to open this library. Then click the "Input" block or "Output" block. The selected block is displayed at the top of the tab.

Explanation of terms: The "Input" block receives a value from outside of the program block, etc. This value is sent to a block in the program block. With the "Output" block it is the other way around: it sends the value "out".

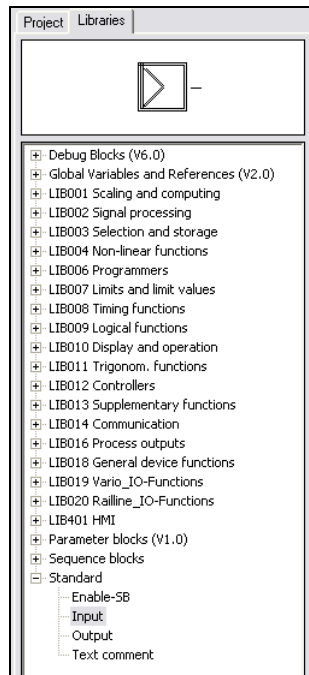


Fig. 142: Select block

4. **Use block:** With the mouse, click the block at the top of the tab. Hold down the left mouse button and drag the block to the right onto the workspace. Then release the left mouse button. Now you see the "Input" dialog.

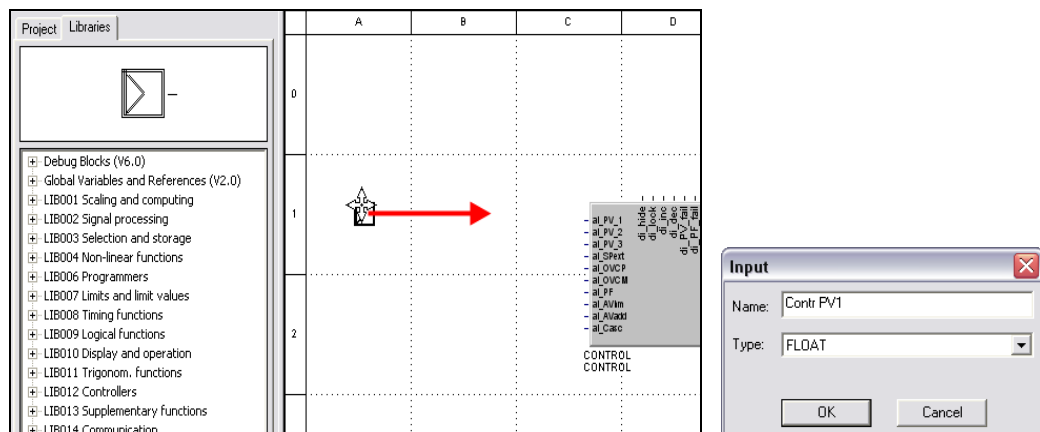


Fig. 143: Using inputs/outputs; "Input" dialog

5. **Specify block name and data type:** Enter the block name in the "Input" dialog and specify the data type ("FLOAT" in this example).

**NOTE!**

It is not possible to change the data type afterwards. If you accidentally use a module with an incorrect data type, you have to delete it and create it again.

6. **Connect input/output with block:** Connect the input or output with a block (further information on connecting blocks can be found in the section //3.5.2 Connecting blocks).

**NOTE!**

Make sure to connect only inputs and outputs that use the same data type. Otherwise, a connection cannot be created and the following error message will appear: "Error! Invalid connection".

II-3.7 Positioning inputs and outputs

Inputs and outputs (interfaces) have to be positioned - they have to be made "outwardly" visible. This is achieved with the following steps (prerequisite: you must define inputs and/or outputs for your program block or macro block as described in section "II-3.6 Specifying inputs and outputs").

1. **Start edit mode:** Make sure that you are in edit mode.
2. **Execute context menu command "Design":** Execute the "Design" command with the right mouse button from the context menu of the respective block.

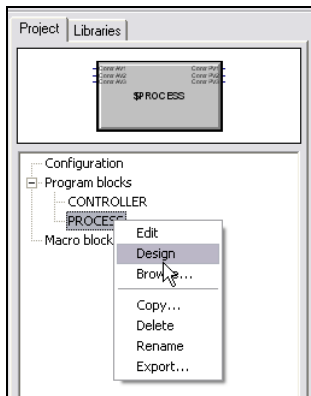


Fig. 144: Select command "Design"

You now see a worksheet on the right side of the screen with the inputs and outputs you have added. The "Inputs" and "Outputs" not used are listed there in two columns. The list is empty if there are no inputs or outputs or if they have all been assigned.

3. **Allocate inputs:** Place the mouse pointer over an element in the "Inputs" or "Outputs" column. The mouse pointer now appears as a hand (see following graphic). Now press the left mouse button. Hold the mouse button down and drag the element to the left edge (for inputs) or to the right edge (for outputs) of the program block symbol. Then release the left mouse button.

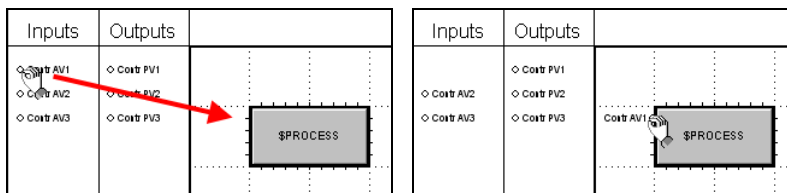


Fig. 145: Allocate inputs

II-3.8 Editing the configuration

In the "Configuration" area you specify

- which program blocks are to be executed and
- the sequence in which they are executed.



NOTE!

Program blocks are executed only if they can be found in the "Configuration" area. A configuration without program blocks makes no sense - the result is a program with no functions (i.e. with no "tasks").

1. **Start edit mode:** Make sure that you are in edit mode.
2. **Display "Configuration" area:** Double click the "Configuration" node on the "Project" tab in order to view the "Configuration" workspace.
3. **Select program block:** Click the program block to be inserted in order to select it (e.g. "PROCESS" in the graphic below). The selected block is displayed at the top of the tab.

4. **Use program block:** With the mouse, click the program block on the tab at the top. Hold down the left mouse button and drag the block to the right onto the workspace. Then release the left mouse button.

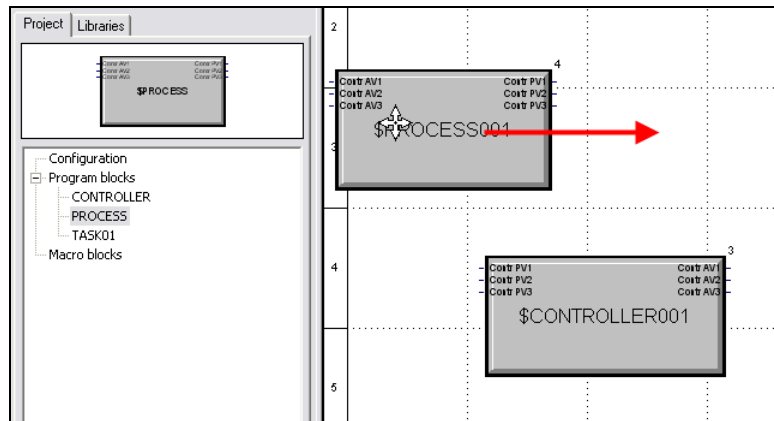


Fig. 146: Creating a configuration

II-3.9 Using parameters

The components of the PMA library have to be configured. They are configured using parameters, which define the behavior and the properties of the blocks. Value lists that offer possible options are displayed for numerous parameters. In addition, inputs are checked. If a value is not permissible, it is corrected to the next permissible value.



NOTE!

In order to understand this section, it is essential that you are familiar with the BlueDesign operating modes. Further information on the operating modes can be found in the section "II-3.1.2 BlueDesign operating modes".

Editing parameters in edit mode

Parameters entered in edit mode can be thought of as a "copy template".

Normally, you should edit parameters in run mode (see the following section). Since the parameters entered in edit mode can be overwritten in run mode at any time, parameters should be assigned in run mode *only* if the parameters are really supposed to be used in all instances ("copies") of a block.

Proceed as follows to edit parameters in edit mode:

1. **Start edit mode:** Make sure that you are in edit mode.
2. **Select block:** Position the mouse on the block you wish to edit.
3. **Open the parameter dialog:** Select the "Parameter dialog..." command with the right mouse button.

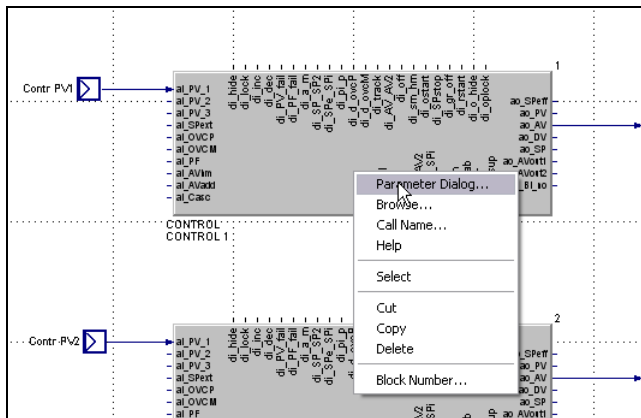


Fig. 147: Start parameter dialog

You now see the "Parameter" dialog, where you can edit the parameters.

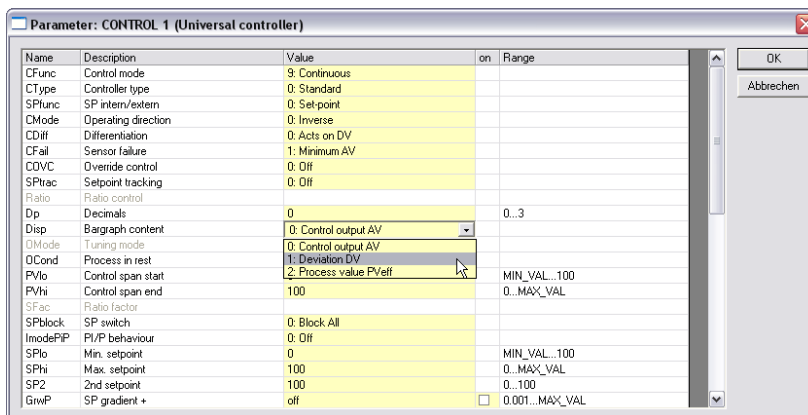


Fig. 148: "Parameter" dialog

- Edit parameters:** You can edit parameters in the fields highlighted in yellow. Depending on the parameter you can enter a free text or select the parameter in a drop-down list (as in the graphic above).
- Save input:** Click the "OK" button to save your input.

**NOTE!**

Macro blocks have no parameters; parameters exist only at the block level (in the macro block). For further information, see section "II-3.10 Working with macro blocks"

Editing parameters in run mode

Parameters that you enter in run mode overwrite parameters entered in edit mode. For this reason, you should normally enter parameters in run mode.

To do this, proceed as follows:

- Start run mode:** Make sure that you are in run mode.
- Select block:** Click the block for which you wish to edit the parameters. You now see the "Parameter" dialog, where you can edit the parameters.

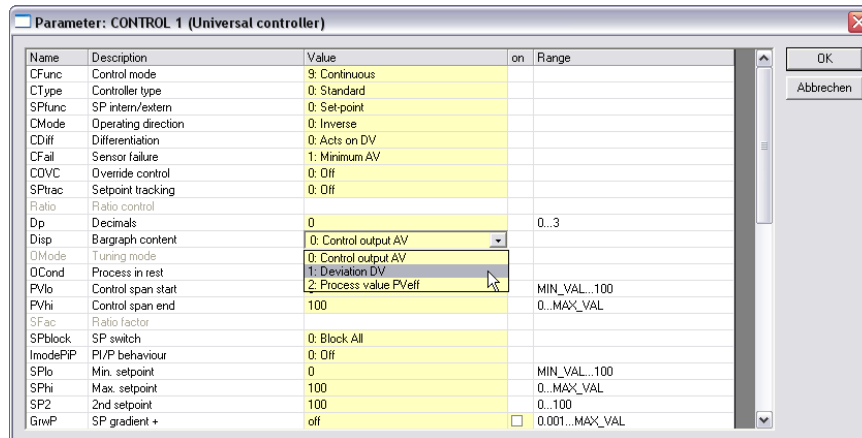


Fig. 149: "Parameter" dialog

- Edit parameters:** You can edit parameters in the fields highlighted in yellow. Depending on the parameter you can enter a free text or select the parameter in a drop-down list (as in the graphic above).
- Save input:** Click the "OK" button to save your input.

**NOTE!**

Macro blocks have no parameters; parameters exist only at the block level (in the macro block). For further information, see section "II-3.10 Working with macro blocks"

Editing parameters in online observation mode

When you enter parameters in editing or run mode, specify parameters that are transferred to the device together with the application. In addition, you can change parameters of the currently running application.

**DANGER!**

Danger of injury or property damage due to unforeseeable plant function sequences and movement sequences!

If the *KS 108 easy* device is used with other devices/equipment/actuators then consequential actions can be induced on these devices/actuators, etc. during transfer of the parameters.

Therefore:

- Consider the effects of a parameter update and ensure that appropriate measures are taken prior to establishing a connection!
- Always ensure that parameters are not faulty prior to establishing a connection.

To do this, proceed as follows:

- Start online observation mode:** Make sure that you are in online observation mode.
- Change parameters:** Change the parameters (as described above).
- Write parameters:** Use the menu command "Run/Write Parameters to Target System" to transfer the parameters.

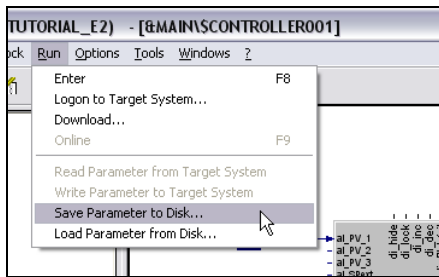


Fig. 150: Call menu command "Write Parameter to Target System"



NOTE!

You can also read parameters from the target system. To do this, use the menu command "Run/Read Parameter from Target System".

II-3.10 Working with macro blocks

Combinations that are frequently used in program blocks can be exported to macro blocks. Macro blocks are copy templates that can be used in program blocks. In the following example, the same construction is used three times (marked red).

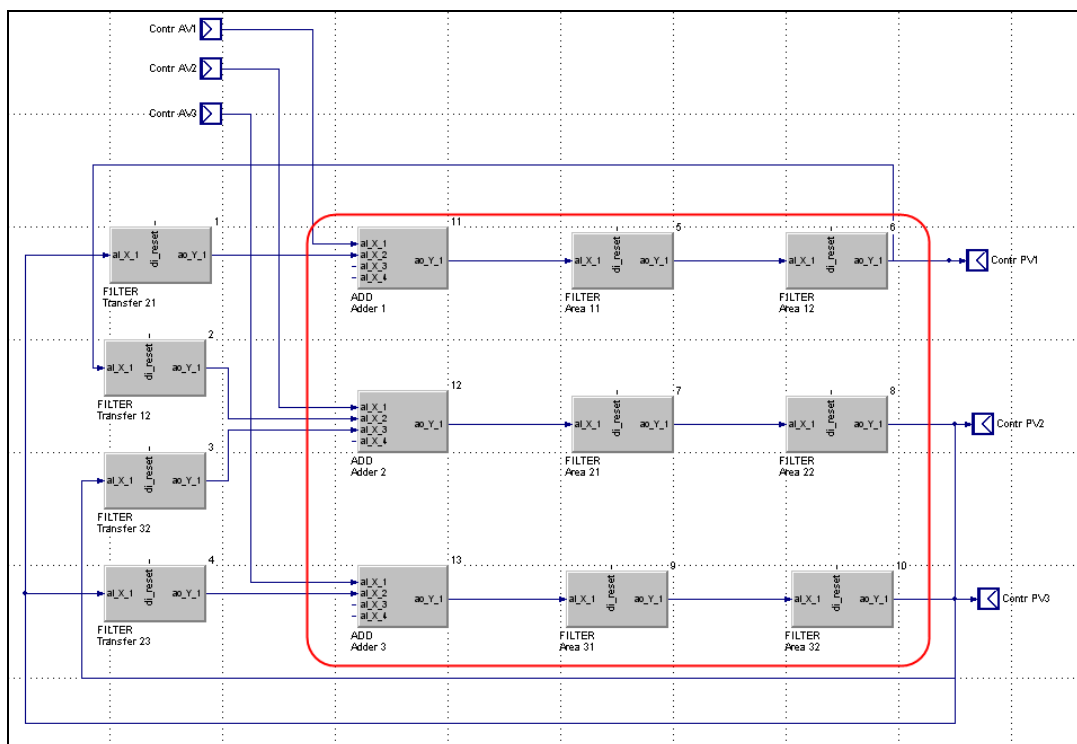


Fig. 151: Example: Multiple use of program structures

The programming block can be simplified by exporting the respective three elements to a macro.

II-3.10.1 Creating a new macro block

Proceed as follows to create macros:

1. **Start edit mode:** Make sure that you are in edit mode.
2. **Select "Macro block" node:** Select the "Macro block" node on the "Project" tab.
3. **Execute context menu command "New Macro Block":** Open the context menu with the right mouse button and select the command "New Macro Block".

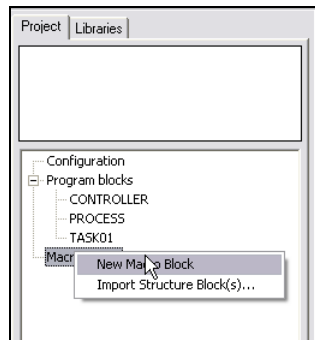


Fig. 152: New Macro Block

4. **Name macro:** The new macro block is given an automatically generated name.
5. **Name:** Give the macro block a new name.

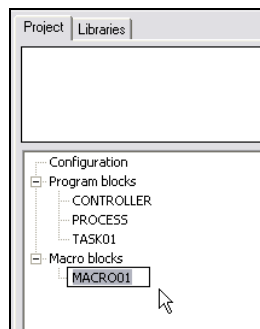


Fig. 153: Assign name for macro

6. **Place blocks in macro block and connect:** Place the required blocks on the workspace of the macro block and connect the blocks. Create the required inputs and outputs and release them.

**NOTE!**

For further information, see the section "II-3.5 Working with blocks".

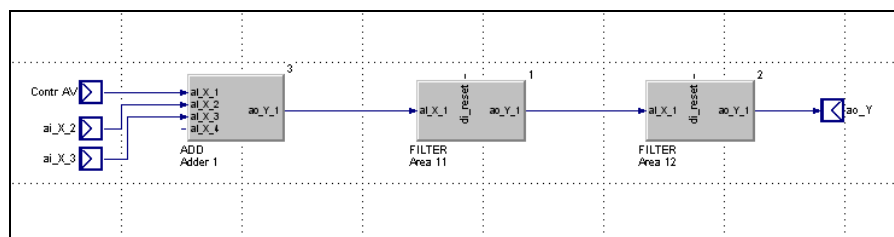


Fig. 154: Place blocks in macro block and connect

II-3.10.2 Using the macro block

Macro blocks are used essentially the same way as blocks from a library:

1. **Start edit mode:** Make sure that you are in edit mode.
2. **Select program block:** Select the program block into which you wish to insert the macro block with a double click on the "Project" tab.
3. **Select "Macro block" node:** Select the "Macro block" node on the "Project" tab.
4. **Using the macro block:** Place and connect the macro block as you would a block from a library (see also: II-3.5.1 Positioning blocks):

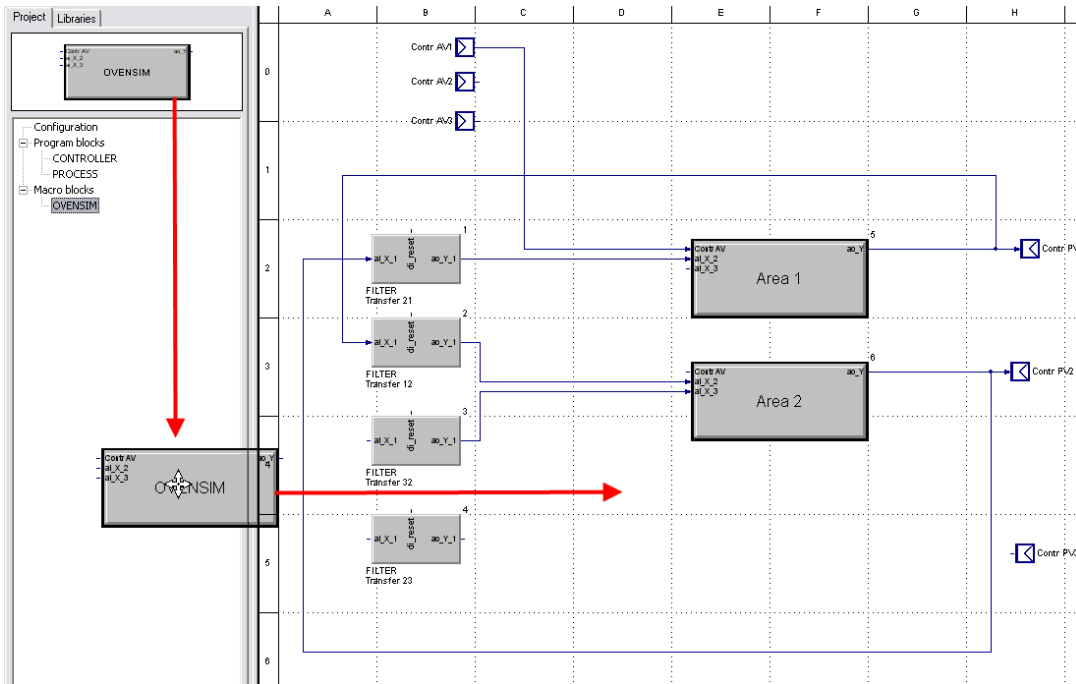


Fig. 155: Using macro blocks

NOTE!
Macro blocks can also call other macro blocks.

II-3.10.3 Editing parameters in macro blocks

Basically, parameters in macro blocks are edited just as in all other blocks (see section II-3.9 Using parameters).

However, parameters can be edited only on the block level (and not on the macro block level).

- **Edit mode:** Therefore, the context menu command "Parameter Dialog ..." is not available in edit mode. Instead, select the "Edit" command to open the detailed view of the macro block.
- **Run mode:** If you click a macro block in run mode, the details of the macro block will be displayed instead of the parameter dialog (see the following graphic).

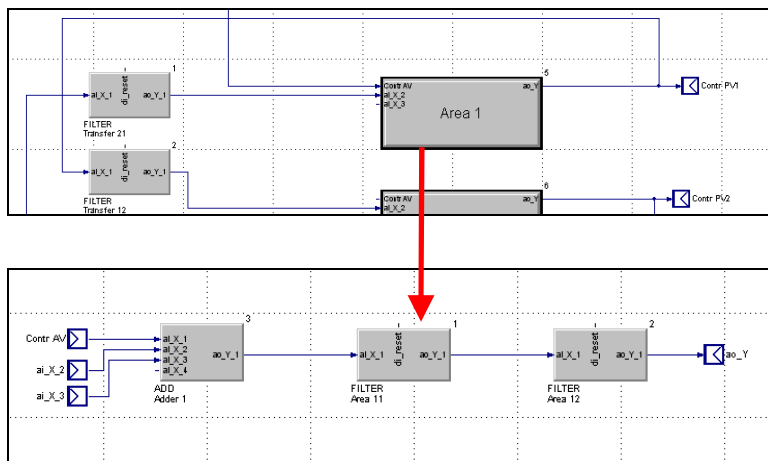


Fig. 156: Editing parameters in macro blocks

II-3.11 Using global variables

Macro blocks and program blocks can communicate with each other via interfaces as described above. However, this assumes that the entire program block or macro block is used in a main program or program block.

If the entire program block or macro block is not used and only a value is needed, for example, then global variables can be used. Global variables can be accessed by all blocks.



NOTE!

Please note that you can use only variables of the type "Float" and "Bit" with the PMA library!

Creating global variables

1. **Start "Global Variables and References" dialog:** Use the menu command "Tools/Global Variables ..." to open the "Global Variables and References" dialog.

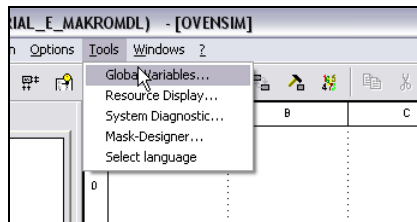


Fig. 157: "Global Variables and References" dialog

2. **Start the "Add Variable" dialog:** Click within the right area of the dialog and start the context menu command "Add" with the right mouse button.

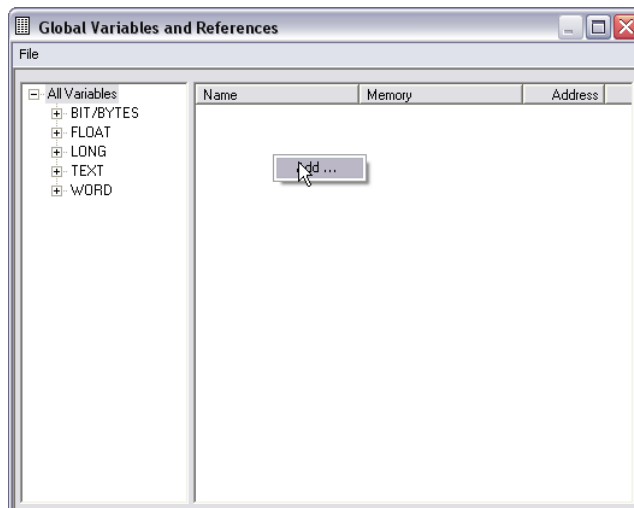


Fig. 158: "Global Variables and References" dialog

3. **Add variable:** Enter the name of the variable in the "Name" field and specify the data type.

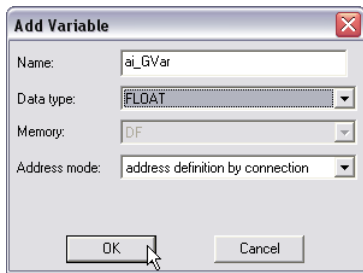


Fig. 159: Add Variable

**NOTE!**

Use only the data types "FLOAT" or "BIT". Always select the option "Address definition by connection" when defining the address.

**CAUTION!****Incorrect address assignments can cause the program to malfunction!**

Incorrect addressing cause the uncontrolled writing of data to memory or uncontrolled reading of data from memory. Both can cause the program to crash or to function incorrectly.

Therefore:

- Make sure that the option "Address definition by connection" is selected in the "Address mode" field. Never use the option "Permanent address definition".
- Test the application intensively to rule out malfunctions.

4. **Save input:** Save your input by clicking "OK".

II-3.11.1 Using backup copies

BlueDesign creates a complete backup of your project each time the project is saved. This means that you can easily restore an earlier project version.

To do this, proceed as follows:

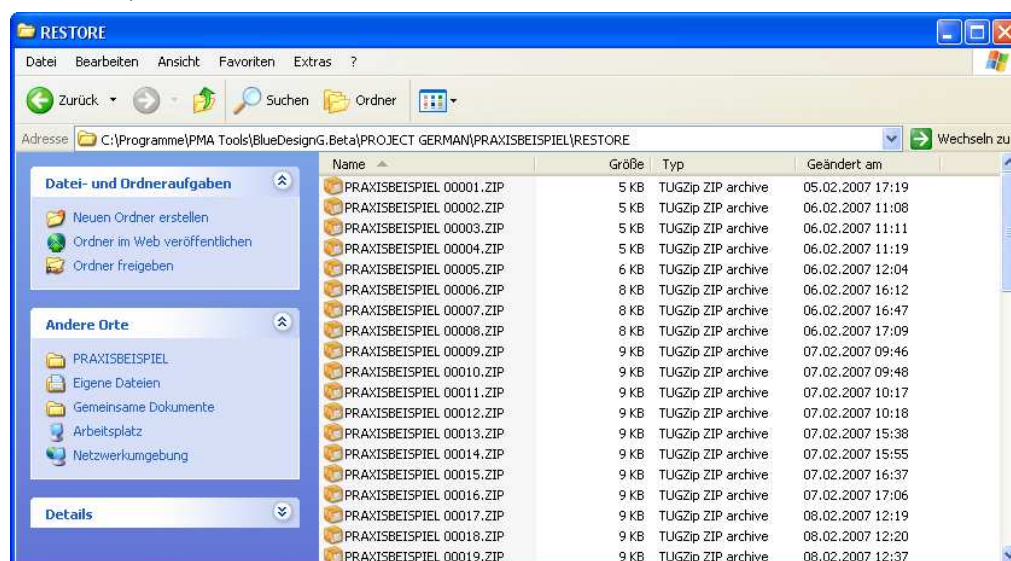


Fig. 160: "RESTORE" directory

1. **Start the "Project/Restore ..." menu command:** Start the "Project/Restore ..." command to open the "Select Project Archive" dialog.

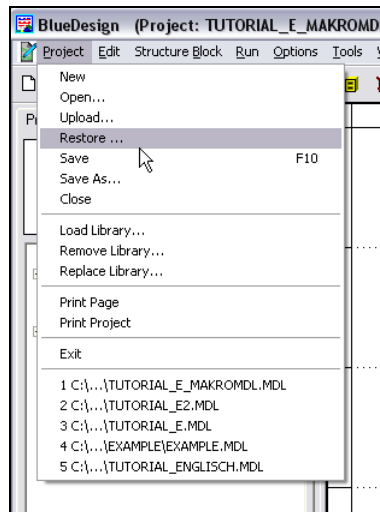


Fig. 161: Start "Project/Restore" command

2. **Select project archive:** Select the "RESTORE" directory for your project. Here you will find the backups of your project that were saved. Select the desired backup and click the "Open" button.

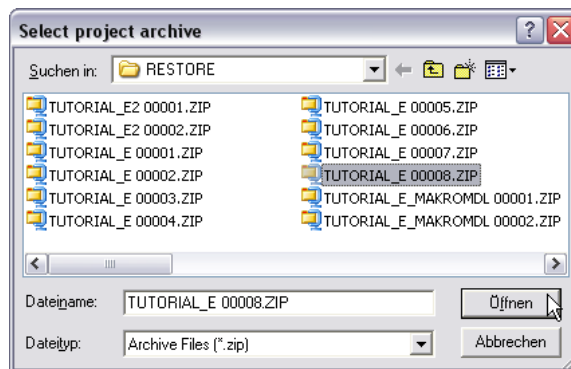


Fig. 162: Select Project Archive dialog

3. **Select project directory:** In the "Project directory" dialog, select the directory where the project is to be saved.



Fig. 163: "Project directory" dialog

If there is already a project in the directory, the following prompt will appear:

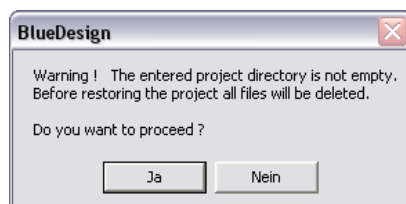


Fig. 164: "Overwrite project?" dialog

If you click "Yes" here, the existing project will be deleted before the backup copy is saved. If you click "No", the project will not be saved.

II-3.12 Using the development environment

II-3.12.1 Logon to target system

In principal, you can use the development environment to access different target systems. However, only one target system can be used at a given time. In order to access the target system, you have to tell the development environment the location of the target system.

An IP number and a port address are used for addressing.

Digression: Addressing

The IP address is a number used for addressing (locating) computers in a network. The IP address is similar to a telephone number. Just as a telephone number is used to establish a connection to a particular telephone, the IP address is used to establish a connection to a particular computer.

It is important that two computers never have the same IP address. In local (company) networks, this is the responsibility of the network administrator. He will either provide you with an IP address, which you can use for the KS 108, or he will use a DHCP server. This "Dynamic Host Configuration Protocol" server automatically assigns an IP address to every computer in the network. Further configuration is not necessary (provided the automatic assignment of IP addresses is permitted on the computer).

The port number indicates an address within a computer (comparable to the extension of a telephone number).

Logon to target system

Proceed as follows to logon to a target system:

1. **Start run mode:** Make sure that you are in run mode.
2. **Execute "Logon to target system" command:** Start the menu command "Run/Logon to Target System".

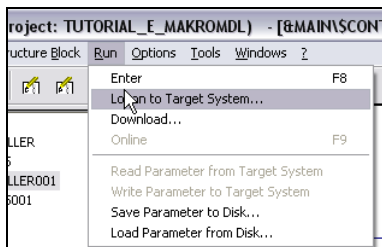


Fig. 165: Logon to target system

Now you see the "Logon to Target System" dialog.

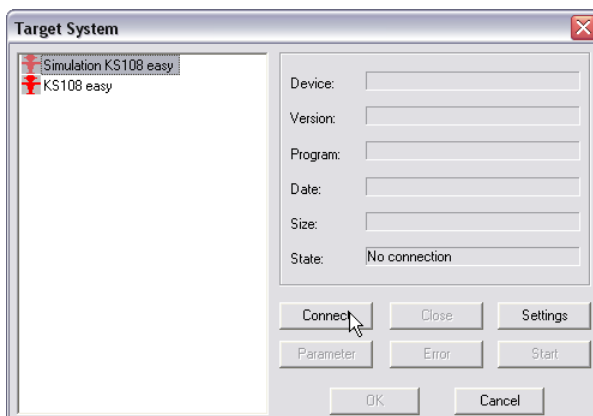


Fig. 166: "Logon to Target System" dialog

3. **Select target system:** Click the desired target system (in the example above: "Simulation KS108 easy") at the left of the "Logon to Target System" dialog.

4. **Open the "Settings" dialog:** Click the "Settings" button to start the "IntraCom Configuration" dialog.

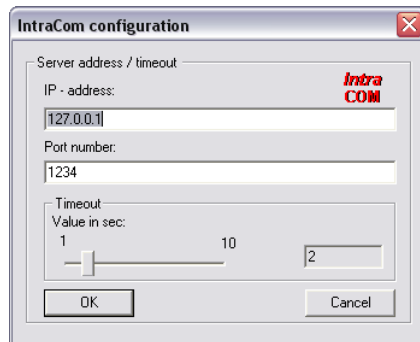


Fig. 167: "IntraCom Configuration" dialog

5. **Enter the IP address:** In the "IntraCom Configuration" dialog enter the IP address and the port number of your target system.



NOTE!

If you are working with the Simulator, you have to use the pre-set IP address "127.0.0.1" (as shown in the graphic above).

Here you can also set the "timeout". The "timeout" is the time period in which the attempt is made to establish a connection to the target system. If the timeout has elapsed, you will see an error message on the "General" tab. If the connection is relatively poor, it may be useful to increase the timeout. This prevents inapplicable error messages.

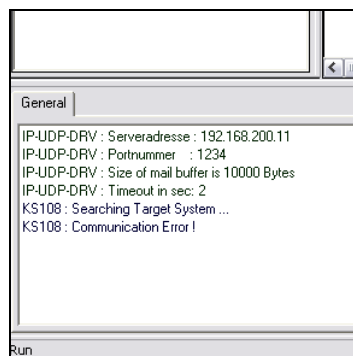


Fig. 168: System messages on the "General" tab

6. **Save input:** Save your entry by clicking "OK".
7. **Establish connection:** Click the "Connect" button to establish the connection. Make sure that the correct target system is selected in the left column (see following graphic).

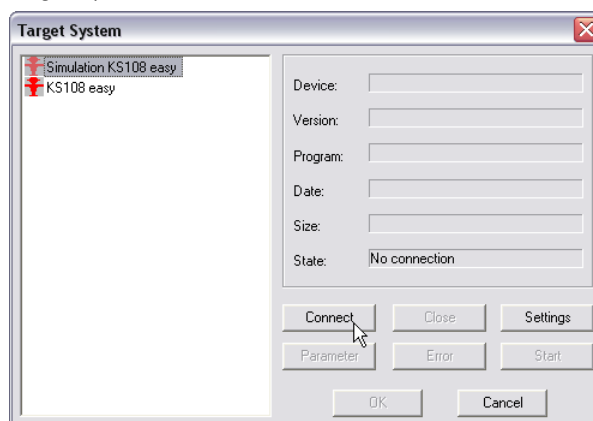





Fig. 169: Connecting to target system.

8. **Save settings:** Click the "OK" button to save your settings.

Status display

The connection status to the target system is displayed with symbols (the example used here is the display for the simulator):

| Symbol | Meaning |
|---|---|
|  Simulation KS108 easy | The connection is active. |
|  Simulation KS108 easy | The connection is not in use - although it is configured. |
|  Simulation KS108 easy | The connection is not active. |

II-3.12.2 Disconnecting from target system

Click the "Close" button to disconnect the connection.

1. **Execute "Logon to target system" command:** Start the menu command "Run/Logon to Target System".

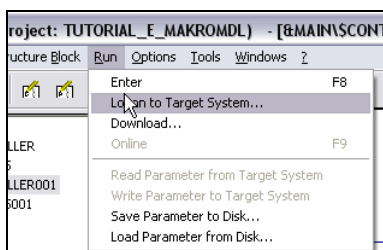


Fig. 170: Logon to target system

Now you see the "Logon to target system" dialog.

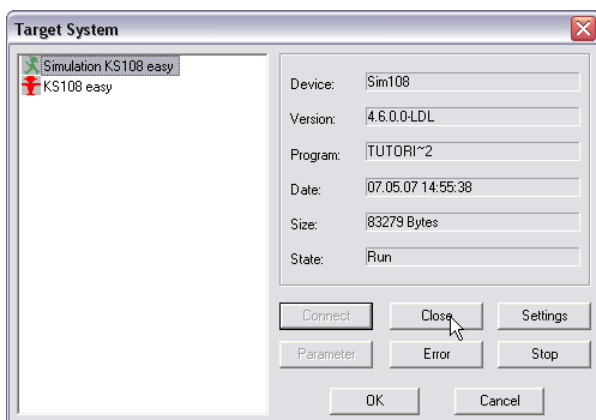


Fig. 171: Disconnecting from target system

2. **Close connection:** Click the "Close" button to disconnect the connection to the target system.
3. **Close dialog:** Close the dialog by clicking "OK".

II-3.12.3 Download...



DANGER!

Danger of injury or property damage due to unforeseeable plant function sequences and movement sequences!

If the *KS 108 easy* device is used with other devices/equipment then consequential actions can be induced on these devices/actuators, etc. by transferring the program.

Therefore:

- Consider the effects of a program update and ensure that appropriate measures are in place prior to establishing a connection!
- Prior to establishing a connection it is strictly necessary to ensure that the correct program is loaded.
- Prior to establishing a connection the error-free status of the program must be ensured.

1. **Call "Download ..." menu:** Call the "Run/Download ..." menu command in order to export the program to the target system.

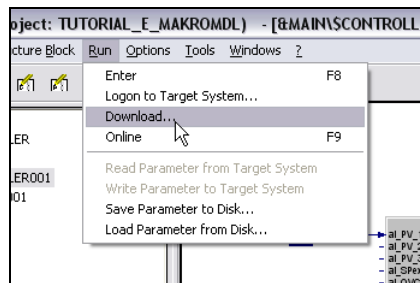


Fig. 172: Download...

If a connection to the target system already exists, the program will now be transferred. If no connection exists to the target system, the "Logon to Target System" dialog appears.

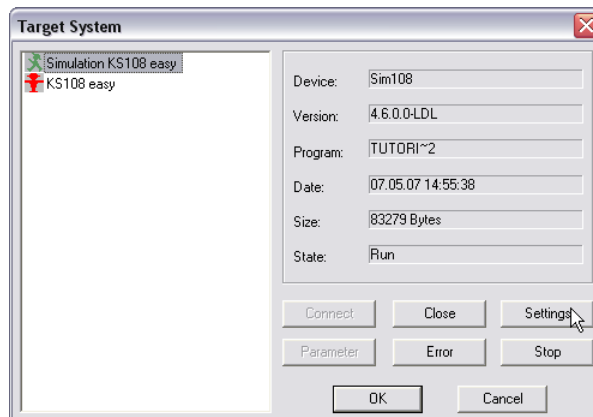


Fig. 173: "Logon to target system" dialog



NOTE!

Further information on establishing a connection can be found in the section "II-3.12.1 Logon to target system".

II-3.12.4 Changing worksheet size

The worksheet contains the blocks and connections for your application. You can change the size of the worksheet in six steps. This is especially useful if you are planning to print the worksheet.

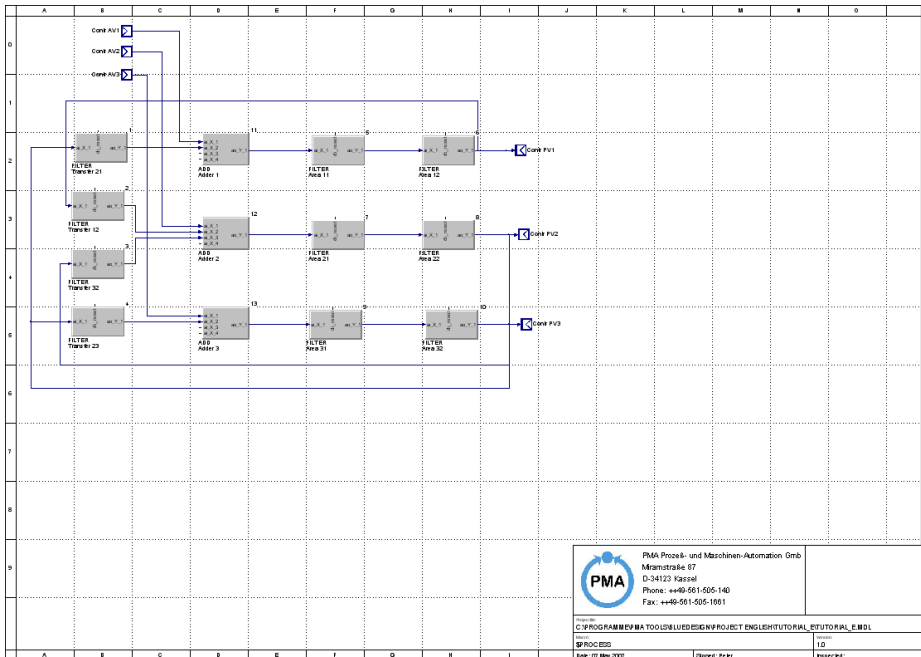


Fig. 174: Worksheet

Proceed as follows to change the size of the worksheet:

1. **Call "Worksheet ..." menu:** Call the menu command "Options/Worksheet".

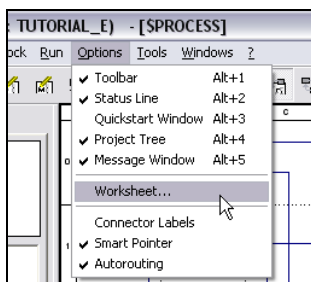


Fig. 175: Call "Worksheet" menu

You now see the "Worksheet Setup" dialog (see following graphic).

2. **Select format:** Select the desired format by clicking the corresponding button in the "Format" area.

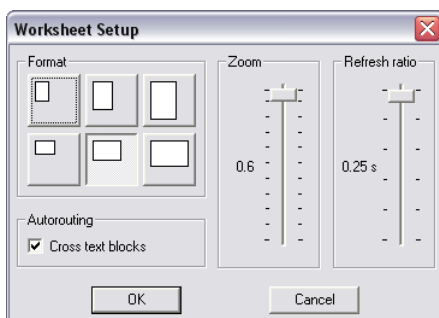


Fig. 176: "Worksheet Setup" dialog

3. **Apply input:** Apply your input by clicking "OK".

II-3.12.5 Adapting worksheet view

You can enlarge or decrease the size of the worksheet view.

To do this, proceed as follows:

1. **Call "Worksheet ..." menu:** Call the menu command "Options/Worksheet".

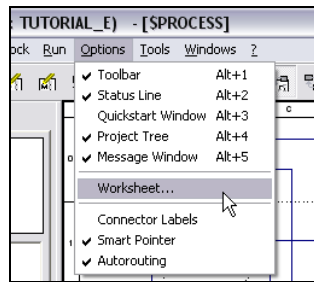


Fig. 177: Call "Worksheet" menu

You now see the "Worksheet Setup" dialog (see following graphic).

2. **Select zoom:** Define the view with the slide control in the "Zoom" area. Click the control and hold down the left mouse button in order to change the setting.

Note: The slide control in the "Refresh Rate" area is used to specify how often the monitor display is updated.

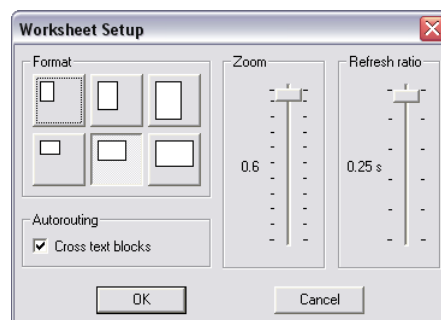


Fig. 178: "Worksheet Setup" dialog

3. **Apply input:** Apply your input by clicking "OK".



NOTE!

You can also enlarge or decrease the view of the worksheet with the "+" and "-" keys on the numeric keypad.

II-3.12.6 Moving a worksheet

You can easily move all elements of a worksheet.

1. **Call context menu command "Move Worksheet":** Click within a free area of the worksheet and call the context menu command "Move Worksheet".

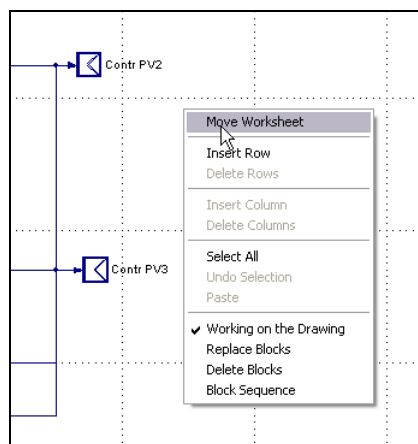


Fig. 179: Call "Move Worksheet" command

2. **Move elements:** Move the mouse in the desired direction. The new position of the elements is displayed together with the old position. Click the left mouse button to complete moving.

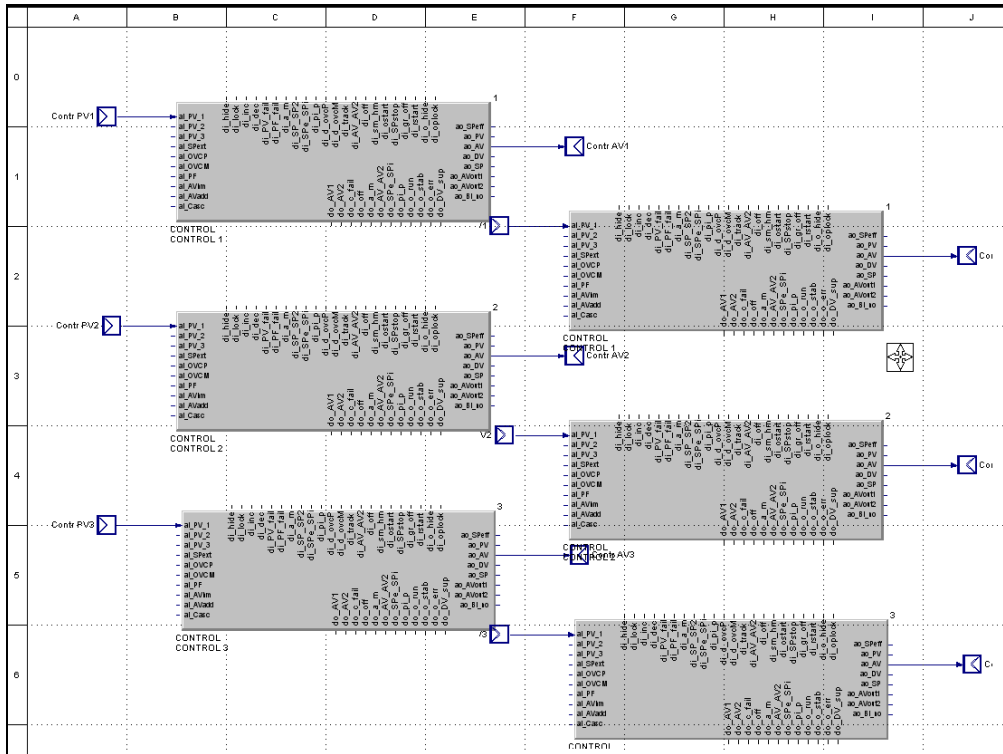


Fig. 180: Moving a worksheet

II-3.12.7 Printing the project

You can print an individual worksheet or the entire project.

1. **Start the "Print Project" menu command:** Start the "Project/Print Project" menu command or the "Print Page" menu command (if you wish to print only a worksheet) to start the "Printer Setup" dialog.

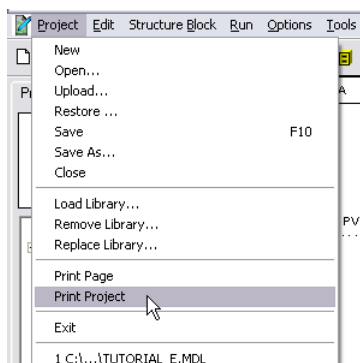


Fig. 181: Print Project

2. **Start printing:** Select the printer and click "OK" to start printing.

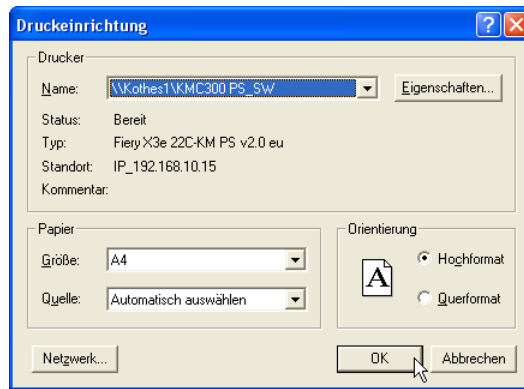


Fig. 182: "Printer Setup" dialog

**NOTE!**

Make sure that the worksheet size is suitable for the printout. Further information on adapting the worksheet can be found in the section "II-3.12.4 Changing worksheet size".

II-3.12.8 Using symbols

You can adapt the standard appearance of the blocks. For example, you can assign a background picture to a block.

To do this, proceed as follows:

1. **Start edit mode:** Make sure that you are in edit mode.
2. **Select program block:** Select the program block you wish to edit with a double click on the "Project" tab.
3. **Execute context menu command "Design":** Start the "Design" command from the context menu of the respective block.

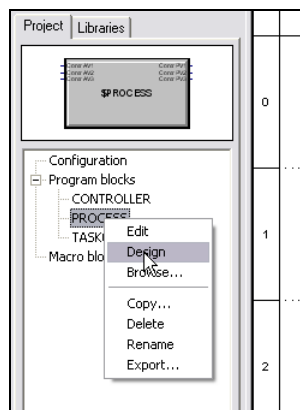


Fig. 183: Start design mode

4. **Deactivate "Button style" option:** Deactivate this option by clicking the "Button style" button (see following graphic). The "Button style" option causes blocks to be displayed three-dimensionally.
5. **Start "Bitmap for symbol" dialog:** Click the "Bitmap for structure block symbol" button (see following graphic) to start the dialog for inserting a bitmap.

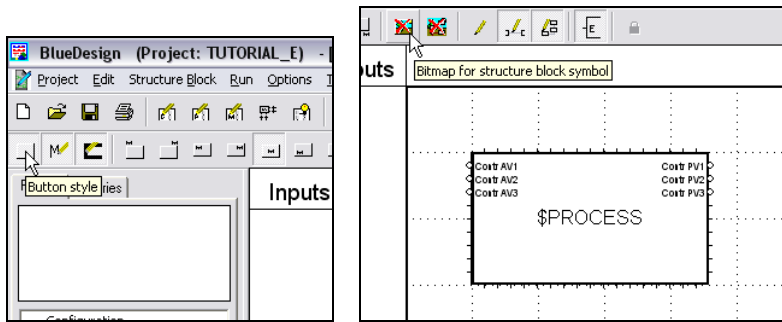


Fig. 184: "Button style" button/add symbol

6. **Select bitmap:** Select the desired symbol in the "Bitmap" area. Click the "Import" button to add a new bitmap to the list.
By default, the bitmap is changed to the current block size. Activate the checkbox "Change block size to bitmap size" to select the opposite process.

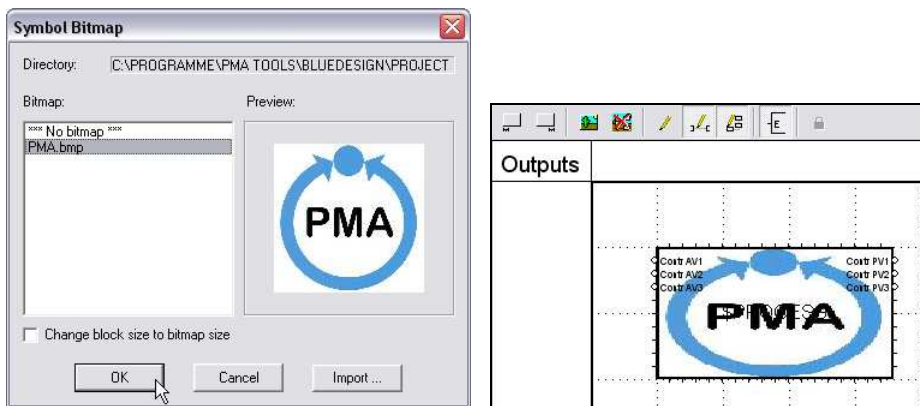


Fig. 185: Select "Bitmap for symbol display" dialog

7. **Apply input:** Click the "OK" button to apply your selection.

II-3.12.9 Using background pictures

You can place background pictures on the worksheet.

To do this, proceed as follows:

1. **Start edit mode:** Make sure that you are in edit mode.
2. **Select program block:** Select the program block you wish to edit with a double click on the "Project" tab.
3. **Execute context menu command "Design":** Start the "Design" command from the context menu of the respective block.

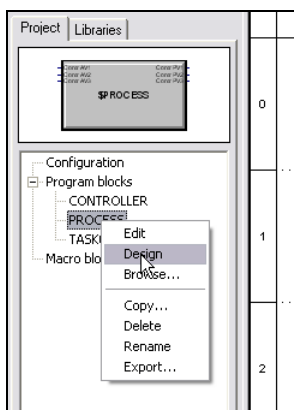


Fig. 186: Start design mode

4. **Start "Bitmap for background picture" dialog:** Click the "Background picture" button (see following graphic) to start the dialog for inserting a bitmap.
5. **Select bitmap:** Select the desired symbol in the "Bitmap" area. Click the "Import" button to add a new bitmap to the list.
By default, the bitmap is adapted to the current block size.

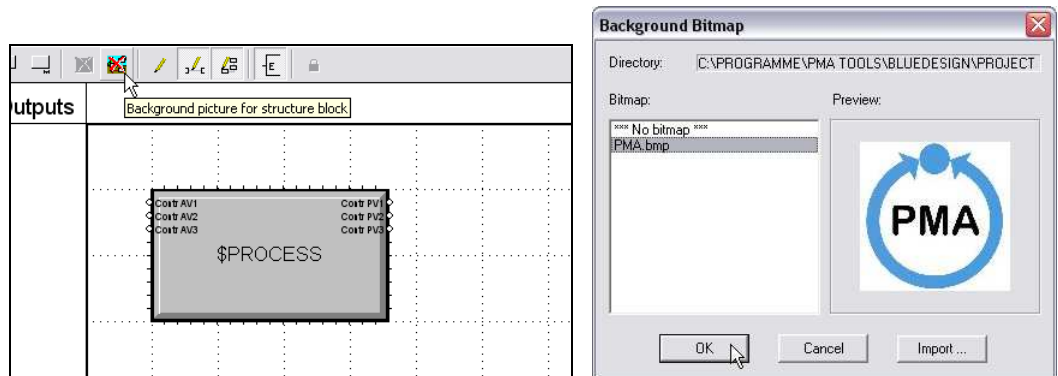


Fig. 187: "Background picture" button

6. **Apply input:** Click the "OK" button to apply your selection.

II-3.12.10 Formatting structure blocks

To do this, proceed as follows:

1. **Start edit mode:** Make sure that you are in edit mode.
2. **Select program block:** Select the program block you wish to edit with a double click on the "Project" tab.
3. **Execute context menu command "Design":** Start the "Design" command from the context menu of the respective block.

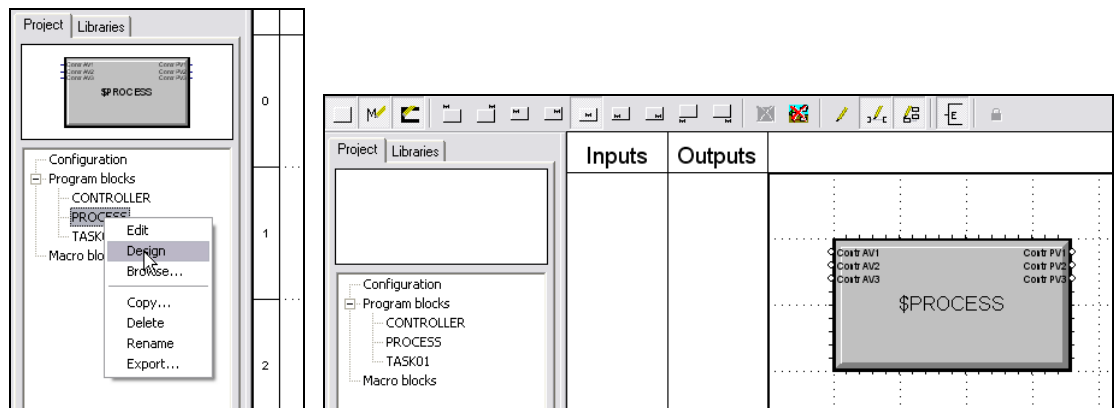









Fig. 188: Start design mode

The following formatting options are available here:

| Buttons | |
|---------|---|
| | Display blocks three-dimensionally (as buttons) |
| | Display block name |
| | Surround block with frame |
| | Display name at top left outside of the block |

| | |
|---|--|
|  | Display name at top right outside of the block |
|  | Display name at top left |
|  | Display name at top right |
|  | Display name at center |
|  | Display name at bottom left |
|  | Display name at top right |
|  | Display connection name inside or outside of block |

II-3.12.11 Using keyboard commands

Numerous commands can also be executed using keyboard shortcuts. However, not all commands are available in all modes. The following table provides an overview ("E" = edit mode, "R" = run mode, "O" = online observation mode).

| Keyboard shortcut | Available in | | | Command |
|-----------------------|--------------|---|---|---|
| | E | R | O | |
| + (on numeric keypad) | X | X | X | Worksheet zoom, enlarge view |
| - (on numeric keypad) | X | X | X | Worksheet zoom, reduce view |
| F1 | X | X | X | Open online help |
| F5 | X | X | | Open "Configuration" area |
| F6 | X | X | | Open "Program block" area |
| F7 | X | X | | Open "Macro block" area |
| F8 | X | | | Start run mode |
| F9 | | X | X | Switch between online and offline mode |
| F10 | X | X | | Save current project: |
| CTRL+F1 | X | X | X | Help on block where mouse is currently positioned |
| SHIFT+F4 | X | X | X | Arrange windows next to each other |
| SHIFT+F5 | X | X | X | Arrange windows on top of each other |
| CTRL+1 | X | | | Working on the drawing |
| CTRL+2 | X | | | Replace block |
| CTRL+3 | X | | | Deleting a block |
| CTRL+4 | X | | | Block sequence |
| CTRL+A | X | | | Select all |
| CTRL+C | X | | | Copy |
| CTRL+U | X | | | Deselect |

| | | | | |
|------------|---|---|---|-----------------------------------|
| CTRL+V | X | | | Insert |
| CTRL+X | X | | | Cut |
| CTRL+DEL | X | | | Delete |
| CTRL+INS | X | | | Copy selected object to clipboard |
| ALT+1 | X | X | X | Show / hide toolbar |
| ALT+2 | X | X | X | Show / hide status line |
| ALT+3 | X | X | X | Show / hide status line |
| ALT+4 | X | X | X | Show / hide project tree |
| Arrow keys | X | X | X | Scroll worksheet |
| POS1 | X | X | X | Jump to top of worksheet |
| END | X | X | X | Jump to end of worksheet |

II-3.13 Application visualization

In nearly all cases it is useful to provide applications with convenient operating pages. These operating pages are referred to as "HMI" (Human-Machine Interface) in the BlueDesign terminology.

The PMA library provides pre-defined operating pages for some blocks (e.g. for controllers or programmers). These operating pages can be used "out of the box" - as is.

In addition, it is often useful to create application-specific operating pages. To do this, use the "Mask Designer" in BlueDesign. You simply "drag & drop" the components of your user interface to the desired position. All elements of your HMI that should interact with the application have to be connected with interface blocks at a later time.

Pages created using the Mask Designer are provided to the user as the "main page". This is the first page the user sees when the *KS 108 easy* is started.

II-3.13.1 Start Mask-Designer

1. **Start edit mode:** Make sure that you are in edit mode.
Although the mask editor can also be started in run mode, the interface blocks can be created in that mode.
2. **Start Mask Designer:** Use the menu command "Tools/Mask Designer ..." to start the mask designer.

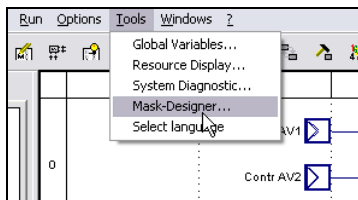


Fig. 189: Start Mask-Designer

II-3.13.2 Structure of mask editor

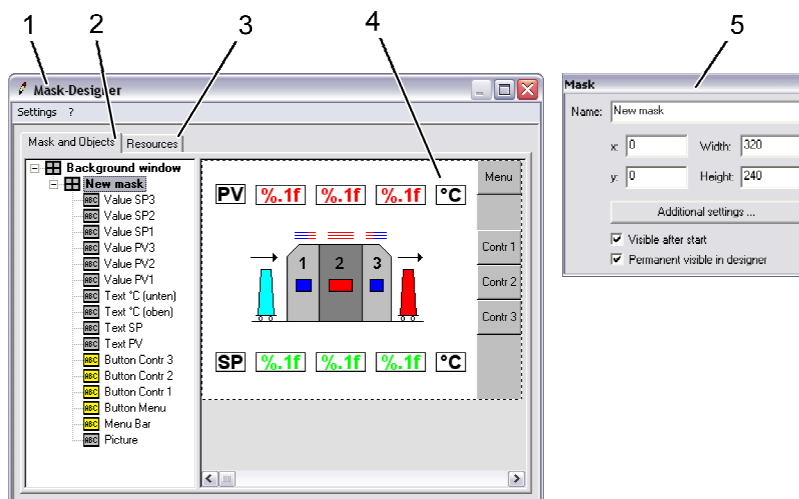


Fig. 190: Structure of HMI editor

- 1 **Mask Designer dialog window:** Main window for designing the HMI
- 2 **"Masks and Objects" tab:** Overview of components ("Objects") of the mask
- 3 **"Resources" tab:** Directory of integrated images and texts
- 4 **Worksheet:** workspace for designing the dialog
- 5 **"Properties" dialog window:** Dialog window for defining the properties of an element ("Object")

The following terms are important when working with the mask editor:

- **Background window:** Each HMI has a background window. All mask elements of your choosing can be placed on the background window.
The background window cannot be hidden. However, other masks can be placed in front of the background window.
- **Mask:** A mask comprises several blocks. It is similar to a dialog in Windows. Masks can be placed one on top of the other. Masks can be shown and hidden (together with all blocks placed on the masks). In this way, for example, popup windows can be implemented using masks.
- **Object:** The individual components of a dialog are referred to as "objects". "Bargraph", "bitmap", "button" and "rectangle" are objects.

II-3.13.3 Editing properties

All objects of your user interface can be adapted by means of properties dialogs. The dialogs are context-dependent.

Editing properties of the background window

1. **Select "Background window" entry:** Select the "Background window" entry in the tree structure on the "Masks and Objects" tab (see graphic below).

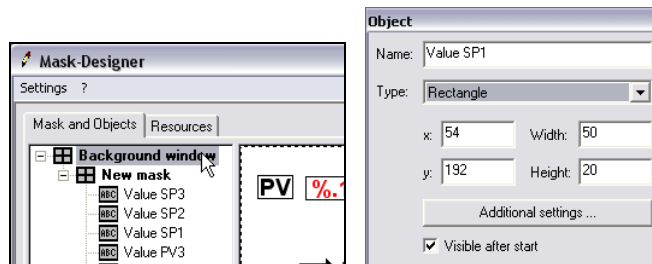


Fig. 191: Editing properties of the background window

2. **Edit properties:** The properties can be edited in the properties window (see graphic above). The following properties are available.
 - **Name:** Name of the background window (as displayed in the tree structure).
 - **Width/height:** Size of the background window (in pixels).
The size must correspond to the size of the display in the target device. For the *KS 108 easy* this value is 320 x 240 pixels.
 - **Additional settings:** Click the "Additional settings" button in order to change the color of the background window or to select a language or a background bitmap.
 - **Permanent visible in Designer:** Select this option if the background window should be visible also when another mask is activated.

Editing properties of masks

1. **Select "Mask" entry:** Select the mask in the tree structure on the "Masks and Objects" tab (see graphic below).

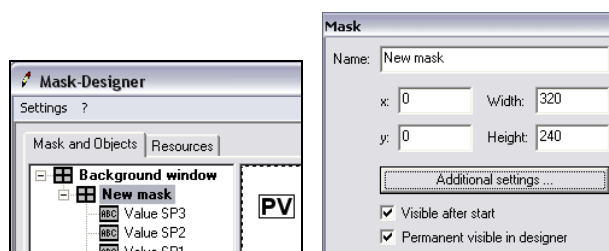


Fig. 192: Editing properties of the background window

2. **Edit properties:** The properties can be edited in the properties window (see graphic above). The following properties are available.
 - **Name:** Name of the mask (as displayed in the tree structure).
 - **Width/height:** Size of the mask (in pixels).
The size of the mask must not be larger than the size of the display in the target device. For the *KS 108 easy* this value is 320 x 240 pixels.
 - **Additional settings:** Click the "Additional settings" button in order to change the color of the background window or to select a language or a background bitmap.
 - **Visible after start:** Specifies whether the mask is visible after the application is started. However, please note: A mask can cover another mask. Therefore, if several masks are displayed simultaneously, it is possible that parts of one mask or the entire mask will not be visible.
 - **Permanent visible in Designer:** Select this option if the mask should be visible also when another mask is activated.

Editing properties of objects

1. **Select object:** Select the object in the tree structure on the "Masks and Objects" tab (see graphic below).

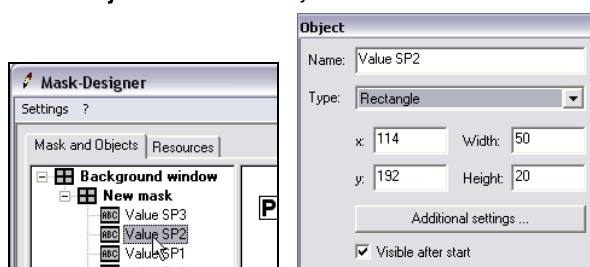


Fig. 193: Editing object properties

2. **Edit properties:** The properties can be edited in the properties window (see graphic above). The following properties are available.
 - **Name:** Name of the object (as displayed in the tree structure).
 - **Type:** Indication or selection of object type (" bargraph", " bitmap", " button" and " rectangle").
 - **Width/height:** Size of the object (in pixels).
The size of the object must not be larger than the size of the display in the target device. For the *KS 108 easy* this value is 320 x 240 pixels.
 - **Additional settings:** Click the "Additional settings" button in order to define properties for an object (for further information, see section *II-3.13.3 Editing properties*).
 - **Visible after start:** Specifies whether the object is visible after the application is started. However, please note: An object can cover another object. Therefore, if several objects are displayed simultaneously, it is possible that parts of one object or the entire object will not be visible.

"Rectangle" properties

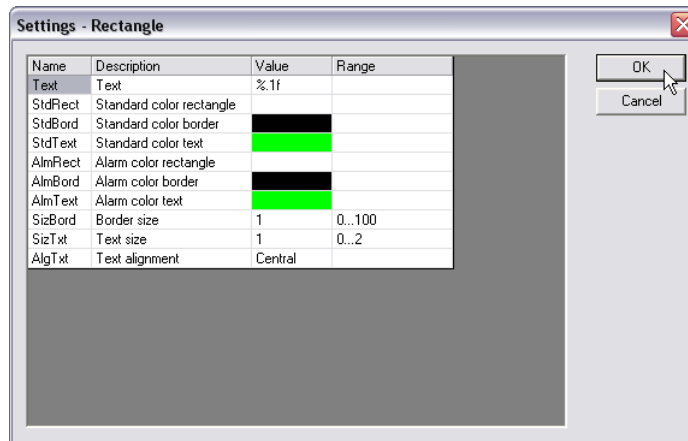


Fig. 194: "Rectangle" properties dialog

- **Text:** In the "Rectangle" object you can display a value. It can be a fixed value or a value determined via the interface block.
The fixed value (constant) is entered in the "Value" column. If you wish to use a value determined via an interface block, on the other hand, you have to enter the character string "%.1f". This string defines *that* a value should be output. It also defines *how* the value should be formatted. The "f" specifies that the number is of the type "Float". The number ("1" in this example) specifies how many decimal places should be displayed.
- **Standard view formatting:** The properties "StdRect", "StdBord" and "StdText" define the colors for the normal view of the rectangle. Click the "Value" column to specify a color.
- **Alarm view formatting:** The properties "AlmRect", "AlmBord" and "AlmText" define the colors for the "Alarm view" of the rectangle. This option allows you to signal an alarm status. The display of the alarm status is started via the digital input "di_Alarm" of the interface block.
Click the "Value" column to specify a color.
- **SizBord:** Size of the border of the rectangle.
- **SizTxt:** Size of text displayed in the rectangle (the normal value is "1").
- **AlgTxt:** Alignment of text displayed in the rectangle (available options are "left-justified", "right-justified" and "center").

"Button" properties

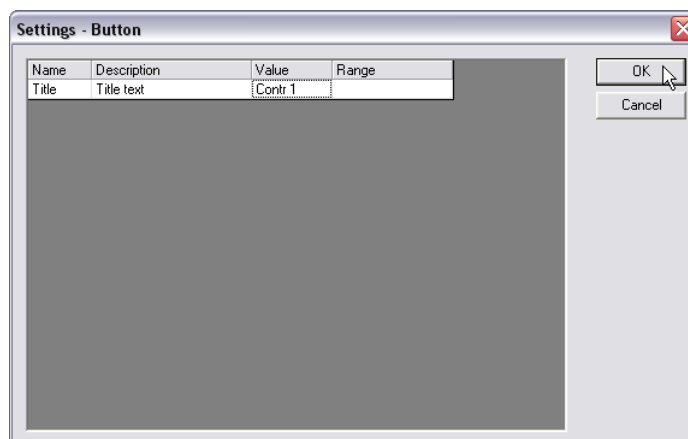


Fig. 195: "Button" properties dialog

- **Title:** Enter the button title ("Caption") in the "Value" column.

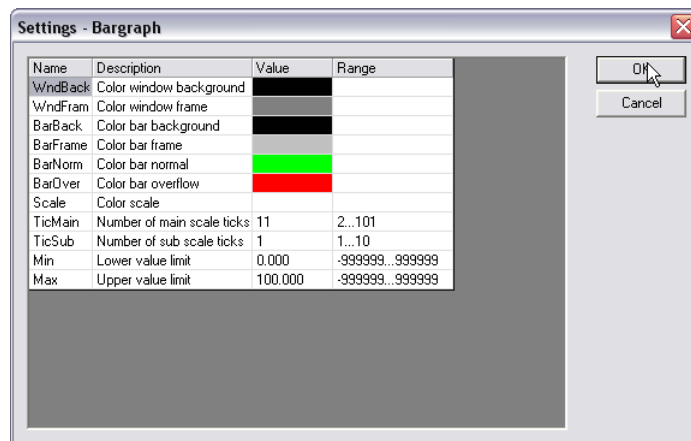
"Bargraph" properties

Fig. 196: "Bargraph" properties dialog

- **Color:** The lines 1 through 7 ("WindBack" through "Scale") are used to define the color of the bargraph.
- **Scale division:** "TicMain" and "TicSub" are used to specify how many main scale tics should be displayed on the bargraph ("TicMain") and how many sub-scale tics ("TicSub") are used to sub-divide it.

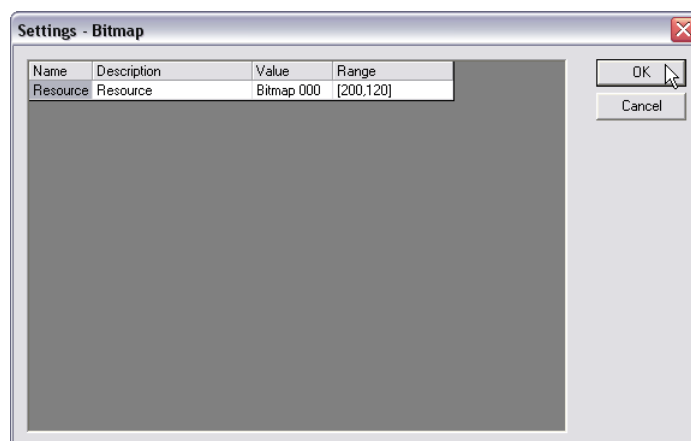
Bitmap properties

Fig. 197: "Bitmap" properties

- **Resource:** In the "Value" field, enter the name of the bitmap to be inserted.

**NOTE!**

The bitmap must first be imported to the "Resources" tab (see following section).

II-3.13.4 Adding resources (bitmap)

Bitmaps to be displayed on the HMI must first be added to the project. This is done on the "Resources" tab. To do this, proceed as follows:

1. **Select the "Resources" tab:** Select the "Resources" tab with a mouse click.
2. **Select resource type:** Select the "Bitmaps" entry in the tree structure.
3. **Add new resource:** Start the context menu command "New" at the right of the dialog with the right mouse button (see following graphic). Now you see the "New Bitmap Resource" dialog.

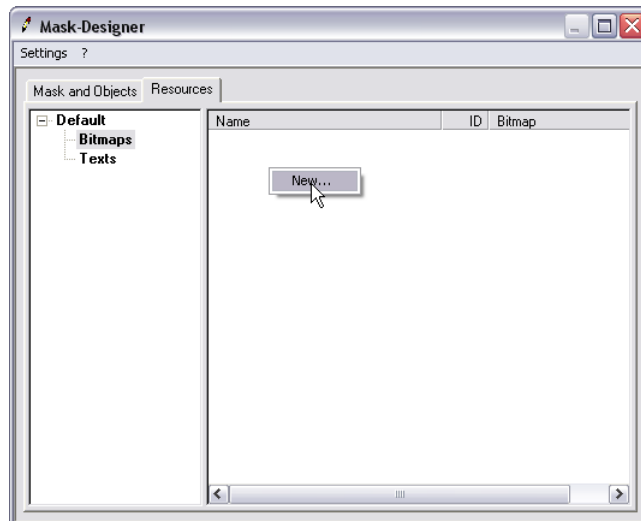


Fig. 198: Add resources

4. **Import bitmap:** Click the "Import" button in the "New Bitmap Resource" dialog to import a new bitmap. The "Import Bitmap" dialog is displayed.

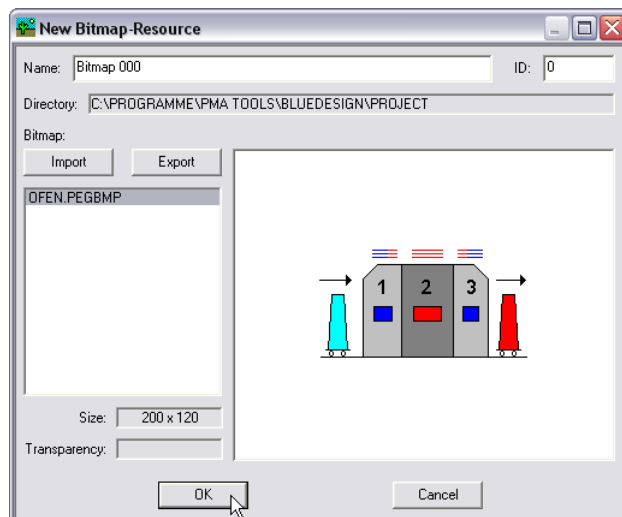



Fig. 199: "New Bitmap Resource" dialog

5. **Select bitmap:** Click the  button to select the bitmap to be inserted. After making your selection, a preview of the bitmap is displayed. Give the bitmap a name (in the "Name" field). This name is used to select the bitmap in the mask editor (see section II-3.13.3 *Editing properties*). You have the possibility to display parts of the bitmap transparently. To do this, click the selection field "Transparency" and then select the color for the areas to be hidden via the "Color" button.

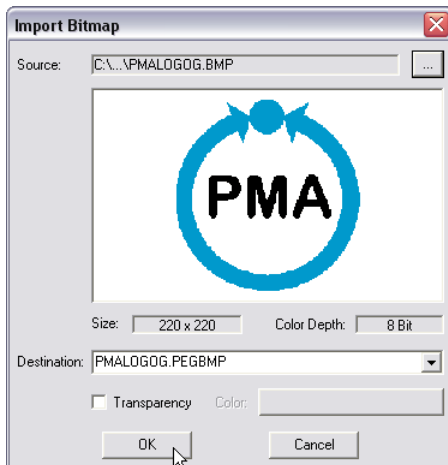


Fig. 200: Start "Import Bitmap" dialog

6. **Save input:** Click the "OK" button to save your input. Under some circumstances you may see the following warning:

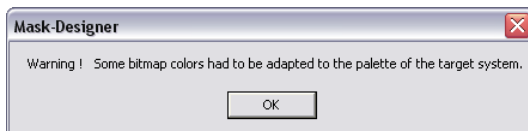


Fig. 201: Adapt bitmap warning

Since the target device can display only bitmaps with 256 colors, bitmaps containing colors not within the color range of the *KS 108 easy* will be corrected. This can result in the incorrect display of colors. If the result of the conversion results in an unusable bitmap, it is advisable to first edit it in a graphic editor.

7. **New bitmap resource:** Click the "OK" button in the "New Bitmap Resource" dialog to apply your selection.

II-3.13.5 Editing resources

Click on the resource to be edited in the right side of the dialog. Then select the desired command in the context menu ("New ...", "Edit ..." or "Delete").

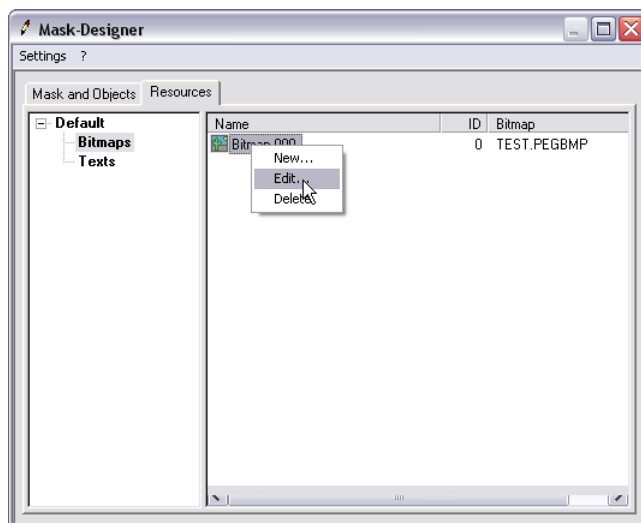


Fig. 202: Editing resources

II-3.13.6 Creating a new mask

1. **Select the "Masks and Objects" tab:** Select the "Masks and Objects" tab with a mouse click.

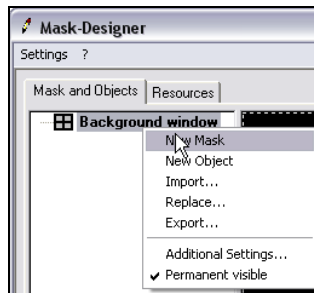


Fig. 203: Creating a new mask

2. **Create new mask:** Start the "New Mask" command in the context menu. A new mask (highlighted in black in the graphic below) is created in a standard size on the workspace. If necessary, change the appearance of the mask in the corresponding properties dialog (see *II-3.13.3 Editing properties*).

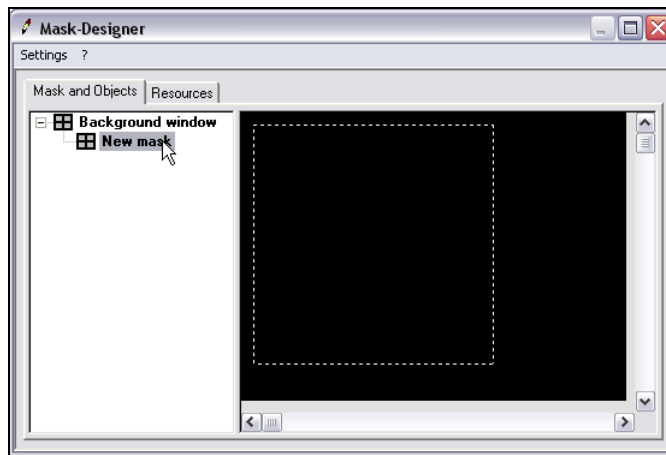


Fig. 204: New mask

3. **Move mask:** If the mask needs to be moved: position the mouse on the mask. Press the left mouse button. The mouse pointer now appears as four arrows. Move the mask and then release the left mouse button.

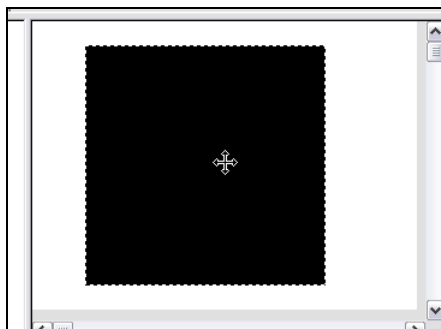


Fig. 205: Move mask

4. **Change size:** The size of the mask can be changed via the properties dialog (see *II-3.13.3 Editing properties*) or with the mouse. Position the mouse pointer on the lower or left border of the mask; the mouse pointer appears as a double arrow. Press the left mouse button and change the size of the mask.



Fig. 206: Change size

II-3.13.7 Creating a new object

1. **Select the "Masks and Objects" tab:** Select the "Masks and Objects" tab with a mouse click.

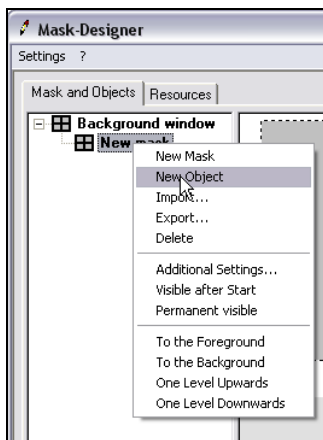


Fig. 207: Creating a new object

2. **Create new object:** Start the "New Object" command in the context menu. A new object (highlighted in black in the graphic below) is created in a standard size on the workspace. If necessary, change the appearance of the object in the corresponding properties dialog (see *II-3.13.3 Editing properties*).

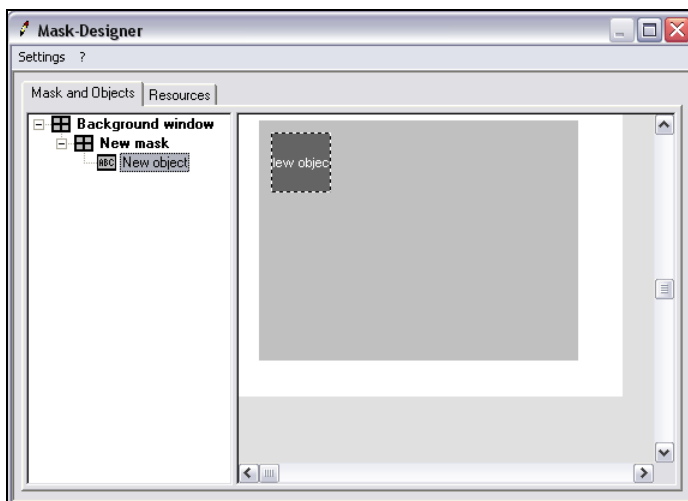


Fig. 208: New object

3. **Move object:** If the object needs to be moved: position the mouse on the object. Press the left mouse button. The mouse pointer now appears as four arrows. Move the mask and then release the left mouse button.



Fig. 209: Move object

4. **Change size:** The size of the object can be changed via the properties dialog (see *II-3.13.3 Editing properties*) or with the mouse. Position the mouse pointer on the lower or left border of the object; the mouse pointer appears as a double arrow. Press the left mouse button and change the size of the mask.

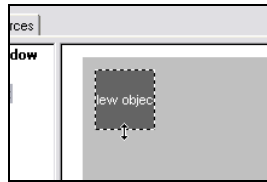


Fig. 210: Change size

5. **Define object type:** By default, a new object has the type "unknown". Therefore, define the type of block in the properties dialog (see II-3.13.3 *Editing properties*).

II-3.13.8 Using interface blocks

An HMI initially has no functions. The functions are assigned to the HMI by interface blocks. Therefore, you have to create interface blocks for all masks and objects of the HMI - in any case for those that should respond to input, display values or interact with other blocks.

To do this, proceed as follows:

1. **Select program or macro block:** On the "Project" tab, select the program or macro block into which the interface blocks are to be inserted.



NOTE!

It is always a good idea to save the interface blocks for an HMI in a separate program or macro block.

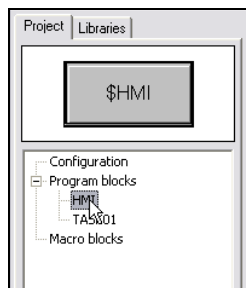


Fig. 211: Select program or macro block

2. **Library "LIB401 HMI":** Select the library "LIB401 HMI" on the "Libraries" tab.
3. **Select block type:** Select the block type that you wish to use (e.g. "Rectangle") with a mouse click.
4. **Drag block to worksheet:** Drag the block with the mouse to the worksheet (corresponding to graphic below).

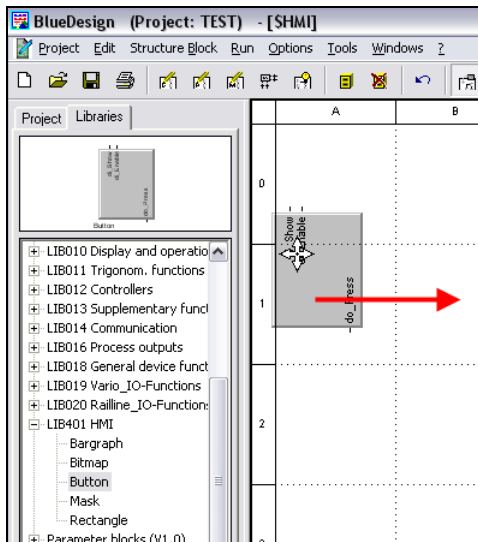


Fig. 212: Library "LIB401 HMI"

5. **Link interface block to HMI:** As soon as you have placed the block on the worksheet, the Mask Designer is started. (Alternatively, you can start the Mask Designer by positioning the cursor on the interface block and starting the context menu command "Parameter Dialog" with the right mouse button.) Now you have to assign the interface block to an element of the HMI. Click the corresponding HMI element (the "OK" button in the graphic) in the tree structure.

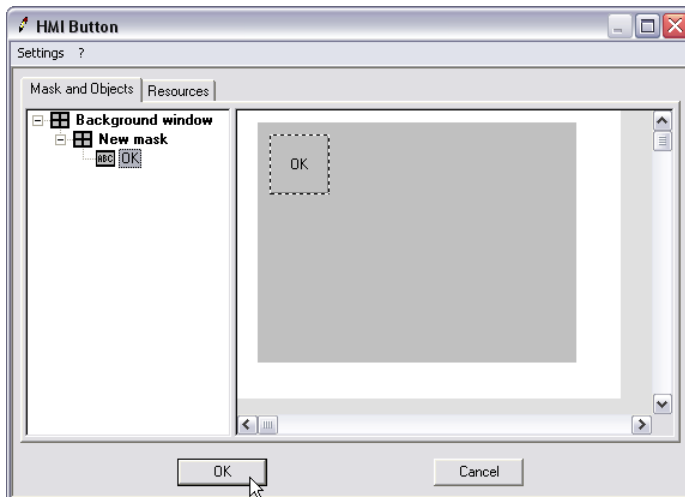


Fig. 213: Link interface block to HMI


6. **Apply input:** Then click the "OK" button to save your input.

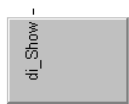

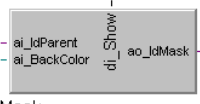
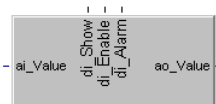


NOTE!

The "OK" button is active only if the interface block is the same type as the selected HMI element.

II-3.13.9 Properties of the interface blocks

| Block | Properties |
|---|---|
|  <p>Bargraph</p> | <ul style="list-style-type: none"> ■ Input "ai_Pos": Value to be displayed in the bargraph. ■ Input "di_Show": If the input is assigned the value "true" (0), the block is displayed. |

| | |
|---|---|
|  <p>Bitmap</p> | <ul style="list-style-type: none"> ■ Input "di_Show": If the input is assigned the value "true" (0), the block is displayed. |
|  <p>Button</p> | <ul style="list-style-type: none"> ■ Input "di_Show": If the input is assigned the value "true" (0), the block is displayed. ■ Input "di_Enable": If the input is assigned the value "true" (0), the block is activated. If the block is assigned the value "false" (1), it is displayed, but not activated. ■ Output "do_Press": The output has the value "true" (0) if the button is pressed. |
|  <p>Mask</p> | <ul style="list-style-type: none"> ■ Input "ai_IdParent": Masks can be connected with other masks. This is useful if all child masks should be hidden when the parent mask is hidden. These masks are connected via the input "ai_IdParent", which is connected with the output "ao_IdMask" of the parent mask. ■ Input "ai_BackColor": The mask can be assigned a new color via this mask. ■ Input "di_Show": If the input is assigned the value "true" (0), the block is displayed. ■ Output "ao_IdMask": The output is used to connect a child mask (see <i>Input "ai_IdParent"</i>). |
|  <p>Rectangle</p> | <ul style="list-style-type: none"> ■ Input "ai_Value": Value to be displayed in the rectangle. ■ Input "di_Show": If the input is assigned the value "true" (0), the block is displayed. ■ Input "di_Enable": If the input is assigned the value "true" (0), the block is activated. If the user clicks this block in the HMI, a dialog opens in which a new value can be entered. ■ Input "di_Alarm": If the input is assigned the value "true" (0), the block is switched to alarm mode. This mode causes the display of the block to change according to the settings in the corresponding properties dialog (see:). II-3.13.3 <i>Editing properties</i>). ■ Output "ao_Value": This output transfers the value of the block. |

II-3.13.10 Calling operating pages

Frequently it is necessary to call the operating pages of blocks (e.g. of a controller) from the user-designed HMI.

To do this, use the "CALLPG" ("Call Page") block from the library "Supplementary functions".



Fig. 214: "CALLPG" block

This requires two pieces of information:

- **Input "ai_BlockNo"**: If the value "true" (0) is assigned here, the operating page is changed.
- **Input "di_d_1"**: The number of the operating page to be displayed. The number of the operating page can be obtained via the Output "ao_BI_no" of the calling operating page.

**NOTE!**

Leaving the block that was called returns the user to the calling block.

II-3.13.11 Switching between masks

If you work with several masks, then you normally have to prevent one mask from covering another mask. This can easily be achieved with the "FLIPM" block (from the "Logical Functions" library). If an input is assigned the value "true", the corresponding output delivers the same value. All other outputs deliver the value "false".

Example:

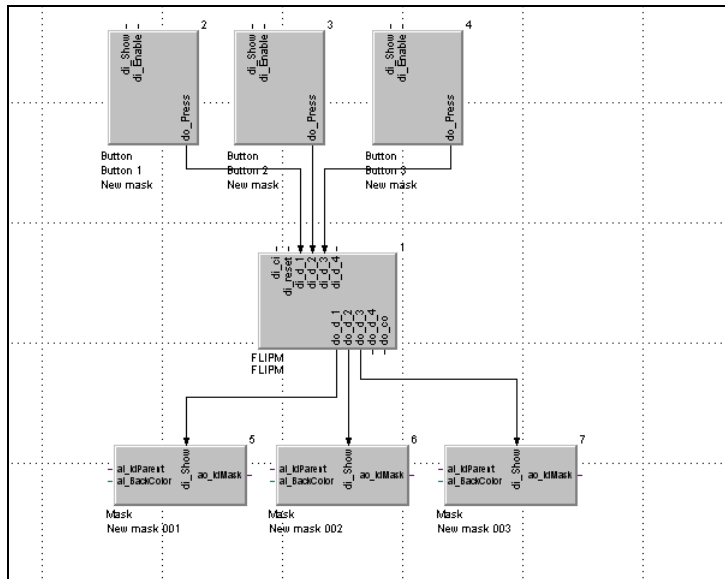


Fig. 215: Example: Use of the FLIPM block

Goal: There are three buttons ("Button 1", "Button 2" and "Button 3"). Each button should call the assigned mask ("Mask 1", "Mask 2", "Mask 3"). At the same time, the inactive masks should be hidden.

Solution: A "FLIPM" block is used. The outputs of the buttons are connected with the inputs ("di_D_1", "di_D_2", "di_D_3") of the FLIPM block. The outputs of the "FLIPM" block are connected with the inputs "di_Show" of the blocks of type "Mask".

If the button "Button 1" is now pressed, the assigned "Mask 1" is displayed; all other masks are hidden.

II-3.13.12 Displaying general data

In many cases it is necessary for the user to set the IP address of the *KS 108*, the system date or the time. The "General Data" menu is used for this purpose.

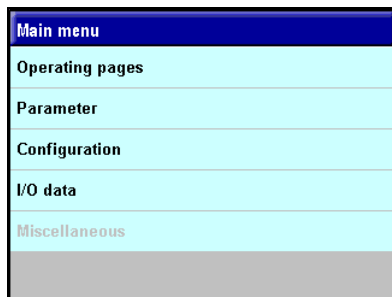


Fig. 216: "General Data" menu

However, this menu exists only if the function block "GENERAL_EASY" is used in your application.

Proceed as follows to embed the blocks:

1. **Start edit mode:** Make sure that you are in edit mode.

2. **Select the "Libraries" tab:** Select the "Libraries" tab.
3. **Select the "GENERAL_EASY" block:** Select the "GENERAL_EASY" block in the library "LIB018 General device functions".

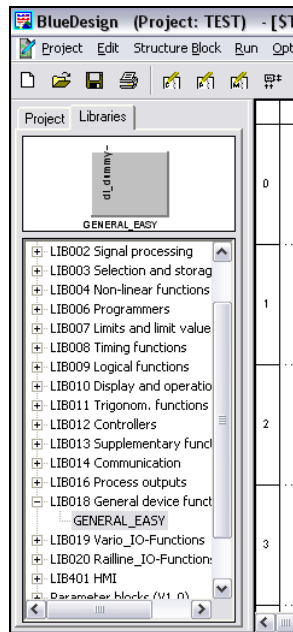


Fig. 217: Select "GENERAL_EASY" block

4. **Use program block:** With the mouse, click the program block on the tab at the top. Hold down the left mouse button and drag the block to the right onto the workspace. Then release the mouse button.

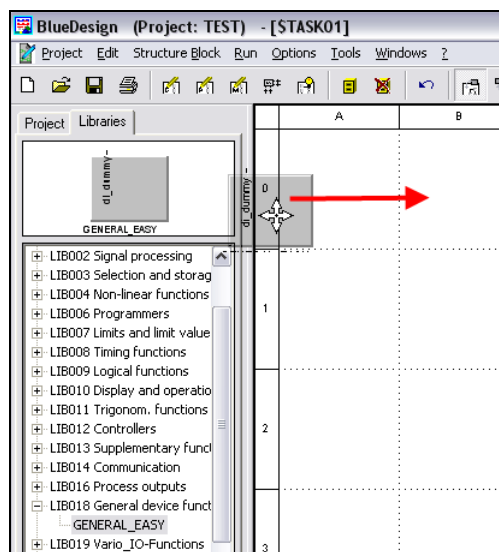


Fig. 218: Use "GENERAL_EASY" program block

After loading the application to the target system, the "General Data" menu is available in the application.

II-4 Working with the Simulator

A software simulator for the *KS 108 easy* is included in the scope of delivery. The simulator allows you to develop and test applications for the *KS 108 easy* without actually having access to the device.

II-4.1 Starting the simulator

Start the simulator from your Windows start menu. The simulator is now ready for operation.

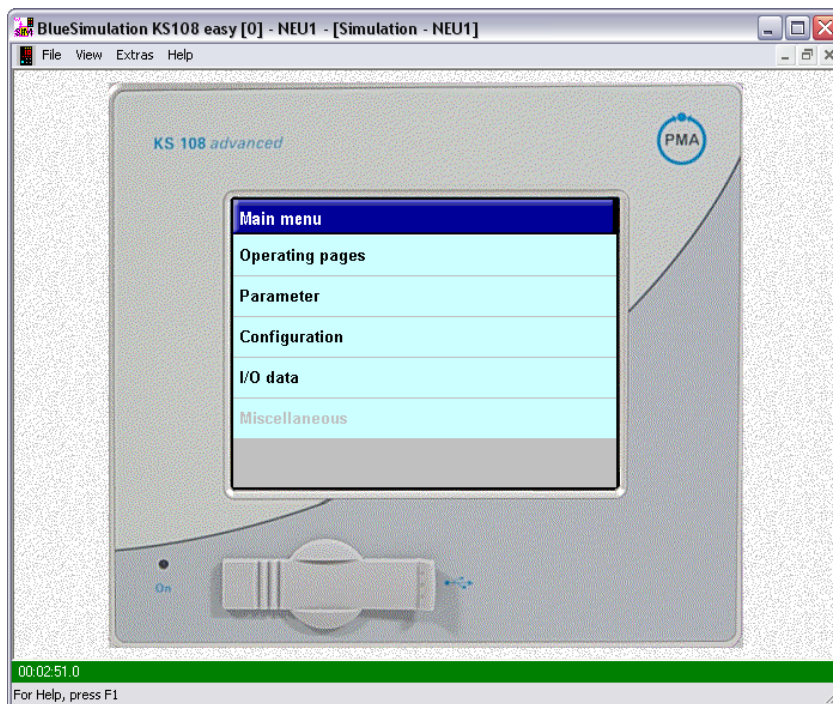


Fig. 219: Start Simulator

II-4.2 Adapting the language

You can switch the language of the simulator between German and English. To do this, open the "Extras/Language" menu and select the desired language.



Fig. 220: Adapt simulator language

II-4.3 Establishing connection to simulator

Proceed as follows to establish a connection with the simulator from the BlueDesign development environment:

1. **Start run mode:** Make sure that you are in run mode.
2. **Execute "Logon to target system" command:** Start the menu command "Run/Logon to Target System".

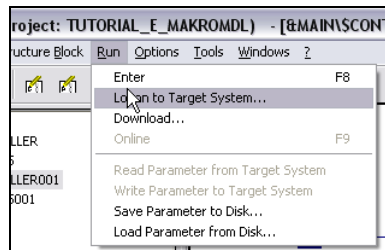


Fig. 221: Logon to target system

Now you see the "Logon to Target System" dialog.

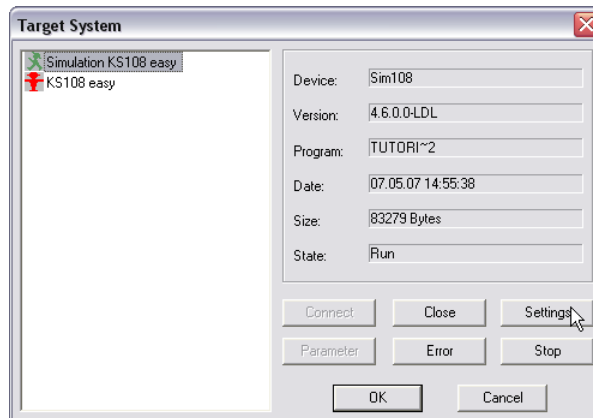


Fig. 222: "Logon to Target System" dialog

3. **Select target system:** Click at the left of the "Logon to Target System" dialog on the entry "Simulation KS108 easy".
4. **Open the "Settings" dialog:** Click the "Settings" button to start the "IntraCom Configuration" dialog.

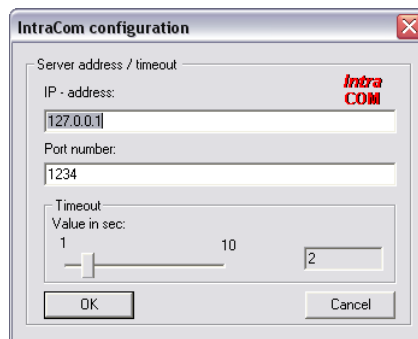


Fig. 223: "IntraCom Configuration" dialog

5. **Enter the IP address:** In the "IntraCom Configuration" dialog enter the IP address and the port number of your target system (see illustration above).
6. **Apply input:** Click the "OK" button to apply your input.

II-4.4 Debugging

Errors in applications can be debugged in two different ways.

- **Popup:** If the mouse is positioned on a connecting line, the current values are displayed.
- **Debug blocks:** Debug blocks are specially developed debugging blocks that can be integrated in your application.

II-4.4.1 Value display with popup windows

1. **Start run mode:** If you have not already done so, start run mode.
2. **Load application program:** Use the command "Run/Download" to load the application to your target device. You may be prompted to register your target device first.

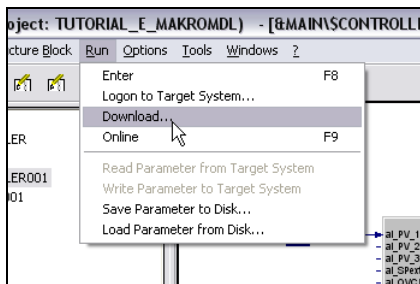


Fig. 224: Load application program

3. **Select connecting line:** Click a connecting line. In a popup window you will see the current value of the connection. The display is structured as follows:
 - Left: Internal name of the connection (in the graphic below: "DF24"). This name is generated automatically and cannot be changed by you.
 - Right: Current value of the connection (in the graphic below: "66.5328").

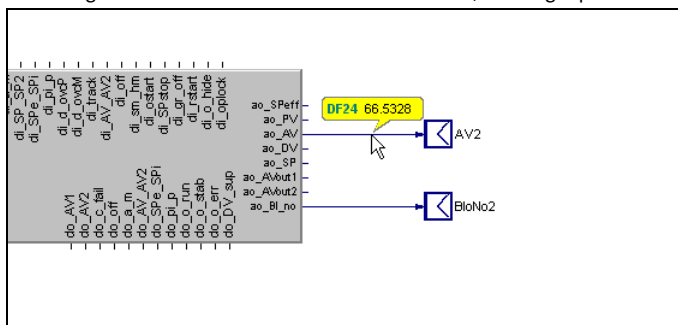









Fig. 225: Display current value of a connecting line

II-4.4.2 Value display with debug blocks

The "Debug blocks" library contains six blocks.

| Block | Name | Description |
|---|-------------------|---|
|  | 7-segment display | Displays numerical values and bit values. |
|  | Bargraph | Displays numerical values as bargraph. |

| | | |
|---|---------------------------|--|
|  | LED | Displays bit values. The color of the block is changed based on the bit (1 = on, 0 = off). |
|  | Numeric display | Display of numerical values. |
|  | Trend-Writer | Trend view of numeric values. This block displays not only the current value, but also the progression of values. |
|  | XY-Writer | Progressive view of numeric values. This block displays not only the current value, but also the progression of values. |
|  | Pointer instrument | Display of numeric values as pointers |

**NOTE!**

All debug blocks can be configured with parameters. To do this, click the block and select the context menu command "Parameter dialog".

II-4.4.3 Using debug blocks

1. **Start edit mode:** If you have not already done so, start edit mode.
2. **Select the "Debug blocks" library:** Select the "Debug blocks" library on the "Libraries" tab (see graphic below).
3. **Drag block to worksheet:** Use the mouse to drag the desired debug block to your worksheet (see following graphic).

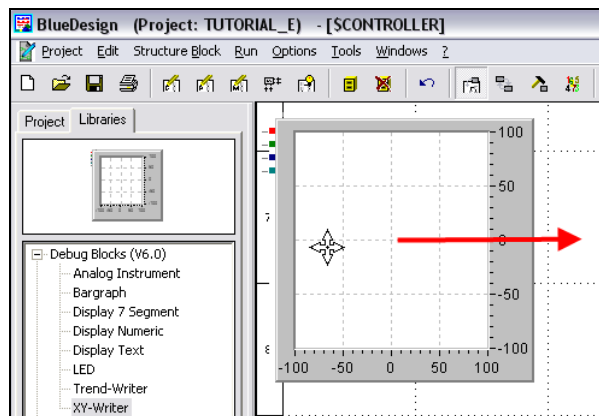


Fig. 226: Using "Debug block" library

4. **Connect debug blocks:** Connect the debug blocks with the outputs whose values are to be displayed (see following graphic; the "Trend" block is used for this example).

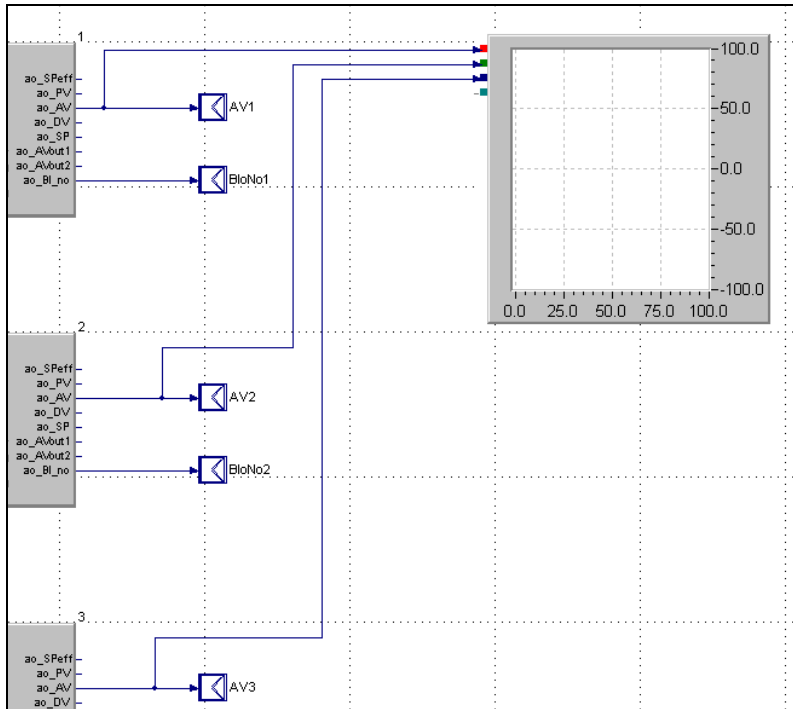


Fig. 227: Connect debug blocks

5. **Start run mode:** If you have not already done so, start run mode.
6. **Load application program:** Use the command "Run/Download" to load the application to your target device. You may be prompted to register your target device first.

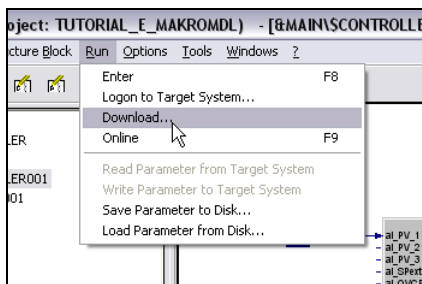


Fig. 228: Load application program

7. **Start application program:** Start the application program. After starting the program, the values are displayed in the debug block (see following graphic for example):

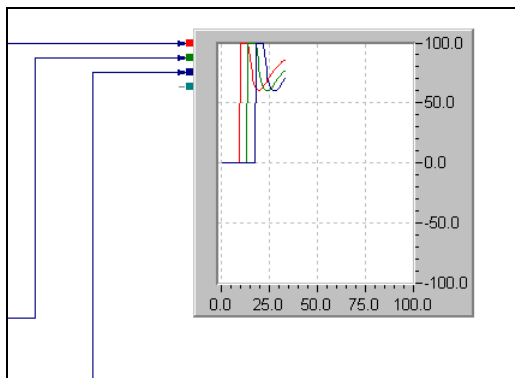


Fig. 229: Debug block display

**NOTE!**

Please note that the debug blocks are displayed only in BlueDesign and never on the target device.

II-5 A practical project

The *KS 108 easy* is a powerful and flexible multi-functional unit. The unit is adapted to your individual application using the *BlueDesign* software development environment. This is an easy-to-operate and powerful software development environment for regulation technology applications. You create applications in *BlueDesign* by selecting and integrating the pre-fabricated components (e.g. controllers or filters) to be used. The pre-fabricated components can be selected from a "library". No programming knowledge is needed in order to develop applications with *BlueDesign*.

Content

In this chapter, based on a practical example you will learn to create an application for the *KS 108 easy* using the PMA library and the *BlueDesign* software development environment.



NOTE!

Basic information on working with *BlueDesign* can be found in the sections "The components of the development environment" and "Working with the development environment".

Specifically you will learn the following in this section:

- Using the PMA library in *BlueDesign*
- Setting parameters for the application using *BlueDesign*
- Transferring the application to the *KS 108 (Simulator)*
- Working with macros

It takes two to three hours to work through the example.



NOTE!

The application and the files necessary for the practical project (bitmaps for instance) are installed automatically. You will find them in the directory
... anywhere ...

Requirements

You should have the following skills:

- Basic knowledge of the operating system *Microsoft Windows™*
- Knowledge of the programming languages standardized in DIN EN 61131-3 (particularly the function block and sequential function chart language)
- Knowledge of regulation technology

Technical prerequisites

The following technical prerequisites must be fulfilled:

- *BlueDesign* in the version 4.2.1.0 (or newer)
- The software simulator *BlueSimulation 108* must be installed (see chapt. *Installing software on the PC*)

The components

First, however, you will need a short overview of the *BlueDesign* software development environment and the PMA library:

BlueDesign

The *BlueDesign* software development environment supports the entire development cycle of a project:

- **Create the application development:** Applications are compiled from application modules. These application modules are selected from a library. An application can be structured according to various criteria. In addition, copy templates can be created, so that frequently needed application elements can easily be re-used. These reusable modules are referred to as macro blocks.

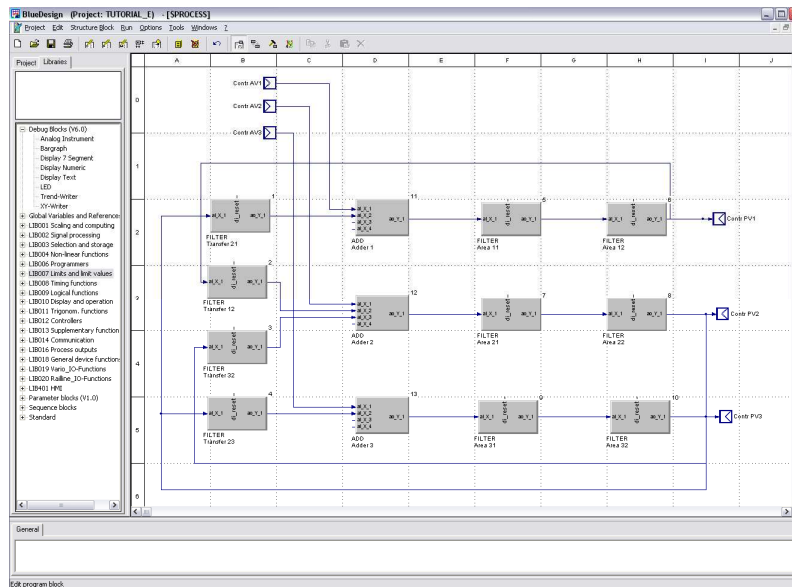


Fig. 230: Practical project: Example for creating the application development

- **Create the user interface:** The user interface for the application is created using the graphical editor. Operating elements, displays and images are positioned on the workspace and configured here. The presentation is very similar to the later appearance on the user interface of the *KS 108*.

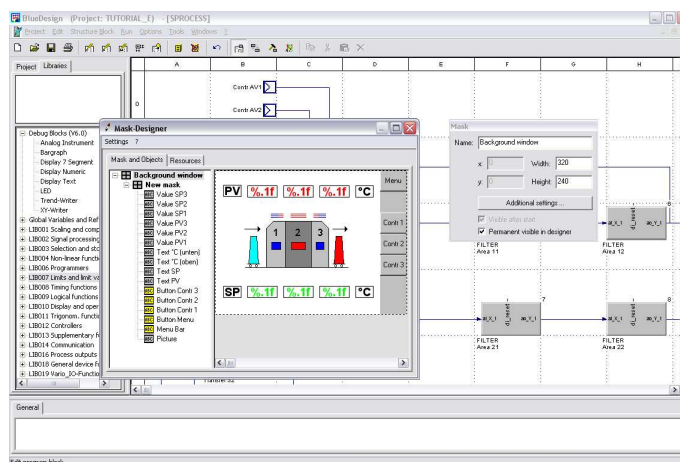


Fig. 231: Practical project: Example of creating the user interface

- **Set parameters:** The PMA library functions can be used "out of the box" - as is. However in order to adapt them to your specific requirements, they must be configured. The functions are configured by means of parameters, which can easily be entered or selected in *BlueDesign*. Value lists that offer possible options are displayed for numerous parameters. In addition, inputs are checked. If a value is not permissible, it is corrected to the next permissible value.

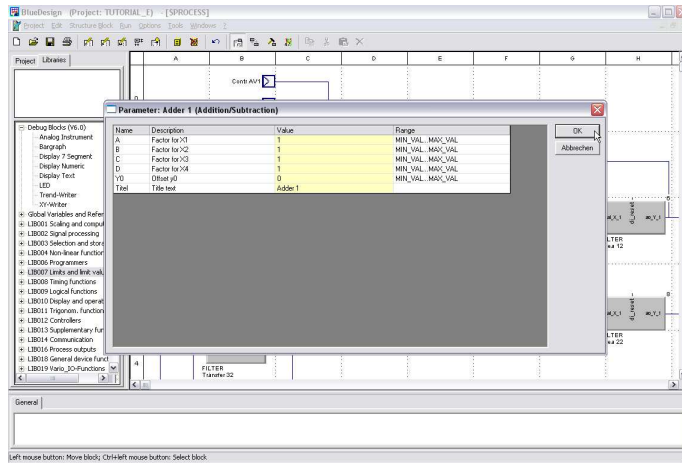


Fig. 232: Practical project: Parameter dialog

- Test the application:** Errors in applications can be debugged in two different ways: If the mouse is positioned on a connecting line, the current values are displayed (No. 2 in Fig. 701). Alternatively, specialized debugging modules from the PMA library can be used. Since the search for errors in applications is referred to as debugging, these blocks are called "debug blocks". These blocks can be used to display various information about the program during runtime (1 and 3 in Fig. 701). The display is shown only in the development environment (not on the display of the unit).

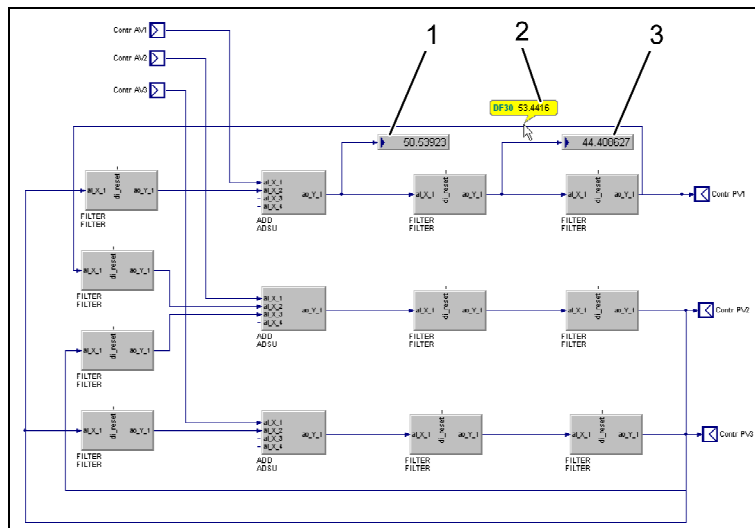


Fig. 233: Practical project: Example of debugging

PMA library

The PMA library consists of a large number of function blocks. It covers all functions that are normally required for operating a plant. These include:

- Mathematical functions**
- Logical functions**
- Alarm and limit value functions**
- Controllers**
- Program control elements**

In addition, a visualization for numerous library functions is provided, for example for the "Control" controller that we use in our example:

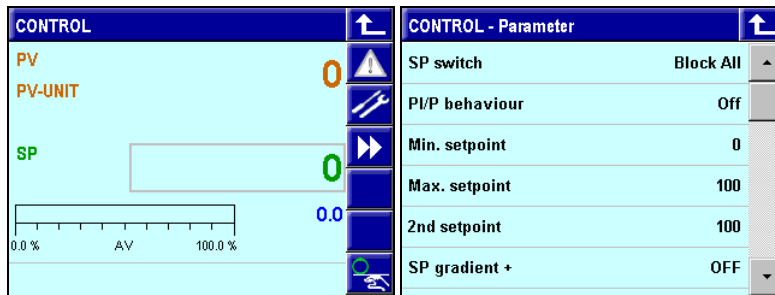


Fig. 234: Dialog "KS 108_Control", sample graphics

The PMA library helps you with program development in two ways:

- **Speed:** Applications can be more quickly developed, since they can be setup almost completely with functions from the library. For the most part the user interface is created automatically in this process.
- **Quality:** In the PMA library you will find optimized functions that you can consider as "black boxes". Thus possible programming errors are avoided.

The example

Our sample application is designed to permit controlling of an oven, and to also simulate this oven. The oven is a three-chamber oven with three chambers that can be regulated independently from each other. On the main operator page the user should be able to see the current temperature of the chambers, in addition the user should also be able to simply input the target temperature.

The application's main operator page should look approximately like the following:

- 1 Display of the process value (PV) for each of the three oven chambers
- 2 Symbolic presentation of the control process
- 3 Display of the and input of the setpoint (SP) for each of the three oven chambers.
- 4 Button: Call main menu
- 5 Buttons: Call operating pages of the controller

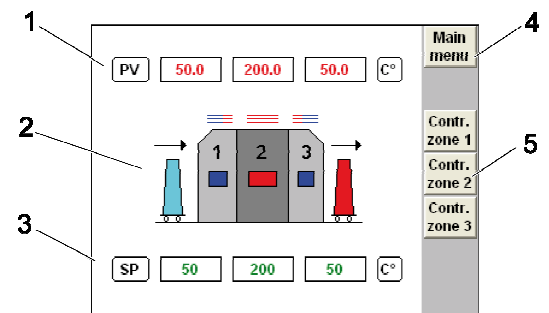


Fig. 235: Practical project: three-chamber oven, main operating page

Description

The oven's controller should have the following properties:

- Process value and setpoint will be displayed separately for all three oven chambers.
- One click on a setpoint field will open an input dialog where the setpoint can be input.
- The setpoint can range from 0 to 400.

Simulation

The behavior of the oven will be simulated. Two basic assumptions are the basis of the simulation:

- Heat transfer of 20% occurs from one oven to the adjacent oven.
- Changes to the setpoint are implemented by the oven with time delay.

II-5.1 Step 1: Creating a project

First, you must create a new project and select the target system. By selecting the target system, the development environment for cooperation with this system is configured.

First create a new project:

1. **Create project:** Proceed as follows to create a project:

- Click the menu item "Project/New".
- Select the target system (here "KS108easy") and give the project a name (here: "PRACTICAL EXAMPLE").

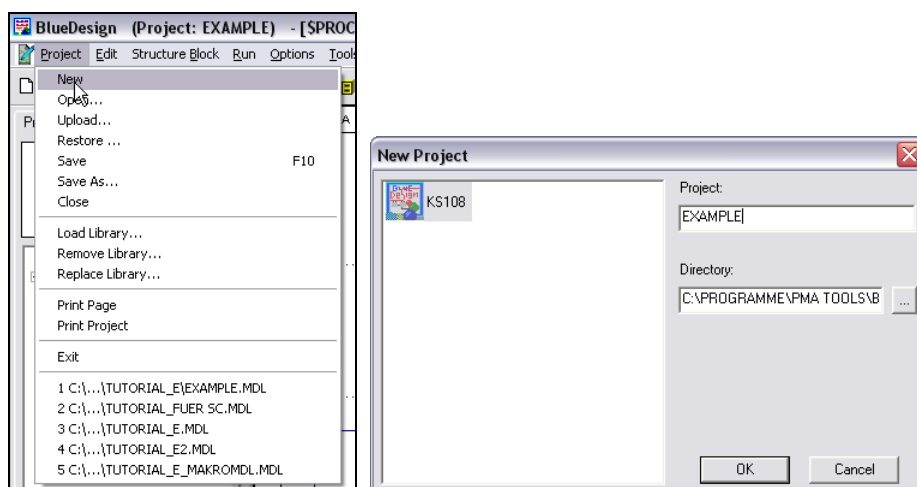


Fig. 236: Practical project: BlueControl dialog "New"

2. **Create new program block:** Now create a (new) program block. Select the "New Program Block" command in the context menu. Enter the name "PROCESS" and press the enter button. In the tree view on the left, a *program block* with the name "PROCESS" has been created.

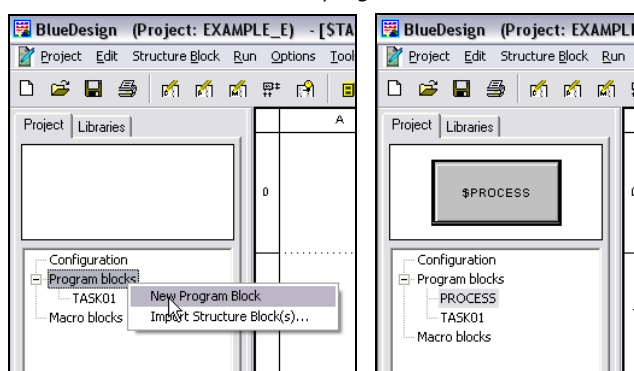


Fig. 237: Practical project: BlueControl dialog "New Program Block", project overview

3. **Delete program block PROG01:** The automatically generated program block "PROG01" is no longer needed. Highlight the block with the mouse and select the "Delete" command in the context menu.

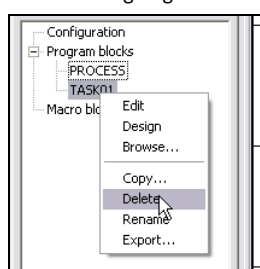


Fig. 238: Practical project: BlueControl dialog "Delete program block"

Digression: How are projects in *BlueDesign* structured?

As can be seen in the graphic, you will find three main categories on the "Project" tab that subdivide your project.

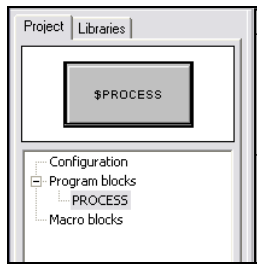


Fig. 239: Project structure in BlueDesign

- **Program blocks:** Program blocks are the basic units which make up your project. Program blocks contain blocks (e.g. controllers, filters, adders), connections between blocks and inputs and outputs. An example of such a program block can be found in Fig. 701. Your application can contain a maximum of 15 program blocks.
- **Configuration:** In the "Configuration" area you can make settings for the interaction of your program blocks. Each program block is designed as a separate sub-application or task, which means that it receives its own computing time from the runtime environment. The basic procedure used by the runtime environment is to start the program blocks one after the other (see below). However, you can also specify via the "Cycle time" that (some) program blocks are called at greater intervals (i.e. they are sometimes "skipped").

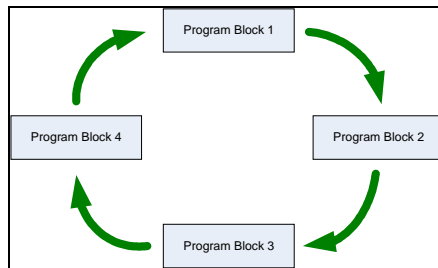


Fig. 240: Starting program components

You can also specify the sequence ("Priority") in which the program blocks are started. These settings are made in the "Configuration" area.

- **Macro blocks:** Frequently you will use identical combinations of blocks in your program blocks. In such cases, macro blocks facilitate your work. You can use macro blocks to create a copy template. You can use the copy template just like any other block in your projects.

Perhaps the overview of the project structure seems somewhat abstract and incomprehensible. However, the importance and advantages of the project structure will become clear during the further course of the example.

II-5.2 Step 2: Creating controllers



NOTE!

This chapter deals with the properties of the components used (controllers, filters, etc.) only to the extent necessary for the example. More extensive information about these components is available in the chapter "Function library".

Each chamber of the oven should be controlled a separate controller. For controlling the temperature of the three chambers of the oven, therefore, three controllers will be used from the PMA library.

Execute the following steps in this regard:

1. **Create a program block:** Create a new program block with the name "CONTROLLER". Proceed as described in "Step 1". The result should look like this:

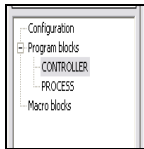


Fig. 241: Practical project: New program block "CONTROLLER"

2. **Select the "Libraries" tab:** Now select the "Libraries" tab in order to select the required blocks in the next step.

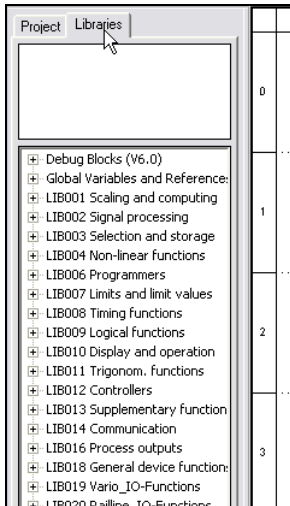


Fig. 242: Practical project: Libraries tab

Now you see an overview of the available libraries. Libraries that start with the letters "LIB" make up the PMA library. The remaining libraries provide general blocks, such as input and output fields or debug or comment fields.

3. **Select controller:** The desired controller is located in the library "LIB012 Controllers and control functions". Click the \oplus symbol in order to open the library. Then select the controller "CONTROL".

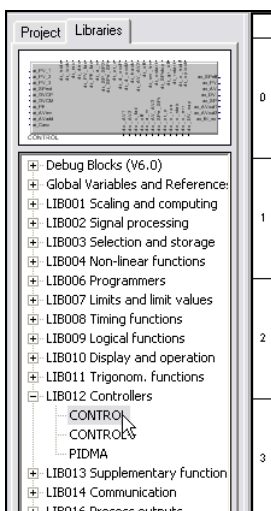


Fig. 243: Practical project: Select controller "Control"

The selected block is displayed at the top of the tab.

4. **Use controller:** Click with the mouse on the controller symbol on the tab at the top. Hold down the left mouse button and drag the block to the right onto the workspace. Then release the left mouse button.

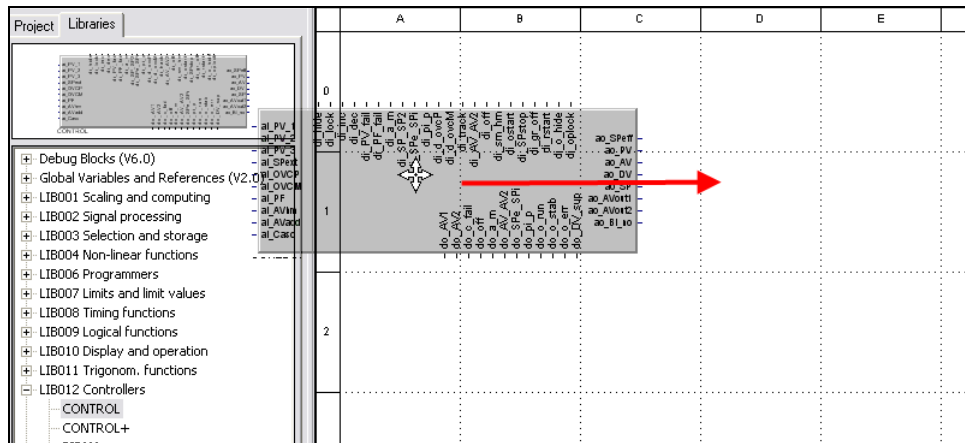


Fig. 244: Practical project: Use controller "Control"

- 5. Insert two more controllers:** In the same manner, add two more controllers of the type "CONTROL" to your program block.

The result should look something like this:

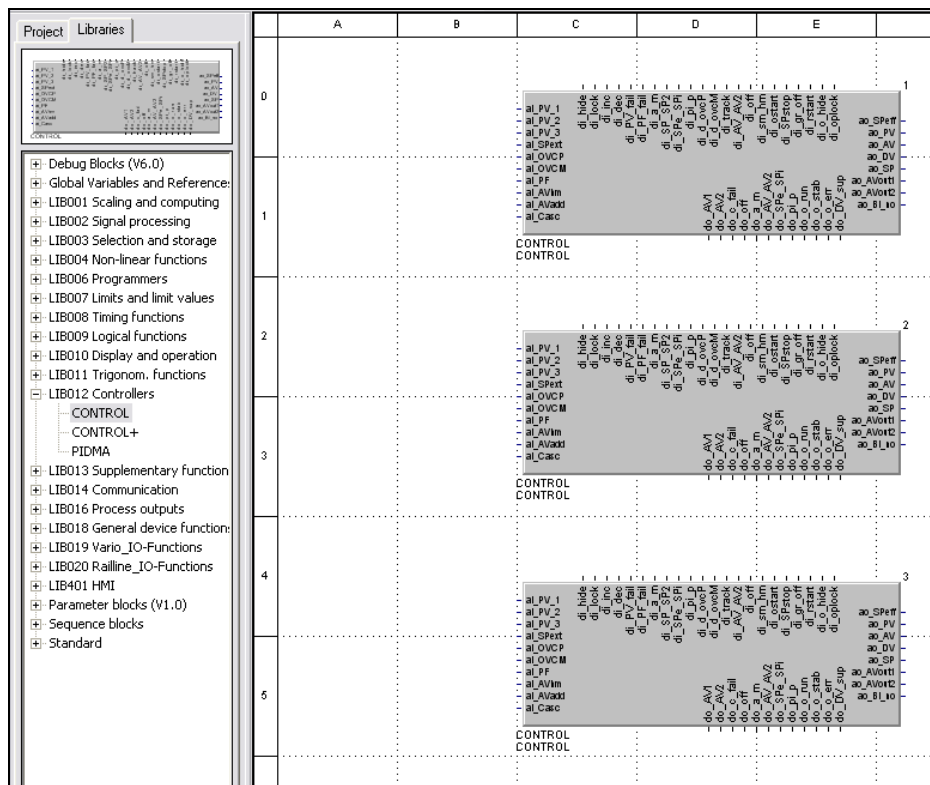


Fig. 245: Practical project: Use blocks "Control"

- 6. Name:** Click with the left mouse button on the top controller. Start the "Parameter dialog ..." command in the context menu. Scroll down in this dialog until you see the entry "Title". In the "Value" column (yellow background) enter the name "CONTROL 1" (since it is a controller for the first chamber of the oven). Save your entry by clicking "OK".

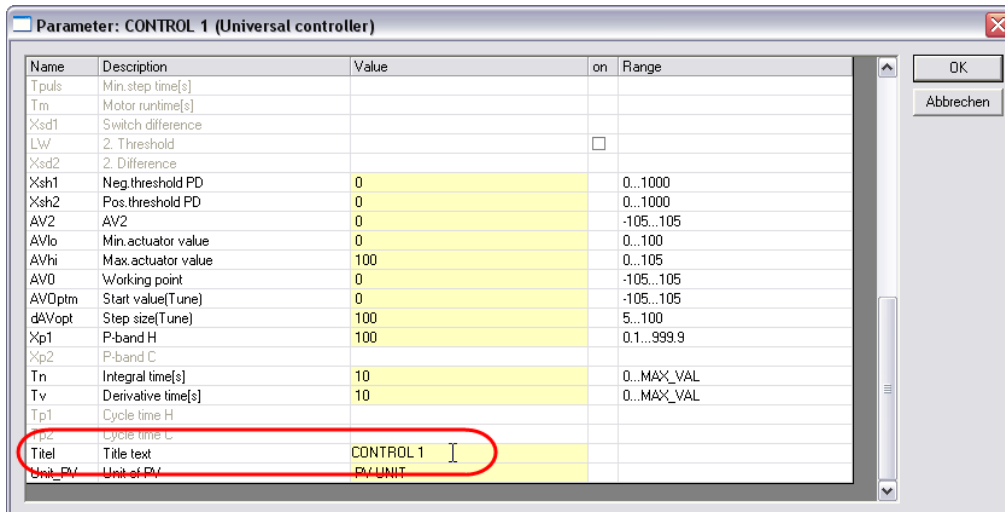


Fig. 246: Practical project: Parameter dialog

Enter the name "CONTROL 2" for the middle controller and the name "CONTROL 3" for the bottom controller.

The result should look like this:

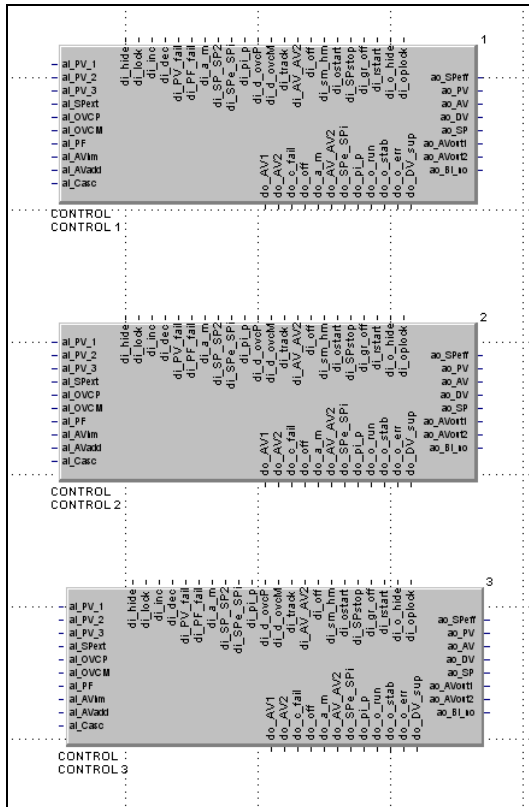


Fig. 247: Practical project: Name controllers

- Specify inputs and outputs:** The controller input and output now have to be connected. Since the actual application logic of our project should be found in the program block "PROCESS", a method has to be created for passing an input value to the controllers and for applying the output value. For this purpose, we will use the "Input" and "Output" blocks from the "Standard" library.

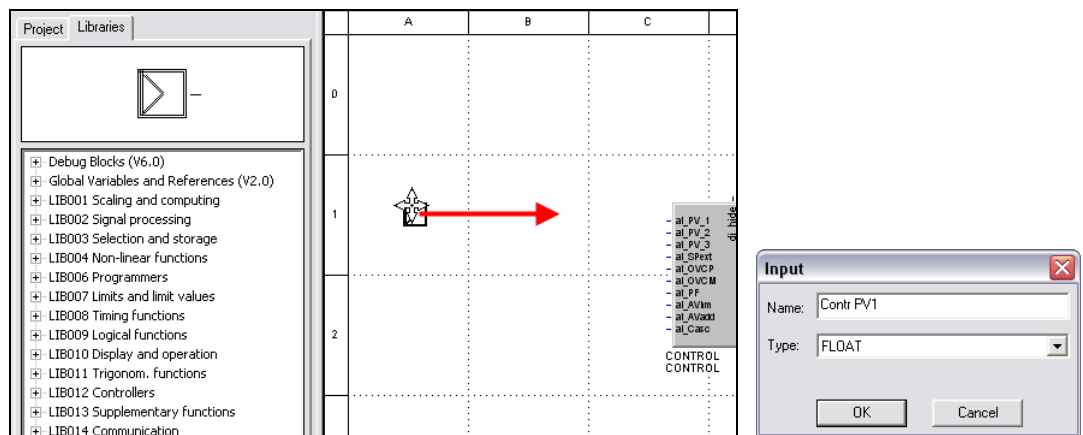


Fig. 248: Practical project: Use of inputs

- First, select the "Input" block and drag the corresponding symbol to the workspace. Now you see the "Input" dialog.
 - Give the block the name "Contr PV1".
 - Select the data type "FLOAT". Then click the "OK" button to save your input.
 - Create two more inputs (with the names "Contr PV2" and "Contr PV3").
 - Then create three outputs. To do this, use the Output block. The outputs should have the names "Contr AV1", "Contr AV2" and "Contr AV3".
- The result should look like this:

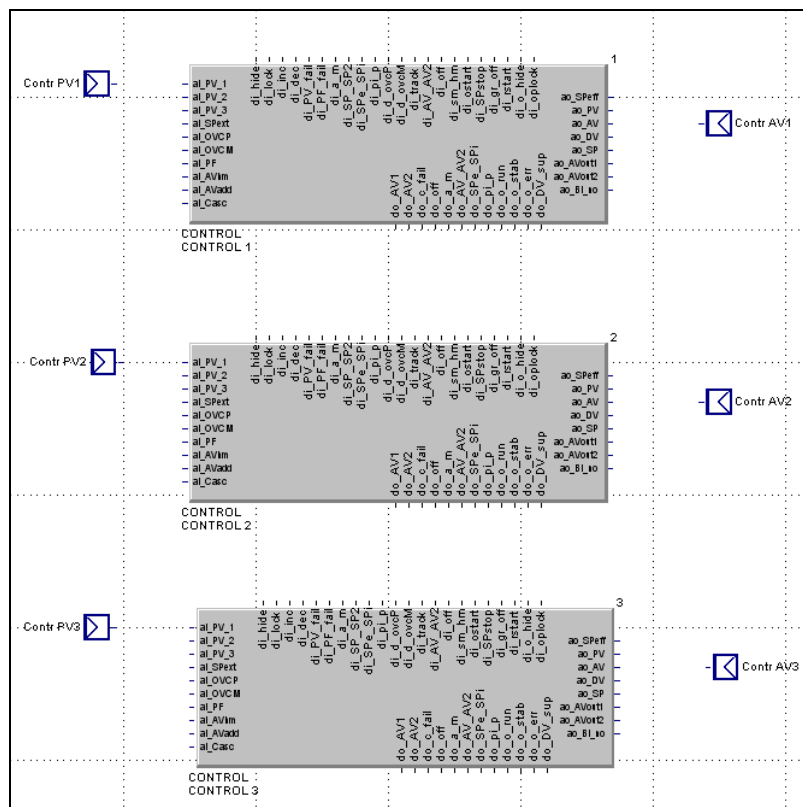


Fig. 249: Practical project: Create inputs and outputs



NOTE!

The default data type is "Bit/Bytes". For the inputs and outputs, however, we need the data type "Float". Always make sure to select the correct data type. Otherwise it will not be possible to connect the input with the block, for example.

8. **Connect inputs and outputs:** Now you have to connect the inputs and outputs with the controllers.

- To do this, position the mouse on the connecting element of the input. Now the mouse pointer appears as a pencil (see below).
- Click with the left mouse button. Now the mouse pointer appears as a cross.
- Now move the mouse from the input block to the input "ai_PV_1" of the controller. Then click on the target point of the connection, in order to create the line.

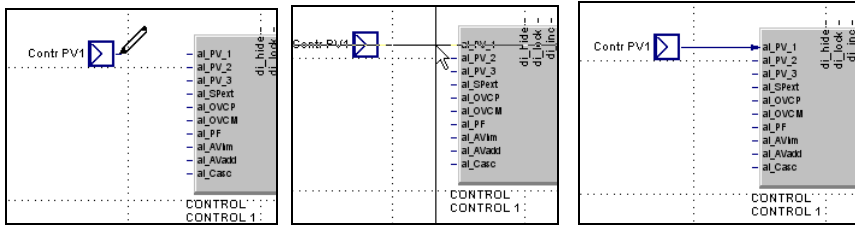


Fig. 250: Practical project: Draw connecting line.

- Now connect the remaining inputs with a controller.
- Then connect one controller output "ao_AVout1" with a block of the type "Output". The result should look like this:

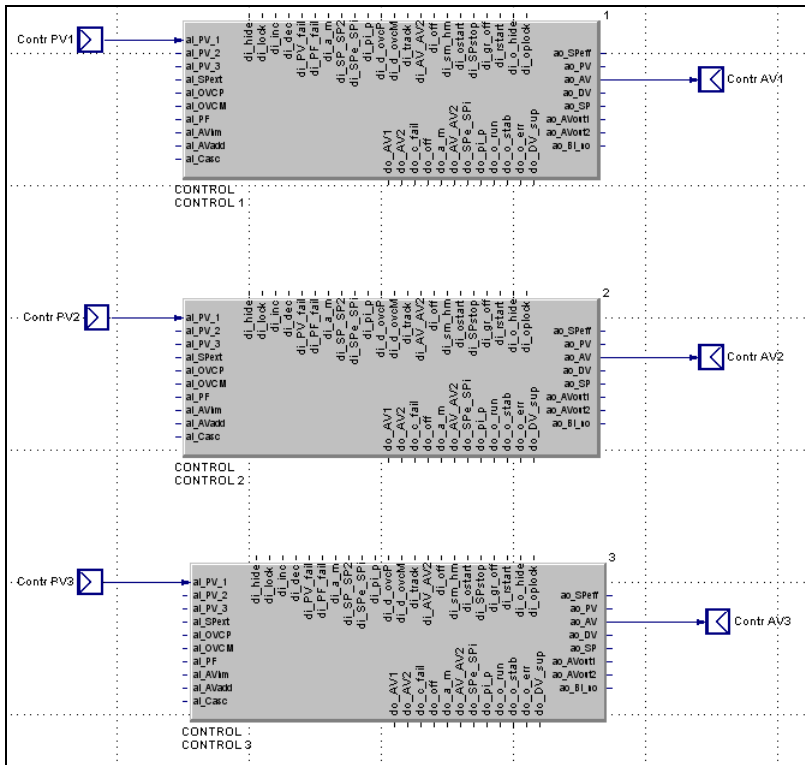


Fig. 251: Practical project: Connection of inputs and outputs

9. **Save changes:** Save your changes. To do this, click on the button "Save project".

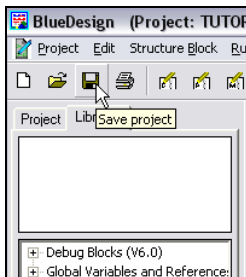


Fig. 252: Practical project: Save project

II-5.3 Step 3: Creating a simulation

As previously mentioned above: The behavior of the oven should be simulated as part of the sample program. The behavior of the oven is simulated with filters and adders from the PMA library. The filter has the task of forwarding the input value with a delay (yet to be defined) as output value. We will specify the delay by means of parameters (further information on this can be found in section "III-24.7 Step 7: Determining parameters").



NOTE!

More extensive information about these components is available in the chapter "Function library".

1. **Select program block PROCESS:** Select the program "PROCESS" with a *double click* in the "Project" tab.

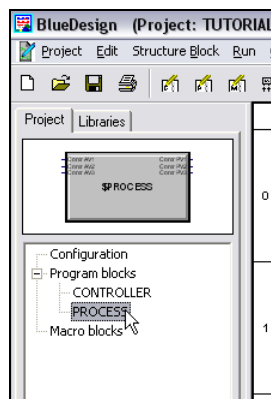


Fig. 253: Practical project: Select program block "PROCESS"

2. **Add filter:** First the behavior of the first oven chamber will be simulated. We will use two filters switched in series to simulate an oven chamber. The task of the filter is to smooth the step-wise changes used as the input (which can be very erratic) so that the output value of the filters approximates the behavior of an oven chamber. A graphic for purposes of illustration (the upper curve shows the output value, the middle curve the value after the first filter, the bottom curve the end value (after the second filter)):

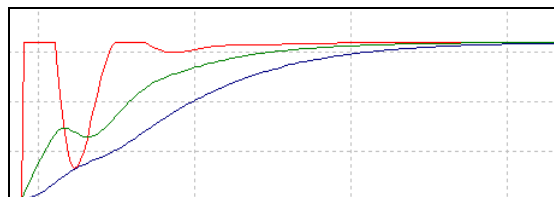


Fig. 254: Practical object: Graphs of filter use

Proceed as follows to add the filters:

- Select the "Libraries" tab and open the library "LIP 0008 Timing functions".
- Click the "Filter" block and drag (as described above) two filter blocks to your worksheet.
- Give the first filter the name "Area 11" and the second filter the name "Area 12". You can give the filter a name by clicking the filter and selecting the command "Parameter Dialog ..." in the context menu. Then enter the name in the "Title" area of this dialog.
- Connect the two filters as shown in the graphic below.

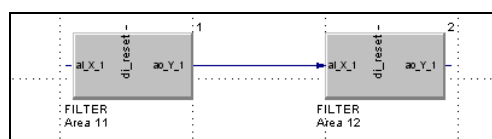


Fig. 255: Practical project: Two filters switched in series

3. **Insert adder:** An adder must be inserted in the simulation for the following reasons: First it will be used later to simulate the heat transfer from one chamber to the adjacent chamber. Second it is required to convert the actual controller output supplied by the controller into a temperature specification. The control variable is specified in percent (values between 0 and 100 are possible), the oven temperature is specified in degrees (here values between 0 and 400 are possible). To depict this situation the control variable is now multiple with the factor 4 in the adder (the adder can also multiply). You are justified in wondering where the adder obtained this information, in other words how it knows that the input value must be multiplied with the factor 4. These settings will be applied (later) as parameters (see section "Step 4: Specify parameters").

Add and an adder to the program and connect it, then name it as shown in the following graphic.

Note: the adder can be found in library "LIB001 Scaling and calculating"; the block has the name "ADD".

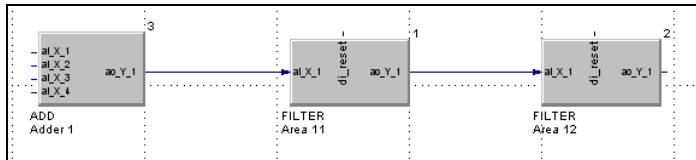


Fig. 256: Practical project: Adder and filter

4. **Prepare for connection to controller:** The control variable from the appropriate controller (in this case from the controller "CONTROL 1") should be used as the input value for the simulation. For transfer of the data, a block of the type "INPUT" is used again.

Therefore, select a block of the type "INPUT" from the "STANDARD" library. Give the block the name "Contr AV1" and select "FLOAT" as the data type.

Connect the block with the input "ai_X_1" of the adder.

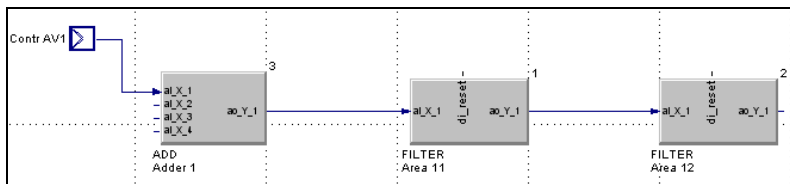


Fig. 257: Practical project: Program with input

5. **Create simulation for chambers 3 and 4:** Now create the simulation for chambers 2 and 3 of the oven in the same manner. The result should look something like the following graphic:



NOTE!

You can speed up your work by copying and inserting the blocks you have already used. You can copy the blocks by placing the mouse pointer next to the input at the top left and highlighting the blocks while holding down the left mouse button. Then you can copy them using the standard Windows commands.

After copying the blocks, they have to be renamed.

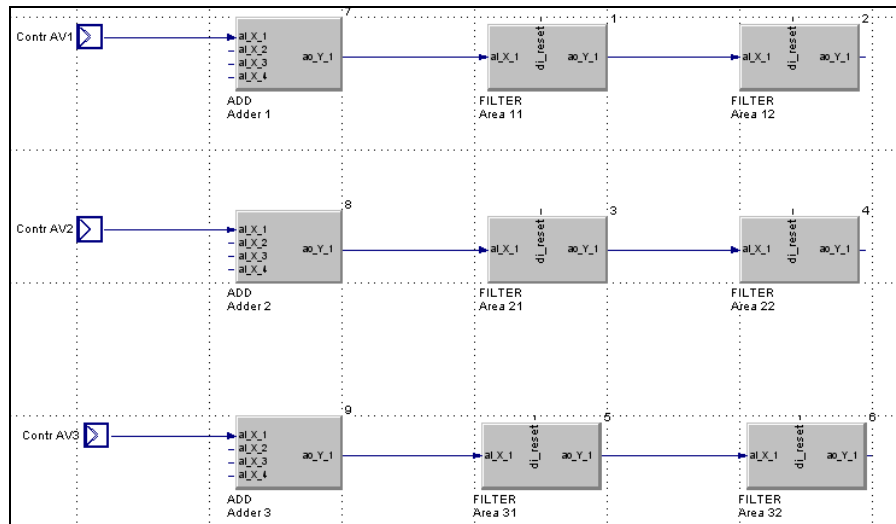


Fig. 258: Practical project: Simulation for three oven chambers

6. **Implement the simulation of the heat transfer:** As previously mentioned above: A feedback will be simulated between the oven chambers. Therefore, warming in one chamber should also cause warming of the adjacent chambers.

We will simulate this by connecting the output of one chamber simulation (for example "Area_12") with the input of the neighboring chamber simulation (for example "Adder_2").

Naturally a complete heat transfer should not take place from one chamber to the next. To keep things simple in the sample we have assumed that a heat transfer of 20% takes place. Consequently the input value is multiplied in the adder with the factor 0.2. These configurations are made by means of parameters. However, more information will be provided later in this regard.

Moreover the adder also has the task of doing what the name implies: the two values are added so that the feedback from the adjacent oven chamber, as well as the control variable requested by the controller are taken into account in the simulation.

To also simulate the time-delayed behavior for the feedback, an additional filter is switched between the output of the one chamber simulation and the input of the adder.

As a result supplement the program as follows:

- Supplement the project four filters and give the filters names corresponding to the following graphic.
- Supplement each oven chamber simulation with one output ("Contr PV 1", "Contr PV 2" and "Contr PV 3") of the type "Float".
- Connect the new filters with the adders.
- Connect the outputs with the oven chamber simulations.

The program should now have the following appearance (new connections are highlighted in yellow):

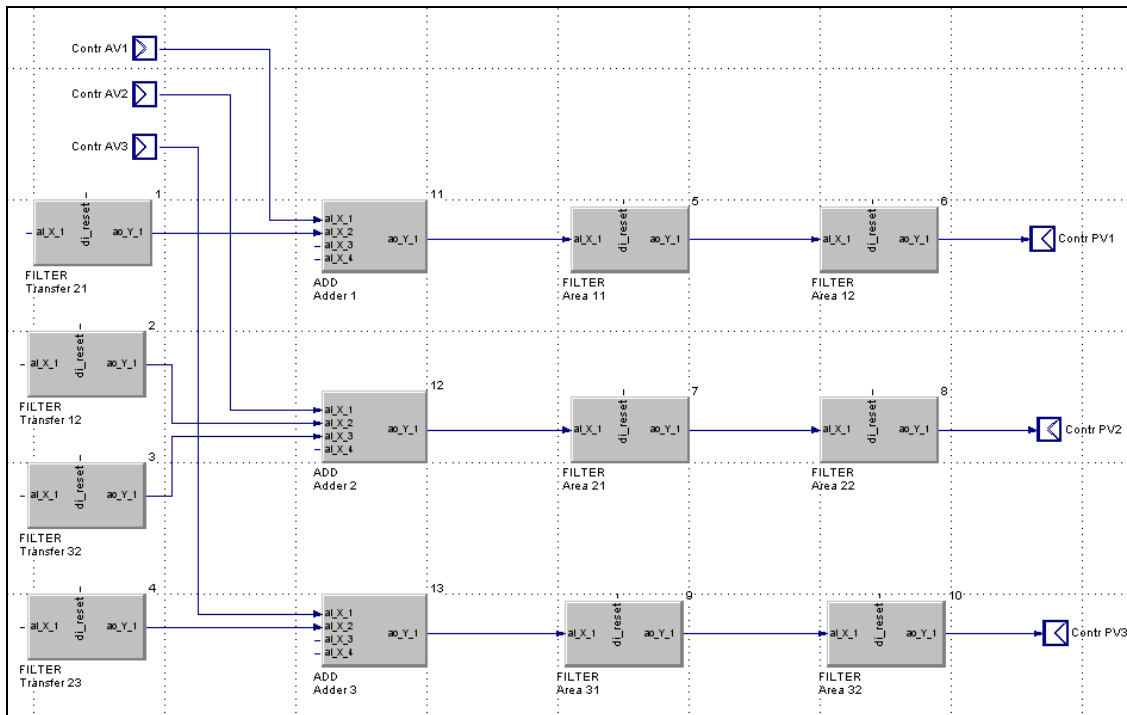


Fig. 259: Practical project: Oven simulation



NOTE!

If the error message "Error! Invalid connection" appears when attempting to establish a connection between two blocks, then you probably selected an incorrect data type for an output. In this case, delete the respective output and create it again. You can delete a block by placing the mouse cursor on the block and selecting the "Delete" command in the context menu.

To simulate the heat transfer from one oven chamber to the next, the outputs of the chamber simulations now have to be connected with the simulation start (the adders) of the adjacent oven chambers.

- Therefore, add the connections marked in color corresponding to the following graphic. The connections highlighted in yellow simulate the heat transfer from the outer chambers to the middle chamber. The connections highlighted in green simulate the heat transfer from the middle chamber to the outer chambers.

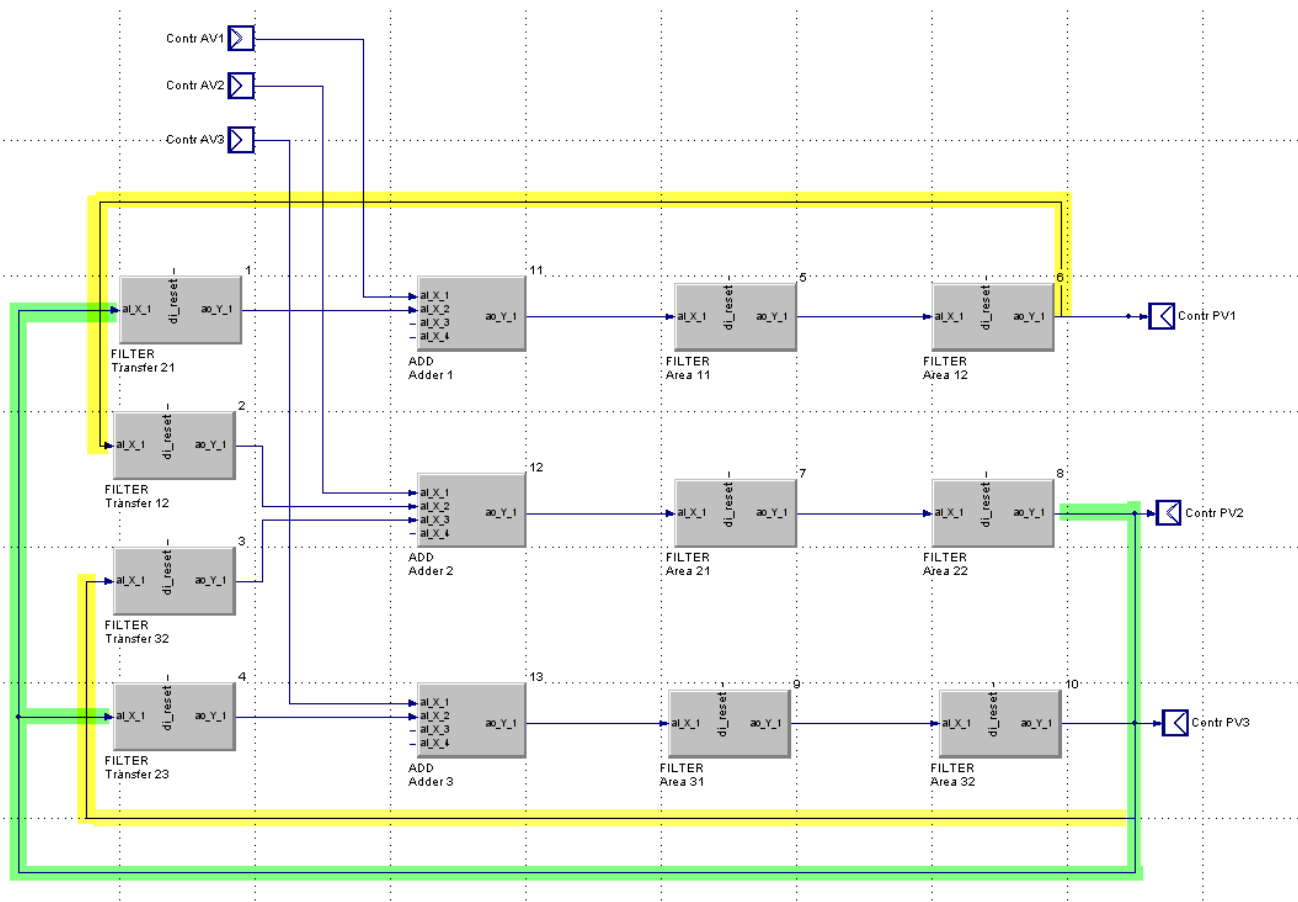


Fig. 260: Practical project: Oven simulation 2

- Save changes:** Save your changes. To do this, click on the button "Save project".

Examples

Two simple examples illustrate the functional principle of the simulation.

Moreover: For the sake of simplicity in the "oven simulation" example we will assume that the oven's ambient temperature is 0 °C. If an oven is not heated (control variable = 0), then after a short time it has an actual temperature of 0 °C.

- Example 1:** For the first example, let's assume that chamber 2 should reach a temperature of 400 °C, chamber 1 on the other hand should reach a temperature of 0 °C (the control variable thus is 0%). The heat feedback for chamber 1 is calculated as follows:

$$(0 * 4) + (400 * 0,2) = 80$$
- Example 2:** In the second example, chamber 2 should likewise reach a temperature of 400 °C, chamber 1 on the other hand should reach a temperature of 200 °C (the control variable thus is 50%). The formula for heat feedback for the chamber is:

$$(50 * 4) + (400 * 0,2) = 280$$

Since the setpoint for chamber 1 is 200 °C and a regulator (and not a controller) is used here, the regulator will reduce the control variable for this chamber accordingly so that a temperature of 200 °C is achieved.

II-5.4 Step 4: Specifying interfaces, connecting blocks

We have thus far created the program blocks "CONTROLLER" and "PROCESS". Both program blocks have three input and three output interfaces. However, these inputs and output cannot yet be accessed from "outside". In addition, the program blocks have not yet been started in our project.

Therefore, the following steps are now necessary:

1. **Create main program:** The "Configuration" area is the uppermost level of a project. All program blocks must be able to be found here.
 - Double click the "Configuration" node on the "Project" tab in order to view the "Configuration" workspace.
 - Click the "CONTROLLER" node.
 - Drag the program block "CONTROLLER" to the configuration area.
 - Then repeat these steps for the program block "PROCESS".

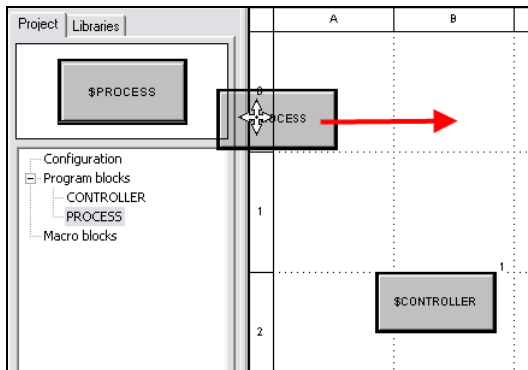


Fig. 261: Practical project: Create "Configuration" area



NOTE!

Program blocks are executed only if they can be found in the "Configuration" area. A configuration without program blocks makes no sense - the result is a program with no functions (i.e. with no "tasks").

Both program blocks have no inputs and outputs (no interfaces). The interfaces must first be released. To do this, proceed as follows:

2. **Select program block "PROCESS":** Select the program block "PROCESS" in the tree view by double clicking it.
3. **Execute context menu command Design:** Start the "Design" command from the context menu.

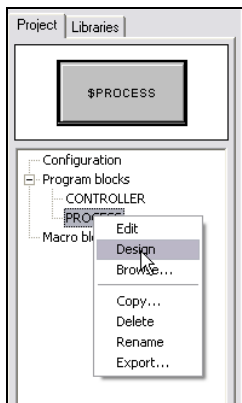


Fig. 262: Practical project: Select command "Design"

You now see a worksheet on the right side of the screen for release of the inputs and outputs.

4. **Allocate inputs:** First the inputs should be allocated. Position the mouse pointer over the input "Contr AV1" in the "Inputs" column. The mouse pointer now appears as a hand (see following graphic). Now press the left mouse button. Hold down the mouse button and drag the input to the left edge of the program block symbol. Then release the left mouse button.

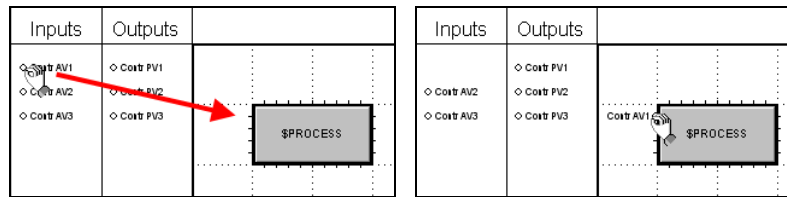


Fig. 263: Practical project: Allocate inputs

Then place the outputs on the right side of the dialog in the same manner. The result should look like this:

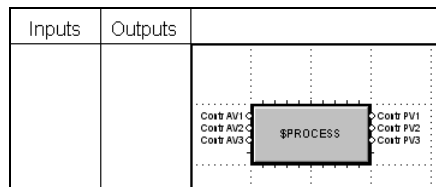


Fig. 264: Practical project: Inputs allocated

5. **Customize labeling:** By default, the names of the inputs and outputs are displayed outside of the program block symbol. However, a better overview is achieved if they are displayed inside the block. To do this, click the button "Connection names inside or outside".

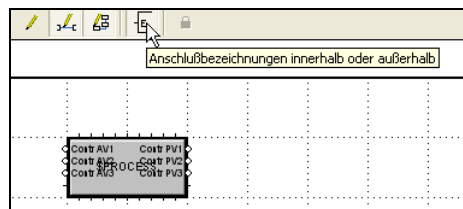


Fig. 265: Practical project: Customize layout

6. **Customize block size:** The appearance of the program block is obviously not acceptable. Therefore, position the mouse pointer on the lower right edge of the block. The mouse pointer is now displayed as a double arrow. Hold the left mouse button down and enlarge the symbol (see below).

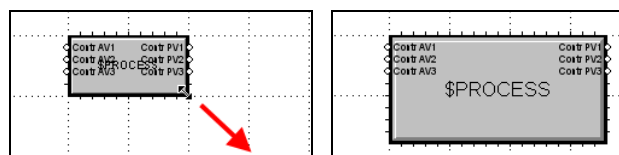


Fig. 266: Practical project: Customize block size

7. **Configure program block "CONTROLLER":** Proceed in the same manner for the program block "CONTROLLER". The result should look like this:

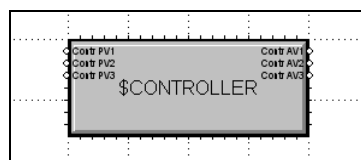


Fig. 267: Practical project: Configure program block "CONTROLLER"

8. **Connect program blocks:** Thus far the program blocks are next to each other, but unconnected. The controllers know nothing of the simulation (and vice versa). In the following step, therefore, the program blocks will be connected.

The outputs of the controllers have to be connected to the input of the respective simulation and the outputs of the simulation have to be connected with the input of the controller.

To do this, proceed as follows:

- Double click in the "Project" tab on the "Configuration" entry in the tree view.
- Connect the two program blocks corresponding to the following graphic.

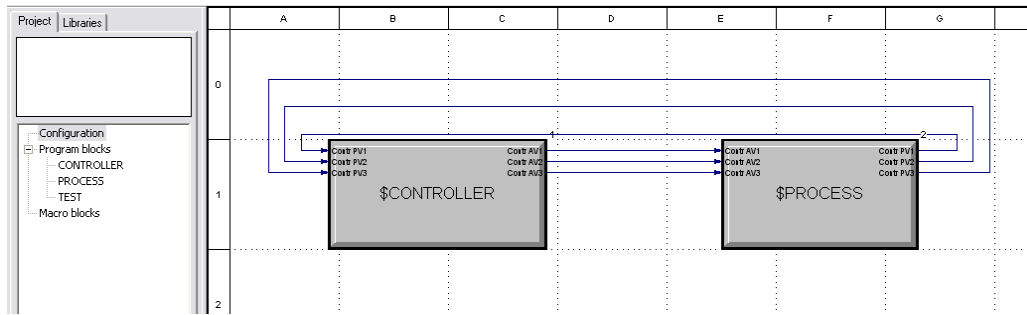


Fig. 268: Practical project: Connect program blocks

9. **Save changes:** Save your changes. To do this, click on the button "Save project".

II-5.5 Step 5: Defining parameters

The components of the PMA library used for the example now have to be configured. Configuration of the components is carried out by the use of parameters. The parameters define the behavior and the properties of the blocks.

Digression: Parameter types

In *BlueDesign* there are two basic possibilities for working with parameters. One possibility has already been explained - the names of blocks or the file type of inputs and outputs were defined by the use of parameters. For this purpose, we worked in the "edit mode" of *BlueDesign*. In this mode, program blocks, macro blocks and connections are edited; parameters can also be entered here. However, this involves one problem: program blocks can be used more than once in a project. If, for example, two ovens should be controlled separately in a project, this could be done as follows:

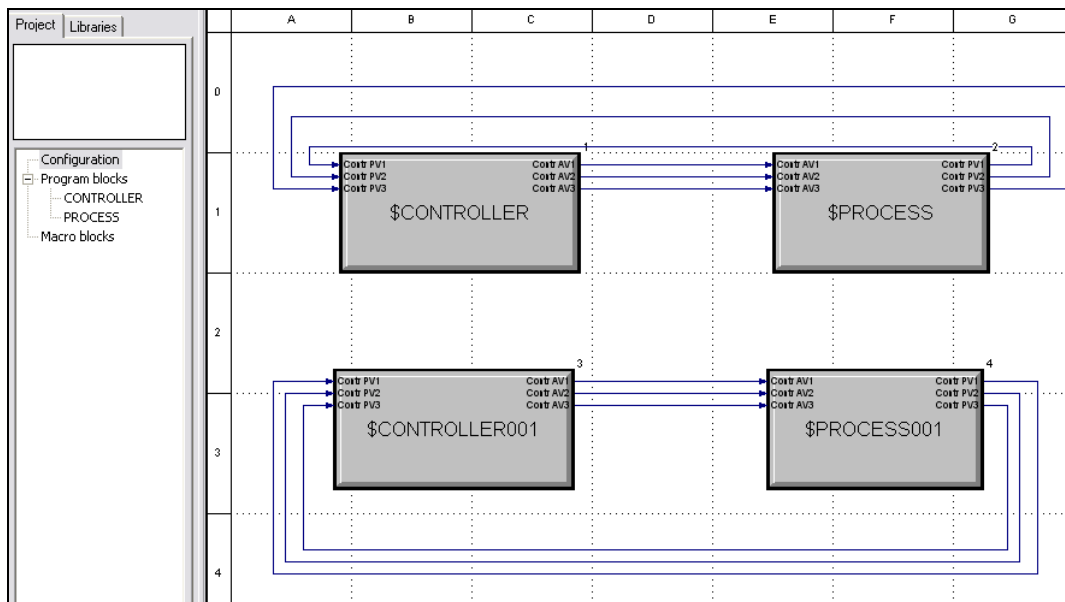


Fig. 269: Practical example: double use of program blocks

The program blocks "CONTROLLER" and "PROCESS" are used double here. Now the maximum oven temperature and many characteristics of the simulation, etc. are defined by means of parameters. If we were to define the parameters in edit mode, each copy of a program block type would have the same parameters. This is obviously not desirable. After all, our two ovens could have different properties (e.g. maximum temperature) or could display different behavior.

BlueDesign offers a simple solution to this problem: Parameters can be edited not only in edit mode, but also in "run mode". In this mode, individual parameters can be defined for each copy of a block or program block. A note on terminology: The copy template is also referred to a "class" and the copy as an "instance".

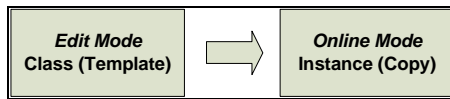


Fig. 270: Edit mode and online mode

Parameters entered in online mode refer only to the respective instance - *never* to the class ("copy template"). This can be compared with the following situation: If you copy a page from a book, any marks you make on the copy are not in the book itself.

In summary:

- **Edit mode:** Parameters entered in edit mode are applied in all instances ("copies") of a block. These parameters can be overwritten in run mode.
Use: They should only be used for defining general properties, e.g. file types.
- **Run mode:** Parameters entered in run mode apply only to the respective instance ("copy") of a block. In the respective instance ("copy") they overwrite the parameters entered in edit mode.
Use: Definition of the (actual) parameters for a block.



NOTE!

The following basic rule applies: enter parameters in run mode! Use the edit mode only if you are certain that a parameter should be used for all instances of a block.

Enter parameters

Proceed as follows in order to define parameters:

1. **Start run mode:** Start run mode with the menu command "Run/Enter".

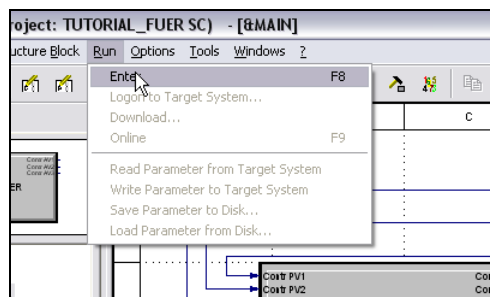


Fig. 271: Practical example: Start run mode

An overview of the instances of the project is displayed in the tree structure at the left. Click the entry "\$Controller".

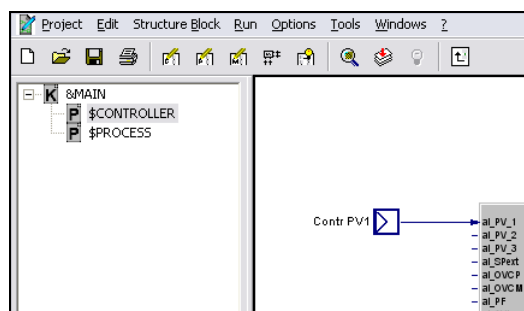


Fig. 272: Practical example: run mode

2. **Configure controller "CONTROL 1":** Now position the mouse pointer on the controller "CONTROL 1". Start the "Parameter dialog ..." command in the context menu.
The maximum temperature of an oven chamber should be 400 °C. Consequently assign the value "400"

to the parameter "SPHi".
Then click the "OK" button to save your input.



NOTE!

Further information on the meaning of the parameters can be found in the chapter "Function block reference".

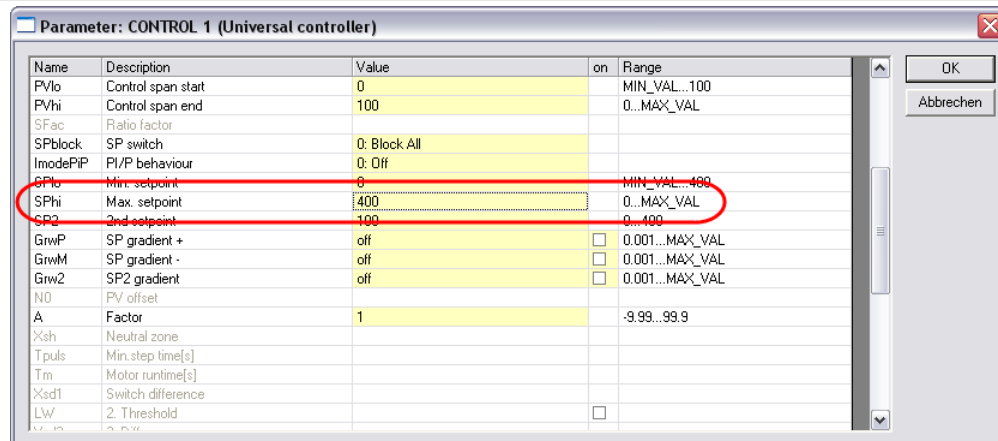


Fig. 273: Practical example: Configure the setpoint for controllers

3. **Configure the remaining controllers:** Assign the value "400" to the parameter "SPHi" for the controllers "CONTROL 2" and "CONTROL 3" as well.
4. **Switch to the program block "PROCESS":** Now switch to the program block "\$PROCESS" by clicking the program block name in the tree view.
5. **Configure filters:** 10 filters are used to simulate oven behavior. To (halfway) simulate realistic oven behavior, the input value of the filter will be transferred to the output with a delay of 10 seconds. Therefore, assign the value 10 to time constant "T" every 10 filters.

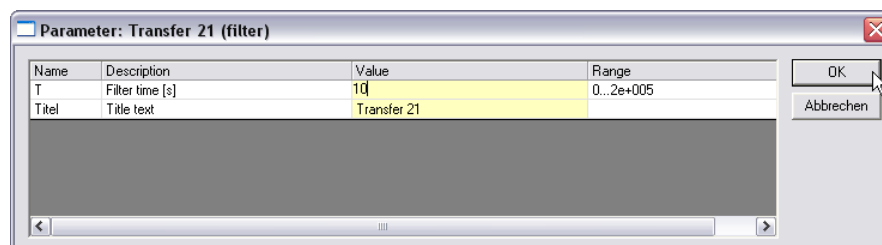


Fig. 274: Practical example: Configure time constant for filters

6. **"Configure "Adder 1" and "Adder 3":** As explained above the adders are required for two reasons: First it will be used to simulate heat transfer from one chamber to the adjacent chamber. Second they are required to convert the actual control variable supplied by the controller into a temperature specification. The control variable is specified in percent; the oven temperature is specified in degrees Celsius (maximum 400). To depict this situation the control variable is now multiplied by the factor 4 in the adder.

Enter the following parameters for the input "X1" and "X2".

- A/multiplication factor for x1: "4"
- B/ multiplication factor for x2: "0.2"

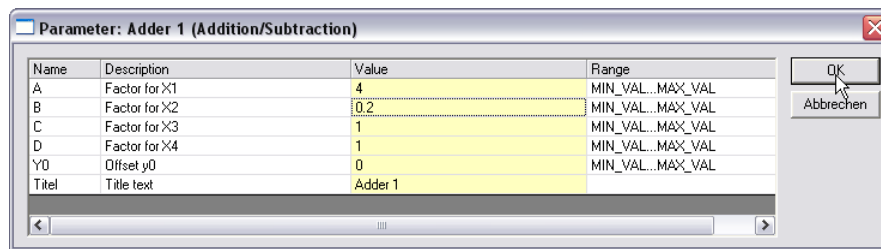


Fig. 275: Sample project: Configure adder 1 and 3

7. **Configure "adder 2":** "Adder 2 belongs to the middle oven chamber. For this adder - as opposed to the two other adders - three inputs are used (since heat transfer should be simulated from the right and left over chambers).

Enter the following parameters for the inputs:

- A/multiplication factor for x1: "4"
- B/ multiplication factor for x2: "0.2"
- C/multiplication factor for x3: "0.2"

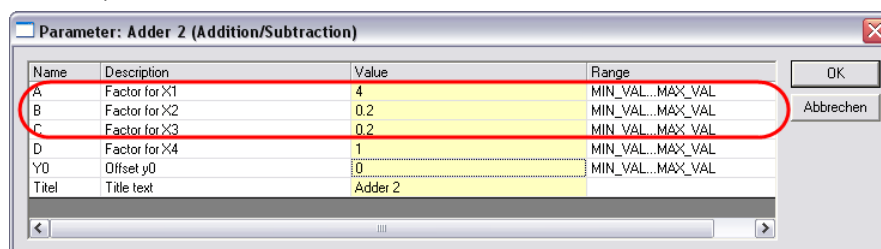


Fig. 276: Sample project: Configure adder 2

8. **Save changes:** Save your changes. To do this, click on the button "Save project".

II-5.6 Step 6: Creating a visualization

In the previous steps we created a controller and simulation of the oven. For the user however the application is somewhat unsatisfactory - convenient operating pages are missing.

As mentioned at the beginning the main operating page of the application should look like this:

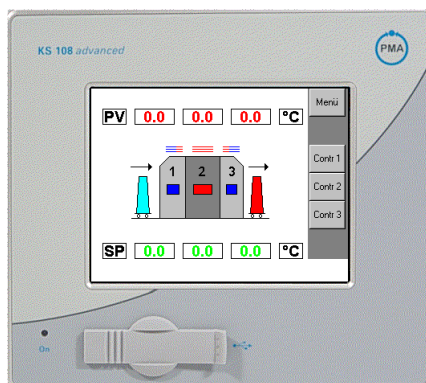


Fig. 277: Sample project: Design of main operating page

The buttons on the right should be used to open the main menu ("Menu") and the operating pages of the controller. The operating pages of the controllers are automatically provided by the PMA library and can be used "out of the box".

This is not so with the main operating page. Since it is application-specific, it must be created by you. BlueDesign provides an editor for this purpose. You can position the elements of the user interface by means of "drag & drop" on the user interface. Once you have created the user interface, you must connect the user interface with your application in a second step. To do this, use the interface blocks from the library "LIB401 HMI".

The user interface will be referred to in the future as **HMI** ("Human-Machine-Interface").

Design the user interface

Start Mask Designer

1. **Start edit mode:** Make sure that you are in edit mode.
Although the mask editor can also be started in run mode, the interface blocks can be created in that mode.
2. **Start Mask Designer:** Use the menu command "Tools/Mask Designer ..." to start the mask designer.

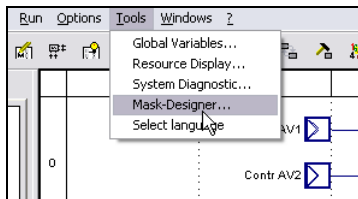


Fig. 278: Sample project: Start Mask-Designer

Now you see the Mask Designer.

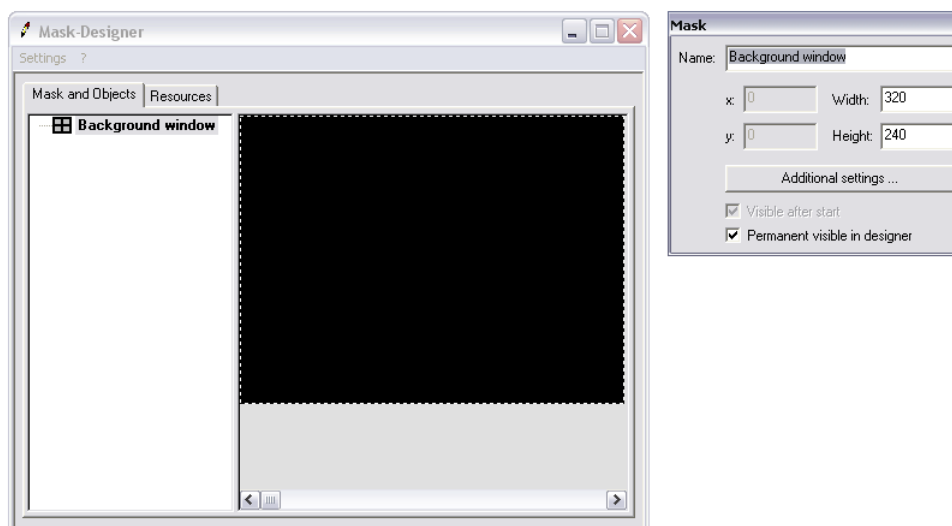


Fig. 279: Sample project: Mask-Designer

Edit background window

When you create a new mask, a "background window" is automatically generated. The background window is the lowest level of your HMI. A background window cannot be deleted; it can only be adapted.

By default the background window is black (see above). For our application, however, it should be white.

1. **Start the "Additional settings" dialog:** Click on the button "Additional settings" to start the "Additional settings" dialog.
2. **Start the "Background color" dialog:** Click the "Background color" button.
3. **Select color:** Select the color "white" in the "Background color" dialog (see following graphic).



NOTE!

The KS 108 easy can display a maximum of 256 colors. Therefore, select a color from the "Basic colors" area. Do not use "User-defined colors".

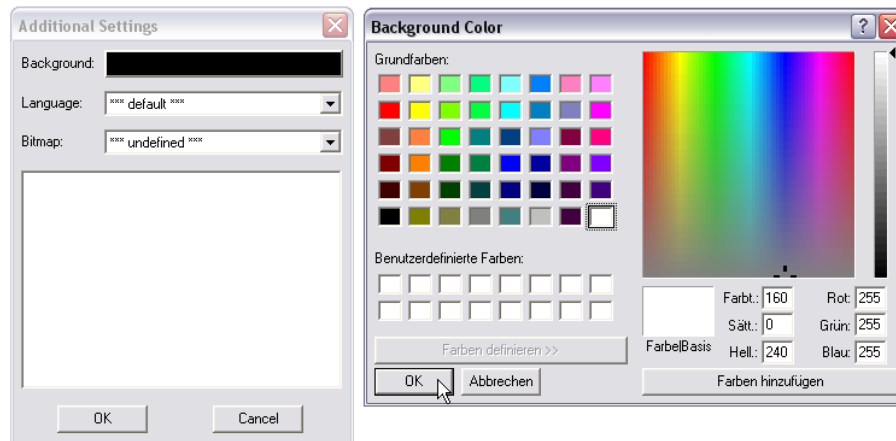


Fig. 280: Sample project: Editing background window

4. **Save input:** Then click the "OK" button to save your input.

Import background picture

In the Mask Designer only images can be inserted that were previously added to the "Resources" of the HMI. In the "Resources" area, images and extended texts are stored that are used in the HMI.

1. **Select the "Resources" tab:** Click the "Resources" tab to select the resources view.
2. **Select "Bitmaps" entry:** Select the "Bitmaps" entry in the tree structure (left).
3. **Add new resource:** Start the context menu command "New" at the right of the dialog (see following graphic).

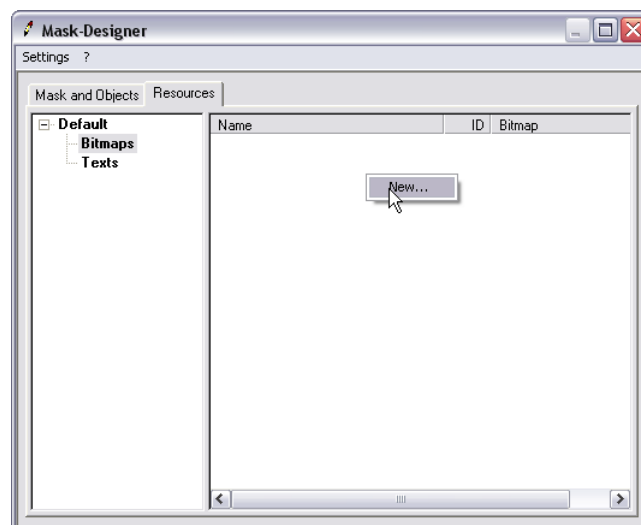


Fig. 281: Practical example: Select the "Resources" tab

Now you see the "New Bitmap Resource" dialog.

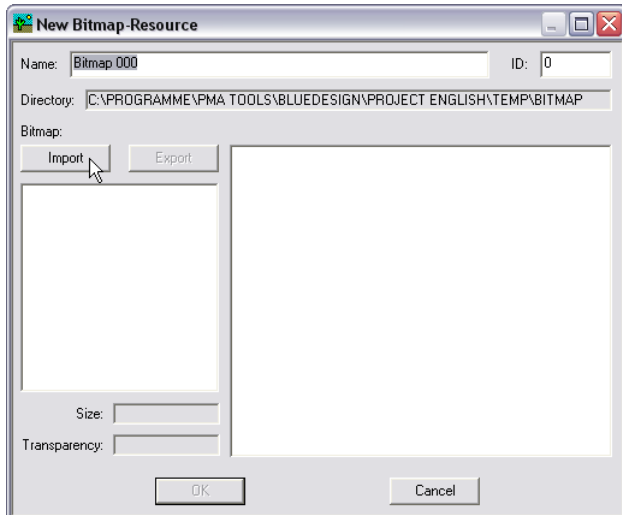


Fig. 282: Practical example: Start "Import Bitmap" dialog

- 4 **Start "Import bitmap" dialog:** Click the "Import" button to start the "Import Bitmap" dialog.

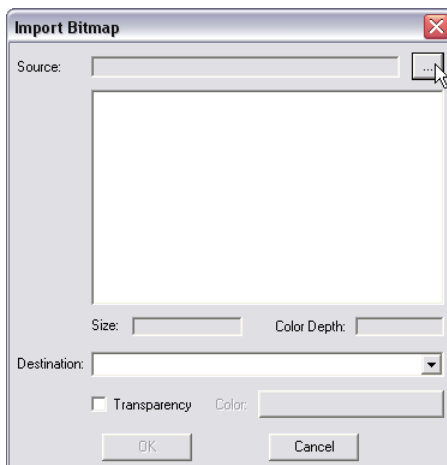


Fig. 283: Practical example: Start "Import Bitmap" dialog

- 5 **Start "Import File" dialog:** Click the (...) button to start the "Select Import File" dialog.
- 6 **Select file:** Select the bitmap "oven.bmp" in the "Bitmap" directory. Then click "Open" to apply your selection.

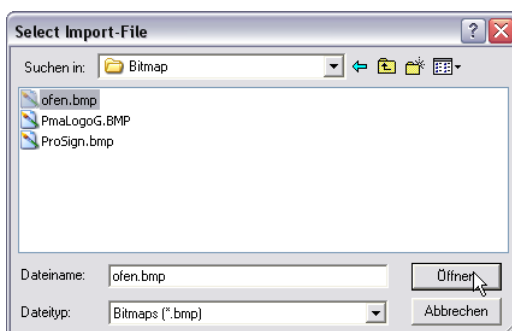


Fig. 284: Practical example: Select import file

- 7 **Save input:** Click the "OK" button in the "Import Bitmap" dialog to apply your selection.
- 8 **Give the bitmap a name:** Give the bitmap the name "Oven" (see following graphic).

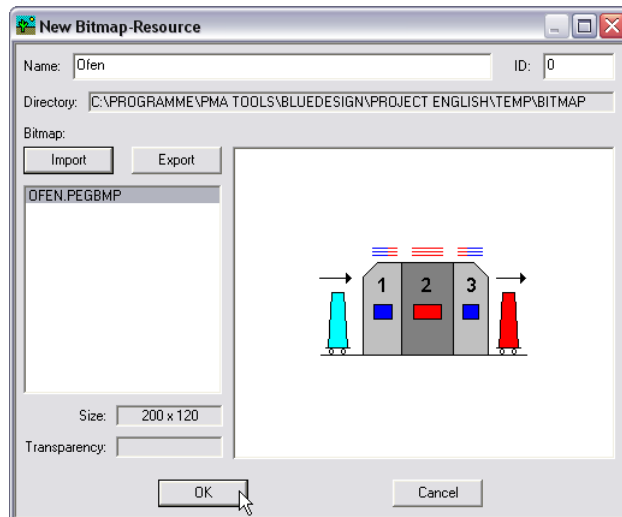


Fig. 285: Practical example: Edit bitmap resources, assign name

9. **Save input:** Click the "OK" button to save your input. You now see a new entry in the resource overview.

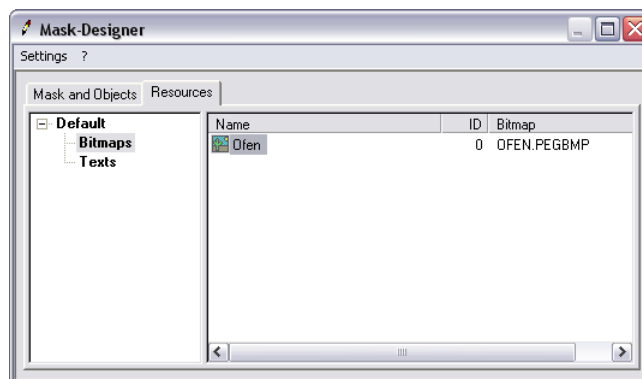


Fig. 286: Practical example: New bitmap inserted

Insert background picture

After you have added the image to the resources, you can insert it into your HMI:

1. **Select the "Masks and Objects" tab:** Click on the "Masks and Objects" tab.
2. **Add new object:** Click the entry "Background window". Then Start the context menu command "New Object" with the right mouse button (see following graphic):

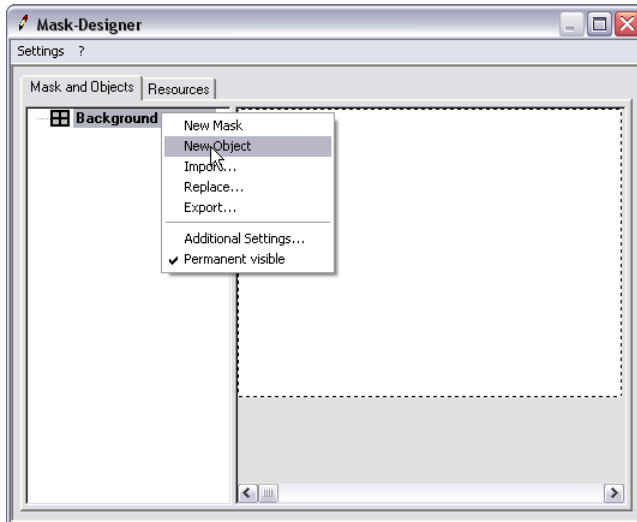


Fig. 287: Practical project: Context menu "New Object"

The new object has been placed on the workspace:

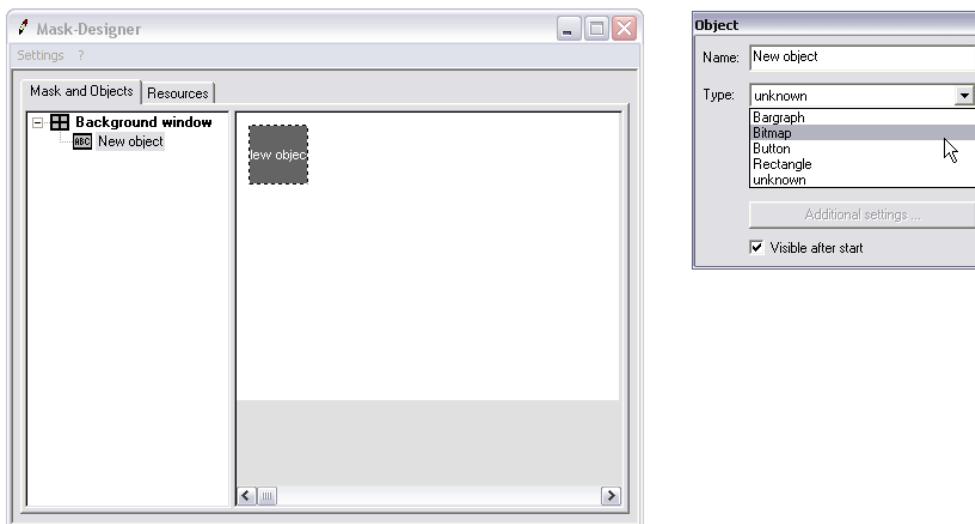


Fig. 288: Practical project: Define object type

3. **Define object type:** Make sure that the entry "New Object" is selected in the tree structure. There is a second dialog next to the Mask Designer. The properties of the respective object are displayed here. After creating a new object, it has the type "unknown". Therefore, the type of object must first be specified. Click the combination field and select the type "Bitmap" (see graphic above).
4. **Open the "Settings" dialog:** Click the "Additional Settings ..." button to open the "Settings" dialog.

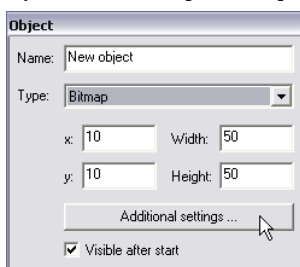


Fig. 289: Practical project: Start "Additional Settings" button

5. **Assign bitmap:** Now you have to assign the bitmap to be displayed in the newly created object. To do this, click within the "Value" column and select the bitmap "Ofen". Note: The name you gave the bitmap will appear here.

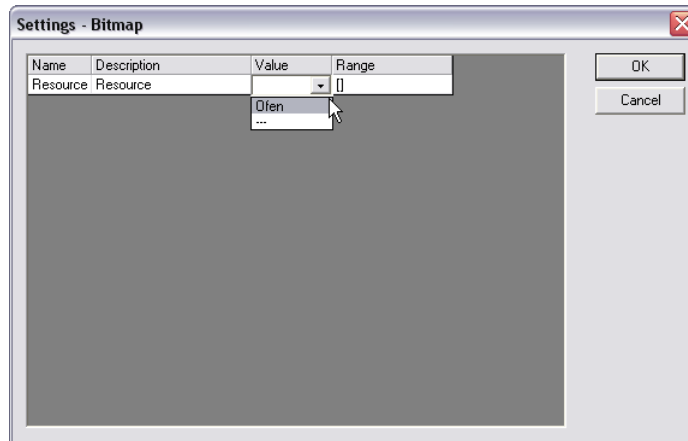


Fig. 290: Practical project: "Settings" dialog

6. **Save input:** Click the "OK" button to apply your input. The bitmap is displayed in the Mask Designer.

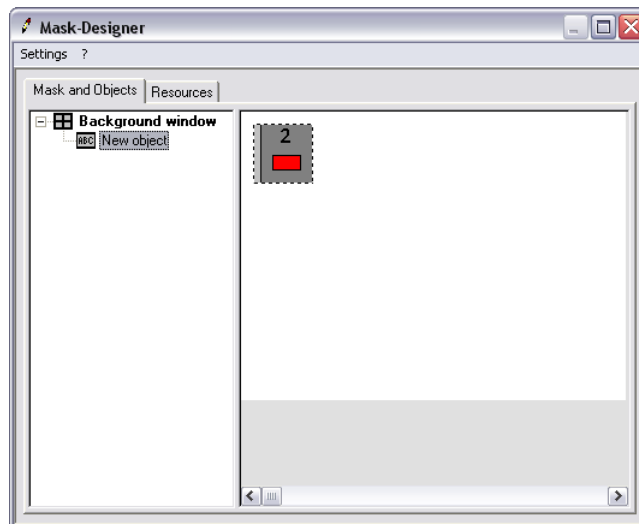


Fig. 291: Practical example: New bitmap

7. **Adapt bitmap:** The bitmap is displayed, however only part of it and also at the wrong location. The object has to be adapted. Therefore, enter the following values in the properties dialog:

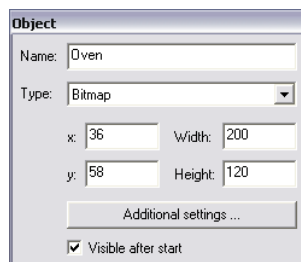


Fig. 292: Practical example: Adapt object properties

The result should look like this:

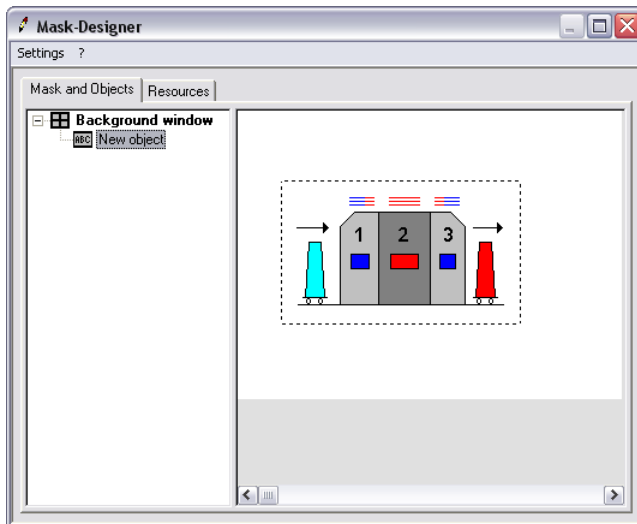


Fig. 293: Practical example: Oven image inserted

Insert menu bar

On the right side of the user interface, a menu bar with four buttons should be created. The background should be gray. The background should be created with an object of the type "Rectangle".

Proceed as follows in order to create the background for the menu bar:

1. **Add new object:** Click the entry "Background window". Then start the context menu command "New Object" with the right mouse button.
2. **Add properties:** Click the object in the tree view and assign it the following properties in the properties dialog (you can start the "Settings" dialog by clicking the button "Additional Settings ..."):

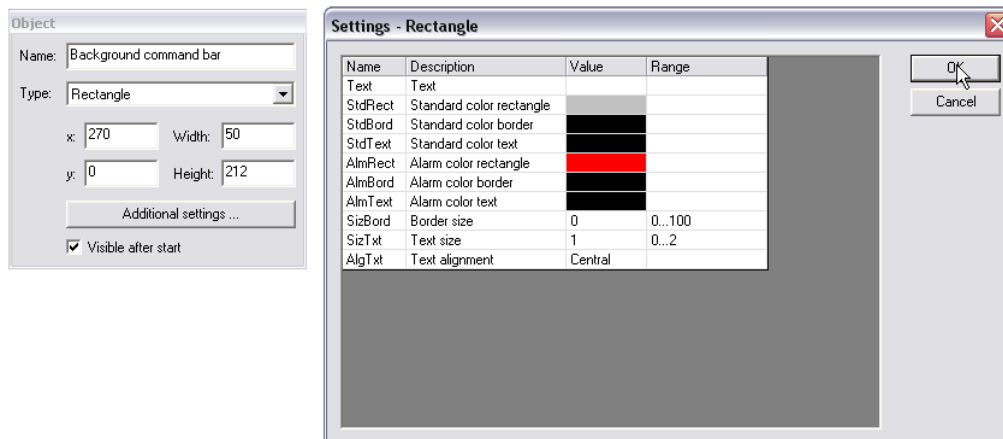


Fig. 294: Sample project: Create menu bar background

Note: In the dialog "Settings - Rectangle" you only have to delete the "Value" entry of the "Text" line and assign the color "dark gray" to the "Standard color rectangle". The result should look like this:

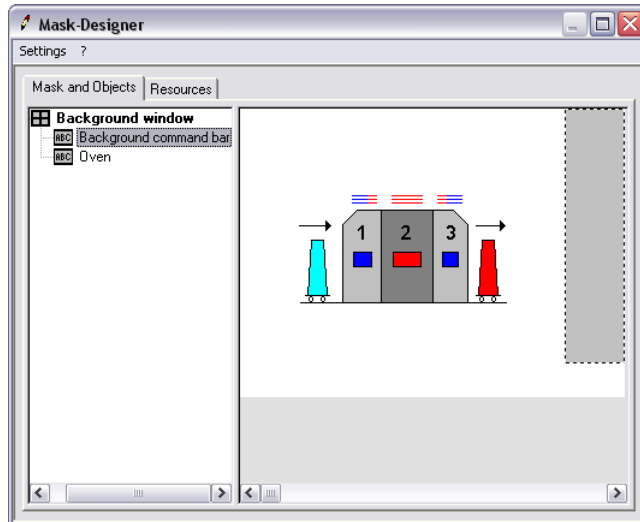


Fig. 295: Sample project: Menu bar created

Create buttons

On the menu bar, 4 buttons should now be inserted, starting with the "Menu" button. To do this proceed as follows:

1. **Create object:** Create a new object as described above.
2. **Specify properties:** Specify the properties of the first button corresponding to the following graphic (the property "Title text" in the "Title" dialog defines the labeling of the button).

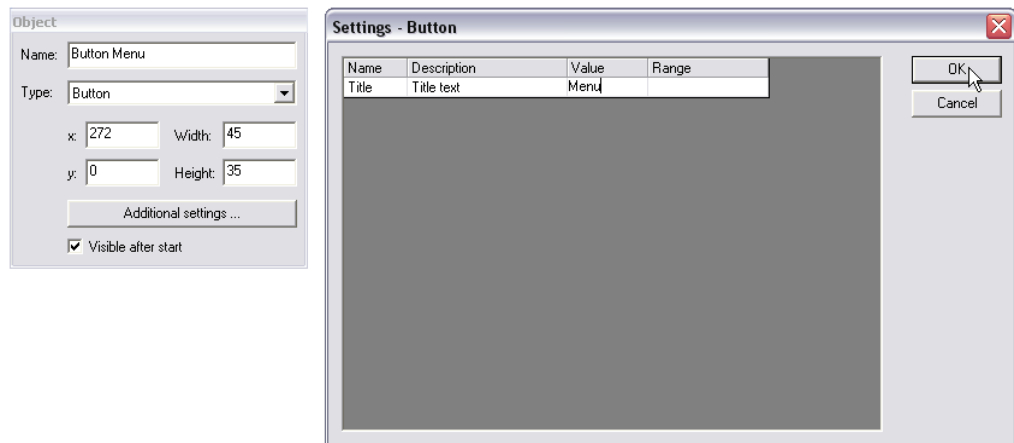


Fig. 296: Sample project: Adapt "Button" object properties

3. **Create remaining buttons:** Create the remaining 3 buttons with the following properties:

| Property | Button Contr. 1 | Button Contr. 2 | Button Contr. 3 |
|-----------------------------------|-----------------|-----------------|-----------------|
| "Object" properties dialog | | | |
| Name | Button Contr. 1 | Button Contr. 2 | Button Contr. 3 |
| Type | Button | Button | Button |
| x | 272 | 272 | 272 |
| y | 70 | 105 | 140 |
| Width | 45 | 45 | 45 |
| Height | 35 | 35 | 35 |
| Visible after start | yes | yes | yes |

"Settings" dialog

| | | | |
|------------|----------|----------|----------|
| Title text | Contr. 1 | Contr. 2 | Contr. 3 |
|------------|----------|----------|----------|

The result should look like this:

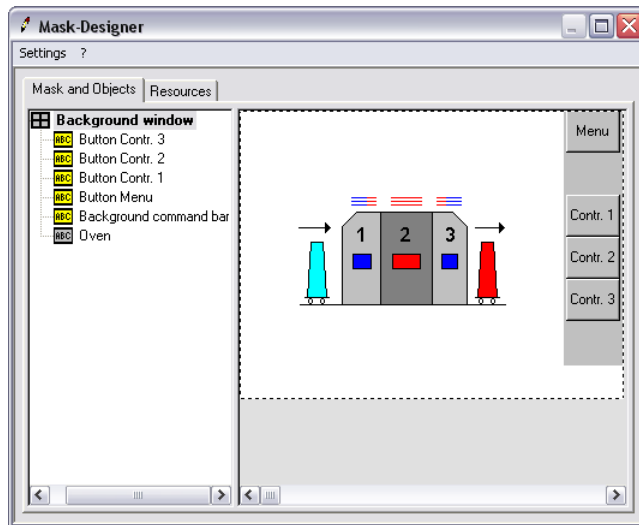


Fig. 297: Sample project: Buttons inserted

Insert actual value display

The actual value of the oven chambers should be displayed by the bitmap "Oven". For this purpose, we need 5 objects of the type "Rectangle".

The three middle rectangles should display the value of the chambers; the left rectangle should receive the character string "PV" and the right rectangle the character string "°C".

1. **Insert new objects:** Click on the tree structure at the left of the dialog and create five new objects.
2. **Assign properties:** Assign the following properties to the new objects:

| Property | Text PV | Value PV1 | Value PV2 | Value PV3 | Text °C (top) |
|-----------------------------------|-----------|-----------|-----------|-----------|---------------|
| Properties dialog "Object" | | | | | |
| Name | Text PV | Value PV1 | Value PV2 | Value PV3 | Text °C (top) |
| Type | Rectangle | Rectangle | Rectangle | Rectangle | Rectangle |
| x | 14 | 54 | 114 | 174 | 234 |
| y | 25 | 25 | 25 | 25 | 25 |
| Width | 30 | 50 | 50 | 50 | 30 |
| Height | 20 | 20 | 20 | 20 | 20 |
| Visible after start | yes | yes | yes | yes | yes |
| "Settings" dialog | | | | | |
| Text | PV | %. 1f | %. 1f | %. 1f | °C |
| Standard color rectangle | white | white | white | white | white |
| Standard color text | black | red | red | red | black |
| Alarm color rectangle | white | white | white | white | white |
| Alarm color text | black | red | red | red | black |
| Border size | 1 | 1 | 1 | 1 | 1 |
| Border color | black | black | black | black | black |

Explanation: In the objects, both constant texts ("PV" and "°C") and the content of variables ("PV1", "PV2", "PV3") should be displayed. Enter the content of the constant texts (as shown in the above table) in the "Text" field. For the variables, you indicate with the command "%. 1f" how the text output should appear. "%" stands for the integer share of the value, and "1f" indicates that one decimal place should be displayed.

The setpoint values should be displayed in the font color "red". This setting is made in the "Standard color text" or "Alarm color text" areas.

The result should look like this:

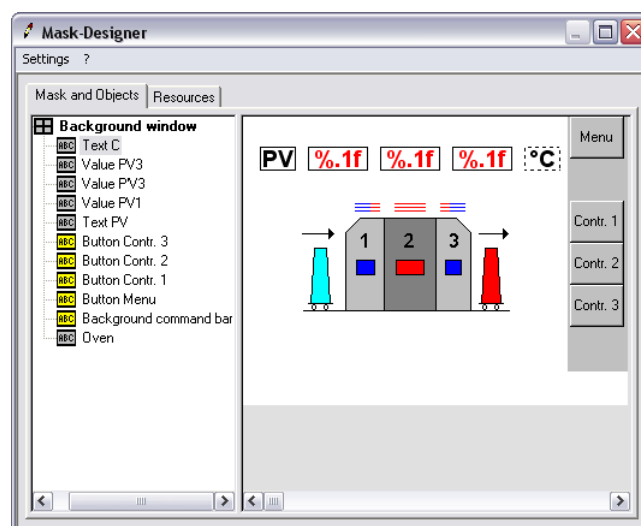


Fig. 298: Sample project: PV display inserted

Insert setpoint value display

Now insert the setpoint value display. Proceed as described in the above section for the actual value display. Assign the following properties to the five objects of the type "Rectangle":

| Property | Text PV | Value SP1 | Value SP2 | Value SP3 | Text °C (bottom) |
|----------------------------|-----------|-------------|-------------|-------------|------------------|
| Properties dialog "Object" | | | | | |
| Name | Text SP | Value SP1 | Value SP2 | Value SP3 | Text °C (bottom) |
| Type | Rectangle | Rectangle | Rectangle | Rectangle | Rectangle |
| x | 14 | 54 | 114 | 174 | 234 |
| y | 192 | 192 | 192 | 192 | 192 |
| Width | 30 | 50 | 50 | 50 | 30 |
| Height | 20 | 20 | 20 | 20 | 20 |
| Visible after start | yes | yes | yes | yes | yes |
| "Settings" dialog | | | | | |
| Text | SP | %. 1f | %. 1f | %. 1f | °C |
| Standard color rectangle | white | white | white | white | white |
| Standard color text | black | light green | light green | light green | black |
| Alarm color rectangle | white | white | white | white | white |
| Alarm color text | black | light green | light green | light green | black |
| Border size | 1 | 1 | 1 | 1 | 1 |
| Border color | black | black | black | black | black |

Perhaps you remember: clicking the setpoint value is supposed to start an editor, with which the setpoint value can be entered. This function is not defined here, but in the following step.

Implement HMI functions

The behavior of the user interface (the HMI) and its interaction with the remaining application is implemented by means of interface blocks. You have to create interface blocks for all masks and objects of the HMI - in any case for those that should respond to input, display values or interact with other blocks.

1. **Start edit mode:** Make sure that you are in edit mode.
2. **Select the "Project" tab:** Click the "Project" tab to activate it.
3. **Create "HMI" block:** In order to maintain a good overview of the application, the HMI blocks should be saved in a separate program block. Therefore, create a new program block with the name "HMI".

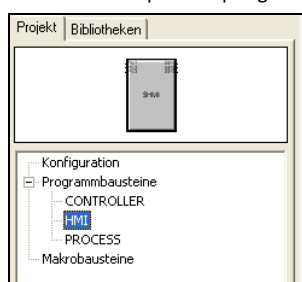


Fig. 299: Sample project: Insert "HMI" block

4. **Select the "Libraries" tab:** Click the "Libraries" tab.
5. **Select the library "LIB401 HMI":** You will find the interface functions in the library "LIB 401 HMI". Therefore, click the "+" sign in front of the library name in order to display the elements of the library. As you can see in the following graphic, a suitable block exists for each object type of the HMI.

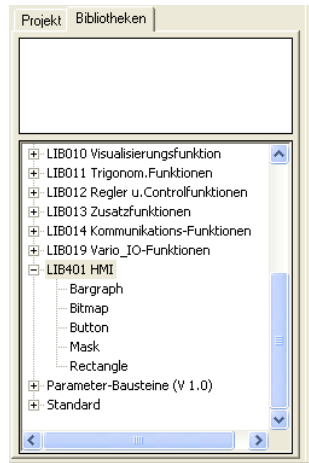


Fig. 300: Sample project: Library "LIB401 HMI"

Actual value display

The actual value of the oven chambers should be displayed with three objects of the type "Rectangle". Although we specified above how the values should be displayed, the data to be displayed has not yet been supplied.

1. **Use "Rectangle" block:** Drag a block of the type "Rectangle" to the workspace. The Mask Designer opens automatically.
2. **Assign block:** Now you have to assign the corresponding HMI element to the block. Click the object "Value PV1" (see following graphic).

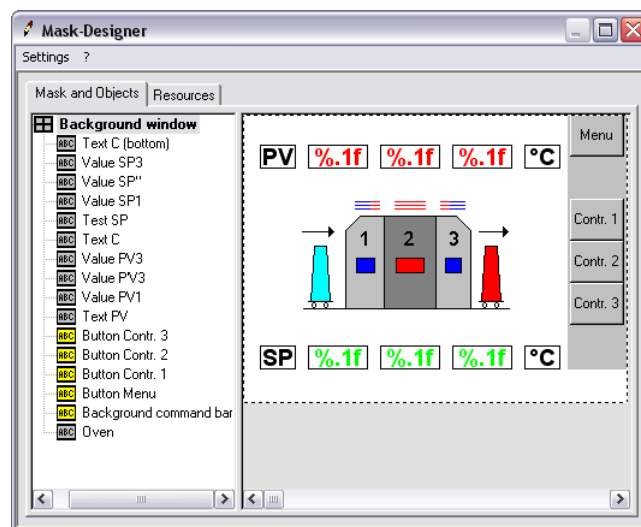


Fig. 301: Sample project: Assign elements

3. **Save assignment:** Click the "OK" button to save your input.
Note: The "OK" button is only active if you have selected an object of the correct type. This prevents you from accidentally assigning an HMI object of the type "Rectangle" to an interface object of the type "Bargraph".
4. **Insert additional blocks:** Now insert two more blocks of the type "Rectangle" in the worksheet in the same manner and connect them with "Value PV2" and "Value PV3".
5. **Prepare connection:** The actual value display should show the actual value for the respective oven chamber.
Therefore, insert three inputs ("PV1", "PV2", "PV3") with the data type "Float" in the worksheet. Connect these three inputs (corresponding to the graphic below) with the input "ai_Value" of the "Rectangle" block.

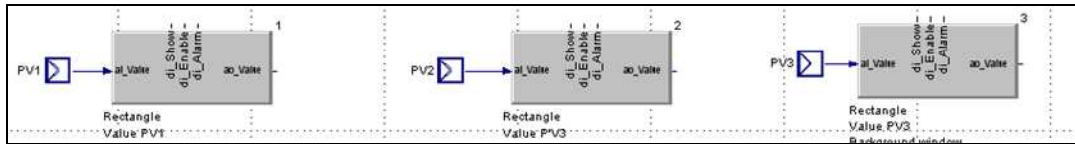


Fig. 302: Sample project: Connect actual value display

Create setpoint value display

- 1. Insert blocks:** Insert three additional blocks of the type "Rectangle" in the program block "HMI". Connect them with the HMI objects "Value SP1", "Value SP2", "Value SP3".
- 2. Prepare connection:** Users can enter a new setpoint value in the HMI. This value can be read from the output "ao_Value". Therefore, insert three outputs of the type "Float" in the worksheet. Give the outputs the names "SP1", "SP2", "SP3".
Connect the three outputs with the outputs ("ao_Value") of the blocks corresponding to the following graphic.

Later we will modify the program block "Controller" so that it can process this input.

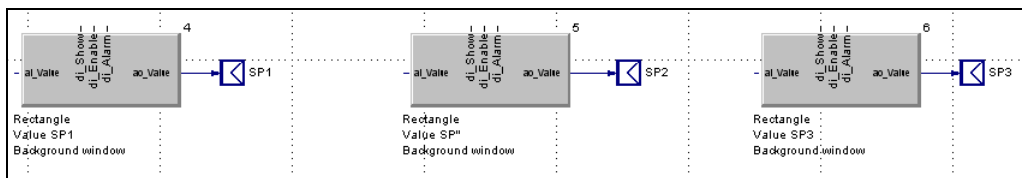


Fig. 303: Sample project: Connect setpoint value display

By clicking the setpoint value display, users should be able to start an editor for entering the setpoint value. For this purpose, the value "true" must be assigned to the input "di_Enable" of the "Rectangle" blocks.

- 3. Insert parameter blocks:** Insert three blocks of the type "Parameter (Bit)" from the "Parameter blocks" library. Assign these blocks the value "on".

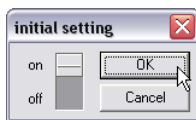


Fig. 304: Sample project: Define default value for parameter blocks

- 4. Connect parameter blocks:** Connect each of the newly inserted parameter blocks of the type "Bit" with the input "di_Enable" of a block of the type "Rectangle".
The result should look like this:

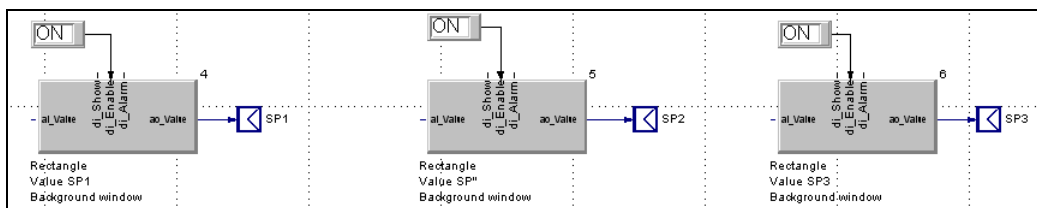


Fig. 305: Sample project: Connect setpoint value display with parameter blocks

Link buttons

- 1. Insert blocks:** Insert four additional blocks of the type "Button" in the program block "HMI". Connect them with the HMI objects "Button Menu", "Button Contr. 1", "Button Contr. 2" and "Button Contr. 3".
- 2. Insert "CALLPG" block:** Insert a block of the type "CALLPG" from the library "LIB013 Supplementary Functions". This block can be used to start an operating page or the main menu.
- 3. Insert "MONO" block:** Insert the "MONO" block from the library "LIB009 Logical Functions".

4. **Connect "Button Menu":** Connect the output "do_Press" of the "Button Menu" block with the input "di_neg" of the "MONO" block. Connect the output "do_z_3" of the "MONO" block with the output "di_d_1" of the "CALLPG" block (see following graphic).
If the user presses the button, the desired page will not be opened until after the button has been released.
5. **Specify page to be opened:** The page to be opened is signaled to the "CALLPG" block via the input "ai_BlockNo". Since the main menu always has the number "0", insert a block of the type "Parameter (Float)" here from the library "ai_BlockNo" and assign it the value "0".
Then connect the parameter block with the "CALLPG" block.

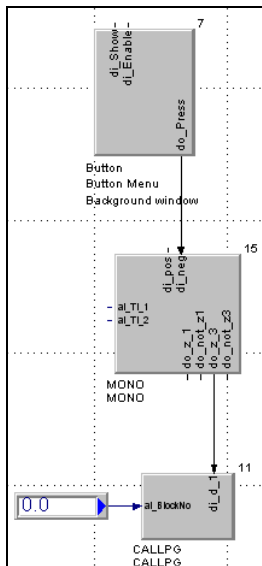


Fig. 306: Sample project: Call main menu

Note: If the user leaves the page that was called (here the main menu), he automatically returns to the calling page.

- 6 **Call operating pages of the controllers:** The remaining three buttons should call the operating pages of the controllers. The numbers of the controller operating pages should be made available by means of inputs.
Therefore, insert three additional blocks of the type "CALLPG" and three inputs of the type "Float" and connect them corresponding to the following graphic:

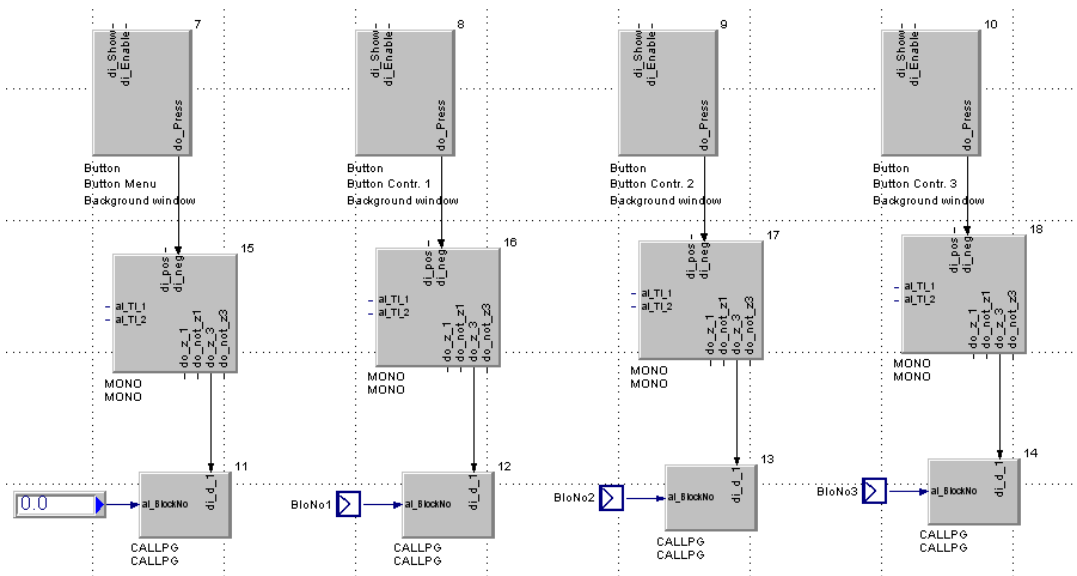


Fig. 307: Sample project: Call operating pages

7. **Define interfaces:** Click the "HMI" program block in the tree structure and release the inputs and outputs with the context menu command "Design" (further information on this command can be found in section II-5.4 Step 4: Specifying interfaces, connecting blocks).

The result should look like this:

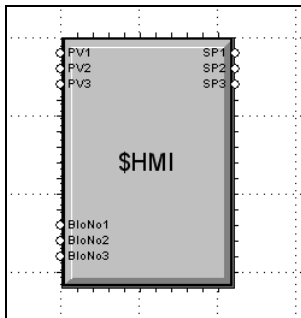


Fig. 308: Sample project: "HMI" program block interfaces

Configure "Controller" program block

Additional interfaces must be added to the "Controller" program block.

- **Block number:** In order to open a page, the "CALLPG" function of the "HMI" program block needs the block number of the calling page. It must be available to the "Controller" program block.
- **Setpoint value:** The setpoint values entered by the user must be sent to the "Controller" program block.

To do this proceed as follows:

1. **Select "Controller" program block:** Select the "Controller" program block on the "Project" tab with a double click.
2. **Insert inputs and outputs:** Insert three inputs and three outputs of the type "Float". Connect and name them corresponding to the following graphic:

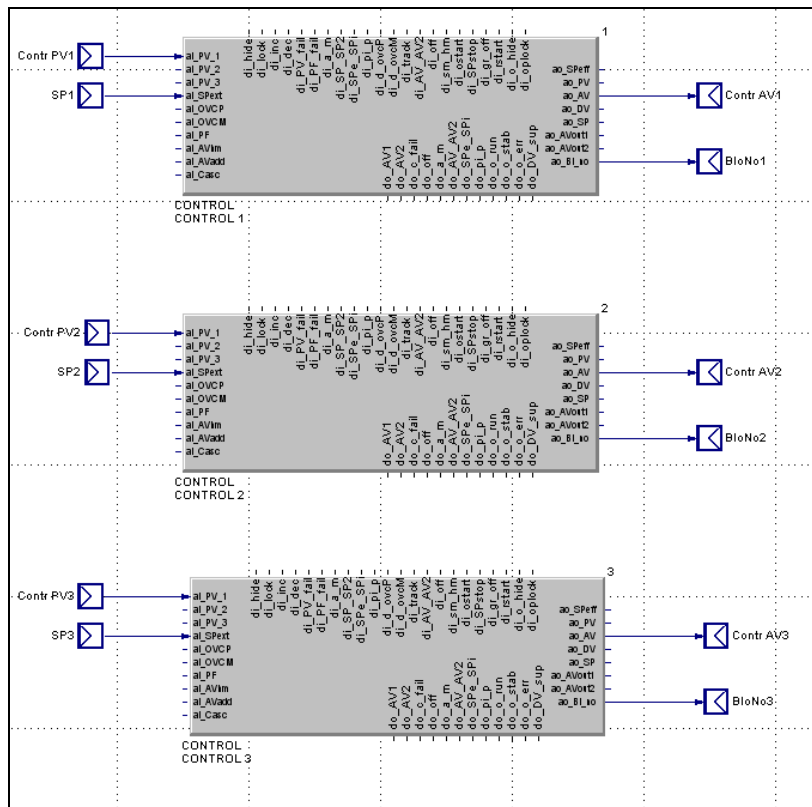


Fig. 309: Sample project: Configuring "Controller" program block

3. **Define interfaces:** Click the "Controller" program block in the tree structure and release the inputs and outputs with the context menu command "Design" (further information on this command can be found in section II-5.4 Step 4: *Specifying interfaces, connecting blocks*).

The result should look something like this:

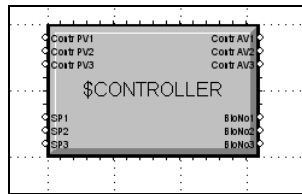


Fig. 310: Sample project: "Controller" program block interfaces

Connect "HMI" program block

Finally, you have to connect the "HMI" program block with the "PROCESS" and "CONTROLLER" blocks.

1. **Select "Configuration" entry:** Select the "Configuration" entry in the tree view with a double click.

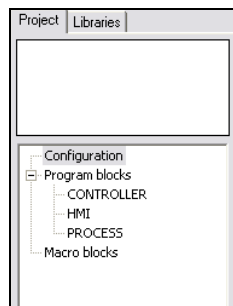


Fig. 311: Sample project: Connect "HMI" program block

2. **Connect program blocks:** Connect the program blocks corresponding to the following graphic.

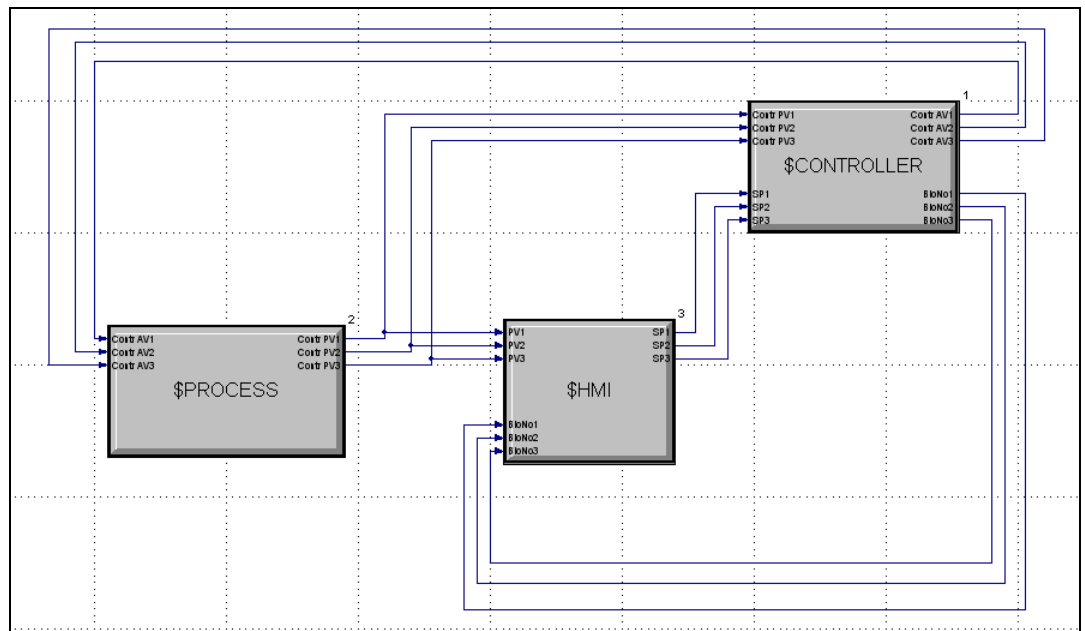


Fig. 312: Sample project: Connect "HMI" program block

II-5.7 Step 7: An application test

Time for an application test. The application should be executed in the *KS 108* simulation.

1. **Start BlueSimulation:** Start the application BlueSimulation.

2. **Start run mode:** Start your project in BlueDesign in run mode. To do this, use the menu command "Run/Enter".

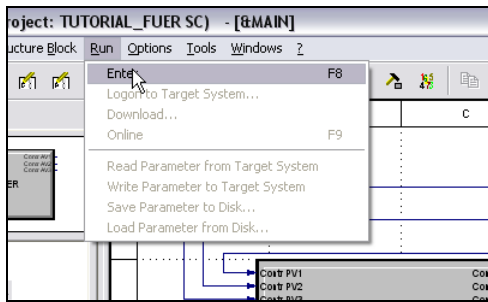


Fig. 313: Practical example: Start run mode

3. **Start "Logon to Target System" dialog:** The application should now be transferred to the simulator. To do this, use the menu command "Run/Download ...". Now you see the "Logon to Target System" dialog.

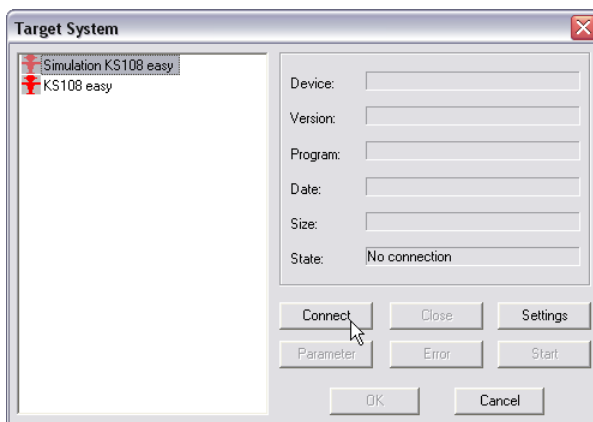


Fig. 314: Practical example: "Logon to Target System" dialog

4. **Connect to target system:** If several target systems are displayed in the left area of the dialog, select the entry "Simulation KS108 easy". Then click the "Connect" button.
 Note: Additional settings are not necessary for working with the simulator.
 After the connection has been established to the target system, information about the application is displayed in the dialog (name and size of the application, status, etc.).

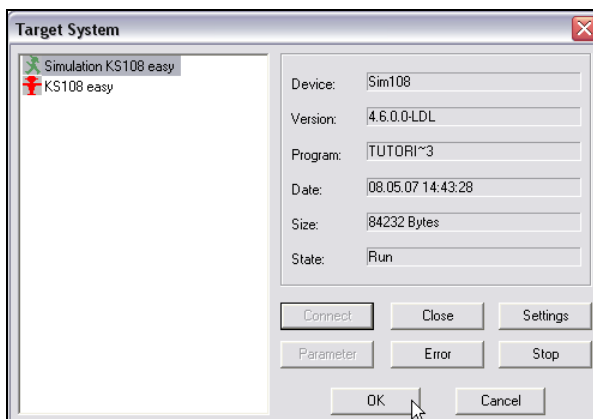


Fig. 315: Practical example: "Logon to Target System" dialog (2)

5. **Close dialog:** Close the dialog by clicking "OK". The application is now transferred. Status messages on the transfer process are displayed in the "General" tab.

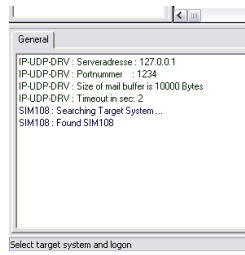


Fig. 316: Practical example: Status messages

6. **Test application:** Now you can test the application in the "BlueSimulation".

II-5.8 Step 8: Working with macros

Our project is now able to function; however, especially the "PROCESS" program block is relatively disorganized - although the simulated situation is fairly simple.

The highlighted part of the program block in the following graphic can be exported to a macro relatively easily.

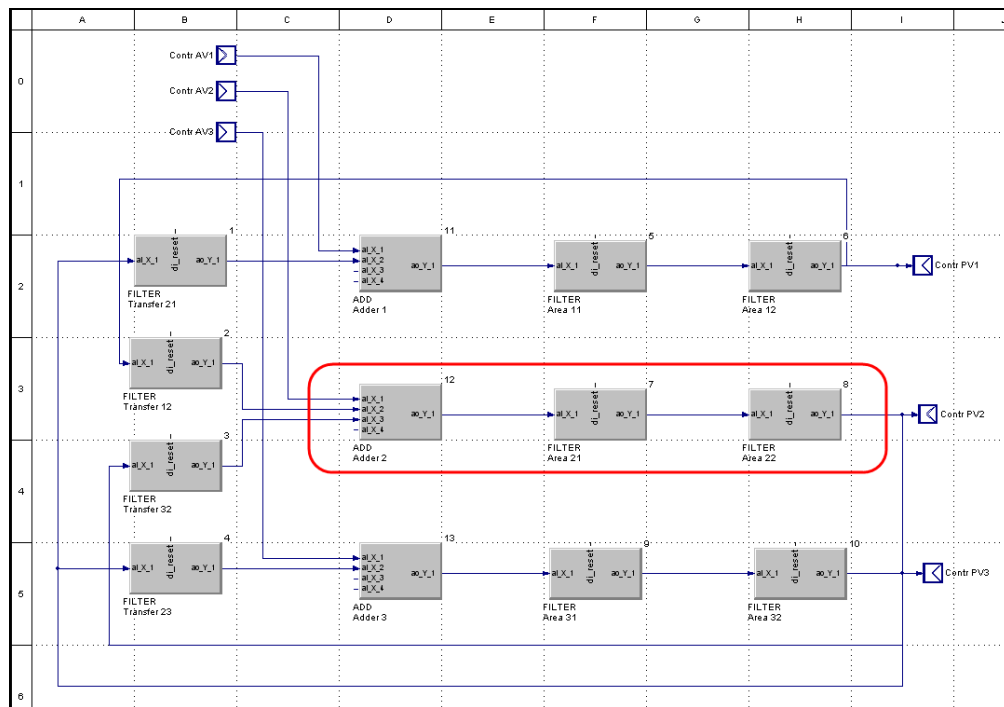


Fig. 317: Practical example: Program block "PROCESS"

Macros are a convenient way of creating your own copy templates. Proceed as follows in order to create and use a macro.

1. **Start edit mode:** Switch to edit mode.
2. **Create macro:** Click the "Macro blocks" node in the tree structure on the "Project" tab. Select the context menu command "Create new macro" and name the macro "OVENSIM".

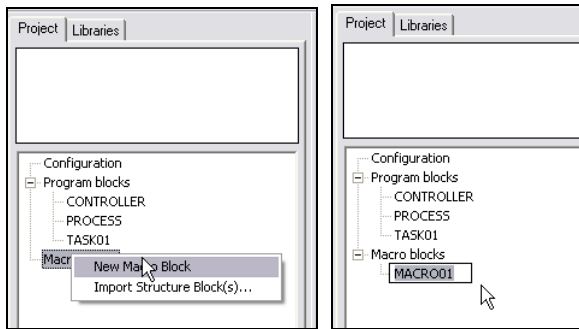


Fig. 318: Practical example: Create new macro

- Copy blocks:** Copy the blocks highlighted in the graphic above (Fig. 317) and insert them in the macro. Name the blocks corresponding to the graphic below.

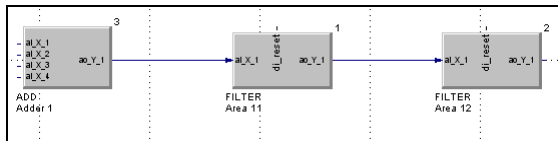


Fig. 319: Practical example: Insert macro

- Add inputs and outputs:** The macro is started the same way as a program block. In order to transfer values to it or to accept values from it, inputs and outputs must be defined. Create them corresponding to the graphic below (Note: use the data type "FLOAT"). Then position the inputs and the output.

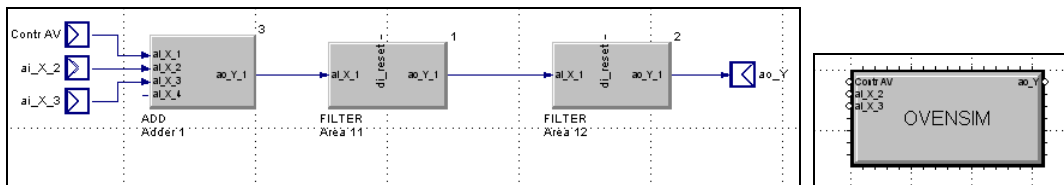


Fig. 320: Practical example: Define inputs and outputs for macro

- Delete blocks from the "PROCESS" program block:** Delete the blocks to be replaced by the macro from the program block.

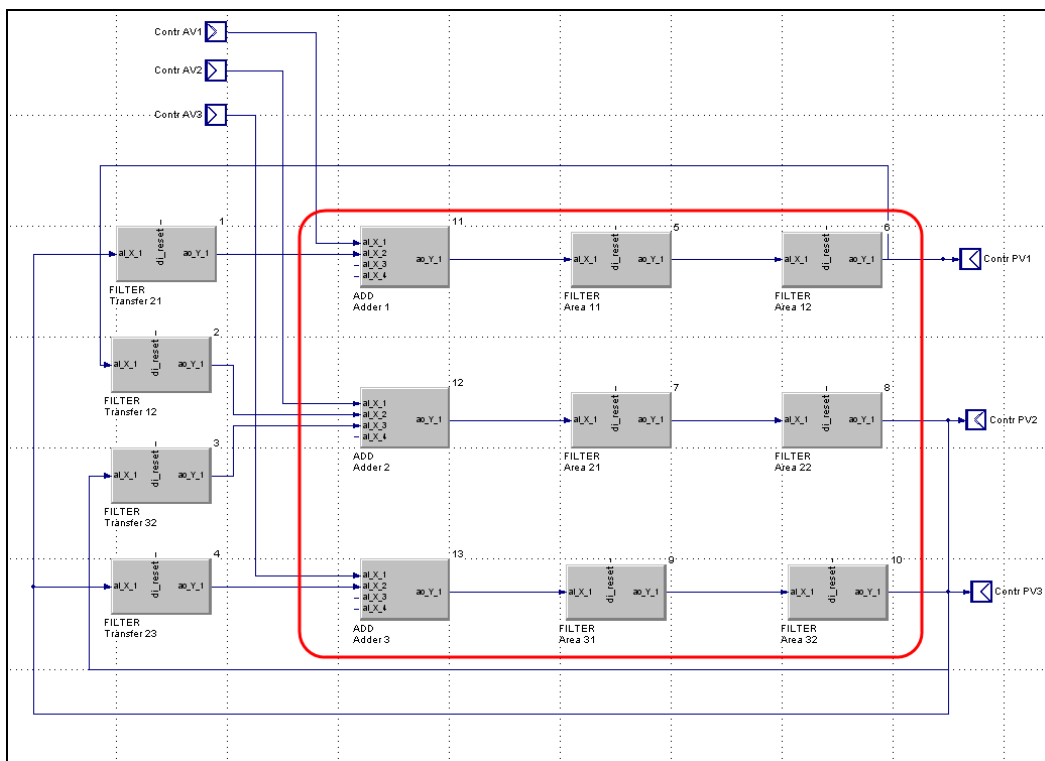


Fig. 321: Practical example: Delete blocks

- 6. **Insert macro:** Click the macro "OVENSIM" in the "Project" tab. Drag three copies of the macro into the "PROCESS" program block.

Connect the macros corresponding to the following graphic:

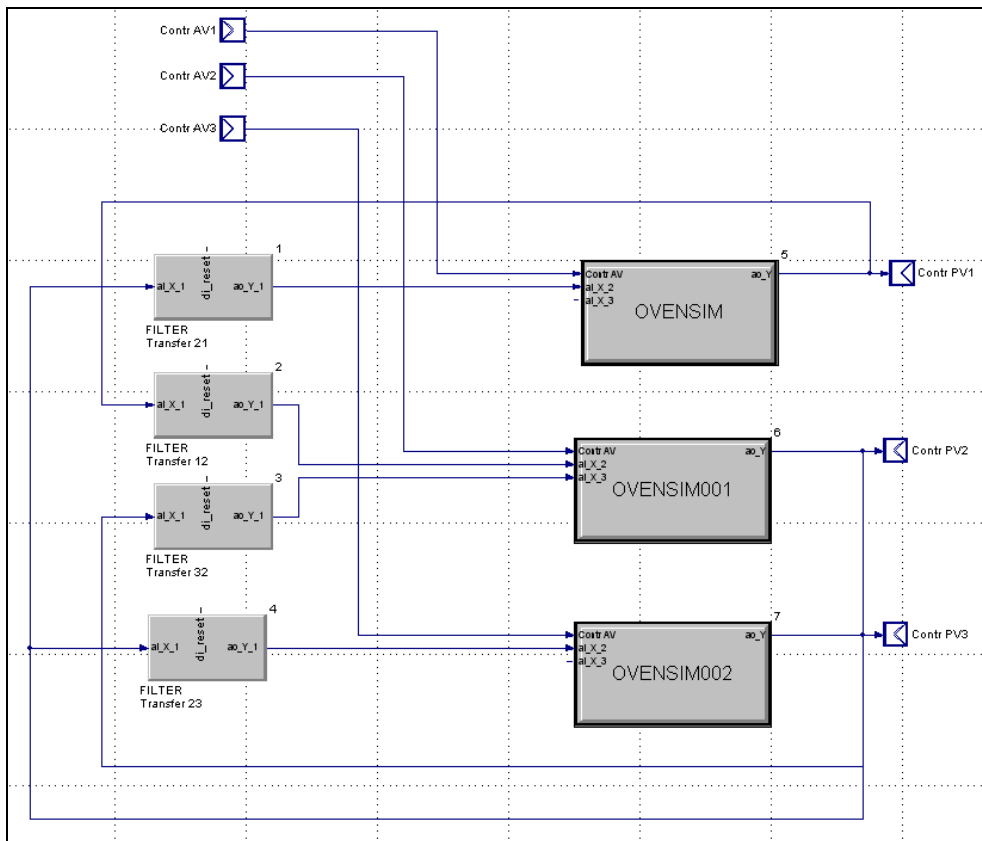


Fig. 322: Practical project: Insert macros in "PROCESS" program block

7. **Change call name:** Macros are numbered consecutively ("OVENSIMU", OVENSIMU001", etc.). These names should be changed to more meaningful names. Position the mouse pointer on the macro symbol and start the context menu command "Call Name". Give the macros the names "Area 1", "Area 2", "Area 3".

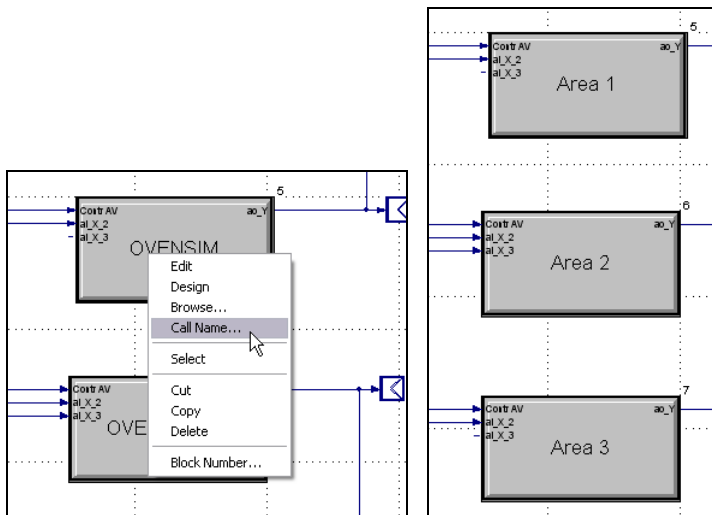


Fig. 323: Practical project: Change call name

8. **Run mode:** Switch the system to run mode and change the parameters of the macros, so that they correspond to the definitions in chapter "Step 7: Determining parameters". Note: You can select the macros in the tree structure on the left by selecting the corresponding macro. Then click the respective blocks in order to edit the parameters.

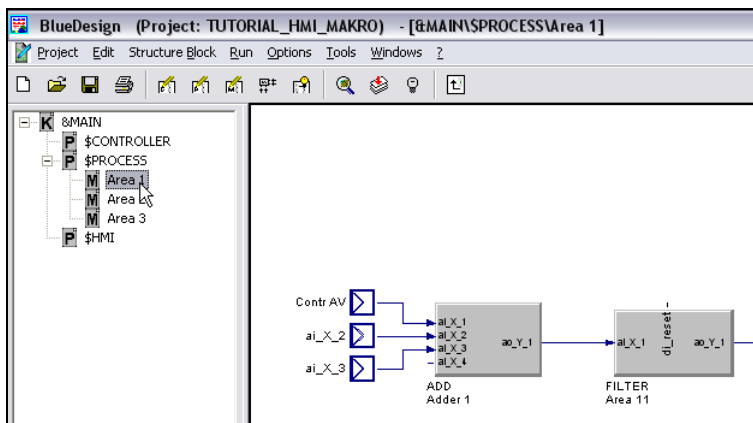


Fig. 324: Select macro

III Function blocks

The KS 108 easy function library contains all functions which are normally used for plant operation. These include:

- Functions for calculation of mathematic formulas from simple addition to exponent function
- Logic functions and functions for realization of control sequences
- Numerous selection and storage functions are helpful for signal processing.
- Alarm and limit value functions are indispensable for plant safety.
- Interface functions facilitate communication with adjacent and supervisory systems.
- Functions for implementation of complex and flexible control and program sequences as well as profile control tasks meet the most exacting requirements.

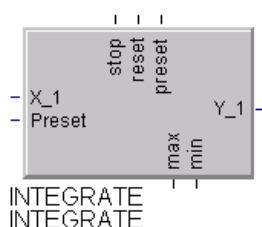
The wiring principle of composed functions such as programmer, controller cascades and stepping control are explained in the relevant basic function descriptions in this manual.

General KS 108 easy function block features

The features of multi-function unit KS 108 easy are determined by purposeful connection of standard function blocks with adjustable parameters.

In the KS 108 easy engineering, a function block represents a black box with analog inputs (from left), digital control inputs (from top), analog outputs (downward) and control or status outputs (to right), as shown in the integrator example.

The following abbreviations are used for general inputs which mean process values and outputs which mean function results:



analog inputs: X_1, Preset, ...

analog outputs: Y_1, ...

digital inputs: stop, ...

digital outputs: max, ...

Abb. 325

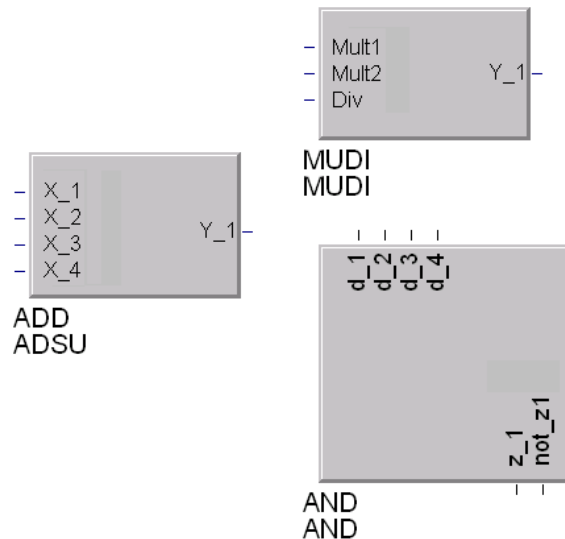


HINT!

The abbreviations for inputs and outputs with special signification are derived from their function.

Not all inputs and outputs of a function block need to be connected. The following rule is applicable: open inputs are without effect. Examples: totalizer, multiplier, AND gate. In some cases, the connection of an input has an additional effect, if e.g. priority handling is concerned (programmer control inputs).

Some function blocks may only be used once, this is noted at the relevant blocks. For some block timing is important (e.g. controller, programmer, time functions), here you'll also find a hint in the relevant functionblock chapter.



Example

Abb. 326

The parameters of each function block are adjustable. Apart from an individual description for documentation purposes, the majority of blocks is provided with function-specific parameters. In addition to very special ones, some parameters are quite frequent. For these general values, always the same identifiers are used:

| | |
|--------------------|--|
| A, B, C, D | factors without special signification |
| A0, B0, ... X0, Y0 | appended 0 as a symbol for an offset x0 = offset of an input, y0 = offset of an output |
| T, Ti | times in seconds (delays, pulse or pause duration) |
| Mode | These parameters are used to select function parameter setting by means of the described parameter or by means of an analog input (dynamic parameter setting). |

In the following parameter tables for parameters and configurations, there are listed type, access, default, a range if special, and a switch off – value if existing. With type a general range is defined:

| | |
|----------|---|
| bool: | 0 ; 1. |
| float: | $-3.4 \cdot 10^{38} \dots + 3.4 \cdot 10^{38}$ |
| integer: | $-32768 \dots + 32767$ |
| time: | $0 \dots 1.8 \cdot 10^6$ min or in hours $0 \dots 30000:00:00$ h |
| enum: | list of elements for selection. No general range. |
| text: | texts are used for description, e.g. name of function block or in text block. Usually, texts can be set in engineering tool only. |

Access is read-only (r) or read and write access (r/w).

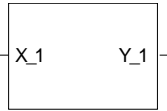
Default is the start value, which is preset for the function.

If the function has a special range, which is not variable and depending on other parameters, it is given in the table.

Switch – off value: 32000.

III-1 Scaling and calculating functions

III-1.1 ABSV (Absolute value (No. 01))



ABSV

Abb. 327

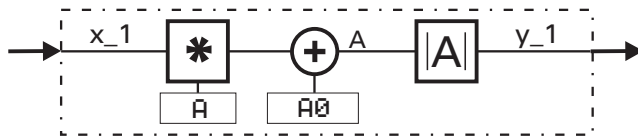


Abb. 328

$$Y_1 = |A \cdot X_1 + A_0|$$

The absolute value of a number is its number without polarity sign. This is the best solution for scaling a value that can't become negative, in reference to calculating time. This function block should be used, when scaling must not use a lot of calculating time.

Input variable X_1 is multiplied by factor A (parameter). Now, constant A_0 is added. The absolute value of the resulting value is formed and output at Y_1 .

Example:

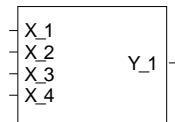
$Y_1 = \text{ABS}(A \cdot X_1 + A_0)$ $A=5$ $X_1=2$ $A_0=+5$ results in $Y_1 = 15$
 $Y_1 = \text{ABS}(fA \cdot X_1 + A_0)$ $A=5$ $X_1=2$ $A_0=-20$ results in $Y_1 = 10$

| In-/Outputs | | |
|-------------|-------|--|
| Name | Type | Description |
| X_1 | Float | Number the value of which is required without sign |

| Name | Type | Description |
|------|-------|---|
| Y_1 | Float | Number without sign. Related to the calculation time, this value is the optimum solution for scaling of a value which cannot be negative. |

| Parameter | | | | | | | |
|-----------|---------------|-------|---|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| A | Factor for X1 | Float | Multiplication factor for input scaling | r/w | 1.0 | | |
| A0 | Offset a0 | Float | Offset for input correction | r/w | 0.0 | | |

III-1.2 ADD (Addition/subtraction (No. 03))



ADD

Abb. 329

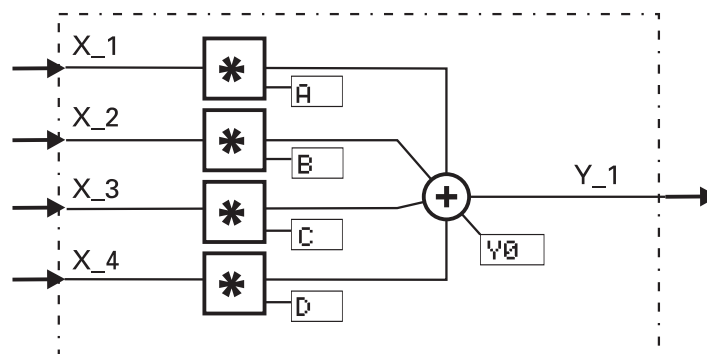


Abb. 330

$$Y_1 = A \cdot X_1 + B \cdot X_2 + C \cdot X_3 + D \cdot X_4 + Y_0$$

Input variables **X1**...**X4** are multiplied by factors **A**...**D**.

Constant **Y0** is added to the sum of evaluated inputs. Value "0" is assigned automatically to unused inputs.

In-/Outputs

| Name | Type | Description |
|-------------|-------|---------------|
| X_1 ... X_4 | Float | Input value 1 |

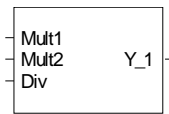
| Name | Type | Description |
|------|-------|---|
| Y_1 | Float | Totalizer result, after scaling and with offset |

Parameter

| ID | Name | Type | Description | Access | Default | Range | Off |
|----|---------------|-------|---|--------|---------|-------|-----|
| A | Factor for X1 | Float | Multiplication factor for input value 1 | r/w | 1.0 | | |
| B | Factor for X2 | Float | Multiplication factor for input value 2 | r/w | 1.0 | | |
| C | Factor for X3 | Float | Multiplication factor for input value 3 | r/w | 1.0 | | |
| D | Factor for X4 | Float | Multiplication factor for input value 4 | r/w | 1.0 | | |

| | | | | | | | |
|----|-----------|-------|--|-----|-----|--|--|
| Y0 | Offset y0 | Float | Offset y0, is added to the result of totalization before output. | r/w | 0.0 | | |
|----|-----------|-------|--|-----|-----|--|--|

III-1.3 MUDI (Multiplication / division (No. 05))



MUDI

Abb. 331

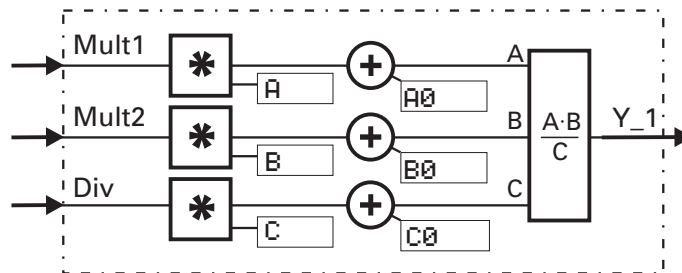


Abb. 332

$$Y_1 = \frac{A \cdot B}{C} = \frac{(A \cdot Mult_1 + A_0) \cdot (B \cdot Mult_2 + B_0)}{(C \cdot Div_3 + C_0)}$$

Input variables **Mult1**, **Mult2** and **Div** are multiplied by factors **A**, **B**, **C**. The relevant constants **A0**, **B0**, **C0** are added.

The output variable corresponds to the product. Value "1" is assigned automatically to unused inputs. With divisions by "0" ($C = C \cdot Div + C_0 = 0$) output **Y_1** is set to $1.5 \cdot 10^{37}$.

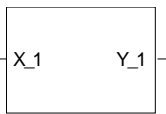
| In-/Outputs | | |
|-------------|-------|---|
| Name | Type | Description |
| Mult1 | Float | Input variable 1 for the numerator (multiplier) of the function |
| Mult2 | Float | Input variable 2 for the numerator (multiplier) of the function |
| Div | Float | Input variable for the denominator (divisor) of the function |

| Name | Type | Description |
|------|-------|------------------------|
| Y_1 | Float | Result of the function |

| Parameter | | | | | | | |
|-----------|---------------|-------|---|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| A | Factor for X1 | Float | Multiplication factor for input variable 1 of the numerator | r/w | 1.0 | | |
| B | Factor für X2 | Float | Multiplication factor for input variable 2 of | r/w | 1.0 | | |

| | | | | | | | |
|----|----------------|-------|---|-----|-----|--|--|
| | | | the numerator | | | | |
| C | Factor für X3 | Float | Multiplication factor for the input variable of the denominator | r/w | 1.0 | | |
| A0 | Offset on X1*A | Float | Offset for input variable 1 of the numerator | r/w | 0.0 | | |
| B0 | Offset on X2*B | Float | Offset for input variable 2 of the numerator | r/w | 0.0 | | |
| C0 | Offset on X3*C | Float | Offset for the input variable of the numerator | r/w | 0.0 | | |

III-1.4 SQRT (Square root function (No. 08))



SQRT

Abb. 333

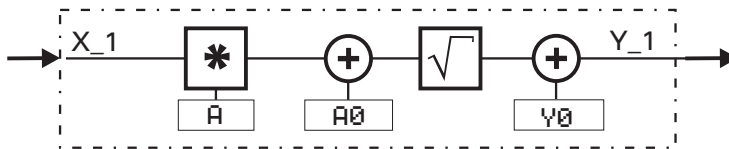


Abb. 334

$$Y_1 = \sqrt{A \cdot X_1 + A_0} + Y_0$$

Constant **A0** is added to input variable **X_1** multiplied by **A**. The result is subjected to square root extraction. Constant **Y0** is added to the result of square root extraction.

If the expression under the root is negative, the square root expression is set to 0. As a result: **Y_1** = 0.

If the input is not connected, this is interpreted as **X_1** = 0.

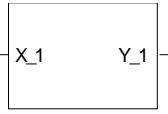
| In-/Outputs | | |
|-------------|-------|--|
| Name | Type | Description |
| X_1 | Float | Input value from which the square root must be extracted |

| Name | Type | Description |
|------|-------|---|
| Y_1 | Float | Square root of the scaled input variable. When the expression under the root (X_1 * A + A0) is negative, the output is set to output offset Y0. |

| Parameter | | | | | | | |
|-----------|---------------|-------|--|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| A | Factor for X1 | Float | Multiplication factor for the input variable | r/w | 1.0 | | |

| | | | | | | | |
|----|----------------|-------|--|-----|-----|--|--|
| A0 | Offset on X1*A | Float | Input offset, for the product of input variable X_1 and multiplication factor A. | r/w | 0.0 | | |
| Y0 | Offset on Y1 | Float | Output offset, the offset Y0 is added to the result of square root extraction. | r/w | 0.0 | | |

III-1.5 SCAL (Scaling (No. 09))



SCAL

Abb. 335

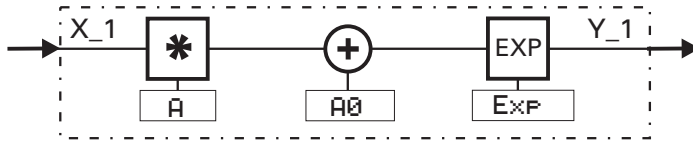


Abb. 336

$$Y_1 = (A \cdot X_1 + A0)^{Exp}$$

Input variable **X_1** is multiplied by factor **A** and added to constant **A0**. The result (**A * X_1 + A0**) is set to the power **EXP**.

If **X_1** is not used, this is interpreted as **X_1=0**. With **Exp = 0** **SCAL** outputs 1

| In-/Outputs | | |
|-------------|-------|-----------------------------|
| Name | Type | Description |
| X_1 | Float | Input variable to be scaled |

| Name | Type | Description |
|------|-------|-----------------|
| Y_1 | Float | Output variable |

| Parameter | | | | | | | |
|-----------|----------------|-------|---|--------|---------|---------------------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| A | Factor for X1 | Float | Multiplication factor for the input variable | r/w | 1.0 | | |
| A0 | Offset on X1*A | Float | Offset for the product of input variable and multiplication factor | r/w | 0.0 | | |
| Exp | Exponent | Float | The sum of product and offset is exponentiated with the fractional rational exponent Exp (also square root extraction, also 1/X). | r/w | 1.0 | -29999 ... 99999 | |

Example: $Y_1 = \sqrt[3]{X_1^2} = X_1^{\frac{2}{3}} = X_1^{0,\bar{6}}$

**HINT!**

*This function block should be used only, if the exponential function is needed. Factor \bar{A} and the offset $\bar{A}0$ are also available with functions that need less calculating time (e.g. **ADD**, **MUDI**, **ABSV**).*

III-1.6 10EXP (10s exponent (No. 10))

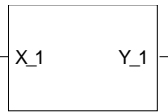
**10EXP**

Abb. 337

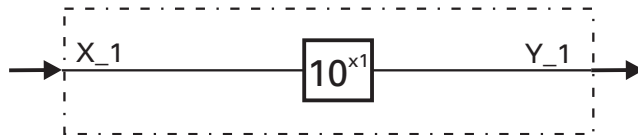


Abb. 338

$$Y_1 = 10^{X_1}$$

Input value X_1 is calculated according to formula $Y_1 = 10^{X_1}$ and output at the Y_1 output.

An unwired X_1 is interpreted as $X_1 = 0$ (Y_1 then is 1).

If the value at input X_1 is higher than 36,7, an overflow may occur.
In this case, output Y_1 is set to $1.5 \cdot 10^{37}$ rather than forming the power.

*Hint!**10EXP is the reversal function of function LG10.*

In-/Outputs

| Name | Type | Description |
|------|-------|--|
| X_1 | Float | Exponent, for calculation of 10 to the power of exponent |

| Name | Type | Description |
|------|-------|---|
| Y_1 | Float | Result of 10 to the power of exponent(= X_1) |

III-1.7 EEXP (e-function (No. 11))

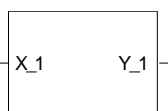
**EEXP**

Abb. 339

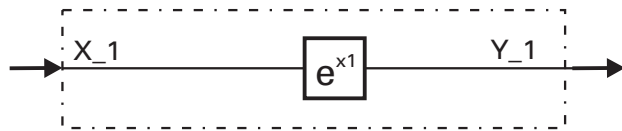


Abb. 340

$$Y_1 = e^{X_1}$$

The e-function is calculated. If input signal x_{-1} is higher than 85, there may be an overflow. In this case, $y_{-1} = 1,5 \cdot 10^{37}$ is output rather than forming the power. If x_{-1} is not wired, this is interpreted as $x_{-1} = 0$ and thus as $y_{-1} = 1$.



Hint!

EEXP is the reversal function of function LN

Examples:

With an input value of $x_{-1} = 5$, output value $y_{-1} = 148,413159$.

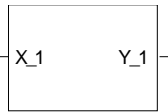
With an input value of $x_{-1} = 0,69314718$, output value $y_{-1} = 2$.

In-/Outputs

| Name | Type | Description |
|------|-------|--|
| X_1 | Float | Exponent, for calculation of the exponent function (base e = 2.718...) |

| Name | Type | Description |
|------|-------|--|
| Y_1 | Float | Result of the e function with exponent X_1 |

III-1.8 LN (Natural logarithm (No. 12))



LN

Abb. 341

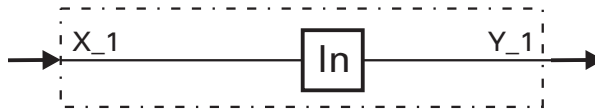


Abb. 342

$$Y_1 = \ln(X_1)$$

The natural logarithm of input variable X_1 is formed.

The basis of natural logarithms is constant $e = (2,71828182845904)$.

If X_1 is not wired, this is interpreted as $X_1 = 1$. In this case Y_1 is 0.

With a negative input variable X_1 , $Y_1 = -1,5 \cdot 10^{37}$ is set.

*Hint!*

LN is the reversal function of function EEXP.

Examples:

The result of input value $X_1 = 63$ is an output value of $Y_1 = 4,143134726$.

The result of input value $X_1 = 2,71828182845904$ is an output value of $Y_1 = 1$.

In-/Outputs

| Name | Type | Description |
|------|-------|--|
| X_1 | Float | Input variable X_1 the natural logarithm of which is formed. The base of natural logarithms is constant $e = 2.71828182845904$. |

| Name | Type | Description |
|------|-------|---|
| Y_1 | Float | Output variable is the natural logarithm of input variable X_1. |

III-1.9 LG10 (10s logarithm (No. 13))

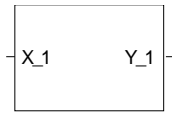
**LG10**

Abb. 343

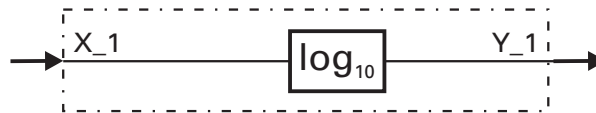


Abb. 344

$$Y_1 = \log(X_1)$$

The common logarithm of input variable X_1 is formed.

LG10 provided the logarithm of a number to base 10.

If X_1 is not wired, this is interpreted as $X_1 = 1$. In this case, Y_1 is 0.

With a negative input variable X_1 , $Y_1 = -1,5 \cdot 10^{37}$ is set.

*Hint!*

LG10 is the reversal function of function 10EXP.

Examples:

The result of input value $X_1 = 63$ is an output value of $Y_1 = 1,799340549$.

The result of an input value $X_1 = 10$ is an output value of $Y_1 = 1$.

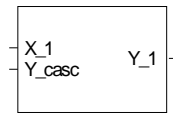
In-/Outputs

| Name | Type | Description |
|------|-------|---|
| X_1 | Float | Input value, the base 10 logarithm of this value is formed. |

| Name | Type | Description |
|------|-------|--|
| Y_1 | Float | Result of the base 10 logarithm of X_1 |

III-2 Non-linear functions

III-2.1 LINEAR (Linearization function (No. 07))



LINEAR

Abb. 345

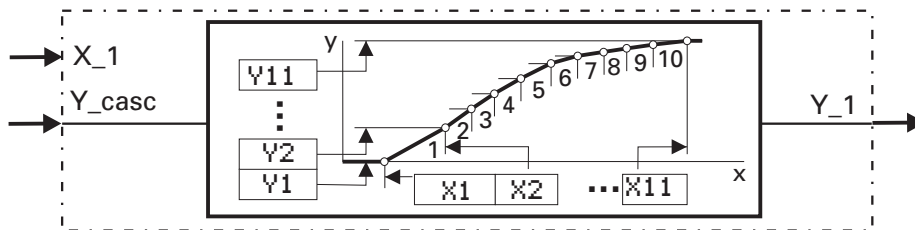


Abb. 346

The Block **LINEAR** calculates $y = f(x)$.

Max. 11 adjustable segment points for simulation or linearization of non-linear functions are available. Each segment point comprises input X_{-1} and output Y_{-1}

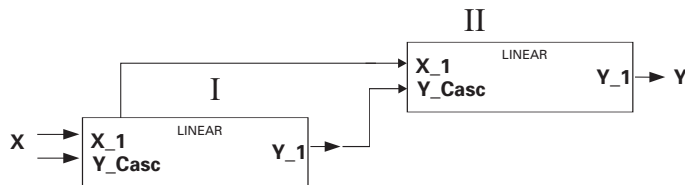
The segment points are connected automatically by straight lines. Each input value X_{-1} has a defined output value Y_{-1} . With an input value X_{-1} smaller than parameter X_{-1} , the output value is equal to the Y_{-1} value. If input value X_{-1} exceeds the highest parameter $X(n)$ value, the output value is equal to the relevant $Y(n)$ value.

The condition for input of configuration parameters is that values are input in ascendant order ($X(1) < X(2) < \dots < X(11)$). The end of value pairs is marked by the "OFF" value in the next input value $X(n+1)$

This function block is cascable. It has 2 inputs: The 1st input provides the variable which must be linearized. The 2nd input (Y_{casc}) is used to connect the previous **LINEAR**-block.

| In-/Outputs | | |
|-------------|-------|---|
| Name | Type | Description |
| X_1 | Float | Input variable which must be linearized. When it is smaller than parameter X_1, the output value is set to the Y_1 value. When it is higher than the parameter X_(n) used last, the output value is set to the Y_(n) value used last. |
| Y_casc | Float | Cascade input to which the output of the previous LINEAR block is connected in a cascade. |
| Name | Type | Description |
| Y_1 | Float | Linearization result |

| Parameter | | | | | | | |
|--------------|----------------|-------|-----------------------------------|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| X_1 ... X_11 | Input value 1 | Float | Input variable for curve point 1 | r/w | 0.0 | | ja |
| Y_1 ... Y_11 | Output value 1 | Float | Output variable for curve point 1 | r/w | 0.0 | | |



Example: Linear as cascade

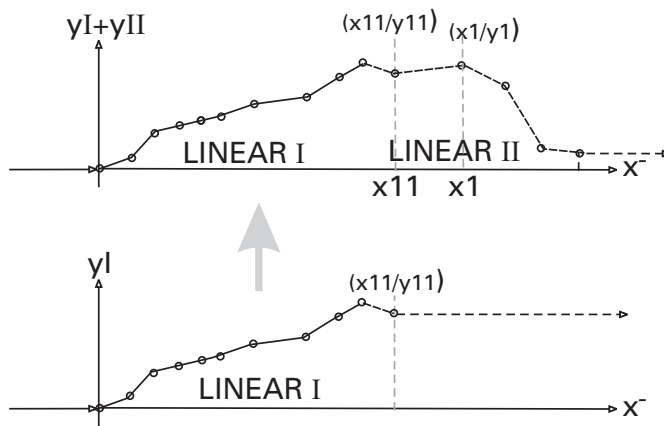
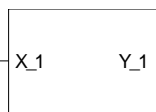


Abb. 347

III-2.2 GAP (Dead band (No. 20))



GAP

Abb. 348

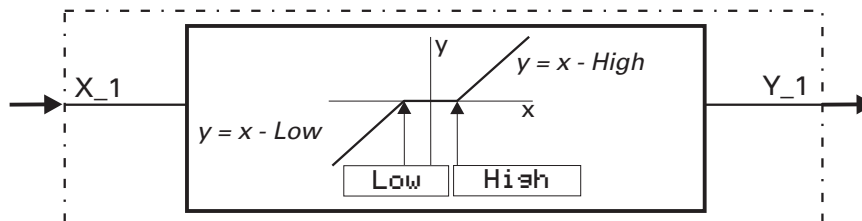


Abb. 349

$$Y_1 = X_1 - Low \quad \text{at } X_1 < Low$$

$$Y_1 = 0 \quad \text{at } X_1 = \text{Low} \dots \text{High}$$

$$Y_1 = X_1 - \text{High} \quad \text{at } X_1 > \text{High}$$

The range of the dead band is adjusted with parameters **Low** (lower limit) and **High** (upper limit). If input value X_1 is within the dead band ($\text{Low} \leq X_1 \leq \text{High}$), output value $Y_1 = 0$. If X_1 is not used, this is interpreted as $X_1 = 0$.

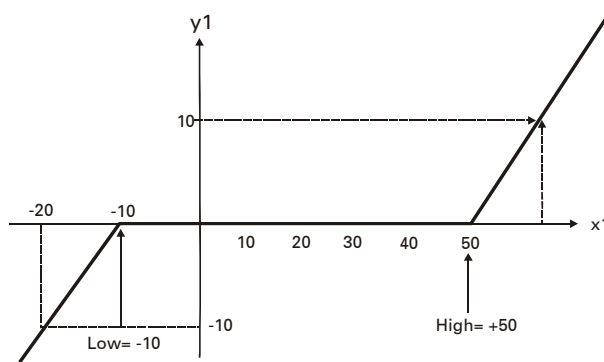


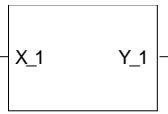
Abb. 350

Example:

In the following example, -10 for **Low** and 50 for **High** was used.

| In-/Outputs | | | | | | | |
|-------------|------------|---|-------------------|--------|---------|-------|-----|
| Name | Type | Description | | | | | |
| X_1 | Float | Input signal | | | | | |
| Name | Type | Description | | | | | |
| Y_1 | Float | When the input value X_1 is in the dead band (X_1 is higher than Low and smaller than High) the output value Y_1 = 0. | | | | | |
| Parameter | | | | | | | |
| ID | Name | Type | Description | Access | Default | Range | Off |
| Low | Low limit | Float | Lower limit value | r/w | 0.0 | | |
| High | High limit | Float | Upper limit value | r/w | 0.0 | | |

III-2.3 CHAR (Function generator (No. 21))



CHAR
Abb. 351

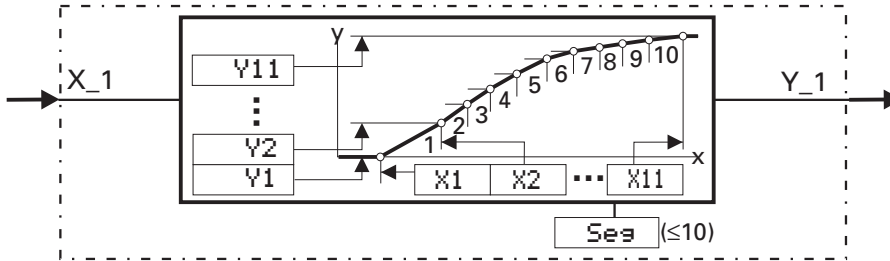


Abb. 352

With max. 11 adjustable value pairs, non-linear functions can be simulated or linearized. Each value pair comprises input X_{-1} and output Y_{-1} . The number of value pairs is determined using configuration parameter Seg (= number of segments +1 corresponds to the number of value pairs).

The value pairs are connected automatically with straight lines so that each input value X_{-1} provides a defined output value Y_{-1} . If input value X_{-1} is smaller than parameter X_{-1} , the output value is equal to the value of Y_{-1} . If input value X_{-1} is higher than the highest parameter $X(n)$, the output value is equal to the corresponding $Y(n)$ value.

During entry of the configuration parameters, the condition is that the assigned values stand in ascending order ($X(1) < X(2) < \dots < X(11)$).

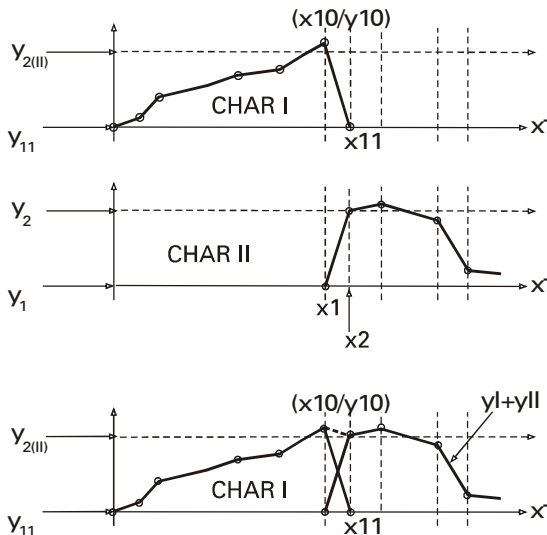
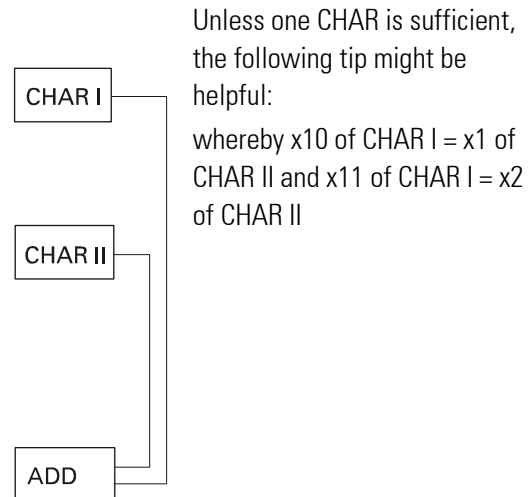


Abb. 353



In-/Outputs

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

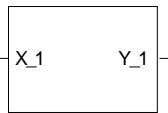
| | | |
|-----|-------|---|
| X_1 | Float | Input variable to be linearized. When it is smaller than configuration X_1 the output value is set to the value of Y_1. When it is higher than the configuration X_(n) used last the output value is set to the value of Y_(n) used last. |
|-----|-------|---|

| Name | Type | Description |
|------|-------|--|
| Y_1 | Float | Output is the linearized input variable. |

| Parameter | | | | | | | |
|--------------|----------------|-------|---|--------|---------|----------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Seg | No of segments | Int | Number of segments for linearization; the segment points are connected by straight lines automatically. | r/w | 2 | 1 ... 10 | |
| X_1 ... X_11 | Input value 1 | Float | Input value for curve point 1 | r/w | 0.0 | | |
| Y_1 ... Y_11 | Output value 1 | Float | Output value for curve point 1 | r/w | 0.0 | | |

III-3 Trigonometric functions

III-3.1 SIN (Sinus function (No. 80))



SIN

Abb. 354

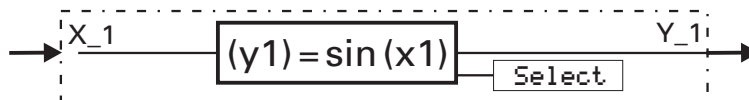


Abb. 355

$$y = \sin(x_1)$$

The function provides the sinus of the input value, i.e. X_1 is the angle the sinus of which is calculated. Parameter **Select** is used to adjust, if the angle is provided in degree of angle [°] or in radian.

Example degree of angle:

$$Y_1 = \sin(X_1), \quad X_1 = 30^\circ \quad \Rightarrow \quad Y_1 = 0,5$$

Example radian:

$$Y_1 = \sin(X_1), \quad X_1 = 90\text{rad} \quad \Rightarrow \quad Y_1 = 0,89399666$$

In-/Outputs

| Name | Type | Description |
|------|-------|------------------------------------|
| X_1 | Float | Input angle for the sinus function |

| Name | Type | Description |
|------|-------|-----------------------------|
| Y_1 | Float | Sinus of the input variable |

Parameter

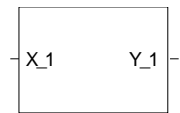
| ID | Name | Type | Description | Access | Default | Range | Off |
|--------|-------------|-----------------|--|--------|---------|-------|-----|
| Select | Select unit | Enum | Setting: Input value Angle is available as an angular degree [°] or as a radian. | r/w | 0 | | |
| | | Degree of angle | 1 degree is defined as the 360th fraction of a full angle, i.e. 1 full angle = 360° (corresponding to 2 Pi rad). | | 0 | | |

| | | | | | |
|--|--------|---|--|---|--|
| | Radian | A full angle comprises 2 Pi radiant: 1 full angle = 2 Pi rad (corresponding to 360°). | | 1 | |
|--|--------|---|--|---|--|

$1 \text{ rad} = 180^\circ/\pi = 57,296^\circ$
 $1^\circ = \pi/180^\circ = 0,017453 \text{ rad}$

Control with the pocket calculator:
 The function for the calculation in "rad" with the pocket calculator is limited to e.g. $\pm 8 \pi$. $\rightarrow 90/\pi = 28,6479$: $\sin(0,6479 \cdot \pi) = 0,893996664$
 Also during input in "°" usually a limitation is effective in the pocket calculator (e.g. $< 1440^\circ$)!

III-3.2 COS (Cosinus function (No. 81))



COS
 Abb. 356

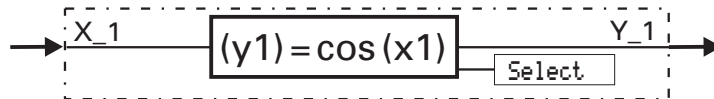


Abb. 357

$y = \cos(x1)$

The function provides the cosinus of the input value, i.e. **X_1** is the angle the cosinus of which is calculated. Parameter **Select** is used to adjust, if the angle is provided in degree of angle [°] or in radian.

Example degree of angle:

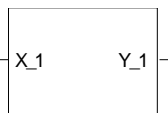
$Y_1 = \cos(X_1), \quad X_1 = 60^\circ \quad \Rightarrow \quad Y_1 = 0,5$

$Y_1 = \cos(X_1), \quad X_1 = 45\text{rad} \quad \Rightarrow \quad Y_1 = 0,525321988$

| In-/Outputs | | |
|-------------|-------|-------------------------------|
| Name | Type | Description |
| X_1 | Float | Angle for cosinus computation |
| Name | Type | Description |
| Y_1 | Float | Cosinus of the input variable |

| Parameter | | | | | | | |
|-----------|-------------|-----------------|--|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Select | Select unit | Enum | Setting: Input value Angle is available as an angular degree [°] or as a radian. | r/w | 0 | | |
| | | Degree of angle | 1 degree is defined as the 360th fraction of a full angle, i.e. 1 full angle = 360° (corresponding to 2 Pi rad). | | 0 | | |
| | | Radian | A full angle comprises 2 Pi radiant: 1 full angle = 2 Pi rad (corresponding to 360°). | | 1 | | |

III-3.3 TAN (Tangent function (No. 82))



TAN
Abb. 358



Abb. 359

$$y1 = \tan(x1)$$

valid for x1: $-90^\circ < X1 < +90^\circ \quad \left(-\frac{\pi}{2} < X1 < \frac{\pi}{2} \right)$

The function provides the tangent of the input value, i.e. **X_1** is the angle the tangent of which is calculated. Parameter **Select** is used to adjust, if the angle is provided in degree of angle [°] or in radian.

For calculation clarity the argument range is limited to the 1st or 4th quadrant (-90° ... 90° oder $-\pi/2 \dots +\pi/2$).

If input value **X_1** is out of this range, output **Y_1** is set to $-1,5 \cdot 10^{37}$ ($x1 \leq -90 [-\pi/2]$)

e.g. to $1,5 \cdot 10^{37}$ ($x1 \geq 90 [\pi/2]$).

Example degree of angle:

$$Y_1 = \tan(X_1) \quad X_1 = 60^\circ \quad \text{is} \quad Y_1 = 1,73205$$

Example radian:

$$Y_1 = \tan(X_1) \quad X_1 = 1,53\text{rad} \quad \text{is} \quad Y_1 = 24,498$$

In-/Outputs

| Name | Type | Description |
|------|-------|---|
| X_1 | Float | Angle for computation of the tangent function |

| Name | Type | Description |
|------|-------|---------------------------|
| Y_1 | Float | Tangent of input variable |

Parameter

| ID | Name | Type | Description | Access | Default | Range | Off |
|--------|-------------|-----------------|--|--------|---------|-------|-----|
| Select | Select unit | Enum | Setting: Input value Angle is available as an angular degree [°] or as a radian. | r/w | 0 | | |
| | | Degree of angle | 1 degree is defined as the 360th fraction of a full angle, i.e. 1 full angle = 360° (corresponding to 2 Pi rad). | | 0 | | |
| | | Radian | A full angle comprises 2 Pi radiant: 1 full angle = 2 Pi rad (corresponding to 360°). | | 1 | | |

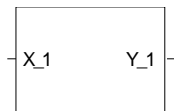
III-3.4 COT (Cotangent function (No. 83))**COT**

Abb. 360

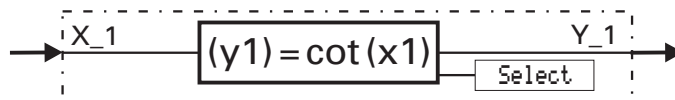


Abb. 361

$$y1 = \cot(x1)$$

valid for x1: $0 < x1 < 180^\circ$ ($0 < x1 < \pi$)

The function provides the cotangent of the input value, i.e. **X_1** is the angle the cotangent of which is calculated. Parameter **Select** is used to adjust, if the angle is provided in degree of angle [°] or in radian.

For calculation clarity, the range for the argument is limited to the 1st and 2nd quadrant

$0^\circ < x1 < 180^\circ$ or ($0 < x1 < \pi$). If input value **X_1** is out of this range, output **Y_1** is set to $1,5 \cdot 10^{37}$
 ($x1 < 0^\circ$) e.g. $-1,5 \cdot 10^{37}$ ($x1 \geq 180^\circ$ or [$x1 \geq \pi$]).

Example degree of angle:

$$Y_1 = \cot(X_1) \quad X_1 = 45^\circ \quad \hat{=} \quad Y_1 = 1$$

Example radian:

$$Y_1 = \cot(X_1) \quad X_1 = 0,1\text{rad} \quad \hat{=} \quad Y_1 = 9,967$$

In-/Outputs

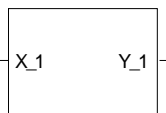
| Name | Type | Description |
|------|-------|---------------------------------|
| X_1 | Float | Angle for cotangent computation |

| Name | Type | Description |
|------|-------|-----------------------------|
| Y_1 | Float | Cotangent of input variable |

Parameter

| ID | Name | Type | Description | Access | Default | Range | Off |
|--------|-------------|-----------------|--|--------|---------|-------|-----|
| Select | Select unit | Enum | Setting: Input value Angle is available as an angular degree [°] or as a radian. | r/w | 0 | | |
| | | Degree of angle | 1 degree is defined as the 360th fraction of a full angle, i.e. 1 full angle = 360° (corresponding to 2 Pi rad). | | 0 | | |
| | | Radian | A full angle comprises 2 Pi radiant: 1 full angle = 2 Pi rad (corresponding to 360°). | | 1 | | |

III-3.5 ARCSIN (Arcus sinus function (No. 84))



ARCSIN

Abb. 362

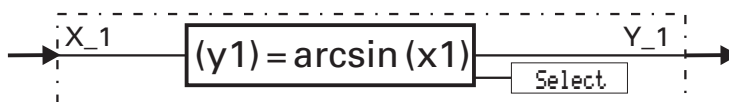


Abb. 363

$$y1 = \arcsin(x1)$$

valid for x1: $-1 \leq x1 \leq +1$

The function provides the arcus sinus of the input value, i.e. X_1 is the angle the arcus sinus of which is calculated. Parameter **Select** is used to adjust, if the angle is provided in degree of angle [°] or in radian. The calculation is output as degree of angle [-90° ... 90°] or as radian $[-\pi/2 \dots +\pi/2]$

With arguments out of the function validity range, output Y_1 is limited to $-1,5 \cdot 10^{37}$

$$(x1 \leq -1) \quad \text{or } 1,5 \cdot 10^{37} \quad (x1 > 1)$$

Example degree of angle:

$$Y_1 = \arcsin(X_1) \quad X_1 = 0,5^\circ \quad \triangleq \quad Y_1 = 30$$

Example radian:

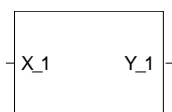
$$X_1 = \arcsin(X_1) \quad X_1 = 1 \text{ rad} \quad \triangleq \quad Y_1 = 1,571$$

| In-/Outputs | | |
|-------------|-------|----------------------------------|
| Name | Type | Description |
| X_1 | Float | Angle for arc cosine computation |

| Name | Type | Description |
|------|-------|-------------------------------|
| Y_1 | Float | Arc cosine of the input value |

| Parameter | | | | | | | |
|-----------|-------------|-----------------|--|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Select | Select unit | Enum | Setting: Input value Angle is available as an angular degree [°] or as a radian. | r/w | 0 | | |
| | | Degree of angle | 1 degree is defined as the 360th fraction of a full angle, i.e. 1 full angle = 360° (corresponding to 2 Pi rad). | | 0 | | |
| | | Radian | A full angle comprises 2 Pi radiant: 1 full angle = 2 Pi rad (corresponding to 360°). | | 1 | | |

III-3.6 ARCCOS (Arcus cosinus function (No. 85))



ARCCOS

Abb. 364



Abb. 365

$$y1 = \arccos(x1)$$

valid for x1: $-1 \leq x1 \leq +1$

The function provides the arcus cosinus of the input value, i.e. X_{-1} is the angle the arcus cosinus of which is calculated. Parameter **Select** is used to adjust, if the angle is provided in degree of angle [°] or in radian.

Calculation is either as degree of angle $[0^\circ \dots 180^\circ]$ or as radian $[0 \dots \pi]$.

With arguments out of the function validity range, output Y_{-1} is set to $1,5 * 10^{37}$ ($X_{-1} \leq -1$) or $-1,5 * 10^{37}$ ($X_{-1} > 1$).

Example degree of angle:

$$Y_{-1} = \arccos(X_{-1}) \quad X_{-1} = 0,5^\circ \quad \triangleq \quad Y_{-1} = 60$$

Example radian:

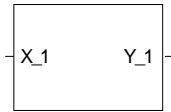
$$Y_{-1} = \arccos(X_{-1}) \quad X_{-1} = 0,5\text{rad} \quad \triangleq \quad Y_{-1} = 1,047$$

| In-/Outputs | | |
|-------------|-------|----------------------------------|
| Name | Type | Description |
| X_1 | Float | Angle for arc cosine computation |

| Name | Type | Description |
|------|-------|-------------------------------|
| Y_1 | Float | Arc cosine of the input value |

| Parameter | | | | | | | |
|-----------|-------------|-----------------|--|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Select | Select unit | Enum | Setting: Input value Angle is available as an angular degree [°] or as a radian. | r/w | 0 | | |
| | | Degree of angle | 1 degree is defined as the 360th fraction of a full angle, i.e. 1 full angle = 360° (corresponding to 2 Pi rad). | | 0 | | |
| | | Radian | A full angle comprises 2 Pi radiant: 1 full angle = 2 Pi rad (corresponding to 360°). | | 1 | | |

III-3.7 ARCTAN (Arcus tangent function (No. 86))



ARCTAN

Abb. 366



Abb. 367

$$y1 = \arctan(x1)$$

The function provides the arcus tangent of the input value, i.e. X_{-1} is the angle the arcus tangent of which is calculated. Parameter **Select** is used to adjust, if the angle is provided in degree of angle [°] or in radian.

The calculation is output either as degree of angle [-90° ... 90°] or as radian $-\pi/2 \dots +\pi/2$.

Example degree of angle:

$$Y_{-1} = \arctan(X_{-1}) \quad X_{-1} = 1 \quad \triangleq \quad Y_{-1} = 45$$

Example radian:

$$Y_{-1} = \arctan(X_{-1}) \quad X_{-1} = 12 \quad \triangleq \quad Y_{-1} = 1,488$$

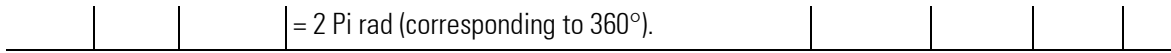
In-/Outputs

| Name | Type | Description |
|------|-------|-----------------------------------|
| X_1 | Float | Angle for arc tangent computation |

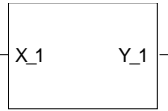
| Name | Type | Description |
|------|-------|--------------------------------|
| Y_1 | Float | Arc tangent of the input value |

Parameter

| ID | Name | Type | Description | Access | Default | Range | Off |
|--------|-------------|-----------------|--|--------|---------|-------|-----|
| Select | Select unit | Enum | Setting: Input value Angle is available as an angular degree [°] or as a radian. | r/w | 0 | | |
| | | Degree of angle | 1 degree is defined as the 360th fraction of a full angle, i.e. 1 full angle = 360° (corresponding to 2 Pi rad). | | 0 | | |
| | | Radian | A full angle comprises 2 Pi radiant: 1 full angle | | 1 | | |



III-3.8 ARCCOT (Arcus cotangent function (No. 87))



ARCCOT
Abb. 368



Abb. 369

$$Y_1 = \text{arccot}(X_1)$$

The function provides the arcus cotangent of the input value, i.e. X_1 is the angle the arcus cotangent of which is calculated. Parameter **Select** is used to adjust, if the angle is provided in degree of angle [°] or in radian.

The calculation is output in degree of angle [0° ... 180°] and in radian[0 ... π].

Example degree of angle:

$$Y_1 = \text{arccot}(X_1) \quad X_1 = 1 \quad \triangleq \quad Y_1 = 45^\circ$$

Example radian:

$$Y_1 = \text{arccot}(X_1) \quad X_1 = -12 \quad \triangleq \quad Y_1 = 3,058$$

| In-/Outputs | | |
|-------------|-------|-------------------------------------|
| Name | Type | Description |
| X_1 | Float | Angle for arc cotangent computation |

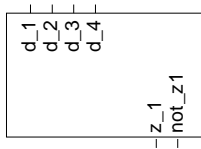
| Name | Type | Description |
|------|-------|----------------------------------|
| Y_1 | Float | Arc cotangent of the input value |

| Parameter | | | | | | | |
|-----------|-------------|-----------------|---|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Select | Select unit | Enum | Setting: Input value Angle is available as an angular degree [°] or as a radian. | r/w | 0 | | |
| | | Degree of angle | 1 degree is defined as the 360th fraction of a full angle, i.e. 1 full angle = 360° (corresponding to 2 | | 0 | | |

| | | | | | | |
|--|--------|---|--|---|--|--|
| | | Pi rad). | | | | |
| | Radian | A full angle comprises 2 Pi radiant: 1 full angle = 2 Pi rad (corresponding to 360°). | | 1 | | |

III-4 Logic functions

III-4.1 AND (AND gate (No. 60))



AND

Abb. 370

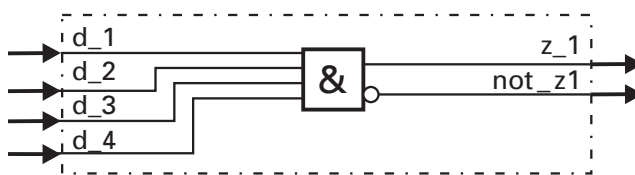


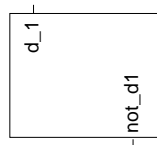
Abb. 371

$$z_1 = d_1 \text{ AND } d_2 \text{ AND } d_3 \text{ AND } d_4$$

Logic function **AND** combines inputs **d_1**...**d_4** according to the truth table given below. Unused inputs are interpreted as logic 1.

| In-/Outputs | | |
|-------------|------|---|
| Name | Type | Description |
| d_1 ... d_4 | Bool | Digital input signal 1 |
| Name | Type | Description |
| z_1 | Bool | Output signal of input signals combined by a logic AND function |
| not_z1 | Bool | Inverted output signal |

III-4.2 NOT (Inverter (No. 61))



NOT

Abb. 372

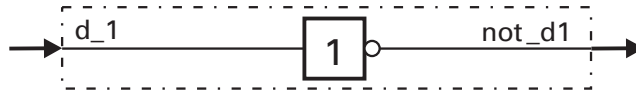


Abb. 373

$$z1 = \overline{d_1}$$

Logic input signal **d_1** is output invertedly at **not_d1**. If **d_1** is not wired, this is interpreted as logic 0.

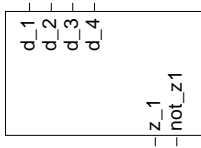
| d_1 | not_d1 |
|-----|--------|
| 0 | 1 |
| 1 | 0 |

In-/Outputs

| Name | Type | Description |
|------|------|-----------------------------|
| d_1 | Bool | Input signal to be inverted |

| Name | Type | Description |
|--------|------|--|
| not_d1 | Bool | Output signal is the inverted input signal |

III-4.3 OR (OR gate (No. 62))



OR

Abb. 374

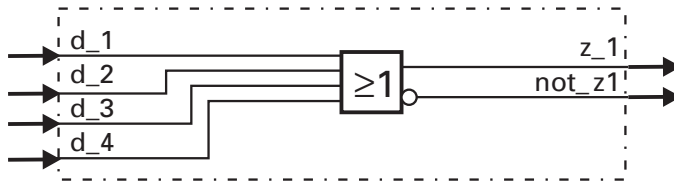


Abb. 375

$$Z_1 = d_1 \text{ OR } d_2 \text{ OR } d_3 \text{ OR } d_4$$

Logic function **OR** combines inputs **d_1 ... d_4** according to the truth table given below. Unused inputs are interpreted as logic 0.

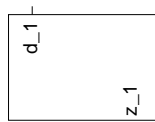
| d1 | d2 | d3 | d4 | z1 | not z1 |
|----|----|----|----|----|--------|
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 |

In-/Outputs

| Name | Type | Description |
|-------------|------|---------------|
| d_1 ... d_4 | Bool | Logic input 1 |

| Name | Type | Description |
|--------|------|--|
| z_1 | Bool | Output signal is the result of the OR function of inputs |
| not_z1 | Bool | Inverted output signal of OR function |

III-4.4 BOUNCE (Debouncer (No. 63))



BOUNCE

Abb. 376

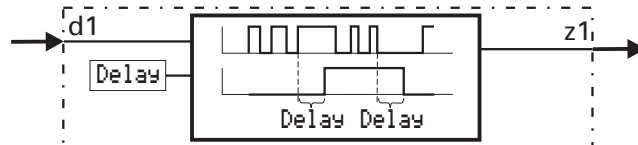


Abb. 377

This function is used for de-bouncing a logic signal. The change of input signal **d_1** is transferred to output **z_1** only, when it remained constant for the time adjusted in parameter **Delay**. The time-out accuracy is dependent of the sampling interval assigned to the function.

Example:

Delay = 0,5s for assignment to

- sampling interval 100ms means that the signal is transferred only after $\geq 0,5$ s.
- Sampling interval 200ms means that the signal is transferred only after $\geq 0,6$ s.
- Sampling interval 400ms means that the signal is transferred only after $\geq 0,8$ s.
- Sampling interval 800ms means that the signal is transferred only after $\geq 0,8$ s.

In-/Outputs

| Name | Type | Description |
|------|------|--|
| d_1 | Bool | Input signal. A change is transmitted to the output only, when it remained constant during the time adjusted in the parameter. |

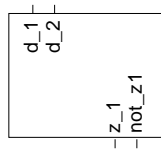
| Name | Type | Description |
|------|------|------------------------|
| z_1 | Bool | Debounced input signal |

Parameter

| ID | Name | Type | Description | Access | Default | Range | Off |
|-------|-------|-------|---|--------|---------|-------|-----|
| Delay | Delay | Float | Switch-on and off delay time in seconds (separate | r/w | 0.0 | >0.0 | |

| | | | | | | |
|--|--|--|--|--|--|--|
| | | adjustment for switch-on and off is not possible). | | | | |
|--|--|--|--|--|--|--|

III-4.5 EXOR (Exclusive OR gate (No. 64))



EXOR

Abb. 378

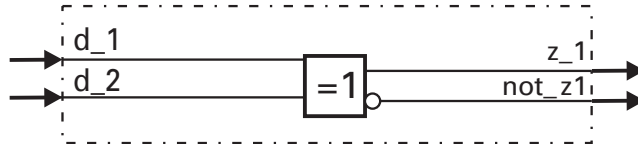


Abb. 379

$$z_1 = d_1 \text{ EXOR } d_2$$

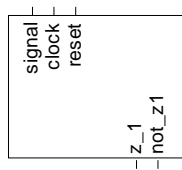
Logic inputs **d_1** and **d_2** are combined into **z_1** according to the truth table given below. Unused inputs are interpreted as logic 0. Output **z_1** is 0, when the two inputs are equal (both 0 or both 1).

| d_1 | d_2 | z_1 | not_z1 |
|-----|-----|-----|--------|
| 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

In-/Outputs

| Name | Type | Description |
|--------|------|---|
| d_1 | Bool | Logic input 1 |
| d_2 | Bool | Logic input 2 |
| Name | Type | Description |
| z_1 | Bool | Output signal = 0, when the two inputs are equal (both 0 or both 1). |
| not_z1 | Bool | Inverted output signal = 1, when the two inputs are equal (both 0 or both 1). |

III-4.6 FLIP (D flipflop (No. 65))



FLIP

Abb. 380

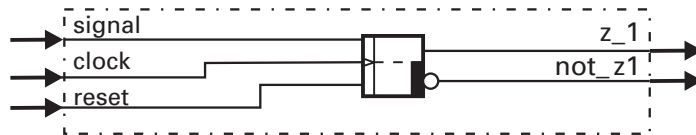


Abb. 381

The digital signal status at static input **signal** is transferred to output **z_1** when the signal at **clock** input clock changes from 0 to 1 (positive flank), and when input **reset** is logic 0. With **reset** = 1, output **z_1** is forced to 0 independent of inputs **signal** and **clock**. **reset** has priority!

Input signals **signal**, **clock** and **reset** must be available at least for the duration of sampling interval T_r selected for this block.

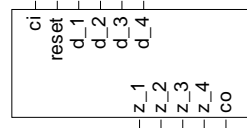
In the switch-on status (initial condition), $z_1 = 0$! Unused inputs are interpreted as logic 0.

In-/Outputs

| Name | Type | Description |
|--------|------|---|
| signal | Bool | D input - This signal is output to Z_1 via input clock with the positive flank (0 to 1), when input reset is not 1. |
| clock | Bool | Clock input - A positive flank transfers the instantaneous status at input signal to output Z_1, when input reset is not 1. |
| reset | Bool | Reset input - sets output z_1 to 0 |

| Name | Type | Description |
|--------|------|--|
| z_1 | Bool | Flipflop output |
| not_z1 | Bool | Flipflop output NOT z1, inverted output signal |

III-4.7 FLIPM (Multi D flipflop (No. 128))



FLIPM

Abb. 382

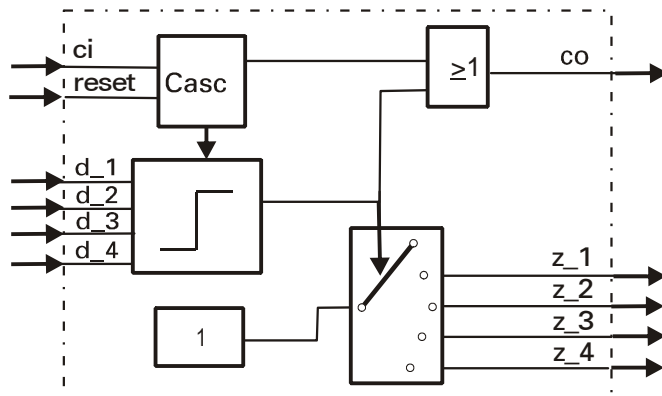


Abb. 383

FlipM is a multi-flipflop with 4 data inputs and outputs.

Depending on operating mode, **FlipM** stores the first falling or rising edge per cycle, which is applied to one of the data inputs (**d_1** ... **d_4**). Unless a flank is detected, the outputs remain unchanged. With simultaneous signal change at several inputs, the output pertaining to the input with the lowest number is set.

The response of input **Reset** is level-sensitive: As long as the input is activated, all outputs are deleted.

The Parameters / modes:

Inverse (Action = 0): The block reacts on the negative flank (1 → 0) at its input.

Direct (Action = 1): The block reacts on the positive flank (0 → 1) at its input.

Additionally, a defined start-up and reset behaviour can be assigned to the block. This is done via parameter **OutputStart**. After start-up or reset, the output selected using the value in **OutputStart** is set to TRUE (=1).

More than 4 inputs:

Several **FlipM** blocks can be synchronized via the cascade inputs. When an input signal changes, the one related output is set, all other outputs of the cascade are deleted. For the cascade, the blocks must be connected into a closed chain, i.e. the last cascade output must be connected to the first cascade input.

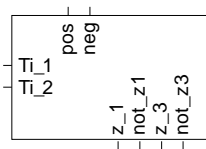
The blocks must be soft-wired according to block numbers. If **FlipM** blocks are cascaded, only 1 block per cycle is active: the one which was the first to detect a flank at the input. After start-up, the first block with **OutputStart** set is activated.

| In-/Outputs | | |
|----------------|------|--|
| Name | Type | Description |
| ci | Bool | Cascade input for FlipM. For synchronizing several FlipM blocks (only 1 output of all blocks is switched on), these must be wired into a circle. |
| reset | Bool | Deletes the outputs. |
| d_1 ... d_4 | Bool | Data input 1 - is output to d_1 at positive flank (0 to 1), the other data outputs are deleted. |

| Name | Type | Description |
|----------------|------|--|
| z_1 ... z_4 | Bool | Data output 1. Only one data output per FlipM block or synchronized FlipM blocks is switched on. |
| co | Bool | Cascade input for FlipM. For synchronizing several FlipM blocks (only 1 output of all blocks is switched on), these must be wired into a circle. |

| Parameter | | | | | | | |
|-------------|------------------|---------|--|--------|---------|---------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| OutputStart | Initial status | Int | Selection of the output which should be switched on at start-up or reset. | r/w | off | 1 ... 4 | ja |
| Action | Edge recognition | Enum | Adjust flank detection: function at positive flank (0 to 1) or at negative flank (1 to 0) at a data input. | r/w | 0 | | |
| | | Inverse | The block responds to a positive flank (0 -> 1) at its input. | | 0 | | |
| | | Direct | The block responds to a negative flank (1 -> 0) at its input. | | 1 | | |

III-4.8 MONO (Monoflop (No. 66))



MONO

Abb. 384

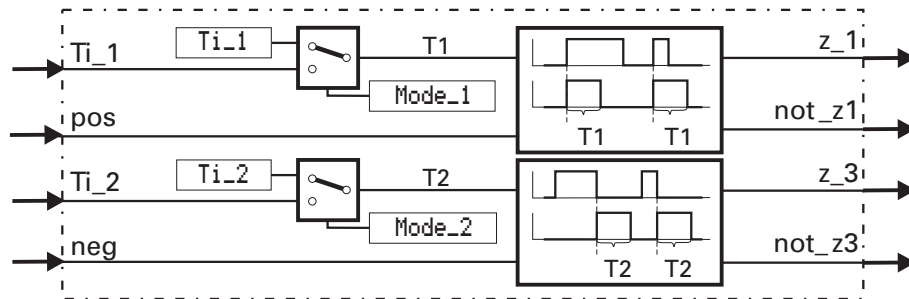


Abb. 385

The function generates a positive pulse of length **Ti1** at output **z_1**, when a positive flank at trigger input **pos** is detected. It generates a positive pulse of length **Ti2**, at output **z_3**, when a negative flank at trigger input **neg** is detected.

Pulse duration **Ti** is adjusted either as parameter **Ti** or read in via inputs **Ti**. The origin of pulse duration is selected via parameter **mode**.

The duration of an output pulse is matched to the new values with changes at inputs **Ti1/Ti2**. With input values **Ti1/Ti2** ≤ 0, the pulse is output for the duration of one scanning cycle.

The function is re-triggerable. I.e., if a new trigger condition is detected during a pulse output, the remaining pulse time to be output is prolonged to a full pulse length.

The pulse duration accuracy is dependent of the sampling time, which is assigned to the function.

Example:

Ti = 0,9s for assignment to

- sampling interval 100ms means that the signal is output during = 0,9s.
- Sampling interval 200ms means that the signal is output during = 1,0s.
- Sampling interval 400ms means that the signal is output during = 1,2s.
- Sampling interval 800ms means that the signal is output during = 1,6s.

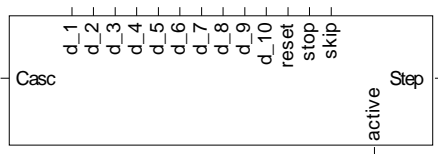
| In-/Outputs | | |
|-------------|-------|---|
| Name | Type | Description |
| Ti_1 | Float | Duration Ti1 [s] of the pulse generated by a flank at input pos, when the input is configured as a source of the pulse duration (Mode_1 = Input Ti1). The accuracy of the pulse duration is dependent on the sampling interval of the function. |
| Ti_2 | Float | Duration Ti2 [s] of the pulse generated by a flank at input neg, when the input is configured as a source of the pulse duration (Mode_2 = Input Ti2). The accuracy of the pulse duration is dependent on the sampling interval of the function. |

| | | |
|-----|------|---|
| pos | Bool | Trigger input: pulse generation at z_1 and not_z1 with positive flank (0 to 1). |
| neg | Bool | Trigger input: pulse generation at z_3 and not_z3 with negative flank (1 to 0). |

| Name | Type | Description |
|--------|------|---|
| z_1 | Bool | Positive pulse of duration Ti_1, when a positive flank at the input pos was detected. |
| not_z1 | Bool | Negative pulse of duration Ti_1, when a positive flank at the input pos was detected. |
| z_3 | Bool | Positive pulse of duration Ti_2, when a negative flank at the input neg was detected. |
| not_z3 | Bool | Negative pulse of length Ti_2, when a negative flank at the input neg was detected. |

| Parameter | | | | | | | |
|-----------|------------------|---------------|---|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Mode_1 | Source of time 1 | Enum | Source of the pulse duration for output z_1 is the parameter Ti_1 or the input Ti_1. | r/w | 0 | | |
| | | Parameter Ti1 | The length of the positive pulse at the z_1 output is adjustable using parameter Ti1. | | 0 | | |
| | | Input Ti1 | The length of the positive pulse at the z_1 output is read by input Ti1. | | 1 | | |
| Mode_2 | Source of time 2 | Enum | Source of the pulse duration for output z_3 is the parameter Ti_2 or the input Ti_2. | r/w | 0 | | |
| | | Parameter Ti2 | The length of the positive pulse at the z_3 output is adjustable using parameter Ti2. | | 0 | | |
| | | Input Ti2 | The length of the positive pulse at the z_3 output is ready by input Ti2. | | 1 | | |
| Ti1 | Pulse duration 1 | Float | Duration Ti1 [s] of the pulse generated by pos, if the parameter is configured as a source of the pulse duration (Mode_1 = Parameter Ti_1). The accuracy of the pulse duration is dependent on the sampling interval of the function. | r/w | 1.0 | >0.1 | |
| Ti2 | Pulse duration 2 | Float | Duration Ti2 [s] of the pulse generated by neg, if the parameter is configured as a source of the pulse duration (Mode_2 = Parameter Ti_1). The accuracy of the pulse duration is dependent on the sampling interval of the function. | r/w | 1.0 | >0.1 | |

III-4.9 STEP (Step function for sequencing (No. 68))



STEP

Abb. 386

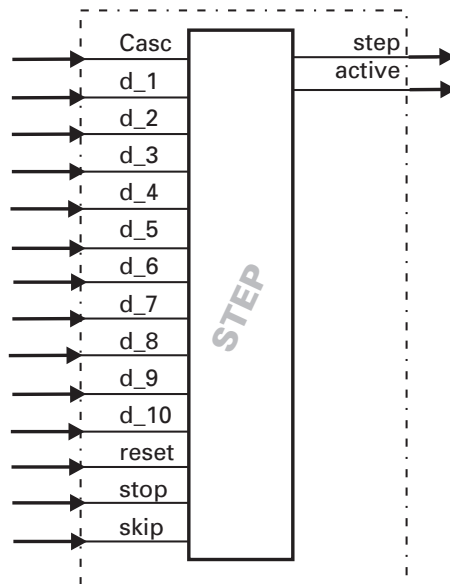


Abb. 387

The **STEP** function realizes the individual steps for sequencing.

The function starts with **RESET** at step 1 and remains at this step, until the relevant condition input **d_1** or the **skip** input is set from 0 to 1. This is followed by switch-over to step 2. The procedure for all further steps is identical. At output **Step** the step number is output as a value.

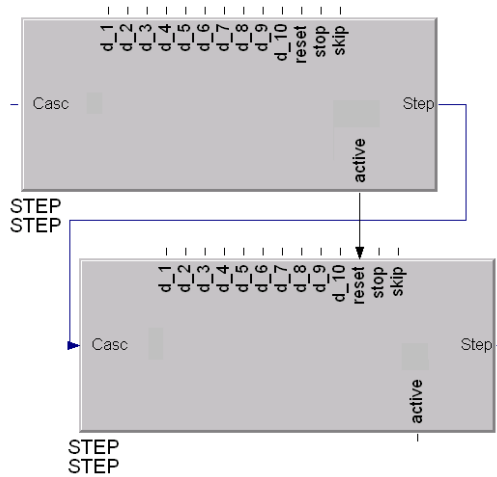
Example:

Switch-over from step 3 (**Step** = 3) to step 4 (**Step** = 4) is only after the condition at **d_3** was met (**d_3** = 1). The condition at **d_4** is checked only when calling up the function for the next time. Thus immediate switch-over is prevented. As long as **d_3** = 0, the value of output **Step** remains 3.

Alternatively, a positive flank at input **skip** also leads to switch-over to the next step (independent of the status at input **d_1**..**d_10**).

When several switch-over conditions are 1 simultaneously (e.g. **d_1**, **d_2**, **d_3**, **d_4** and **d_5**), only the instantaneously effective input is handled. I.e. in each calculation cycle, switch-over is only by one step. For realizing a sequencing with more than 10 steps, the **STEP** function can be cascaded:

The wiring example shows how 2 **STEP** functions are cascaded. With cascading, step number 1...n is output always as a value at output **Step** of the last follow-up step.



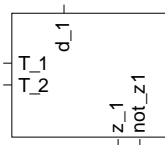
For resetting the cascaded stepping control, reset wiring at the 1st function block is required.

Abb. 388

| In-/Outputs | | |
|-------------|-------|--|
| Name | Type | Description |
| Casc | Float | Cascade input. At the first STEP function of a cascade this input must not be connected. The RESET condition at the first STEP switches the entire chain to step 1. |
| d_1 ... d10 | Bool | Condition input 1 for switching over to the next step (step 2) |
| reset | Bool | reset = 1: output STEP is set to 1 (only with individual function or at the first step of a cascade). With the following steps (cascaded), the output step of the following step is set equal to the input value at casc by reset. reset has the highest priority of all digital inputs. |
| stop | Bool | stop = 1: the function block remains in the current step, the outputs step and active remain unchanged, unless reset is switched to 1. |
| skip | Bool | This input reacts only to a positive flank (0 to 1). At this flank the STEP function switches over to the next step without taking the status at the relevant input d_1 to d_10 into account. |
| Name | Type | Description |
| Step | Float | Indicates the current step of the STEP function. With cascading, the value at Casc is added to this value. |
| active | Bool | active = 1: indicates that the STEP function is still in the active status or in reset. active = 0: indicates that the STEP function has elapsed. |

No parameters!

III-4.10 Delay_D (Timer (No. 69))



DELAY_D

Abb. 389

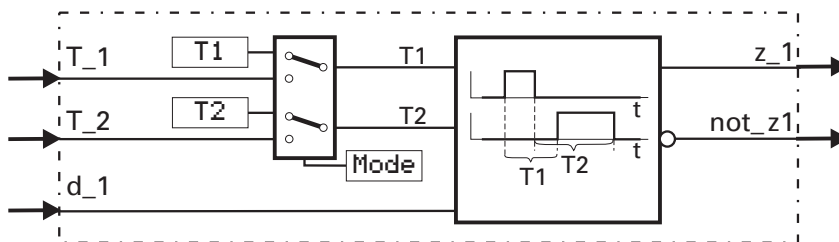


Abb. 390

The function outputs the change of signal status at **d_1** with a delay at **z_1**.

The delay time can be adjusted separately for each change direction of the signal status! (positive and negative flank).

With change from 0 to 1 at input **d_1**, output **z_1** is switched to 1 with a delay of time **T_1**. With change from 1 to 0 at input **d_1**, output **z_1** is switched to 0 with a delay of time **T_2**.

Time **T_1** is adjusted either as parameter **T_1** or read in via input **T_1**.

Time **T_2** is adjusted either as parameter **T_2** or read in via input **T_2**.

The time origin is selected via parameter Mode.

In-/Outputs

| Name | Type | Description |
|------|-------|--|
| T_1 | Float | Delay time T_1 [s], by which the positive signal of d_1 is delayed, when the inputs are configured as a source of the delay times. |
| T_2 | Float | Delay time T_2 [s], by which the negative signal of d_1 is delayed, when the inputs are configured as a source of the delay times. |
| d_1 | Bool | Input signal, is output with delay at z_1 and negatedly at output not_z1. |

Name Type Description

| | | |
|--------|------|-----------------------------------|
| z_1 | Bool | Delayed input signal d_1 |
| not_z1 | Bool | Inverted delayed input signal d_1 |

Parameter

| ID | Name | Type | Description | Access | Default | Range | Off |
|-----|------|-------|---|--------|---------|-------|-----|
| T_1 | T 1 | Float | Delay time T_1 [s], by which the positive signal of d_1 is delayed, when the parameters are configured as a | r/w | 0.0 | >0.0 | |

| | | | | | | | |
|-----|-----|-------|--|-----|-----|------|--|
| | | | source of the delay times. | | | | |
| T_2 | T 2 | Float | Delay time T_2 [s], by which the negative signal of d_1 is delayed, when the parameters are configured as a source of the delay times. | r/w | 0.0 | >0.0 | |

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|------|-----------------|-----------------|---|--------|---------|-------|-----|
| Mode | Source of delay | Enum | Define, if the delay times are determined by parameters or via the analog inputs. | r/w | 0 | | |
| | | Parameter T1/T2 | The delay is set by parameters T_1 and T_2. | | 0 | | |
| | | Inputs T1/T2 | The delay is determined by inputs T_1 and T_2. | | 1 | | |

The pulse duration accuracy is dependent of the time group to which the function is assigned. It is an integer multiple of the sampling interval adjusted for this block.

Example:

T1 = 0,7s with assignment to

- sample time 100ms means, delay time of the positive flank is 0,7s.
- sample time 200ms means, delay time of the positive flank is 0,8s.
- sample time 400ms means, delay time of the positive flank is 1,2s.
- sample time 800ms means, delay time of the positive flank is 1,6s.

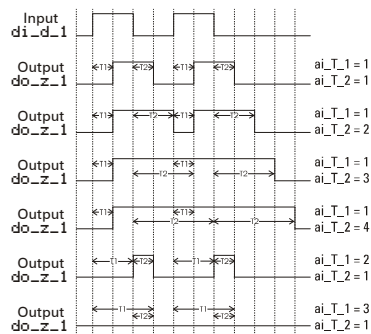
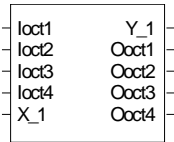


Abb. 391 Examples with different delay times T1 and T2

III-5 Signal converters

III-5.1 AOCTET (Data type conversion (No. 02))



AOCTET

Abb. 392

Function **AOCTET** converts an analog value (**X_1**) into the individual bytes (**loct1-4**) of a data type. The function works in both directions simultaneously (analog > bytes / bytes > analog) with separate data type adjustment in the parameters.

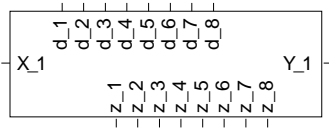
| In-/Outputs | | |
|-----------------|-------|---------------------------|
| Name | Type | Description |
| loct1 ... loct4 | Float | Analog input byte value 1 |
| X_1 | Float | Analog input value |

| Name | Type | Description |
|-----------------|-------|----------------------------|
| Y_1 | Float | Analog output value |
| Ooct1 ... Ooct4 | Float | Analog output byte value 1 |

| Parameter | | | | | | | |
|-----------|---------------|---------------------------|--|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| loct | Input format | Enum | Data type of analog-into-byte conversion | r/w | 0 | | |
| | | UInt8 | Unsigned 8-bit integer, without sign | | 0 | | |
| | | Int8 | 8-bit integer, with sign | | 1 | | |
| | | UInt16 | Unsigned 16-bit integer, without sign | | 2 | | |
| | | Int16 | 16-bit integer, with sign | | 3 | | |
| | | UInt32 | Unsigned 32-bit integer, without sign | | 4 | | |
| | | Int32 | 32-bit integer, with sign | | 5 | | |
| Ooct | Output format | Float | Floating point value | | 6 | | |
| | | Enum | Data type of byte-into-analog conversion | r/w | 0 | | |
| | | UInt8 | Unsigned 8-bit integer, without sign | | 0 | | |
| | | Int8 | 8-bit integer, with sign | | 1 | | |
| | | UInt16 | Unsigned 16-bit integer, without sign | | 2 | | |
| | Int16 | 16-bit integer, with sign | | 3 | | | |

| | | | | | |
|--|--------|---------------------------------------|---|--|--|
| | Uint32 | Unsigned 32-bit integer, without sign | 4 | | |
| | Int32 | 32-bit integer, with sign | 5 | | |
| | Float | Floating point value | 6 | | |

III-5.2 ABIN (Analog <=> binary conversion (No. 71))



ABIN

Abb. 393

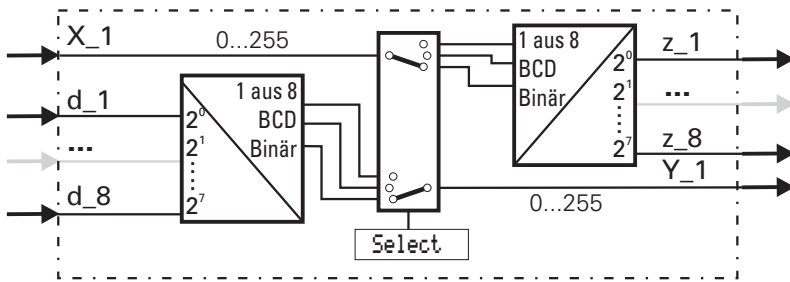


Abb. 394

Analog input variable X_1 is converted into a binary number, a BCD number or a selection "1 out of 8". Thereby, X_1 is always rounded off (down for values $< 0,5$, up for values $\geq 0,5$).

Simultaneously, binary input values $d_1 \dots d_8$ (considered as a binary number or a BCD number) can be converted into an analog output variable. The conversion mode is determined by configuration parameter **Select**.

Analog/binary conversion - binary/analog conversion (**Select** = **ana<->bin**)

Conversion analog value into binary number:

The analog input value at x_1 is converted into a binary variable, which is output in binary form at outputs $z_1 \dots z_8$ ($z_1=2^0 \dots z_8=2^7$). The range is within 0...255.

Out of the range, the output allocation is:

| Input | z_1 | z_2 | z_3 | z_4 | z_5 | z_6 | z_7 | z_8 |
|----------------|-----|-----|-----|-----|-----|-----|-----|-----|
| $X_1 \leq 0$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $X_1 \geq 255$ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Conversion binary number into analog value

A binary number at digital inputs $d_1 \dots d_8$ ($d_1 = 2^0 \dots d_8 = 2^7$) is converted into an analog output variable and output at analog output y_1 . The range is within 0...255.

BCD - conversion (Select = ana<->BCD) Converting a value into a BCD number

The analog input value at x_1 (range 0...99) is output as a BCD number at outputs $z_8 \dots z_5$ and $z_4 \dots z_1$.

Example: $X_1 = 83 = 83 \rightarrow$ the output allocation is:

| Input | z_1 | z_2 | z_3 | z_4 | z_5 | z_6 | z_7 | z_8 |
|----------|-------|-----|-----|-------|-------|-----|-----|-------|
| | 2^0 | | | 2^3 | 2^0 | | | 2^3 |
| X_1 = 83 | 1 | 1 | 0 | 0 | | 0 | 0 | 1 |
| BCD | 3 | | | | 8 | | | |

Out of the range, the output allocation is:

| Inputs | z_1 | z_2 | z_3 | z_4 | z_5 | z_6 | z_7 | z_8 |
|----------|-----|-----|-----|-----|-----|-----|-----|-----|
| X_1 ≤ 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | | | | 0 | | | |
| X_1 ≥ 99 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| | 9 | | | | 9 | | | |

Converting a BCD number into an analog value

BCD input values at inputs d_1...d_4 und d_5...d_8 are converted into a floating point number and available at output Y_1. With a BCD number > 9 at inputs d_1...d_4 or d_5...d_8, output variable Y_1 is limited to 9.

Out of the range, the output allocation is:

| Outputs | d_1 | d_2 | d_3 | d_4 | d_5 | d_6 | d_7 | d_8 |
|---------|-------|-----|-----|-------|-------|-----|-----|-------|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 2^0 | | | 2^3 | 2^0 | | | 2^3 |
| Y_1 = | 0 | | | | 0 | | | |
| | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Y_1 = | 9 | | | | 9 | | | |

Converting a value into selection "1 out of 8" (Select = ana<->1/8)

An analog input value at X_1 (range 0...8) selects none or one of the 8 outputs z_1...z_8.

Example for conversion value (x1 = 5) into selection:

| Input | z_1 | z_2 | z_3 | z_4 | z_5 | z_6 | z_7 | z_8 |
|---------|-----|-----|-----|-----|-----|-----|-----|-----|
| X_1 = 5 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

Out of the range, the output allocation is:

| Input | z_1 | z_2 | z_3 | z_4 | z_5 | z_6 | z_7 | z_8 |
|---------|-----|-----|-----|-----|-----|-----|-----|-----|
| X_1 ≤ 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| X_1 ≥ 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Conversion Selection "1 out of 8" into analog value (Select = ana<->1/8)

Individual digital input allocation d_1...d_8 result in an analog output variable at Y_1 according to the allocated input value.

Example for conversion value ($x_1 = 5$) into selection:

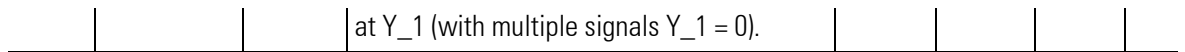
| Output | z_1 | z_2 | z_3 | z_4 | z_5 | z_6 | z_7 | z_8 |
|---------|-----|-----|-----|-----|-----|-----|-----|-----|
| Y_1 = 5 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

If more than one of inputs $d_1 \dots d_8$ is active, output variable Y_1 is set to 0.

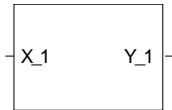
| Inputs/outputs | | |
|----------------|-------|---|
| Name | Type | Description |
| X_1 | Float | Analog input for binary value, BCD value or 1-out-of-8 selection |
| d_1 ... d_8 | Bool | Digital input 1 for binary value, BCD value or 1-out-of-8 selection |

| Name | Type | Description |
|-------------|-------|---|
| Y_1 | Float | Converted analog value |
| z_1 ... z_8 | Bool | Converted binary value, BCD value or selected value |

| Configuration: | | | | | | | |
|----------------|-------------------|-------------------------|--|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Select | Select conversion | Enum | Type of conversion either between analog and binary values (and vice versa), or between analog and BCD-coded values (and vice versa), or 1 out of 8 selection. | r/w | 0 | | |
| | | Analog <-> Binary | Range 0...255. The analog input value at X_1 is converted into an integer variable which is output in binary code at the outputs z_1...z_8 ($z_1=2^0 \dots z_8 = 2^7$). A binary value at the digital inputs d_1...d_8 ($d_1 = 2^0 \dots d_8 = 2^7$) is converted into an analog output variable and output at the analog output Y_1. | | 0 | | |
| | | Analog <-> BCD | Range 0...99. The analog input value at X_1 is output as a BCD-coded value at the outputs z_8...z_5 and z_4...z_1. BCD input values at the inputs d_1...d_4 and d_5...d_8 are converted into a floating point value and output at Y_1. With a BCD value > 9 applied to the inputs d_1...d_4 respectively d_5...d_8, the output variable Y_1 is limited to 9. | | 1 | | |
| | | Analog <-> 1 of 8 | Range 0...8. An analog input value at X_1 selects none or one of the 8 outputs z_1...z_8. Setting one of the digital inputs d_1...d_8 produces the number of the relevant input as an analog output variable | | 2 | | |



III-5.3 TRUNC (Integer portion (No. 72))



TRUNC

Abb. 395

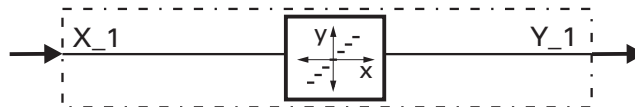


Abb. 396

$$Y_1 = \text{INT}(X_1)$$

The function provides the integer portion (integer) of input variable X_1 without rounding off at output Y_1 .

Example:

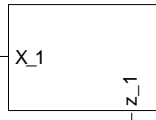
$$X_1 = 1,7 \quad Y_1 = 1,0$$

$$X_1 = -1,7 \quad Y_1 = -1,0$$

| In-/Outputs | | |
|-------------|-------|--|
| Name | Type | Description |
| X_1 | Float | Input variable to be handled |
| Name | Type | Description |
| Y_1 | Float | Integer portion of input variable X_1 without rounding off |

No parameters!

III-5.4 PULS (Analog pulse-conversion (No. 73))



PULS

Abb. 397

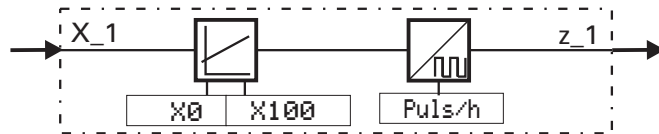


Abb. 398

$$n = Puls/h \cdot \frac{X_1 - X_0}{X_{100} - X_0}$$

n = Number of pulses per hour z_1
 X_0 = Parameter
 X_{100} = Parameter
 X_1 = Analog input

Input variable X_1 is converted into a number of pulses per hour. Parameter $Puls/h$ is used for selecting the maximum number of pulses at $X_1 \geq X_{100}$. For $X_1 \leq X_0$ no pulses are output.

Within range X_0 und X_{100} , input value X_1 is converted linearly into $Puls/h$.

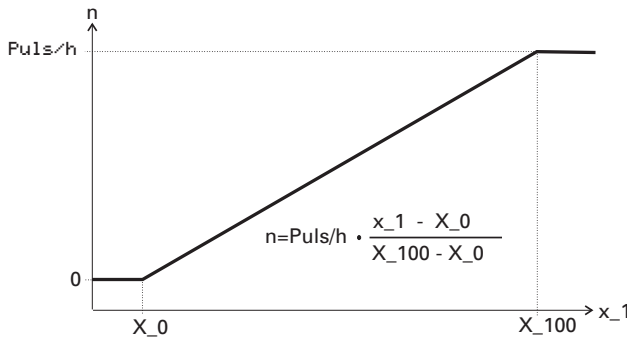


Abb. 399

- $Puls/h$ = max. Pulse number/h
- X_0 = 0% of Pulse/h
- X_{100} = 100% of Pulse/h

The parameter settings result in a straight line between 0 and 3600 pulses /h according to input X_1 . The pulse length corresponds to the sampling interval adjusted for this block. The length of switch-off time between pulses is not always equal and dependent of the configured sampling interval.

The sampling interval allocation also determines the maximum number of pulses/hour, which can be realized. If higher values than can be output due to the sampling interval are entered in parameter $Puls/h$, limiting is to the maximum possible number of pulses.

| Maximum number of pulses: | | |
|---------------------------|---|-----------------|
| 100 ms | = | 18 000 pulses/h |
| 200 ms | = | 9 000 pulses /h |
| 400 ms | = | 4 500 pulses /h |
| 800 ms | = | 2 250 pulses /h |

| In-/Outputs | | |
|-------------|-------|---|
| Name | Type | Description |
| X_1 | Float | Input variable X_1 is converted into a number of pulses per hour. |

| Name | Type | Description |
|------|------|--------------|
| z_1 | Bool | Pulse output |

| Parameter | | | | | | | |
|-----------|-------------|-------|---|--------|---------|-----------------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| X_0 | Scale start | Float | Span start (0 % of pulses/h). When the input value X_1 is smaller or equal, no pulses are generated (span start, creep flow suppression). | r/w | 0.0 | | |
| X_100 | Scale end | Float | Span end (100 % of pulses/h). When the input value X_1 is higher or equal, the number of pulses remains constant at Puls_h | r/w | 1.0 | >0.001 | |
| Puls_h | Pulses/h | Float | Maximum number of output pulses per hour, for input value X_1 higher than or equal to span end X_100. The switch-off times between the pulses are not always equally long and dependent on configured sampling interval. The assignment of the sampling interval also determines the maximum number of pulses per hour which can be realized. | r/w | 0.0 | 0.0 ... 18000.0 | |

$$n = Puls/h \cdot \frac{X_1 - X_0}{X_{100} - X_0}$$

Equation for calculating the momentary impulse number of n per hour

n = momentary impulse number/hour

X_0 = Parameter. With analog input $X_1 \leq X_0$ no pulses are produced (area start, creeping flow suppression)

X_100 = Parameter. If the analog input $X_1 \leq X_{100}$ remains $n = \text{constant} = Puls/h$

Puls/h = Parameter. Pulse number/hour for analog input $X_1 = X_{100}$

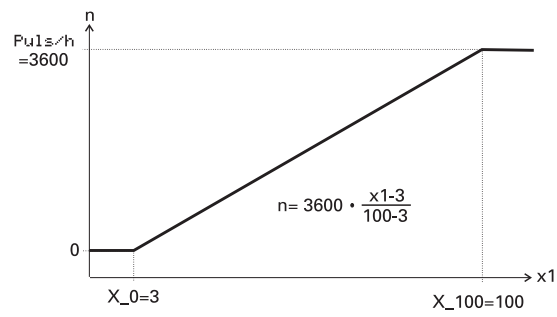


Abb. 400

Example:

$$X_{100} = 3 \dots 100\% \cong 0 \dots 3600/\text{h}$$

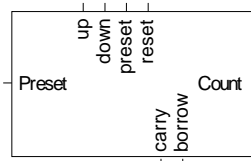
$$X_{0} = 3$$

$$X_{100} = 100$$

$$\text{Puls/h} = 3600$$

$$\text{sampling period} \leq 400 \text{ ms}$$

III-5.5 COUNT (Up/down counter (No. 74))



COUNT

Abb. 401

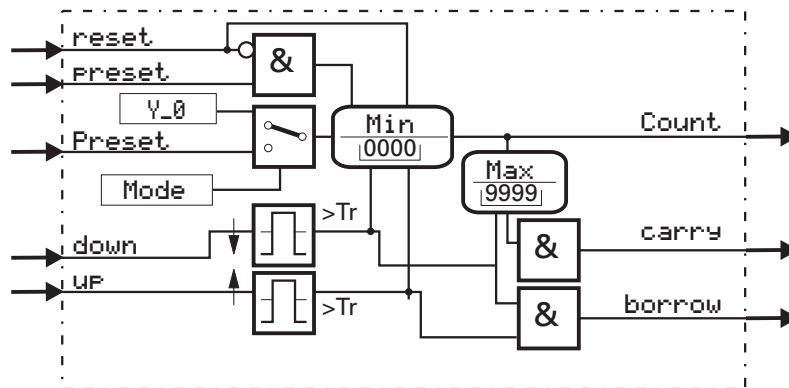


Abb. 402

'COUNT' is an up/down counter and counts the positive flanks at input **UP** or **down**.

| reset | Preset | Mode |
|-------|--------|---|
| 0 | 0 | GO (Default) : COUNT counts. |
| 0 | 1 | Preset : output Count is set to Preset value |
| 1 | 0 | Reset (First Run) : output Count is set to Min. value |
| 1 | 1 | Reset (First Run) : output Count is set to Min. value |

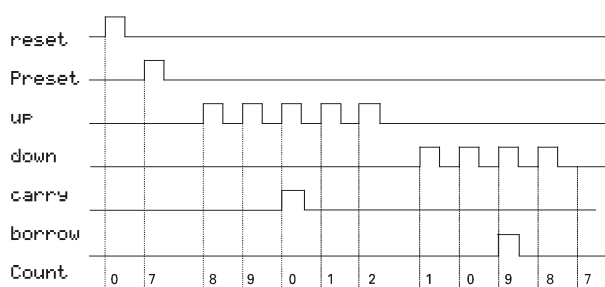


Abb. 403

Pulse diagram of the up/down counter:
 "up, down, carry und borrow" are 1 in active condition

Example: Max. -limit = 9;
 Min. -limit = 0; Preset = 7.

An unwired clock input is set to value 0 internally. If both clock inputs go from 0 to 1 signal simultaneously, counting is omitted.

If parameters for the **Min-** or **Max** limit are changed during operation, the counter can be out of this new range. In order to prevent faulty functions, the counter must be set to a new, defined output status with '**reset**' or '**Preset**'.

Function up counter:

At each positive flank (0 → 1) at input **up**, output **Count** is increased by 1, until the **Max.** limit is reached. With the next pulse, output **Count** returns to the **Min.** value, and carry output **carry** is set to 1 for the duration of the applied pulse. **COUNT** continues counting with the next pulses.

Function down counter:

With each positive flank (0 → 1) at input **down**, output **Count** is decreased by 1, until the **Min.** limit is reached. With the next pulse, output **Count** returns to the **Max.** value, and borrow output **borrow** is set to 1 for the duration of the applied pulse. **COUNT** continues counting down with the next pulses.

Function reset:

A 1 signal at input **reset** has priority over all other inputs. **reset** resets the **COUNT** to the **Min.** value.

Function preset:

A 1 signal at input **Preset**-has priority over inputs **up** and **down**. **Preset** resets the **COUNT** to the **Preset** value.

The origin of the **Preset** value is selected with parameter **Mode**.

Mode = Para.y0 means that the **Preset** value corresponds to parameter **Y_0**.

Mode = InpPreset means that the **Preset** value corresponds to analog input **Preset**.

With a **Preset** value higher than the **Max.** limit, output **COUNT** is set to the **Max.** limit. A **Preset** value smaller than the **Min.** limit is set to the **Min.** limit.

A **Preset** value which is not an integer is rounded off.

In-/Outputs

| Name | Type | Description |
|--------|-------|---|
| Preset | Float | Input for the preset value |
| up | Bool | Input for counting up |
| down | Bool | Input for counting down |
| preset | Bool | Input for the preset mode - output Count goes to value Preset. preset has priority over inputs up and down. |
| reset | Bool | Input for the reset mode - output Count goes to value Min. |

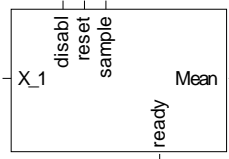
| Name | Type | Description |
|--------|-------|------------------------------|
| Count | Float | Count output |
| carry | Bool | Carry output (clock - up) |
| borrow | Bool | Borrow output (clock - down) |

Parameter

| ID | Name | Type | Description | Access | Default | Range | Off |
|------|-----------|------|-----------------------------------|--------|---------|-------|-----|
| Mode | Source of | Enum | Source of the preset value is the | r/w | 0 | | |

| | | | | | | | |
|-----|---------------|--------------|---|-----|-----|--|--|
| | preset | | preset input or parameter Y_0. | | | | |
| | | Parameter y0 | preset value from parameter | | 0 | | |
| | | Input Preset | preset value from input. When the preset value is higher than the Max limit, output COUNT is set to the Max limit. When the preset value is smaller than the Min limit, it is set to the min limit. Unless the preset value is a whole number, it is rounded off. | | 1 | | |
| y_0 | Preset value | Float | Preset value | r/w | 0.0 | | |
| Max | Maximal value | Float | Maximum limit of the counter | r/w | MAX | | |
| Min | Minimal value | Float | Minimum limit of the counter | r/w | 0.0 | | |

III-5.6 MEAN (Mean value formation (No. 75))



MEAN

Abb. 404

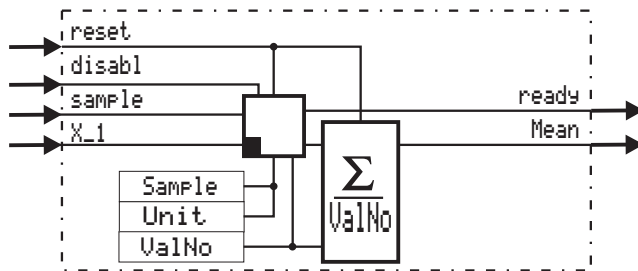


Abb. 405

General

Function **MEAN** forms the floating, arithmetic mean value of the number (**ValNo**) of the values detected last at input **X_1** for output at output **Mean**.

The interval between the individual samplings (interval) is adjustable with **Sample** and **Unit**.

The **Unit** is used to specify the measurement interval (sec = seconds, min = minutes or h = hours).

The **Sample** is used to specify the number of "**Unit**" intervals for measurement.



Hint!

With the **SAMPLE** input wired, the adjusted sample and unit parameter are ineffective.
Only the sample-pulse is used.

Example 1: mean value of the past minute with sampling per second.

Sample = 1 and **Unit** = sec → value sampling per second.

ValNo = 60 → the past 60 values form the mean value (1 minute).

Example 2: mean value of the past day with sampling per hour.

Sample = 1 and **Unit** = h → value sampling per hour.

ValNo = 24 → the past 24 values for the mean value (1 day).

Example 3: mean value of the past day with sampling per quarter of an hour.

Sample = 15 and **Unit** = min → value sampling at intervals of 15 minutes.

ValNo = 96 → the past 96 values form the mean value (1 day).



Hint!

If the **SAMPLE** input is wired, sampling is triggered by a positive flank at this input.
The adjusted sampling interval is invalid.

With **disabl** = 1 the sampling is interrupted, **reset** = 1 deletes the mean value.

Internal calculation:

The number of input values entered in **ValNo** is stored, totaled and divided by the number.

$$Y1 = \frac{Value_1 + Value_2 + Value_3 + \dots + Value_n}{n}$$

Example: ValNo = 5

x1= 11 24 58 72 12

di_reset

Analog input **Mean** outputs value 0 for the duration of the applied reset signal.

The stored values are deleted.

Example: ValNo = 5 output Mean at reset:

x1= x x x x x

Detection that no valid values are available is made. Value 0 is output at output **Mean**.

ValNo = 5 1. sample after reset:

x1= 55 x x x x

Detection that only one valid value is available is made. The only valid value y1 = 55 is available at output **Y_1**.

ValNo = 5 2. sample after reset:

x1= 44 55 x x x

Detection that two valid values are available is made. The mean value of these valid values y1 = 49,5 is output at output **Y_1**.

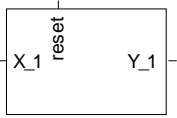
After all memory cells with a value are occupied (ValNr = 5), with every sample a new input value is added, the at this time oldest value subtracted and the result divided by ValNr. = 5. The input values are shifted (like with a shift register).

| In-/Outputs | | |
|-------------|-------|--|
| Name | Type | Description |
| X_1 | Float | Process value, the floating arithmetic mean value of which is formed from a number (ValNo) of values detected last. |
| disabl | Bool | disabl = 1: The mean value formation is interrupted. |
| reset | Bool | The reset input clears the memory and resets the mean value to 0. |
| sample | Bool | A positive flank (0 to 1) is used for sampling a new value. With sample soft-wired, the adjusted sample and unit parameters are ineffective. |
| Name | Type | Description |
| Mean | Float | Calculated mean value |
| ready | Bool | Display of pulse for an elapsed overall cycle |

| Parameter | | | | | | | |
|-----------|------------------|--------|---|--------|---------|-----------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| ValNo | Number of values | Int | Number of values to be detected | r/w | 100 | 1 ... 100 | |
| Unit | Unit | Enum | Time unit of the interval time for mean value formation are seconds, minutes or hours. | r/w | 0 | | |
| | | Second | One value per second should be measured. | | 0 | | |
| | | Minute | Values should be measured at intervals of one minute. | | 1 | | |
| | | Hour | A new value should be measured at intervals of one hour. | | 2 | | |
| Sample | Sample | Float | Interval time. If the sample input is soft-wired, detection or recording of data is triggered by a positive flank at this input, and the adjusted sampling interval is invalid. | r/w | 1.0 | >0.1 | |

III-6 Time functions

III-6.1 LEAD (Differentiator (No. 50))



LEAD

Abb. 406

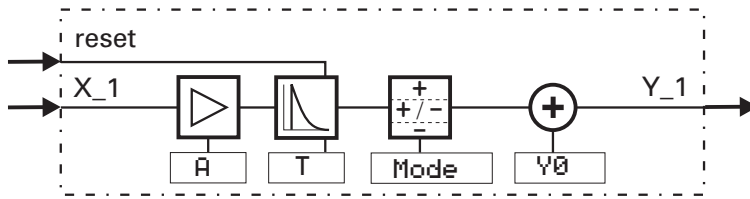


Abb. 407

The differentiator forms the difference quotient according to equation:

$$Y1_{(t)} = \frac{T}{T + t_s} \cdot [Y1(t - t_s) + A \cdot \{X1_{(t)} - X1_{(t-t_s)}\}] + Y0$$

| | | | |
|----|-------------------|----------|------------------|
| ts | sampling interval | x1(t) | instantaneous x1 |
| T | time constant | x1(t-ts) | previous x1 |
| A | gain | y1(t) | instantaneous y1 |
| Y0 | output offset. | y1(t-ts) | previous y1 |

$C = \frac{T}{T + t_s} < 1$ (differentiation constant). The complex transfer function reads: $F_{(p)} = \frac{A \cdot T \cdot p}{T \cdot p + 1}$

| In-/Outputs | | |
|-------------|-------|--|
| Name | Type | Description |
| X_1 | Float | Input variable to be differentiated |
| reset | Bool | reset = 1 causes the output y1 to be set to offset y0, and the difference quotient to be set to 0. reset = 0 starts differentiating automatically. |
| Name | Type | Description |
| Y_1 | Float | Differentiator output |

| Parameter | | | | | | | |
|-----------|---------------|-------|--------------------------|--------|---------|------------------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| A | Factor for X1 | Float | Gain factor | r/w | 1.0 | | |
| Y0 | Offset Y1 | Float | Output offset | r/w | 0.0 | | |
| T | Decay time | Float | Time constant in seconds | r/w | 1.0 | 0.0 ... 199999.0 | |

| Configuration | | | | | | | |
|---------------|----------------|--------------------|---|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Mode | Direction mode | Enum | Operating mode of the differentiator | r/w | 0 | | |
| | | Both directions X1 | Differentiation of all changes | | 0 | | |
| | | Pos. Changes X1 | Differentiation of positive changes only. (Negative changes are set to zero at the input, the output signal is positive.) | | 1 | | |
| | | Neg. changes X1 | Differentiation of negative changes only. (Positive changes are set to zero at the input, the output signal is negative.) | | 2 | | |

Step response:

After a step change of input variable X_1 by $\{x = x(t) - x(t - t_s)\}$, the output changes to maximum value.

$$Y_{\max} = C \cdot \Delta \cdot \Delta X + Y0$$

and decays to 0 according to function

$$Y_{n \cdot t_s} = C^n \cdot \Delta \cdot \Delta X + Y0 = Y_{\max} \cdot C^{n-1}$$

Thereby, n is the number of calculation cycles t_s after the input step change. Number n of required calculation cycles t_s until output variable decaying to $y(n \cdot t_s)$ is $(n \cdot t_s)$ is

$$n = \frac{\lg \frac{Y_{(n \cdot t_s)}}{Y_{\max}}}{\lg C} + 1.$$

Surface area A under the decaying function is:

$$A = Y_{\max} \cdot \left(\frac{T}{T_s} - 1 \right) = \Delta \cdot \Delta X.$$

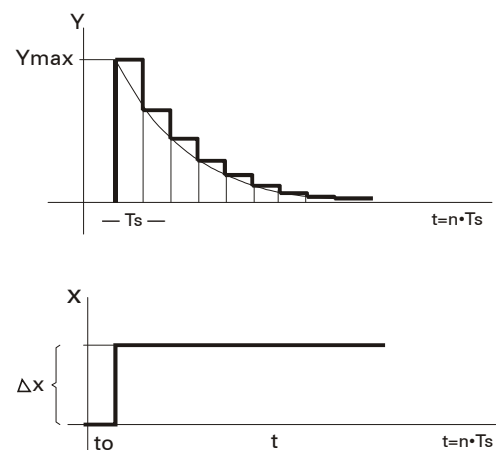


Abb. 408

Ramp response:

After ramp starting, output variable y runs towards the final value of differentiation quotient

$$Y_{\max} = m \cdot \Delta \cdot T$$

according to function $Y_{(n \cdot Ts)} = m \cdot \Delta \cdot T \cdot (1 - C^n)$

Thereby, $m = m = \frac{dx}{dt}$

is the gradient factor of the input function. Relative error F after n calculating cycles T_s referred to the final value is calculated as follows: $F = C^n$ and the number of required calculating cycles, according to which function approaches final value $y = y_{\max}$ to error F is

$$n = \frac{\lg F}{2 \cdot \lg C}$$

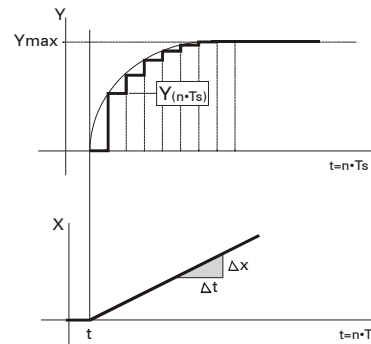
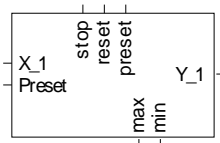


Abb. 409

III-6.2 INTEGRATE (Integrator (No. 51))



INTEGRATE

Abb. 410

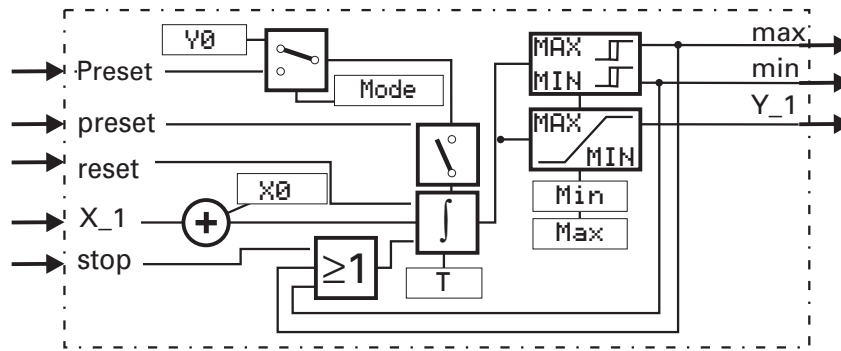


Abb. 411

The integrator forms the integral according to equation:

$$Y1(t) = Y1(t-t_s) + \frac{ts}{T} \cdot [X1(t) + X0]$$

- | | | | |
|----|------------------------------|----------|------------------|
| ts | sampling interval | x1(t) | instantaneous x1 |
| T | Integration constant | y1(t) | y1 after t=n*ts |
| n | Number of calculation cycles | y1(t-ts) | previous y1 |
| x0 | Input offset | | |

The complex transfer function is: $F(p) = \frac{1}{T \cdot p}$

Unused control inputs are interpreted as logic "0". With simultaneous input of several control commands

reset = 1 has priority over **Preset** and **stop**

preset = 1 has priority over **stop**

Integrator output **Y_1** is limited to the preset limits (**Min,Max**): $Min \leq Y_1 \leq Max$.

When exceeding **Min** or **Max**, the integrator is stopped automatically and the relevant control output **min** or **max** is set to logic 1. Limit value monitoring uses a fixed hysteresis of 1 % referred to operating range (**Max - Min**)

| In-/Outputs | | |
|-------------|-------|--|
| Name | Type | Description |
| X_1 | Float | Input variable to be integrated |
| Preset | Float | preset has priority over stop. preset = 1: Set the integration result to the preset value. After cancelation of the preset command, integration starts with the actually effective |

| | | |
|--------|------|---|
| | | preset value. |
| stop | Bool | stop = 1: The integrator is stopped for the duration of the stop command. Output Y_1 does not change. |
| reset | Bool | reset has priority over preset and stop. reset = 1: The integration result is adjusted to lower limit (Min). After cancelation of reset, integration starts at lower limiting. |
| preset | Bool | = 1 The integration result is set either to a preset value y0 (Mode=0) or to a preset variable Preset (Mode= 1). After cancelation of the preset command, integration starts with the actually effective preset value. |

| Name | Type | Description |
|------|-------|-------------------------------------|
| Y_1 | Float | Integrator output |
| max | Bool | max = 1: max. limiting is exceeded. |
| min | Bool | min = 1: min. limiting is exceeded. |

Parameter

| ID | Name | Type | Description | Access | Default | Range | Off |
|------|---------------------|--------------|---|--------|---------|-------|-----|
| Mode | Preset source | Enum | The Preset value can be either adjusted using parameter y0 or determined by input preset. | r/w | 0 | | |
| | | Parameter y0 | The integrator is set to a fixed value using the digital preset input; this value is adjustable via parameter y0. | | 0 | | |
| | | Input Preset | The integrator is set to a fixed value using the digital preset input; this value is read by the analog Preset input. | | 1 | | |
| T | Integration time[s] | Float | Time constant in seconds, integration constant | r/w | 60.0 | >0.1 | |
| X0 | Verschiebung X1 | Float | Offset for X_1, input offset | r/w | 0.0 | | |
| Y0 | Preset value | Float | Preset value, effective with Mode = 0 | r/w | 0.0 | | |
| Max | Max | Float | Maximum limiting | r/w | 1.0 | | |
| Min | Min | Float | Minimum limiting | r/w | 0.0 | | |

Ramp function:

With constant input $x1+x0$, the applicable formulas are

$$Y1(t) = Y(t0) + n \cdot \frac{ts}{T} \cdot (X1 + X0)$$

$$t = n \cdot ts$$

"t" is the time required by the integrator for changing output Y_1 linearly by value $x1 + x0$ after integration start.

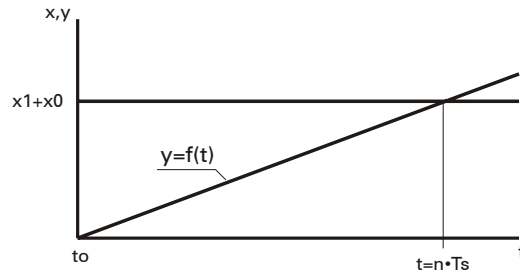
Ramp response:

Abb. 412

Example:

Which is the value of output variable y after $t=20s$ with a time constant of $100s$, if a constant of $x_1 = 10$ Volt is preset. Sampling interval t_s is $100ms$.

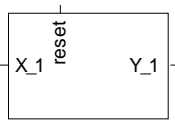
$$n = \frac{t}{t_s}$$

$$n = \frac{20s}{0,1s} = 200$$

$$Y = 0 + 200 \cdot \frac{0,1}{100} \cdot 10 = 2$$

This results in a gradient of $\frac{2}{20s}$ or $\frac{0,1}{1s}$.

III-6.3 FILTER (Filter (No. 52))



FILTER

Abb. 413

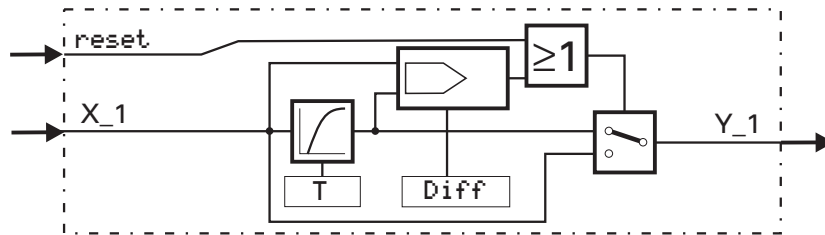


Abb. 414

Dependent of control input **reset**, input variable **X_1** is passed on to output **Y_1** with delay (reset= 0) or without delay (reset = 1). Delay is according to a 1st order e-function (1st order low pass) with time constant **T**(s). The output variable for reset= 0 is calculated according to the following equation:

$$Y1_{(t)} = \frac{T}{T + t_s} \cdot Y1_{(t-t_s)} + \frac{t_s}{T + t_s} \cdot X1_{(t)}$$

| | | | |
|----|------------------------------|----------|-------------------|
| ts | sampling interval | x1 (t) | instantaneous x1 |
| T | time constant | x1(t-ts) | previous x1 |
| n | number of calculation cycles | y1 (t) | y1 after t = n*ts |
| | | y1(t-ts) | previous y1 |

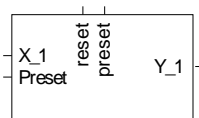
The complex transfer function is: $F(p) = \frac{1}{1 + p \cdot T}$.

| In-/Outputs | | |
|-------------|-------|--|
| Name | Type | Description |
| X_1 | Float | Input variable to be delayed |
| reset | Bool | Delay switch-off. reset = 0: input signal X_1 is output according to the calculated e function at output Y_1. reset = 1: input signal X_1 is output without delay at output Y_1. |
| Name | Type | Description |
| Y_1 | Float | Output variable |

| Parameter | | | | | | | |
|-----------|-----------------|-------|--------------------------|--------|---------|------------------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| T | Filter time [s] | Float | Time constant in seconds | r/w | 1.0 | 0.0 ... 199999.0 | |

No configuration parameters!

III-6.4 DELAY (Delay time (No. 54))



DELAY

Abb. 415

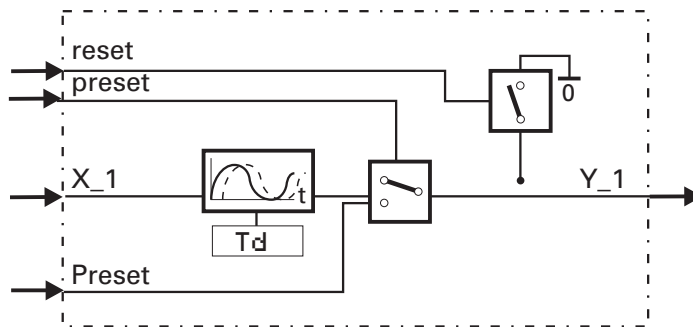


Abb. 416

The function provides calculation

$$Y1_{(t)} = X1_{(t-Td)}$$

Input variable X_1 is output at Y_1 with delay by time Td . The accuracy Td is dependent of the time group (sampling interval), to which the function is assigned.

The shift register has a maximal length of 255, which depends on the setting of parameter Td and sampling interval t_s . The effective length is calculated from Td/t_s (rounded to next higher natural number).

Example:

| | | |
|------------------|--------------------|-------------------------|
| $Td = 0,7s$ with | sampling intervall | 100ms means $Td = 0,7s$ |
| | sampling intervall | 200ms means $Td = 0,8s$ |
| | sampling intervall | 400ms means $Td = 0,8s$ |
| | sampling intervall | 800ms means $Td = 0,8s$ |

The possible delay time is dependent of the configured time slot (sampling interval).

| | | | |
|--------------------|--------|--------------|-------|
| $Td \text{ max} =$ | 25,5s | with $t_s =$ | 100ms |
| $Td \text{ max} =$ | 51,0s | with $t_s =$ | 200ms |
| $Td \text{ max} =$ | 102,0s | with $t_s =$ | 400ms |
| $Td \text{ max} =$ | 204,0s | with $t_s =$ | 800ms |

The priorities are:

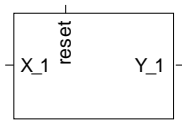
reset = 1 has higher priority than **Preset**.

| In-/Outputs | | |
|-------------|-------|--|
| Name | Type | Description |
| X_1 | Float | Input variable to be delayed |
| Preset | Float | With preset = 1, the preset value is taken to the output. |
| reset | Bool | With reset = 1, output Y_1 is set to zero. Priority over input preset. |
| preset | Bool | = 1 The preset value is taken to the output |

| Name | Type | Description |
|------|-------|-----------------|
| Y_1 | Float | Output variable |

| Parameter | | | | | | | |
|-----------|----------------|-------|---|--------|---------|---------------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Td | Delay time (s) | Float | Delay in seconds. 'The accuracy of the delay time is dependent on the sampling interval assigned to the function. | r/w | 0.0 | 0.0 ... 204.0 | |

III-6.5 BAND_FILTER (Filter with tolerance band (No. 55))



BAND_FILTER

Abb. 417

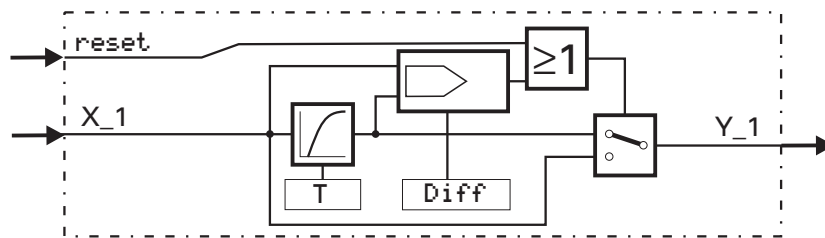


Abb. 418

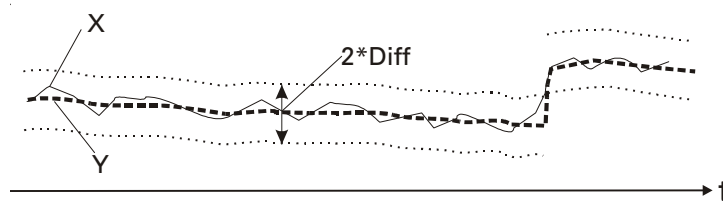
$$(|X_1 - Y_1| \ll \delta) \quad \text{is} \quad F(p) = \frac{1}{1 + p \cdot T}$$

With a difference higher than **Diff** between input **X_1** and output **Y_1** or **reset = 1**, the filter stage is switched off and the output follows the input directly.

With a difference of input **x1** and output **y1** smaller than **Diff** and **reset = 0**, the output follows an e-function with time constant **T**. The output variable is calculated according to the following equation:

$$Y_{1(t)} = \frac{T}{T + t_s} \cdot Y_{1(t-t_s)} + \frac{t_s}{T + t_s} \cdot X_{1(t)}$$

Y



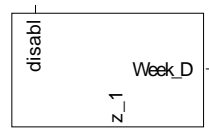
- t_s sampling interval
- T time constant
- $x(t)$ instantaneous x_1
- $x_1(t-t_s)$ previous x_1

Abb. 419

| In-/Outputs | | |
|-------------|-------|--|
| Name | Type | Description |
| X_1 | Float | Input variable to be delayed |
| reset | Bool | Delay switch-off, 1 = delay is ineffective, output is passed to the input immediately. |
| Name | Type | Description |
| Y_1 | Float | Output variable |

| Parameter | | | | | | | |
|-----------|----------------|-------|--|--------|---------|---------------------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| T | Filter time[s] | Float | Time constant in seconds | r/w | 1.0 | 0.0 ... 199999.0 | |
| Diff | Tolerance band | Float | Filtering is effective only, when the separation between input and output variable is smaller than the tolerance band. | r/w | 1.0 | >0.0 | |

III-6.6 Timer (Timer (No. 67))



TIMER

Abb. 420

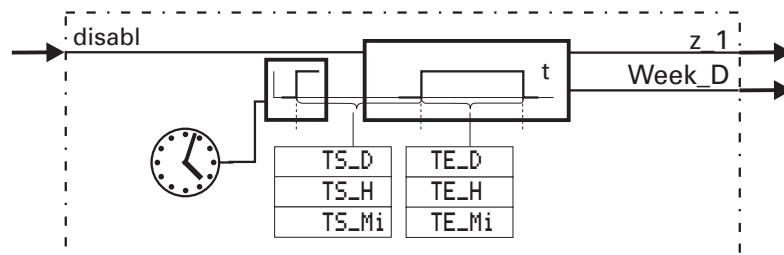


Abb. 421

The function **TIMER** uses data of the real-time clock. Output **z_1** is switched on at absolute time **TS** and switched off again after **TE**. This switching operation can be unique or cyclical (parameter adjustment). Output **Week_D** indicates the actual weekday (0...6 = Su...Sa). **TS_Mo** = 0 and **TS_D** = 0 means actual day.

When the time defined with **TS_H** and **TS_Mi** has elapsed, the 1st switching operation occurs on the following day. With **TS_Mo** = 0 und **TS_D** < actual day, the first switching operation occurs in the following month. With **TS_Mo** ≤ actual month and **TS_D** < actual day, the 1st switching operation occurs in the next year.

In-/Outputs

| Name | Type | Description |
|--------|------|--|
| disabl | Bool | disabl = 0: output z_1 active; is switched on when the time is reached. disabl = 1: output z_1 switched off. The output behaves like "time not yet reached". |

| Name | Type | Description |
|--------|-------|---|
| Week_D | Float | Indicates the current weekday (0...6 = Su...Sa) |
| z_1 | Bool | Output to be switched, is logic 1 between the start and end time. |

Parameter

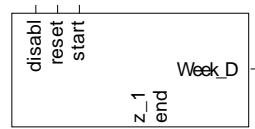
| ID | Name | Type | Description | Access | Default | Range | Off |
|-------|-------------|------|--|--------|---------|----------|-----|
| TS_Mo | Start month | Int | Switch-on time "month". Setting TS_Mo = 0 means "present day". | r/w | 0 | 0 ... 12 | |
| TS_D | Start day | Int | Switch-on time "day". Setting TS_D = 0 means present day. Unless a month was selected (TS_Mo = 0) and if the selected day has already passed (TS_D smaller than "present | r/w | 0 | | |

| | | | | | | | |
|-------|--------------------|-----|--|-----|---|-----------|--|
| | | | day") the 1st switching operation will take place in the next month. If the selected day has already passed (month TS_Mo smaller than or equal to "present month" and day TS_D smaller than "present day") the 1st switching operation will take place in the next year. | | | | |
| TS_H | Start hour | Int | Switch-on time "hour". If the selected hour has already passed the 1st switching operation will take place on the following day. | r/w | 0 | 0 ... 23 | |
| TS_Mi | Start minute | Int | Switch-on time "minute". If the selected minute has already passed, the 1st switching operation will take place on the following day. | r/w | 0 | 0 ... 59 | |
| TE_D | Duration [days] | Int | Switch-on duration "days" | r/w | 0 | 0 ... 255 | |
| TE_H | Duration [hours] | Int | Switch-on duration "hours" | r/w | 0 | 0 ... 23 | |
| TE_Mi | Duration [minutes] | Int | Switch-on duration "minutes" | r/w | 0 | 0 ... 59 | |

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|--------|----------------|---------|--|--------|---------|-------|-----|
| Func_1 | Clock function | Enum | Switch-over between single and cyclical start of the selected function | r/w | 0 | | |
| | | Cyclic | The function is handled cyclically (i.e. is repeated at a fixed interval). | | 0 | | |
| | | Once | Function is handled once (no repetition). | | 1 | | |
| Func_2 | Timer cycle | Enum | Switch-over between various intervals for starting the function. The function can be started daily, or daily except on Saturdays and Sundays, or daily except on Sundays, or weekly. | r/w | 0 | | |
| | | Daily | The function is handled once a day. | | 0 | | |
| | | Mo...Fr | The function is handled from Monday to Friday (not on Saturdays and Sundays). | | 1 | | |
| | | Mo...Sa | The function is handled from Monday to Saturday (not on Sundays). | | 2 | | |
| | | Weekly | The function is handled once a week. | | 3 | | |

III-6.7 TIMER 2 (Timer (No. 70))



TIMER2

Abb. 422

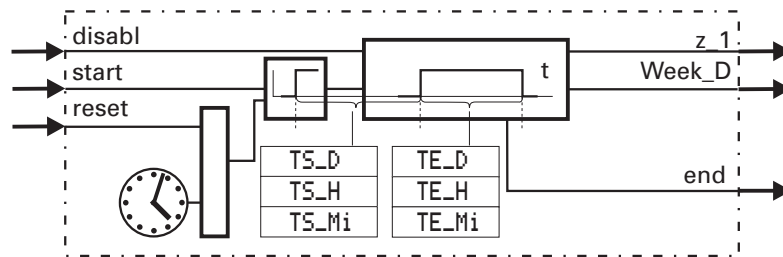


Abb. 423

The function **TIMER2** uses data of the real-time clock. With a positive flank at **start**, **TIMER2** is started and output **z_1** is switched to 1 after elapse of time **TS** and reset to 0 after elapse of time **TE**.

Example:

$TS_D = 2, TS_H = 1, TS_Mi = 30, TE_D = 0, TE_H = 2, TE_Mi = 2$

After the change from 0 to 1 at input **start**, output **z_1** is set to 1 after 2 days, 1 hour and 30 seconds and reset to 0 after 2 hours and 2 seconds.

Cyclic switching operations can be realized by feed-back of the **end** output to the **start** input.

In-/Output

| Name | Type | Description |
|--------|------|---|
| disabl | Bool | disabl = 1: suppresses the switching operation. |
| reset | Bool | reset = 1: finishes an instantaneously running switching operation immediately. |
| start | Bool | With positive flank (0 to 1). TIMER2 is started: after elapse of time TS , the output z_1 is switched to 1 and reset to 0 after elapse of time TE . |

| Name | Type | Description |
|--------|-------|--|
| Week_D | Float | Week_D indicates the actual weekday (0...6 = Su...Sa) |
| z_1 | Bool | z_1 = 1: switching operation running |
| end | Bool | end = 1: switching operation end. Cyclical switching operation can be realized by feedback of the end output to input start. |

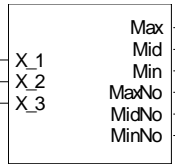
Parameter

| ID | Name | Type | Description | Access | Default | Range | Off |
|------|--------------|------|--------------------------------------|--------|---------|-------|-----|
| TS_D | Delay [days] | Int | Switch-on delay "day" (whole number) | r/w | 0 | 0 ... | |

| | | | | | | | |
|-------|--------------------|-----|--|-----|---|-----------|--|
| | | | | | | 255 | |
| TS_H | Delay [hours] | Int | Switch-on delay "hour" (whole number) | r/w | 0 | 0 ... 23 | |
| TS_Mi | Delay [minutes] | Int | Switch-on delay "minute" (whole number) | r/w | 0 | 0 ... 59 | |
| TE_D | Duration [days] | Int | Switch-on delay "days" (whole number) | r/w | 0 | 0 ... 255 | |
| TE_H | Duration [hours] | Int | Switch-on delay "hours" (whole number) | r/w | 0 | 0 ... 23 | |
| TE_Mi | Duration [minutes] | Int | Switch-on delay "minutes" (whole number) | r/w | 0 | 0 ... 59 | |

III-7 Selecting and storage

III-7.1 EXTREM (extreme value selection (No. 30))



EXTREM

Abb. 424

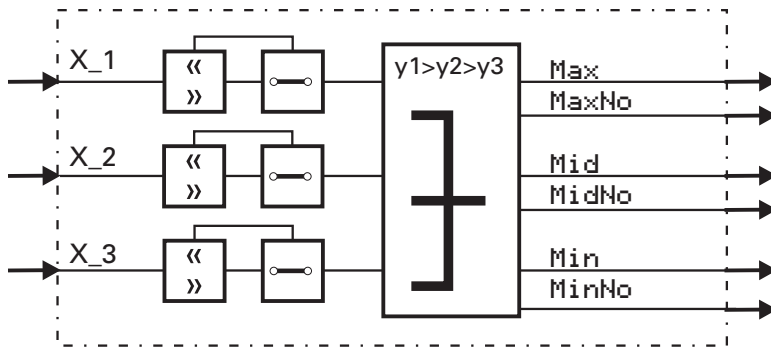


Abb. 425

Analog inputs X_1 , X_2 and X_3 are sorted according to their instantaneous values and provided at outputs **Max**, **Mid** and **Min**. Input value output is at **Max** for the highest one, at **Mid** for the medium one and at **Min** for the smallest one.

The number of the input with the highest value is output at **MaxNo**.

The number of the input with the medium value is provided at output **MidNo**.

The number of the input with the smallest value is provided at output **MinNo**.



Hint!

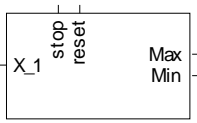
With equality, the distribution is at random. Inputs are not included into the extreme value selection, if: -the input is not wired-or the input value is higher than $1,5 \cdot 10^{37}$ or smaller than $-1,5 \cdot 10^{37}$.

| Number of failed inputs | Max | Mid | Min | MaxNo | MidNo | MinNo |
|-------------------------|---------------------|------|---------------------|---------------------------------|----------------|----------------|
| 0 | xmax | xmid | xmin | number of xmax | number of xmid | number of xmin |
| 1 | xmax | | | number of xmin | | number of xmax |
| 2 | the valid value | | | number of the still valid input | | |
| 3 | $1,5 \cdot 10^{37}$ | | $1,5 \cdot 10^{37}$ | | | |

| In-/Outputs | | |
|--------------------|-------------|---------------------------------|
| Name | Type | Description |
| X_1 | Float | Input variable 1 to be compared |
| X_2 | Float | Input variable 2 to be compared |
| X_3 | Float | Input variable 3 to be compared |

| Name | Type | Description |
|-------------|-------------|---|
| Max | Float | Maximum instantaneous input value |
| Mid | Float | Mean instantaneous input value |
| Min | Float | Minimum instantaneous input value |
| MaxNo | Float | Number of the input which provides the maximum instantaneous input value (1 = x1, 2 = x2, 3 = x3) |
| MidNo | Float | Number of the input which provides the mean instantaneous input value (1 = x1, 2 = x2, 3 = x3) |
| MinNo | Float | Number the input which provides the minimum instantaneous input value (1 = x1, 2 = x2, 3 = x3) |

III-7.2 PEAK (peak value memory (No. 31))



PEAK

Abb. 426

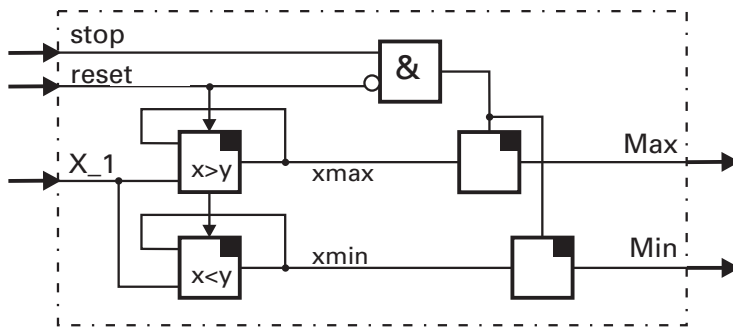


Abb. 427

Maximum input value **xmax** and minimum input value **xmin** are determined, stored and output at **Max** and **Min**. With the **stop** input set to 1, the extreme values determined last remain unchanged.

If the **reset** input is set to 1, the extreme value memory and any applied stop command are cancelled. (**max** and **min** are set to the instantaneous **x1** value and follow input **X_1**, until the **reset** input returns to 0.) Unused inputs are interpreted as 0 or logic 0.

| In-/Outputs | | |
|-------------|-------|---|
| Name | Type | Description |
| X_1 | Float | Process value, the min and max values of which are output. |
| stop | Bool | stop = 1: the max and min instantaneous values are unchanged. |
| reset | Bool | reset = 1: extreme value storage and any applied stop command are cancelled. Outputs max and min are set to the instantaneous X_1 value and follow input X_1, until the reset input returns to 0. |
| Name | Type | Description |

| | | |
|-----|-------|---------------|
| Max | Float | Maximum value |
| Min | Float | Minimum value |

No parameters!

III-7.3 HOLD (hold amplifier (No. 32))

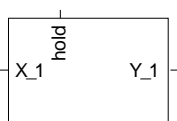
**HOLD**

Abb. 428

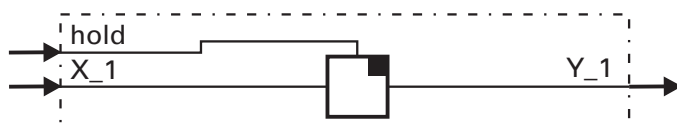


Abb. 429

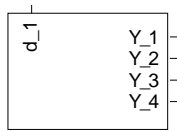
With control input **HOLD** set to 1, instantaneous input value x_1 is stored and output at Y_1 . With control input **HOLD** set to 0, output Y_1 follows input value X_1 .

| In-/Outputs | | |
|-------------|-------|--|
| Name | Type | Description |
| X_1 | Float | Input value |
| hold | Bool | Storage signal for the X_1 input value |

| Name | Type | Description |
|------|-------|--|
| Y_1 | Float | Function output; when hold = 0, the output follows X_1, when hold = 1, the value output instantaneously remains unchanged. |

No parameters!

III-7.4 SEL_C (Constant selection (No. 33))



SEL_C

Abb. 430

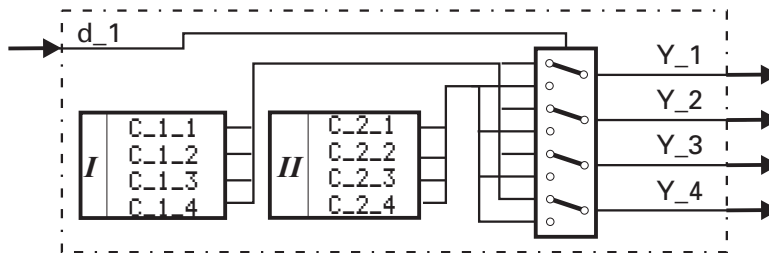


Abb. 431

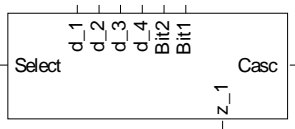
Dependent of control signal **d_1** , the four preset parameters of group 1 or group 2 are output.

| In-/Outputs | | |
|-------------|------|---|
| Name | Type | Description |
| d_1 | Bool | Selecting the constant group (0 = group I; 1 = group II), which should be passed on to the outputs. |

| Name | Type | Description |
|----------------|-------|---|
| Y_1 ... Y_4 | Float | Output 1: For input d1=0 , parameter C1.1 of group I is output; for input d1=1, parameter C2.1 of group II is output. |

| Parameter | | | | | | | |
|--------------------|----------------|-------|---|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| C_1_1 ... C_1_4 | Set 1, value 1 | Float | 1st constant of group I, is output on Y_1 when d1 = 0. | r/w | 0.0 | | |
| C_2_1 ... C_2_4 | Set 2, value 1 | Float | 1st constant of group II, is output on Y_1 when d1 = 1. | r/w | 1.0 | | |

III-7.5 SEL_D (selection of digital variables (No. 06))



SEL_D

Abb. 432

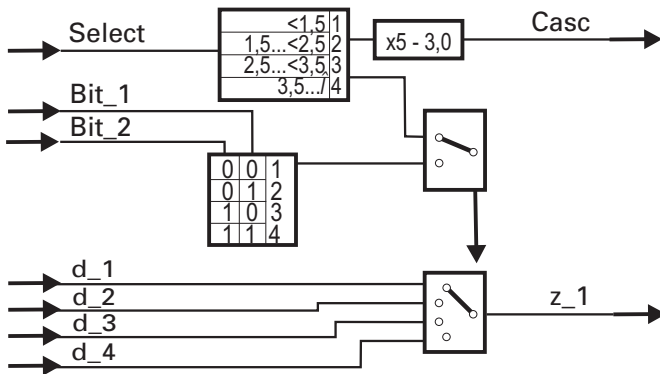


Abb. 433

Selection of one of the 4 digital inputs is either by an analog “**Select**” signal or by the two digital control signals **Bit_1**, **Bit_2**. If analog control signal **Select** is connected, selecting is done using this control signal. If the input is not connected, selection is by means of the two digital control inputs **Bit_1**, **Bit_2**.

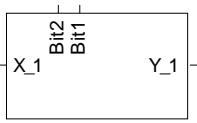
This function block is cascadable. The **Select** input can be linked with the **Casc** output of another SELD block into a choice of 8 digital variables.

| In-/Outputs | | |
|-------------|-------|---|
| Name | Type | Description |
| Select | Float | Dependent on input value, the relevant variable is output at the z_1 output (priority over bit inputs). |
| d_1 | Bool | Input, is output at z_1, when Select = 1, or when bit2 = 0 and bit1 = 0. |
| d_2 | Bool | Input, is output at z_1, when Select = 2, or bit2 = 0 and bit1 = 1. |
| d_3 | Bool | Input, is output at z_1, when Select = 3, or bit2 = 1 and bit1 = 0. |
| d_4 | Bool | Input, is output at z_1, when Select = 4, or bit2 = 1 and bit1 = 1. |
| Bit2 | Bool | 2nd control signal for variable selection (most significant bit). Only valid, when input Select is not soft-wired. |
| Bit1 | Bool | 1st control signal for variable selection (least significant bit). Only valid, when input select is not soft-wired. |

| Name | Type | Description |
|------|-------|---|
| Casc | Float | Cascade output = Select – 3.0. The block can be cascaded for selection of more than 4 digital inputs: Soft-wire Casc of the first SEL_D with Select of the second one and z_1 of the first one with d_1 of the second one (selection in the cascade only via input Select). |

| | | |
|-----|------|--|
| z_1 | Bool | According to the input value of Select (or, without Select, according to the values at inputs bit1, bit2), the relevant input variable d_1, d_2, d_3 or d_4 is output. |
|-----|------|--|

III-7.6 SEL_P (parameter selection (No. 34))



SEL_P

Abb. 434

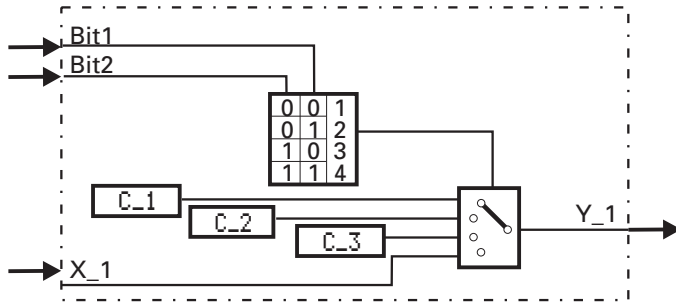


Abb. 435

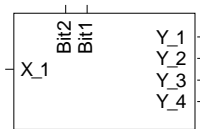
Dependent of control signals **Bit1** and **Bit2**, either one of the three preset parameters **C_1**, **C_2**, **C_3** or input variable **X_1** is connected with output **Y_1**. Unused inputs are interpreted as 0 or logic 0.

| In-/Outputs | | |
|-------------|-------|---|
| Name | Type | Description |
| X_1 | Float | Input, is output at Y_1, when bit2 = 1 and bit1 = 1. |
| Bit2 | Bool | 2nd control signal for variable selection (most significant bit) |
| Bit1 | Bool | 1st control signal for variable selection (least significant bit) |

| Name | Type | Description |
|------|-------|---|
| Y_1 | Float | Output of parameter C1, when bit2 = 0 and bit1 = 0, of parameter C2, when bit2 = 0 and bit1 = 1, of parameter C3, when bit2 = 1 and bit1 = 0, and of the input variable, when bit2 = 1 and bit1 = 1. |

| Parameter | | | | | | | |
|-----------|------------|-------|---|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| C_1 | Constant 1 | Float | Is output at Y_1, when bit2 = 0 and bit1 = 0. | r/w | 0.0 | | |
| C_2 | Constant 2 | Float | Is output at Y_1, when bit2 = 0 and bit1 = 1. | r/w | 0.0 | | |
| C_3 | Constant 3 | Float | Is output at Y_1, when bit2 = 1 and bit1 = 0. | r/w | 0.0 | | |

III-7.7 SEL_OUT (selection of output (No. 36))



SEL_OUT

Abb. 436

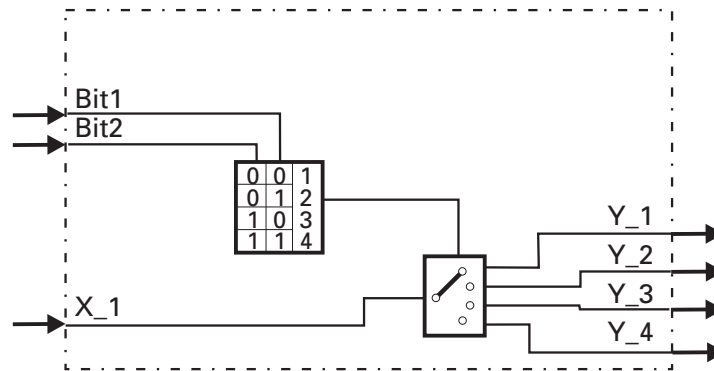


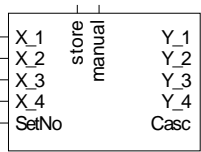
Abb. 437

Dependent of control signals **Bit1** and **Bit2**, the input **X_1** is connected with one of the outputs **Y_1**, **Y_2**, **Y_3** or **Y_4**. Unused inputs are interpreted as 0 or logic 0.

| In-/Outputs | | |
|-------------|-------|---|
| Name | Type | Description |
| X_1 | Float | Input, is output at Y_(n) which is determined by the bit inputs. |
| Bit2 | Bool | 2nd control signal for variable selection (most significant bit) |
| Bit1 | Bool | 1st control signal for variable selection (least significant bit) |
| Name | Type | Description |
| Y_1 | Float | Output of the input variable, when Bit2 = 0 and Bit1 = 0 |
| Y_2 | Float | Output of the input variable, when Bit2 = 0 and Bit1 = 1 |
| Y_3 | Float | Output of the input variable, when Bit2 = 1 and Bit1 = 0 |
| Y_4 | Float | Output of the input variable, when Bit2 = 1 and Bit1 = 1 |

No Parameters

III-7.8 REZEPT (recipe management (No. 37))



REZEPT

Abb. 438

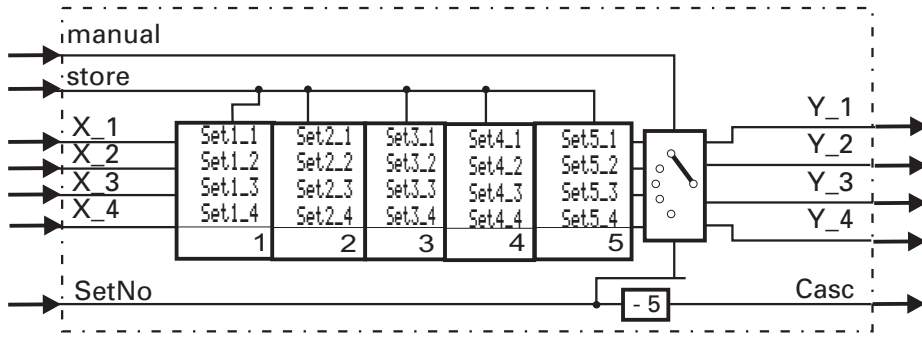


Abb. 439

The function **REZEPT** has 5 groups (recipe blocks) each with 4 memory locations. The recipes can be written via parameter setting and analog inputs. The function parameters are stored in EEPROM with back-up. Selection which recipe block is output at **Y_1 ... Y_4** is determined by the value applied to input **SetNo**. In mode **STORE** (**store** = 1), the values applied to **X_1 ... X_4** are written into the memory addresses of the recipe block selected with input **SetNo**. During manual mode (**manual** = 1), the inputs are directly connected with the outputs. If more than 5 recipes are required, a corresponding number of recipe functions are simply cascaded.



ATTENTION!

Too frequent storing operations may lead to the destruction of the EEPROM! Values of the used analog inputs are stored as parameter values when detecting a positive edge at the **store** - input. This input should be activated only with relevant changes of the input values.

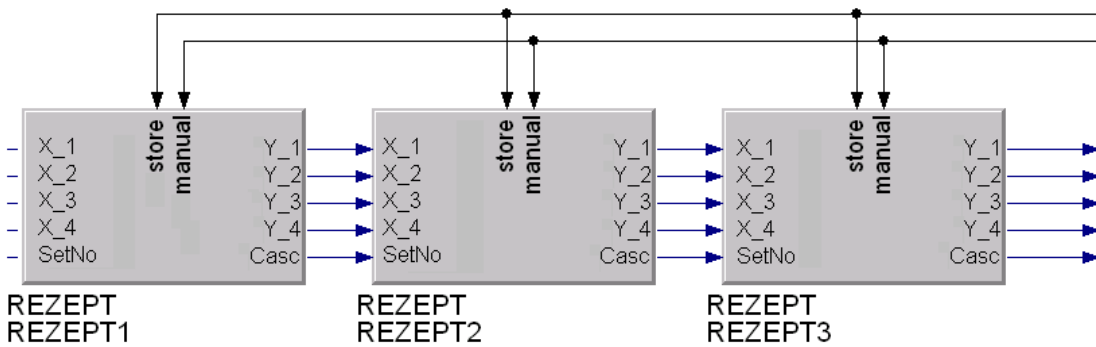


Abb. 440 Example for 15 recipes

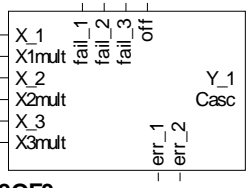
With cascading, the values for the overall recipe are available at outputs **Y_1 ... Y_4** of the last stage.

| In-/Outputs | | |
|----------------|-------|--|
| Name | Type | Description |
| X_1 ... X_4 | Float | In mode STORE (store =1), the input value X_1 is written into the memory location of the group selected with SetNo. The input is connected with output Y_1 directly both in manual mode (manual = 1) and when the SetNo input is out of range 1 to 5. |
| SetNo | Float | Selecting a recipe block: SetNo determines, which one of the 5 recipe blocks is selected. Selection for reading and storage (store). Selection is valid for SetNo within 1 and 5. With SetNo out of range 1...5, the inputs are connected directly with the outputs (independent of the status at input manual). This is required for cascading. |
| store | Bool | Reacts on a positive flank (change from 0 to 1). Input values X_1 to X_4 are stored in the recipe block selected with SetNo and in EEPROM. Storage is also in manual mode (manual = 1). Protection of the EEPROM: this input should be activated only with relevant input value changes. |
| manual | Bool | manual = 0: automatic mode: recipe function active. manual = 1: manual mode: the values of inputs X_1 to X_4 are applied to outputs Y_1 to Y_4 directly. |

| Name | Type | Description |
|----------------|-------|---|
| Y_1 ... Y_4 | Float | The value at Y_1 corresponds either to the recipe block selected with SetNo or to input X_1 in manual mode (store = 1). |
| Casc | Float | The value at output Casc is the value of input "SetNo" reduced by 5 and is used for cascading. |

| Parameter | | | | | | | |
|-------------------|-------------------|-------|--------------------------|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Set1_1 ... Set1_4 | Recipe 1, value 1 | Float | Parameter 1 for recipe 1 | r/w | 0.0 | | |
| Set2_1 ... Set2_4 | Recipe 2, value 1 | Float | Parameter 1 for recipe 2 | r/w | 0.0 | | |
| Set3_1 ... Set3_4 | Recipe 3, value 1 | Float | Parameter 1 for recipe 3 | r/w | 0.0 | | |
| Set4_1 ... Set4_4 | Recipe 4, value 1 | Float | Parameter 1 for recipe 4 | r/w | 0.0 | | |
| Set5_1 ... Set5_4 | Recipe 5, value 1 | Float | Parameter 1 for recipe 5 | r/w | 0.0 | | |

III-7.9 20F3 (2-out-3-selection with mean value formation (No. 38))



20F3

Abb. 441

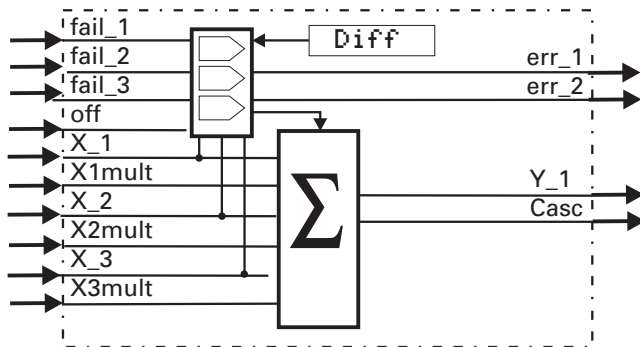


Abb. 442

Function **20F3** forms the arithmetic mean value of input variables $X_1, 2$ und 3 . The difference of $X_1, 2$ and 3 is formed and compared with parameter **Diff**. Inputs of which the value exceeds this limit value are not used for mean value formation.

With 1 applied to **fail_1...fail_3**, faulty inputs are not taken into account either for mean value formation. If all faulty inputs are set to 1, 999999 is output at **Y_1**

err_1 = 1 indicates that one input failed and was not used for mean value formation. If at least 2 inputs do not participate in mean value formation, output **err_2** is set to 1.

With input **off** set to 1 or if output **err_2** = 1 the X_1 value is output at **Y_1**. If all inputs ($X1mult... X3mult$) = 0, "-1" is output at **Y_1**.

Inputs $X1mult... X3mult$ can be used for differentiating the weighting factor of the inputs $X_1..X_3$.

With more than 3 input variables, function **20F3** can be cascaded. Output **Casc** indicates the number of values used for mean value formation. This is important with **20F3** function cascading.

With unwired factor inputs ($X1mult...X3mult$) factor 1 is used automatically. If one of inputs $X_1 .. X_3$ is used, the relevant **Xmult** must be set to 0 or left open!

The **Xmult** input of a following function block is wired with factor output **Casc** of the previous function block.

In this example, **CONST**-output $Y 16 = 0$ is set. The following formulas are calculated:

The left 20F3: $\frac{X1 \cdot 1 + X2 \cdot 1 + X3 \cdot 0}{2}$ and the right 20F3: $\frac{X1 \cdot 1 + X2 \cdot 1 + X3 \cdot 2}{4}$

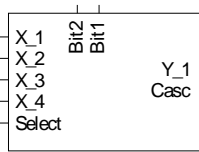
| In-/Outputs | | |
|-------------|-------|--|
| Name | Type | Description |
| X_1 | Float | Signal input 1 |
| X1mult | Float | Factor input pertaining to input 1. Determination of the number of inputs included for calculation of X_1 (required with function block cascading or input X_1 not connected). Non-connected input X1mult is considered as value 1. |
| X_2 | Float | Signal input 2 |
| X2mult | Float | Factor input pertaining to input 2. Determination of the number of inputs included for calculation of X_2 (required with function block cascading or input X_2 not connected). Non-connected input X2mult is considered as value 2. |
| X_3 | Float | Signal input 3 |
| X3mult | Float | Factor input, pertaining to input 3. Determination of the number of inputs included for calculation of X_3 (required with function block cascading or input X_3 not connected). Non-connected input X3mult is considered as value 1. |
| fail_1 | Bool | Error message for input X_1. With fail_1 = 1, input X_1 is not taken into account for mean value formation. |
| fail_2 | Bool | Error message for input X_2. With fail_2 = 1, input X_2 is not taken into account for mean value formation. |
| fail_3 | Bool | Error message for input X_3. With fail_3 = 1, input X_3 is not taken into account for mean value formation. |
| off | Bool | Function switch-off: with off = 1, input X_1 is output at Y_1. |

| Name | Type | Description |
|-------|-------|--|
| Y_1 | Float | Arithmetic mean value or X_1 (off = 1 or several inputs defective) |
| Casc | Float | Factor: number of the values used for mean value formation. Casc = X1mult + X2mult + X3mult. |
| err_1 | Bool | Error message 1 = 1 indicates that at least one of inputs X_1... X_3 is not taken into account with mean value formation. |
| err_2 | Bool | Error message 2 = 1 indicates that mean value formation is omitted. Either several inputs (fail or difference > Diff) are disturbed or function was switched off by input off. |

| Parameter | | | | | | | |
|-----------|------------|-------|--|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Diff | Difference | Float | Limit value for comparison of differences between inputs X_1 ... X_3 for determination of faulty inputs. | r/w | 1.0 | >0.0 | |

No configuration parameters!

III-7.10 SEL_V (cascadable selection of variables (No. 39))



SEL_V

Abb. 443

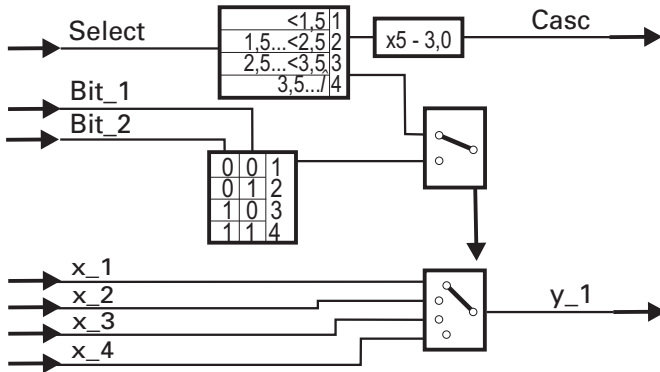


Abb. 444

Dependent on input **Select**, or inputs **Bit1** and **Bit2**, one of the four inputs **X_1...X_4** is connected with output **Y_1**. If input **Select** is connected, the output **Y_1** corresponds to the input **Select**, else to the digital inputs **Bit1** and **Bit2**.

Unused inputs are interpreted as 0. Output **Casc** = input **Casc** = **Block_No**).

The function can be cascaded as shown in the example given below. Dependent of input signal **Select** at the 1st **SEL_V**, the corresponding variable is output at **Y_1** of the 2nd **SEL_V**. Cascades work with input **Select** only (not digital inputs **Bit1** and **Bit2**).

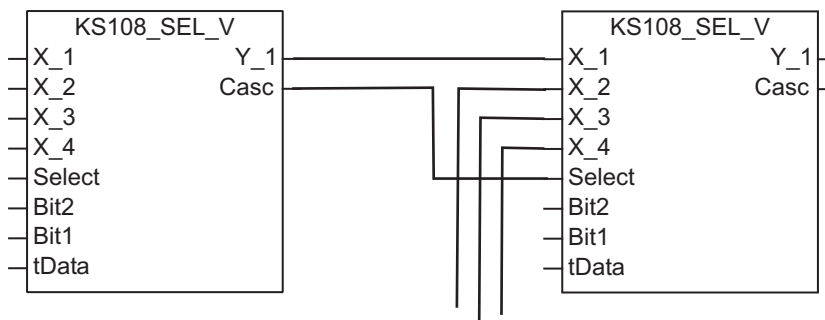


Abb. 445 Cascading

| SEL_V | | y1 output 2nd SEL_V |
|--------------------|----------------|---------------------|
| Select < 1,5 | Bit2=0, Bit1=0 | X_1 of 1st. SEL_V |
| 1,5 ≤ Select < 2,5 | Bit2=0, Bit1=1 | X_2 of 1st. SEL_V |
| 2,5 ≤ Select < 3,5 | Bit2=1, Bit1=0 | X_3 of 1st. SEL_V |
| 3,5 ≤ Select < 4,5 | Bit2=1, Bit1=1 | X_4 of 1st. SEL_V |
| 4,5 ≤ Select < 5,5 | see Cascade | X_2 of 2nd. SEL_V |

| | |
|--------------------------------|---------------------|
| $5,5 \leq \text{Select} < 6,5$ | X_3 of 2nd. SEL_V |
| $\text{Select} \geq 6,5$ | X_4 of 2nd. SEL_V |

In-/Outputs

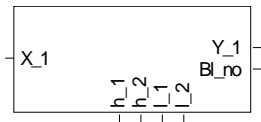
| Name | Type | Description |
|----------------|-------|---|
| X_1 ... X_4 | Float | Input, is output at Y_1, when the value at input Select is smaller than 1.5 or when bit2 = 0 and bit1 = 0. |
| Select | Float | Dependent on input Select, one of the four inputs X_1 to X_4 is connected with output Y_1. Digits behind the decimal point are rounded off to the next higher value from 0.5 and to the next lower value below 0.5. |
| Bit2 | Bool | 2nd control signal for variable selection (most significant bit) |
| Bit1 | Bool | 1st control signal for variable selection (least significant bit) |

| Name | Type | Description |
|------|-------|---|
| Y_1 | Float | According to the input value of Select, or to the input values at bit1 and bit2, the relevant input variable is output. |
| Casc | Float | Cascade output = "Select - 3.0". For selection of more than 4 analog inputs, the block can be cascaded: Soft-wire Casc of the first SEL_V with Select of the second one and Y_1 of the first one with X_1 of the second one (Selection in the cascade only via input Select.) |

No parameters!

III-8 Limit value signalling and limiting

III-8.1 ALLP (alarm and limiting with fixed limits (No. 40))



ALLP

Abb. 446

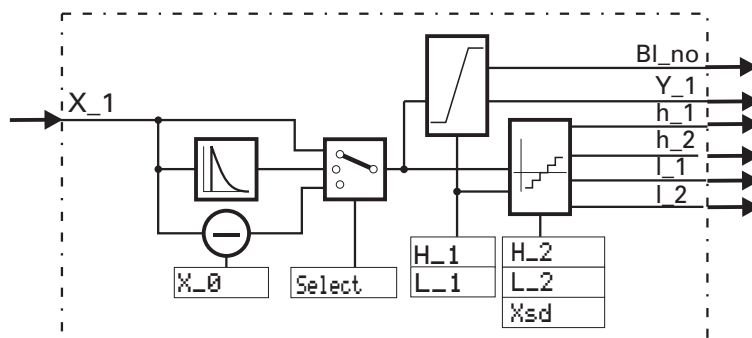
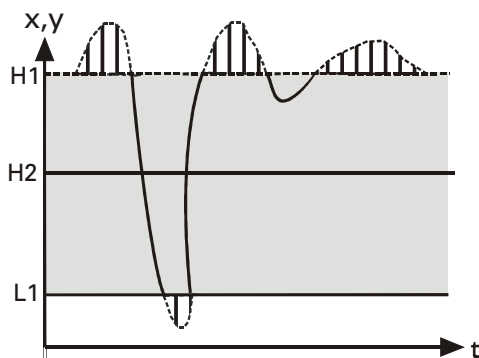


Abb. 447

Signal limiting:

Parameter L_1 determines the minimum, H_1 the maximum limiting, of the output Y_1 ($L_1 \leq y_1 \leq H_1$). With parameter H_1 smaller than L_1 , a higher priority is allocated with H_1 . L_1 is not effective then. This means that $Y_1 \leq H_1$.

Abb. 448: Limiting at $H_1 < L_1$

Limit signaller

The limit signaller has two low and high alarms (L_1 , L_2 , H_1 and H_2). Configuration parameter **Select** can be used to select the variable to be monitored (x_1 , dx_1/dt , $x_1 - x_0$).

The limit values are freely adjustable as parameters and have an adjustable hysteresis of ≥ 0 .

The smallest separation between a minimum and a maximum limit value is 0.

When an alarm is triggered, the corresponding output (L_1 , L_2 , H_1 and H_2) is logic "1".

D -Alarm (dx1/dt)

Value $x_1(t-1)$ measured one sampling interval before is subtracted from instantaneous value $x_1(t)$. This difference is divided by calculation cycle time T_r (e.g. 100ms, cycle time for this block, that is set in engineering).

Thus input variable x_1 can be monitored for its rate of change.

Alarm with offset (x1 - x0):

x_1 can be shifted by means of x_0 . This corresponds to the offset of the adjusted alarm limits (L1, L2, H1 and H2) in parallel to the x-axis.

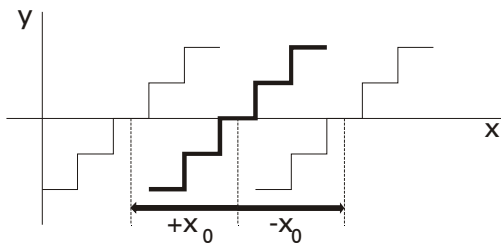


Abb. 449: Offset of the alarm limits

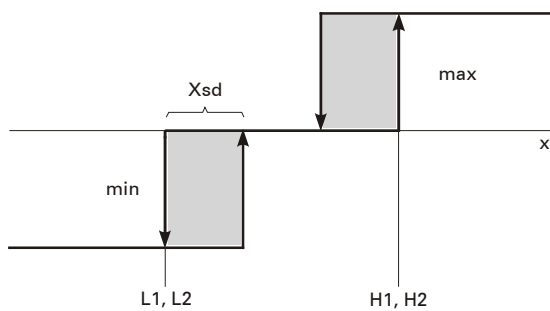


Abb. 450: Switching hysteresis and alarm limits

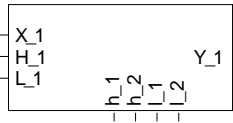
In-/Outputs

| Name | Type | Description |
|-------|-------|---|
| X_1 | Float | Input value to be monitored |
| Y_1 | Float | Calculated and limited input signal X_1. |
| Bl_no | Float | Own block number |
| h_1 | Bool | Logic 1, when input value X_1 is higher than parameter H_1 |
| h_2 | Bool | Logic 1, when input value X_1 is higher than parameter H_2 |
| l_1 | Bool | Logic 1, when input value X_1 is smaller than parameter L_1 |
| l_2 | Bool | Logic 1, when input value X_1 is smaller than parameter L_2 |

| Parameter | | | | | | | |
|-----------|-------------------|-------|---|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| H_1 | H 1 | Float | High alarm 1 and upper output signal limiting | r/w | 9999.0 | | |
| H_2 | H 2 | Float | High alarm 2 | r/w | 9999.0 | | |
| L_1 | L 1 | Float | Low alarm 1 and lower output signal limiting (only valid, if higher than H_1). | r/w | -9999.0 | | |
| L_2 | L 2 | Float | Low alarm 2 | r/w | -9999.0 | | |
| X_0 | Offset X 0 | Float | Offset x0 of the variable X_1 to be monitored | r/w | 0.0 | | |
| Xsd | Switch difference | Float | A switching difference prevents excessively frequent alarm output switching. | r/w | 1.0 | >0.0 | |

| Configuration | | | | | | | |
|---------------|-----------------|--------|---|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Select | Monitoring type | Enum | Selection of the variable to be monitored: either input value X_1, or its rate of change (D-alarm), or input value X_1 shifted by a fixed offset. | r/w | 0 | | |
| | | x1 | X = x1: The value at input X_1 is monitored. | | 0 | | |
| | | dx1/dt | X = dx1/dt: The rate of change of the values at the X_1 input is monitored. | | 1 | | |
| | | x1-x0 | X = x1-x0: The value at the X_1 input shifted by offset X0 is monitored. | | 2 | | |

III-8.2 ALLV (alarm and limiting with variable limits (No. 41))



ALLV

Abb. 451

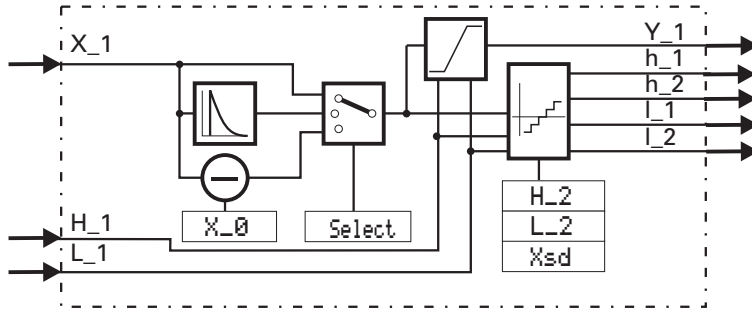


Abb. 452

Signal limiting:

Analog input **H_1** determines the maximum limiting, **L_1** determines the minimum limiting. **Y_1** is limited to the range between **L_1** and **H_1** ($L_1 \leq Y_1 \leq H_1$).

As both **H_1** and **L_1** come from analog inputs, **H_1** can be smaller than **L_1**. In this case, **H_1** is assigned a higher priority. This means that signal **Y_1** is $\leq H_1$!

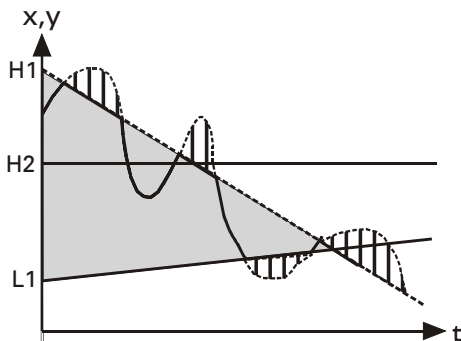


Abb. 453: Limiting at $H_1 < L_1$

Limit signaller:

The limit signaller has 2 low and high alarms (L1, L2, H1 and H2). The variable to be monitored can be selected with configuration parameter **Select** ($x_1, dx_1/dt, x_1 - x_0$).

The limit values are freely adjustable via the analog inputs **H_1** and **L_1** and have an adjustable hysteresis of ≥ 0 . The smallest separation between a minimum and a maximum limit value is 0. With an alarm triggered, the relevant output (L1, L2, H1 and H2) is logic "1".

D -Alarm (dx1/dt)

Value $x_1(t-1)$ measured one sampling interval before is subtracted from instantaneous value $x_1(t)$. This difference is divided by calculation cycle time T_r (e.g. 100ms, cycle time for this block, that is set in the engineering).

Thus input variable **X_1** can be monitored for rate of change.

Alarm mit Offset (x1 - x0):

x_1 can be shifted by means of X_0 . This corresponds to the offset of alarm limits (L_1 , L_2 , H_1 und H_2) in parallel to the x-axis.

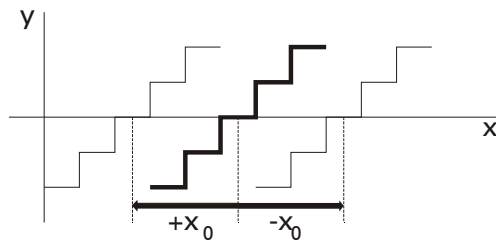


Abb. 454: Offset of the alarm limits

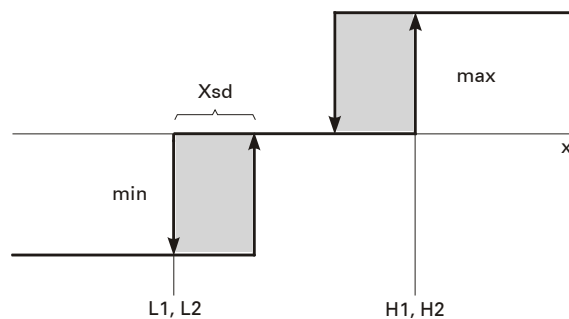


Abb. 455: Switching hysteresis and alarm limits:

In-/Outputs

| Name | Type | Description |
|------|-------|---|
| X_1 | Float | Input value to be monitored |
| H_1 | Float | High alarm 1, upper output signal limiting |
| L_1 | Float | Low alarm 1, lower output signal limiting (valid only, when it is higher than H_1). |

| Name | Type | Description |
|------|-------|---|
| Y_1 | Float | Calculated and limited input signal X_1 |
| h_1 | Bool | High alarm 1 - is logic 1, when input value X_1 is higher than input value H_1. |
| h_2 | Bool | High alarm 2 - is logic 1, when input value X_1 is higher than parameter H_2. |
| l_1 | Bool | Low alarm 1 - is logic 1, when input value X_1 is smaller than input value L_1. |
| l_2 | Bool | Low alarm 2 - is logic 1, when input value X_1 is smaller than parameter L_2. |

Parameter

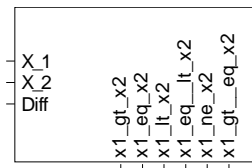
| ID | Name | Type | Description | Access | Default | Range | Off |
|-----|------|-------|--------------|--------|---------|-------|-----|
| H_2 | H 2 | Float | High alarm 2 | r/w | 9999.0 | | |

| | | | | | | | |
|-----|-------------------|-------|--|-----|---------|------|--|
| L_2 | L 2 | Float | Low alarm 2 | r/w | -9999.0 | | |
| X_0 | Offset X0 | Float | Offset x0 of variable X_1 to be monitored | r/w | 0.0 | | |
| Xsd | Switch difference | Float | A switching difference prevents excessively frequent alarm output switching. | r/w | 1.0 | >0.0 | |

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|--------|-----------------|--------|--|--------|---------|-------|-----|
| Select | Monitoring type | Enum | Selection of the variable to be monitored: either input value X_1, or its rate of change (D -alarm), or input value X_1 shifted by a fixed offset X_0. | r/w | 0 | | |
| | | X1 | X = x1: The value at input X_1 is monitored. | | 0 | | |
| | | dX1/dt | X = dx1/dt: The rate of change of the values at the X_1 input is monitored. | | 1 | | |
| | | X1-X0 | X = x1-x0: The value at the X_1 input shifted by offset X0 is monitored. | | 2 | | |

III-8.3 EQUAL (comparison (No. 42))



EQUAL

Abb. 456

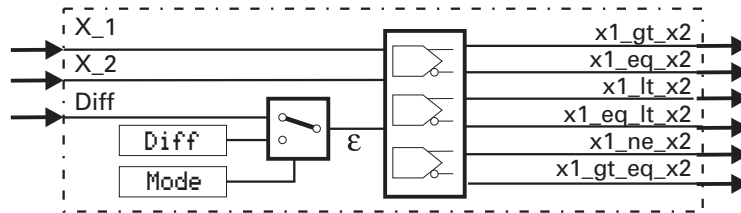


Abb. 457

The function checks the two analog input values **X_1** and **X_2** for equality.

The values are equal, if the amount of their difference is smaller than or equal to the preset tolerance.

| Comparison conditions | x1_gt_x2 | x1_eq_x2 | x1_lt_x2 | x1_eq_lt_x2 | x1_ne_x2 | x1_gt_eq_x2 |
|---------------------------------------|----------|----------|----------|-------------|----------|-------------|
| $X_2 + Diff < X_1$ | 1 | 0 | 0 | 0 | 1 | 1 |
| $X_2 - Diff \leq X_1 \leq X_2 + Diff$ | 0 | 1 | 0 | 1 | 0 | 1 |
| $X_2 - Diff > X_1$ | 0 | 0 | 1 | 1 | 1 | 0 |

The tolerance can be adjusted either as parameter **Diff** (**Mode** = Para. **Diff**) or entered at analog input **Diff** (**Mode** = Inp. **Diff**).

| In-/Outputs | | |
|-------------|-------|--|
| Name | Type | Description |
| X_1 | Float | 1st input value to be compared |
| X_2 | Float | 2nd input value to be compared |
| Diff | Float | Tolerance for comparison operations. If the difference of input values is smaller than Diff, input values are considered as equal. |

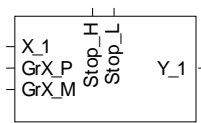
| Name | Type | Description |
|-------------|------|---|
| x1_gt_x2 | Bool | Is logic 1, when input value X_1 is higher than (input value X_2 - Diff). |
| x1_eq_x2 | Bool | Is logic 1, when input value X_1 is higher than or equal to (input value X_2 - Diff) and smaller than or equal to (input value X_2 + Diff). |
| x1_lt_x2 | Bool | Is logic 1, when input value X_1 is smaller than (input value X_2 + Diff). |
| x1_eq_lt_x2 | Bool | Is logic 1, when input value X_1 is smaller than or equal to (input value X_2 + Diff). |

| | | |
|-------------|------|--|
| x1_ne_x2 | Bool | Is logic 1, when input value X_1 is smaller than (input value X_2 - Diff) or higher than (input value X_2 + Diff). |
| x1_gt_eq_x2 | Bool | Is logic 1, when input value X_1 is higher than or equal to (input value X_2 - Diff). |

| Parameter | | | | | | | |
|-----------|------------------|------------------|--|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Mode | Tolerance source | Enum | Setting which determines, if tolerance Diff is defined by the parameter or by the input. | r/w | 0 | | |
| | | Parameter 'Diff' | The two analog input values X_1 and X_2 are considered as equal, when the amount of their difference is smaller than or equal to the predefined tolerance; this difference can be adjusted using parameter Diff. | | 0 | | |
| | | Input 'Diff' | The two analog input values X_1 and X_2 are considered as equal, when the amount of their difference is smaller than or equal to the predefined tolerance; this difference is read by input Diff. | | 1 | | |
| Diff | Diff | Float | Tolerance for comparison operations. If the difference of input values is smaller than Diff, input values are considered as equal. | r/w | 0.0 | >0.0 | |

No Configuration parameters!

III-8.4 VELO (rate-of-change limiting (No. 43))



VELO

Abb. 458

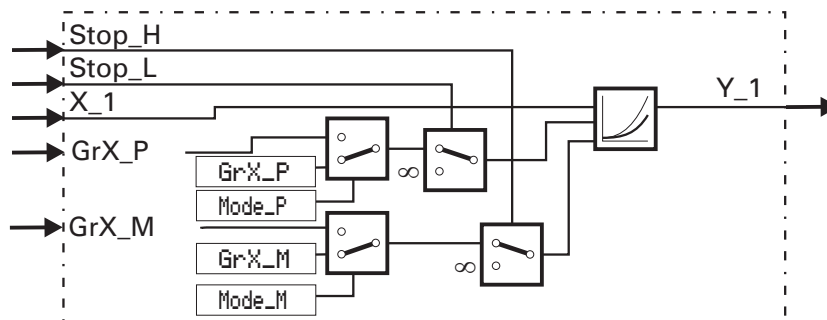


Abb. 459

The function passes input variable X_1 to output Y_1 and limits its rate of change dx_1/dt to a positive and negative gradient.

The gradients can be adjusted either as parameter GrX_P and GrX_M or preset at analog inputs GrX_P and GrX_M . Switch-over between the gradient sources is by parameter $Mode_P$ for the positive gradient and by $Mode_M$ for the negative gradient.

Via digital inputs $Stop_H$ and $Stop_L$, limiting can be switched off separately for positive and negative rates of change. Y_1 then follows X_1 without delay.

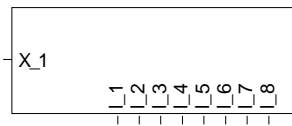
When using the analog inputs for gradient adjustment, the following is applicable:
 $GrX_P \geq 0$ or $GrX_M \leq 0$, otherwise the relevant gradient is set to 0.

| In-/Outputs | | |
|-------------|-------|---|
| Name | Type | Description |
| X_1 | Float | Input variable the rate of change of which is to be limited |
| GrX_P | Float | Positive gradient [physical unit/seconds], when the input is selected as a source of the positive gradient ($Mode_P = \text{Inp. } GrX +$). |
| GrX_M | Float | Negative gradient [physical unit/seconds], when the input is selected as a source of the negative gradient ($Mode_M = \text{Inp. } GrX -$). |
| $Stop_H$ | Bool | Control of positive gradient. $Stop_H = 0$: the selected gradient is effective. $Stop_H = 1$: the gradient is not effective. |
| $Stop_L$ | Bool | Control of negative gradient. $Stop_L = 0$: the selected gradient is effective. $Stop_L = 1$: the gradient is not effective. |
| Name | Type | Description |
| Y_1 | Float | Output value, follows input value X_1 with limited speed |

| Parameter | | | | | | | |
|-----------|---------------------|-------------------|--|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Mode_P | Source pos.gradient | Enum | Source of the positive gradient is either input GrX_P or parameter GrX_P. | r/w | 0 | | |
| | | Parameter 'GrX P' | Gradient Plus, i.e. the limiting for the rate of increase, is adjustable via parameter GrX P. | | 0 | | |
| | | Input 'GrX P' | Gradient Plus, i.e. the limiting for the rate of increase, is read by input GrX P. | | 1 | | |
| Mode_M | Source neg.gradient | Enum | Source of the negative gradient is either input GrX_M or parameter GrX_M. | r/w | 0 | | |
| | | Parameter 'GrX M' | Gradient Minus, i.e. the limiting for the rate of decrease, is adjustable via parameter GrX M. | | 0 | | |
| | | Input 'GrX M' | Gradient Minus, i.e. the limiting for the rate of decrease, is read by input GrX M. | | 1 | | |
| GrX_P | GrX + | Float | Positive gradient [physical unit/seconds], when the parameter is selected as a source of the positive gradient (Mode_P = Para. GrX +). | r/w | 0.0 | >0.0 | |
| GrX_M | GrX - | Float | Negative gradient [physical unit/seconds], when the parameter is selected as a source of the negative gradient (Mode_M = Para. GrX -). | r/w | 0.0 | | |

No configuration parameters!

III-8.5 LIMIT (multiple alarm (No. 44))



LIMIT

Abb. 460

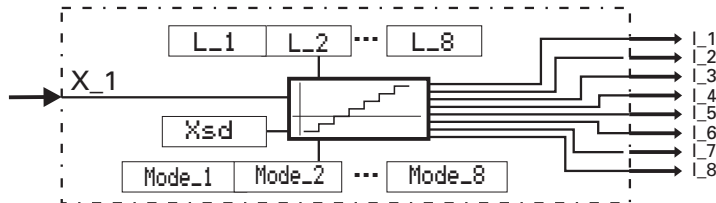


Abb. 461

The function checks input variable X_1 for 8 alarm values $L_1 \dots L_8$. Dependent of configuration by $Mode_1 \dots Mode_8$, the relevant alarm value is evaluated as MAX or MIN alarm.

With MAX alarm configuration, the alarm is triggered when the input signal is higher than the alarm value and finished when it is lower than (alarm value - hysteresis X_{sd}).

With MIN alarm configuration, the alarm is triggered when the input signal is lower than the alarm value and finished when it is higher than (alarm value + hysteresis X_{sd}).

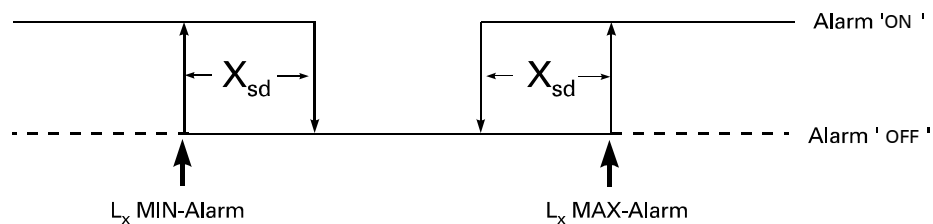


Abb. 462

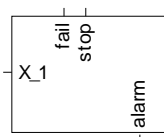
| In-/Outputs | | |
|-------------|-------|--|
| Name | Type | Description |
| X_1 | Float | Input variable, is monitored for 8 alarm values. |

| Name | Type | Description |
|-----------------|------|--|
| $I_1 \dots I_8$ | Bool | Alarm status of alarm1, $I_1 = 0$: no alarm; $I_1 = 1$: alarm case |

| Parameter | | | | | | | |
|-----------------|--------------------|-------|---|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| $L_1 \dots L_8$ | L 1 | Float | Trigger point for alarm 1 | r/w | 0.0 | | |
| Xsd | Switch. Hysteresis | Float | Switching hysteresis Xsd prevents excessively frequent alarm switching. | r/w | 0.0 | >0.0 | |

| Configuration | | | | | | | |
|-------------------------|---------------------|--------------|--|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Mode_1 ... Mode_8 | Alarm alignment1 | Enum | MAX alarm: alarm is triggered, when the input signal exceeds the limit value and finished, when the input signal falls below (limit value - hysteresis Xsd). MIN alarm: alarm is triggered, when the input signal exceeds the limit value and finished, when the input signal is higher than (limit value + hysteresis Xsd). | r/w | 0 | | |
| | | MAX alarm | The alarm is triggered when the input signal exceeds the alarm limit and finished, when the input signal falls below (alarm limit - hysteresis Xsd). | | 0 | | |
| | | MIN alarm | The alarm is triggered when the input signal exceeds the alarm limit and finished, when the input signal is higher than (alarm limit + hysteresis Xsd). | | 1 | | |

III-8.6 ALARM (alarm processing (No. 45))



ALARM

Abb. 463

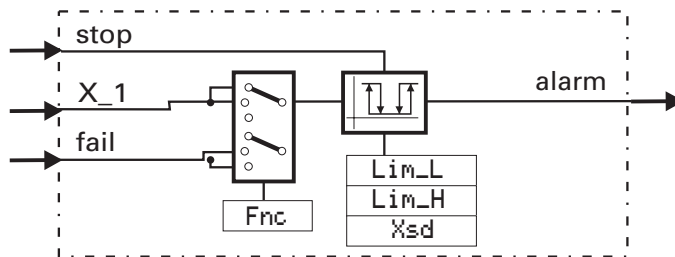


Abb. 464

X_1 is checked for a lower and an upper alarm value. Additionally, digital alarm input **fail** can be used. Configuration parameter **Fnc** can be used to select which signal shall be monitored (**X_1**, **X_1 + fail** or **fail**).

With input **stop** = 1, alarms (**fail** and **X_1**) are suppressed. After removal of this signal, suppression lasts, until the monitored value is again within the limits. This can be used e.g. for suppressing an alarm message with change or for quitting an alarm.

Suppression of the alarm ends with the signal changing directly from high alarm to low alarm or from low alarm to high alarm. In this case the alarm is activated, though the monitored value has not been within the limits.

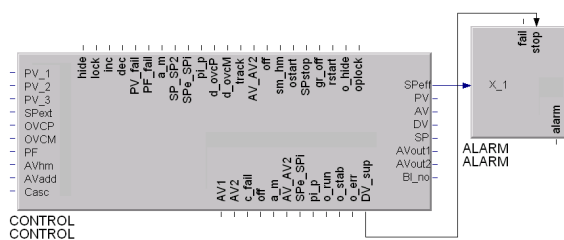


Abb. 465

During value change at the exit **DV_sup** a pulse with the length of a scanning cycle T_s is sent.

| In-/Outputs | | |
|-------------|-------|--|
| Name | Type | Description |
| X_1 | Float | Input variable to be monitored, is checked for a min and a max limit value. |
| fail | Bool | Digital alarm signal e.g. fail signal of an analog input |
| stop | Bool | stop = 1: the alarms (fail and X_1) are suppressed. After stop returned to 0, suppression lasts, until the monitored value is again within the limits. |

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|-------|------|--|
| alarm | Bool | Alarm status: alarm = 0: no alarm; alarm = 1: alarm case |
|-------|------|--|

Parameter

| ID | Name | Type | Description | Access | Default | Range | Off |
|------|-------------------|-------|--|--------|---------|-------|-----|
| LimL | Lower alarm limit | Float | Lower limit value for the alarm | r/w | -10.0 | | |
| LimH | Upper alarm limit | Float | Upper limit value for the alarm | r/w | 10.0 | | |
| Lxsd | Switch difference | Float | A switching difference prevents excessively frequent alarm output switching. | r/w | 0.0 | >0.0 | |

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|-----|----------|-------------------|--|--------|---------|-------|-----|
| Fnc | Function | Enum | The alarm function can be limited to input X_1, or to input fail, or to monitoring both inputs X_1 and fail. | r/w | 0 | | |
| | | Measured value X1 | X_1 is checked for a min. and a max. alarm value. | | 0 | | |
| | | X1 + fail | X_1 is checked for a min. and a max. alarm value. Additionally, the digital alarm input is switched to fail. | | 1 | | |
| | | fail | The digital alarm input fail is monitored. | | 2 | | |

III-9 Human-machine interface

III-9.1 General information

The HMI library can be used to provide applications with individually designed, convenient operating pages. In the BlueDesign terminology, these operating pages are called "HMI" (Human-Machine Interface).

To start with, open BlueDesign and select the "Mask Designer" (main menu "Tools": "Mask Designer", also refer to the information given under "Note"). In the Mask Designer, "drag & drop" the objects of your user interface to position them conveniently. Then, connect all objects of your HMI that should interact with the application to function blocks of the **Lib401 HMI library**.

This chapter describes the function blocks of the **Lib401 HMI library** and the related HMI objects in the mask editor.



NOTE

*Detailed information related to working with the Mask Designer is also given under **Application visualizing** in Chapter II: Development environment.*

Creating a new object in the Mask Designer

The procedure for creating HMI objects is:

1. **Creating:** Open the Mask Designer and select tab "**Masks and Objects**" to create a new object (new object or new mask) using context menu command "New Object" or "New Mask".
2. **Adapting:** A new object (or a new mask) is created with default settings on the worksheet. Change the name and the appearance of the object (or of the mask) in the related property dialogue (see below for the relevant function block: Property table and parameter table). Shift the object (or the mask) into the required position.
3. **Object type:** As default, the type of a new object is "unknown". Determine the type of the block in window "Object" → "Type".

Connecting function blocks and HMI objects

To enable a function block from library "**LIB401 HMI**" to interact with a HMI object they must be assigned to each other. Only if this prerequisite is met, HMI entries are taken over into the function block and the HMI object is able to react on the function block, e.g. for display of function block values in the HMI.

1. **Program block:** Select the program block with the inserted function blocks from library "**LIB401 HMI**".



NOTE

It is always purposeful to save the function blocks related to a particular HMI in an own program or macro block.

2. **Connecting a function block to the HMI:** Open the Mask Designer via the parameter dialog (context menu) of the function block from the "LIB401 HMI" library. In the Mask Designer, the function block must be assigned to a suitable HMI object. Click the required HMI object in the tree structure.

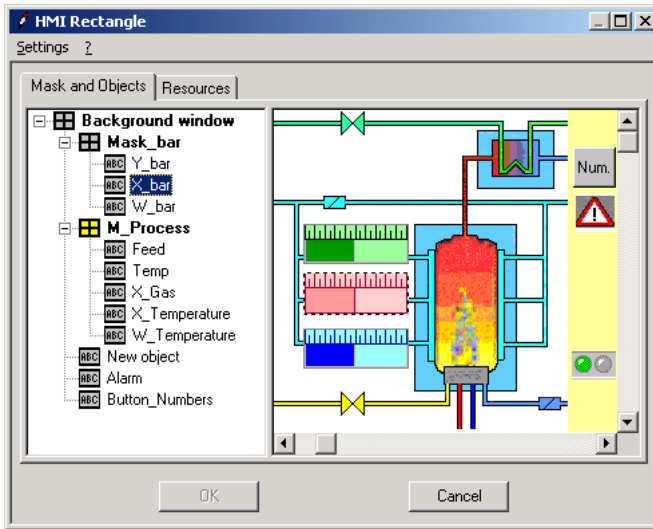


Fig. 1

3. **Saving the entry:** Click button "OK" to store the entry.



NOTE

Button "OK" is active only, if the function block is of the same type as the selected HMI object.

4. **Checking:** In addition to the function block type, the program block on the worksheet contains the name of the HMI object and the mask (or the background window) with the HMI object under the function block.

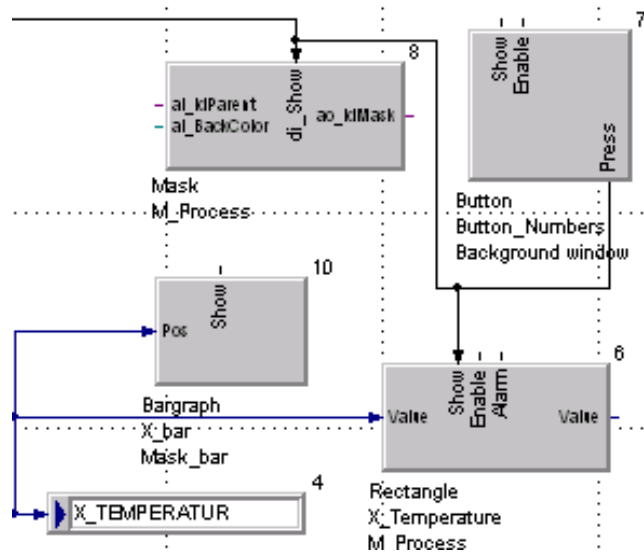
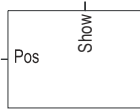


Fig. 2

III-9.2 Bargraf (display of a value as the length of a bar (no. 401))



Bargraph

Abb. 466

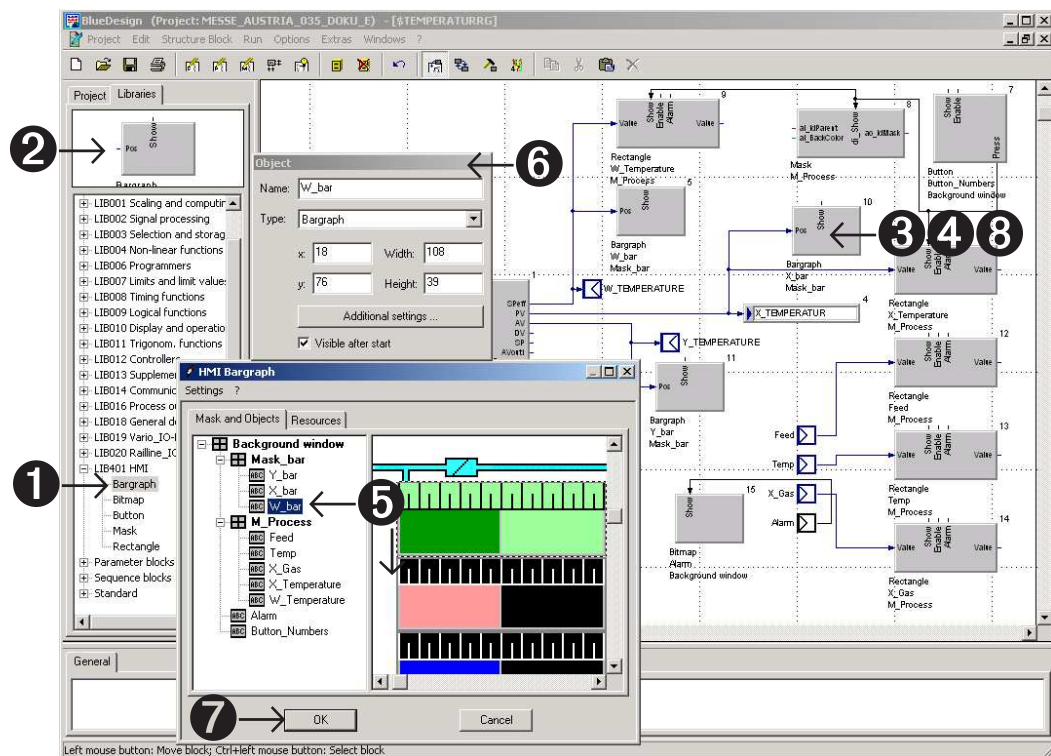


Abb. 467

- ❶ Select the function block from library LIB401 HMI.
- ❷ Click the function block and drag it onto the worksheet.
- ❸ Drop the function block to position it in the engineering as required.
- ❹ Click with the right mouse key to open the Mask Designer via the parameter dialog.
- ❺ Create a "New Object", adjust its size and shift it.
- ❻ Define a name, select a type and continue with the "Additional settings ...".
- ❼ Click "OK" to confirm. The function block name and the name of the background window or of the mask are displayed in the engineering (see item 8.)
- ❽ Function block in the engineering with type, name of the related HMI object and name of the related mask (or of the background window).

The bargraph is used to indicate a variable as a bar.

The bargraph indicates an overflow: When exceeding the max. limit of the display range, the colour changes and the tip of a triangle is displayed.

The colours for the bar, the overflow, the scale, the background and the frames are adjustable freely.

The scale, i.e. the number of graduation ticks and sub-ticks, is adjustable.

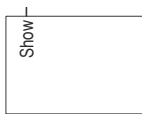
| Inputs | | |
|---------------|-------------|--|
| Name | Type | Description |
| X_1 | Float | Input variable to be delayed |
| reset | Bool | Delay switch-off, 1 = delay is ineffective, output is passed to the input immediately. |

| Properties | |
|---------------------|---|
| Name | Description |
| Name | Name of the object (as displayed in the tree structure). |
| Type | Selection of the object type ("Bargraph"). |
| Width/Height | Size of the object (in pixels). An object must not be bigger than the size of the target instrument display. For the <i>KS 108 easy</i> this value is 320 x 240 pixels. |
| Additional settings | Click button "Additional settings" to determine the properties of an object (see Parameters). |
| Visible after start | Indicates if the object is visible after starting the application. However, note that an object can be hidden by another one. If several objects are displayed at the same time, parts of an object or the overall object may be not visible. |

| Parameters | | | | | | | |
|-------------------|---------------|-------------|---|---------------|--------------------|--------------|------------|
| ID | Name | Type | Description | Access | Default | Range | Off |
| WndBack | ColorWndBack | Int | Background colour of the window, i.e. background colour of the scale. | r/w | HMI_C256_BLACK | 0 ... 255 | |
| WndFrame | ColorWndFrame | Int | Colour of the frame around the bargraph (with scaling). | r/w | HMI_C256_DARKGRAY | 0 ... 255 | |
| BarBack | ColorBarBack | Int | Colour of the background behind the display bar. | r/w | HMI_C256_BLACK | 0 ... 255 | |
| BarFrame | ColorBarFrame | Int | Colour of the inner frame (only around the bargraph, not around the scale). | r/w | HMI_C256_LIGHTGRAY | 0 ... 255 | |

| | | | | | | |
|---------|------------------|-------|---|-----|---------------------|--------------------------|
| BarNorm | ColorBarNormal | Int | Colour for display of the bar as long as the display value is within the scale limits. | r/w | HMI_C256_LIGHTGREEN | 0 ... 255 |
| BarOver | ColorBarOverflow | Int | Colour of the bar in the event of overflow, i.e. when the display value exceeds the full scale. | r/w | HMI_C256_LIGHTRED | 0 ... 255 |
| Scale | ColorScale | Int | Colour of graduation ticks. | r/w | HMI_C256_WHITE | 0 ... 255 |
| TicMain | CountTicsMain | Int | Number of main graduation ticks. | r/w | 11 | 2 ... 101 |
| TicSub | CountTicsSub | Int | Main scale graduation, number of graduation sub-ticks | r/w | 1 | 1 ... 10 |
| Min | ValueMin | Float | Lower end of scale. When exceeded, the bargraph is empty. | r/w | 0.0 | -999999 ... 999999 |
| Max | ValueMax | Float | Upper end of scale. When exceeded, the colour of the bar changes from "normal" to "overflow". | r/w | 100.0 | -999999 ... 999999 |

III-9.3 Bitmap (display of a graph (no. 406))



Bitmap

Abb. 468

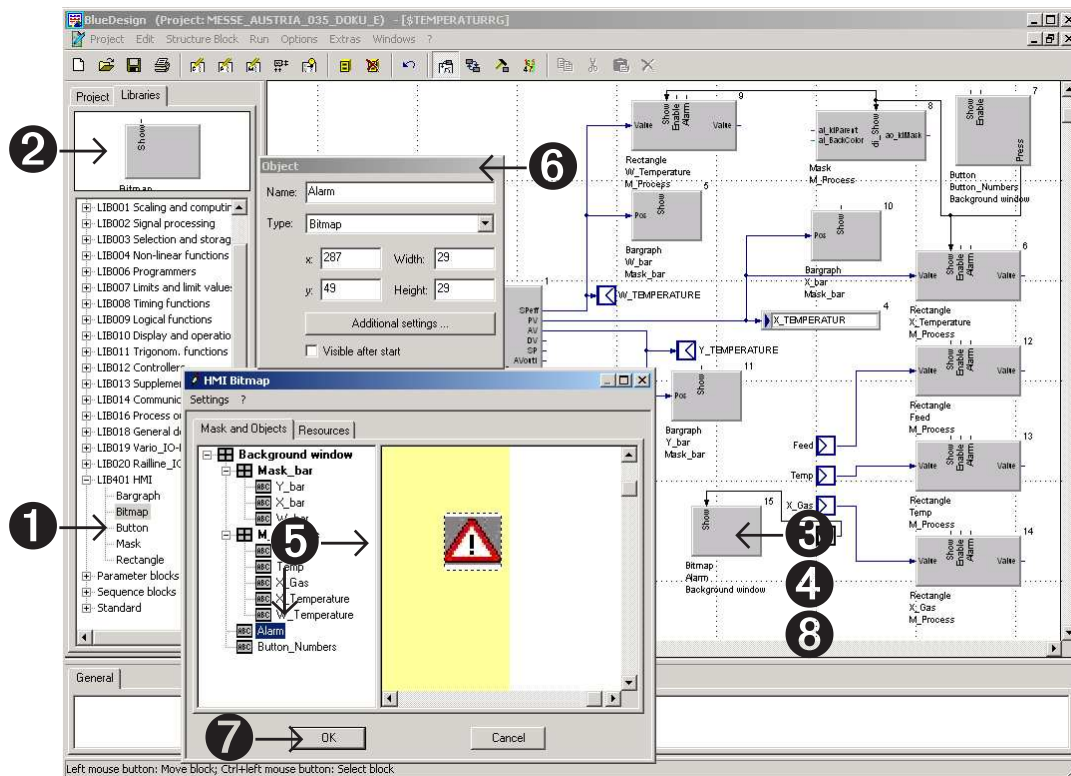


Abb. 469

- ❶ Select the function block from library LIB401 HMI
- ❷ Click the function block and drag it onto the worksheet.
- ❸ Drop the function block to position it in the engineering as required.
- ❹ Click with the right mouse key to open the Mask Designer via the parameter dialogue.
- ❺ Create a "New Object", adjust its size and shift it.
- ❻ Define a name, select a type and continue with the "Additional settings ...".
- ❼ Click "OK" to confirm. The function block name and the name of the background window or of the mask are displayed in the engineering (see item 8.)
- ❽ Function block in the engineering with type, name of the related HMI object and name of the related mask (or of the background window).

HMI block **Bitmap** can be used to include a graph.

Display and display suppression of the graph are possible via digital input Show of function block **Bitmap**.

Graphs must be provided in the required size in "**bmp**" format. They must be imported; when importing a graph, its colours are adapted for the target system. Definition of a "**transparent**" colour is possible.

**NOTE!**

To start with, a **bitmap** must be imported into tab "**Resources**". For this, see also "Adding resources (Bitmap)" in chapter II.

Enter the imported graphs as parameters: click column "**Value**" under "**Additional settings...**" → "**Resources**" to open a list of available **bmp** graphs. Now the graph can be selected.

**NOTE!**

A bitmap cannot be changed in BlueDesign. If the selected area is smaller than the bitmap area, a bitmap cut-out is displayed.

Input

| Name | Type | Description |
|------|------|---|
| Show | Bool | Display of the bitmap via digital input, Show = 1: The bitmap is visible. Only effective, if "Visible after start" (in the parameter dialog -> Object) is de-activated. |

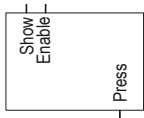
Properties

| Name | Description |
|---------------------|---|
| Name | Name of the object (as displayed in the tree structure). |
| Type | Selection of the object type ("Bitmap"). |
| Width/height | Size of the object (in pixels). An object must not be bigger than the size of the target instrument display. For the <i>KS 108 easy</i> this value is 320 x 240 pixels. |
| Additional settings | Click button "Additional settings" to define the properties of an object (see Parameter). |
| Visible after start | Indicates if the object is visible after starting the application. However, note that an object can hide another one. If several objects are displayed at the same time, parts of an object or the overall object may be not visible. |

Parameter

| ID | Name | Type | Description | Access | Default | Range | Off |
|----------|----------|------|---|--------|---------|-------------|-----|
| Resource | Resource | Int | Selection of the bitmap. After clicking "Value", the bitmaps imported under Resources in the Mask Designer are offered as a list. | r/w | -1 | 0 ... 32767 | |

III-9.4 Button (button field (no. 402))



Button

Abb. 470

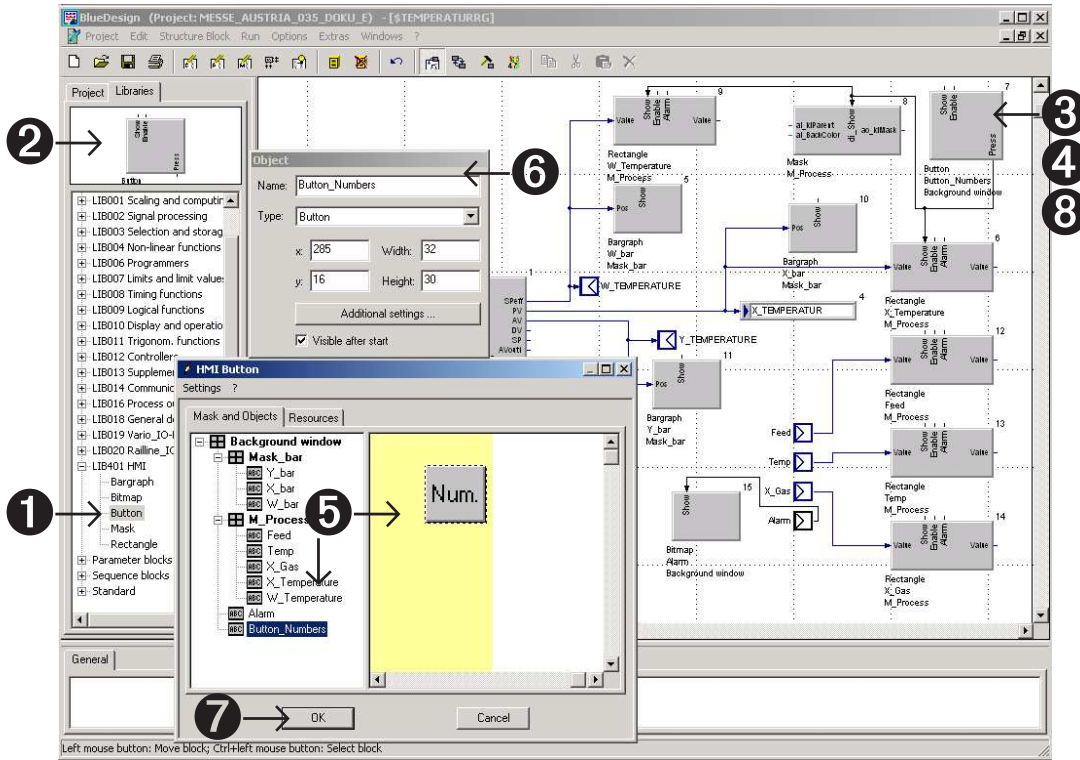


Abb. 471

1. Select the function block from library LIB401 HMI.
2. Click the function block and drag it onto the worksheet.
3. Drop the function block to position it in the engineering as required.
4. Click with the right mouse key to open the Mask Designer via the parameter dialog.
5. Create a "New Object", adjust its size and shift it.
6. Define a name, select a type and continue with the "Additional settings ...".
7. Click "OK" to confirm. The function block name and the name of the background window or of the mask are displayed in the engineering (see item 8.).
8. Function block in the engineering with type, name of the related HMI object and name of the related mask (or of the background window).

HMI block **Button** can be used to select a push-button. The output is switched on by pressing the button and off again by releasing it.

The button can be provided with a text, blanked out and disabled.

| Inputs | | |
|--------|------|--|
| Name | Type | Description |
| Show | Bool | Display of the button via digital input, Show = 1: The button is visible. Effective only, if |

| | | |
|--------|------|---|
| | | "Visible after start" (in the parameter dialog -> Object) is de-activated. |
| Enable | Bool | Enables operation of the button. Enable = 1 : The output of the button can be set. Enable = 0 or not connected : The output can be set. |

Outputs

| Name | Type | Description |
|-------|------|---|
| Press | Bool | The output is set by pressing the button. |

Properties:

| Name | Description |
|---------------------|---|
| Name | Name of the object (as shown in the tree structure). |
| Type | Selection of the object type ("Button"). |
| Width/height | Size of the object (in pixels). An object must not be bigger than the size of the target instrument display. For <i>KS 108 easy</i> this value is 320 x 240 pixels. |
| Additional settings | Click button "Additional settings" to determine the properties of an object. |
| Visible after start | Indicates if the object is visible after starting the application. However, note that an object can hide another one. If several objects are displayed at the same time, parts of an object or the overall object may be not visible. |

Parameter

| ID | Name | Type | Description | Access | Default | Range | Off |
|-------|-------|------|---------------------------------|--------|------------|-------|-----|
| Title | Title | Text | Text to be shown on the button. | r/w | HMI Button | | |

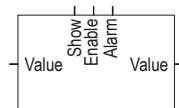
III-9.5 Rectangle (display or input field for value and text (no. 403))**Rectangle**

Abb. 472

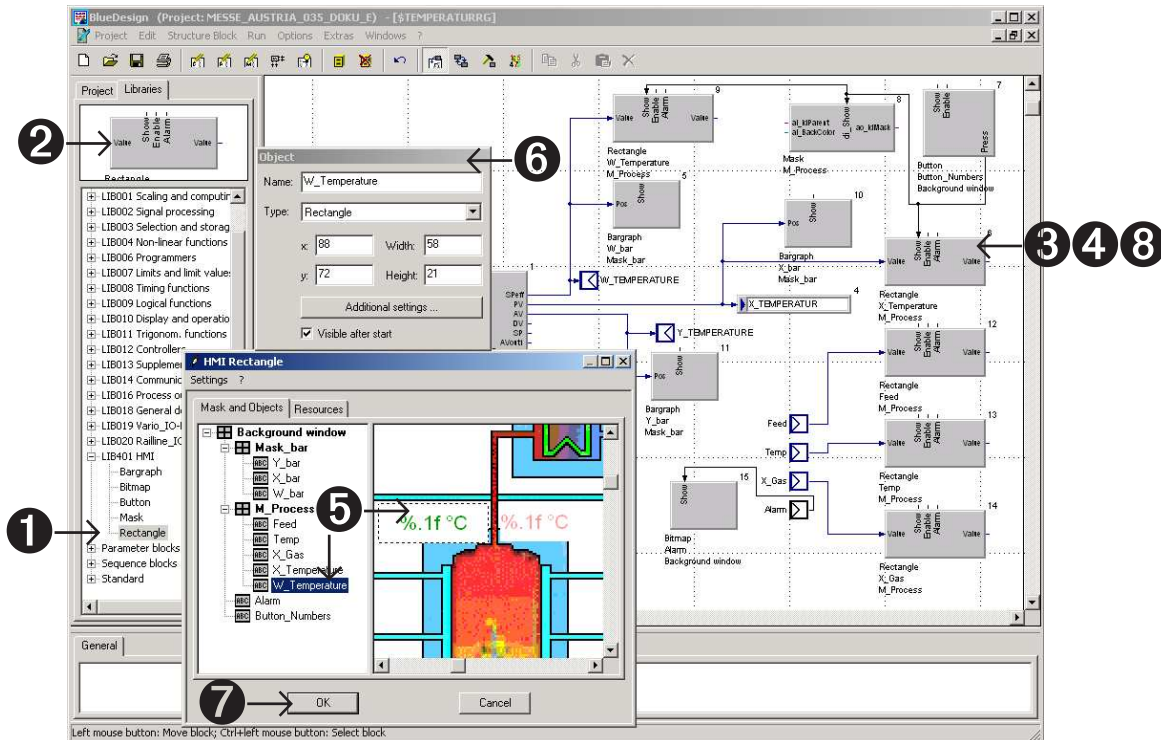


Abb. 473

- ❶ Select the function block from library LIB401 HMI.
- ❷ Click the function block and drag it onto the worksheet.
- ❸ Drop the function block to position it in the engineering as required.
- ❹ Click with the right mouse key to open the Mask Designer via the parameter dialog.
- ❺ Create a "New Object", adjust its size and shift it.
- ❻ Define a name, select a type and continue with the "Additional settings ..."
- ❼ Click "OK" to confirm. The function block name and the name of the background window or of the mask are displayed in the engineering (see item 8).
- ❽ Function block in the engineering with type, name of the related HMI object and name of the related mask (or of the background window).

HMI object **Rectangle** can be used for display of a value. The value can be fixed or transmitted via the function block.

Specify a fixed value (a constant) in column "Value". If you want to use a value transmitted via a function block, however, you must enter "%.1f". These characters indicate *that* a value must be output. Additionally, you specify *how* this value should be formatted.

Format: **%.*(number of digits behind the decimal point)*(format)**

The digit in position (number of digits behind the decimal point) indicates the displayed number of digits behind the decimal point. If necessary, these are completed with "0" at the end. The other digits behind the decimal point are rounded off: up to 5, digits are rounded off to the next lower value and above 5, digits are rounded off to the next higher value.

Example 1: If **%.1f** is selected, value 124.850 is displayed as "124.8". Value 124.852 is displayed as "124.9".

Example 2: Three digits behind the decimal point are specified with **%.3f**. Value 124.8501563 is displayed as "124.850".

The "f" indicates that a value is of the "float" type; this is the default setting. If display with decimal power is required for scientific purposes, "e" can be selected instead of "f". Letter "e" marks the decimal power mode.

For example, a value of 124.84 is displayed as "124.8" if a format of **%.1f** is selected, but as "1.2e+002" if the specified format is **%.1e**. This means $1.2 \cdot 10^{+2}$. Negative values behind "e" indicate values smaller than 1: "3.5e-003" corresponds to a value of 0.0035. Format "g" is also possible. In this case, display is in format "f" or "e" dependent on which format is more convenient for a particular value.

Behind the value or instead of the value, a text can be inserted. For example, a unit or a physical quantity can be specified. For display of e.g. "12.6 kg", format **%.1f kg** or **%.1fkg** for "12.6kg" (shorter, because spaces are omitted) must be specified.

**HINT!**

To set the symbol %-as unit you have to place it twice "%%". For example "12.6%" is formatted with "%.1f%%".

For display of a bit value (0/1), 2 **rectangles** which are switched over must be used. Connect the bit signal to the related **Show** input directly for one **rectangle** and via a **NOT** function block for the other one.

Time display in a single **rectangle** is not possible. To solve the problem, split the time into separate values and combine several **rectangles**.

The colours of the **rectangle** can be switched over via a signal applied to the **alarm** input. Moreover, the text size and alignment can be selected under "Additional settings".

| Inputs | | |
|--------|-------|---|
| Name | Type | Description |
| Show | Bool | Display of the rectangle. Show = 1: The rectangle is visible. Effective only, if "Visible after start" (in the parameter dialog -> Object) is de-activated. |
| Enable | Bool | Enable operation of the rectangle value. Enable = 1 : The displayed value of the rectangle can be adjusted. Enable = 0 or not connected: The value cannot be changed. |
| Alarm | Bool | Activates the alarm colours of the rectangle. Alarm = 1 : Text, frame and background are displayed with the alarm colours rather than with the default colours. |
| Value | Float | Rectangle input value to be displayed, or fixed value. |

| Outputs | | |
|---------|-------|---|
| Name | Type | Description |
| Value | Float | Value of the rectangle used as an input value for a following function block. |

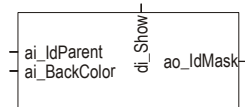
| Properties | |
|--------------|--|
| Name | Description |
| Name | Object name (as displayed in the tree structure). |
| Type | Selection of the object type ("Rectangle"). |
| Width/height | Object size (in pixels). An object may not be bigger than the size of the target instrument display. For <i>KS 108 easy</i> this value is 320 x 240 pixels. |

| | |
|---------------------|---|
| Additional settings | Click button "Additional settings" to define the properties of an object (see Parameters). |
| Visible after start | Indicates if the object is visible after starting the application. However, note that an object can hide another one. If several objects are displayed at the same time, parts of an object or the overall object may be not visible. |

| Parameter | | | | | | | |
|-----------|---------|------|---|--------|---------------------|-----------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Text | Text | Text | Rectangle display text. Value and / or text. Value as "%.1f". Additional format specifications: digit behind point = number of digits behind the decimal point, letter e (instead of f) means decimal power for scientific display. Example: For a value of 124.86, display is "124.9kg" with "%.1fkg" and "1.249e+02" corresponding to "1.249 * 10^2" with "%.3e". | r/w | %.1f | | |
| StdRect | StdRect | Int | Rectangle background colour (standard colour). | r/w | HMI_C256_LIGHTGREEN | 0 ... 255 | |
| StdBord | StdBord | Int | Colour of rectangle frame (standard colour). | r/w | HMI_C256_BLACK | 0 ... 255 | |
| StdText | StdText | Int | Colour of text displayed in rectangle (standard colour). | r/w | HMI_C256_BLACK | 0 ... 255 | |
| AlmRect | AlmRect | Int | Rectangle background colour with active digital input "Alarm". | r/w | HMI_C256_LIGHTRED | 0 ... 255 | |
| AlmBord | AlmBord | Int | Colour of rectangle frame with active digital input "Alarm". | r/w | HMI_C256_BLACK | 0 ... 255 | |
| AlmText | AlmText | Int | Colour of text displayed in rectangle with active digital input "Alarm". | r/w | HMI_C256_BLACK | 0 ... | |

| | | | | | | | |
|---------|---------|---------|---|-----|---|-----------|--|
| SizBord | SizBord | Int | Frame thickness in pixels. | r/w | 0 | 0 ... 100 | |
| SizTxt | SizTxt | Int | Text size. SizTxt = 0 : small letters, SizTxt = 1 : medium-sized letters, SizTxt = 2 : big letters. | r/w | 1 | 0 ... 2 | |
| AlgTxt | JstTxt | Enum | Text is left-adjusted, right-adjusted, or central | r/w | 0 | | |
| | | Central | The text is adjusted centrally, at equal distance from both sides. | | 0 | | |
| | | Left | The text is aligned on the left side. | | 1 | | |
| | | Right | The text is aligned on the right side. | | 2 | | |

III-9.6 Mask (mask field, grouping several HMI objects)



Mask

Abb. 474

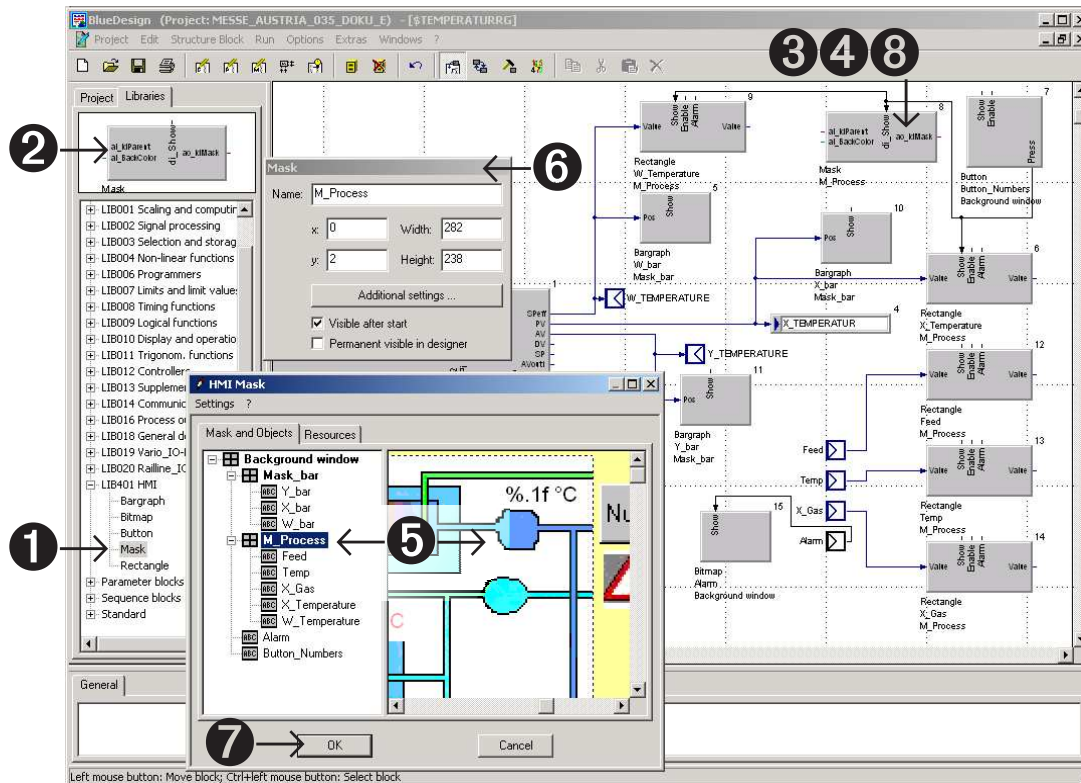


Abb. 475

1. Select the function block from library LIB401 HMI.
2. Click the function block and drag it onto the worksheet.
3. Drop the function block to position it in the engineering as required.
4. Click with the right mouse key to open the Mask Designer via the parameter dialog.
5. Create a "New mask", adjust its size and shift it.
6. Define a name, select a type and continue with the "Additional settings ...".
7. Click "OK" to confirm. The function block name and the name of the background window or of the mask appear in the engineering (see item 8.)
8. Function block in the engineering with type, name of the related HMI object and name of the related mask (or of the background window).

General information

A mask forms a group of several blocks. It is similar to a Windows dialog. Masks can be placed on top of each other, displayed or blanked out (together with all their blocks). For instance, "pop-up" windows can be realized with masks.

Switching over between masks

If you use several masks, you must avoid that a mask hides another one. Display of only a single mask at a time and display suppression of all other ones can be realized conveniently using the **FLIPM** block from the "Logic functions" library. If a "true" value is assigned to one of the data inputs of **FLIPM**, the corresponding data output is set "true" as well: The related mask is displayed. All other data outputs of **FLIPM** provide value "false": display of all other masks is suppressed.

Example: The three buttons ("Button 1", "Button 2" and "Button 3") open the related masks ("Mask 1", "Mask 2" and "Mask 3"). Simultaneously, display of the inactive masks is suppressed.

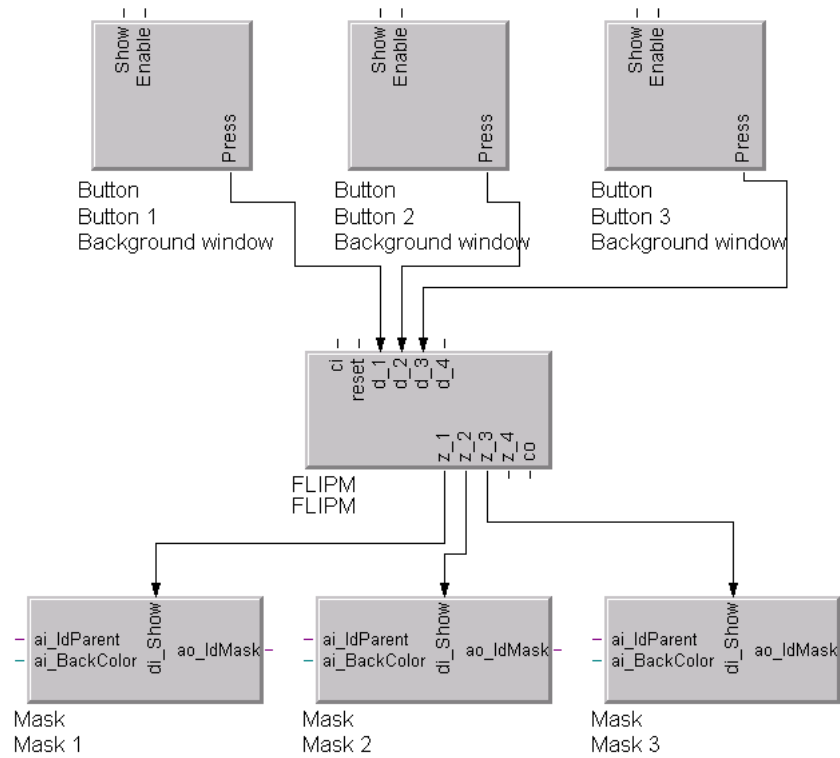


Abb. 476: Example: Use of the FLIPM block

Inputs

| Name | Type | Description |
|--------------|-------|---|
| di_Show | Bool | If value "true" (1) is assigned to the input, the block is displayed. |
| ai_idParent | Float | Masks can be connected with other masks. This is purposeful for display suppression of all subordinated masks when blanking out the superimposed mask. Connection of these masks is done via input "ai_idParent". This input is connected to output "ao_IdMask" of the superimposed mask. |
| ai_BackColor | Float | Via this input, a new colour can be assigned to a mask. |

Outputs

| Name | Type | Description |
|-----------|-------|--|
| ao_IdMask | Float | The output can be used for connecting a subordinated mask (see input "ai_idParent"). |

Properties

| Name | Description |
|--------------|---|
| Name | Name of the mask (as shown in the tree structure). |
| Width/height | Mask size (in pixels). A mask must not be larger than the size of the target instrument display. For KS 108 easy this value is 320 x 240 pixels. |
| Colour | Click button "Colour" to assign a different colour to the background window. |

| | |
|---------------------------------|---|
| Visible after start | Indicates if the mask is visible after starting the application. However, note that a mask can hide another one. If several masks are displayed at the same time, parts of a mask or the overall mask may be not visible. |
| Permanently visible in Designer | Select this option, if a mask should be visible also when another mask is activated. |

Parameter

| Name | Description |
|------------|--|
| Background | Colour of mask background (use only colours provided in the target system!) |
| Language | (Cannot be changed) |
| Bitmap | The mask can show a bitmap. When clicking in the field, a list of the bitmaps imported under Resources is offered. |



NOTE!

Bitmaps must be imported into tab "Resources" to start with. For this, see also "Adding resources" (bitmaps) in chapter II.

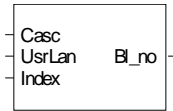


NOTE!

Bitmap changing is not possible in BlueDesign. If the selected area is smaller than the bitmap, a bitmap cut-out is displayed.

III-10 Visualization

III-10.1 TEXT (text container with language-dependent selection (No. 79))



TEXT

Abb. 477

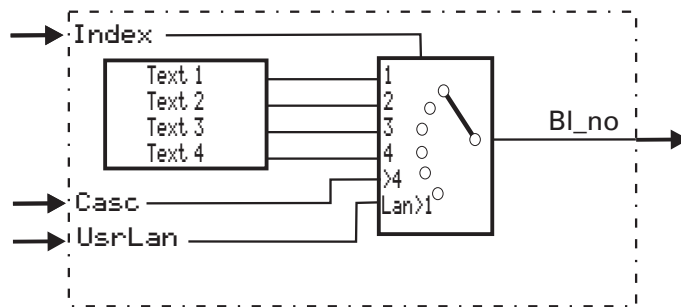


Abb. 478

The text block contains a list of user texts which can be displayed by various operating pages (e.g. A_PROG, V_DISPLAY, V_BAR). These texts can be displayed and adjusted as a selection list on a V_DISPLAY page (e.g. for plain text selection of recipes).

The function block is cascadable, if more than 4 texts are available for selection. Texts can be entered only via engineering: 4 texts of 19 characters.

Output "BI_no" of the last text block in a text cascade must be wired to the block on the operating page of which the texts must be used, e.g. V_DISPLAY. The number of the display text is allocated to the index input of this text block.

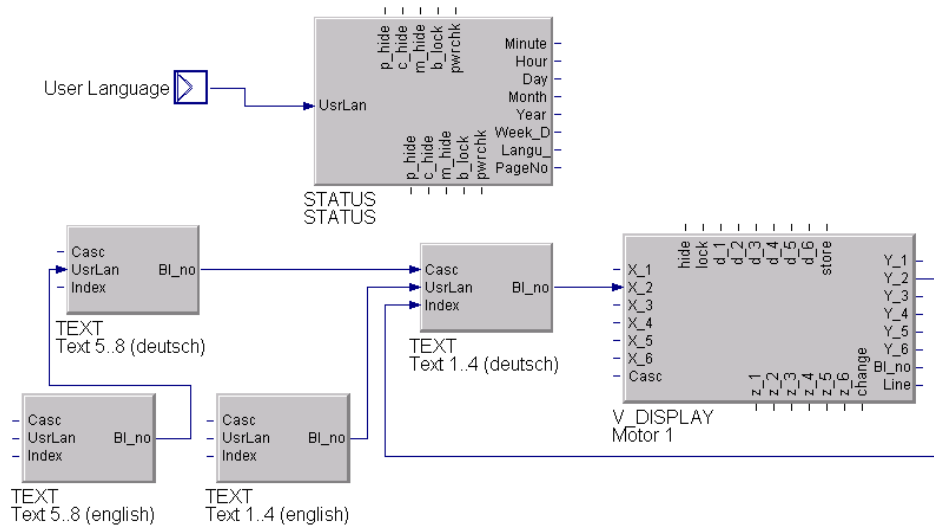


Abb. 479 Example of soft-wired text blocks

The texts can be extended into any number of texts via the cascade input (**Casc**). For this, wire the **BL_no** output of the subordinated block (texts 5 ...8) to input "**Casc**" of the next text block. The index for text selection is adjustable only at the index input of the last block (see example below).

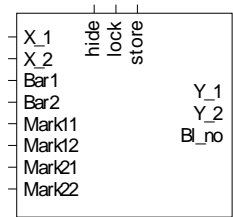
For user language selection, wire the index output of the (language) text block to language input **UsrLan** of the used text block. During selection of the user language, its texts will replace the texts of the first text block. User language selection is central at the status block.

| In-/Outputs | | |
|-------------|-------|--|
| Name | Type | Description |
| Casc | Float | Cascade input for further text blocks in the same language |
| UsrLan | Float | Input for a text block with texts in a further language. Switch-over is via input UsrLan at the status block. |
| Index | Float | Input for text selection. With cascaded text blocks, only the index output of the text block which is connected directly with the calling block is set. (The language-dependent texts are switched over via input UsrLan at the status block.) |

| Name | Type | Description |
|-------|-------|--|
| Bl_no | Float | Own block number. Output for connection to blocks which use the texts, e.g. V_DISPLAY. |

| Parameter | | | | | | | |
|-------------------------|--------|------|---|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Text_1 ... Text_4 | Text 1 | Text | User text 1. Language-dependent: also language-dependent texts when connecting to input UsrLan of another text block; these are switched over via input UsrLan at the status block. | r/w | TEXT_1 | | |

III-10.2 V_BAR (bargraf-display (No. 97))



V_BAR

Abb. 480

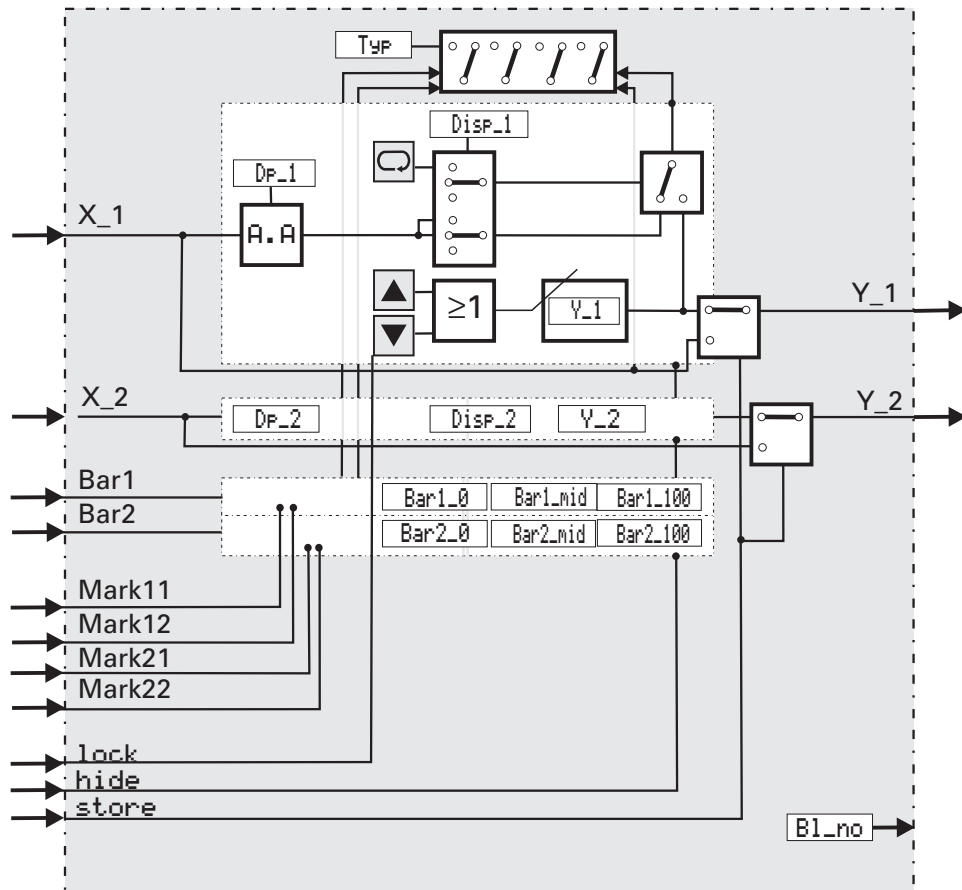


Abb. 481

General

This function permits the display of 2 analog input signals as bargraphs, and of 2 analog input signals as numeric values. Moreover, two analog output signals can be defined.

Another 4 analog inputs can be used to position 2 markings at each side of the bar within the bargraph range. These markings can be e.g. alarm limits or reference values. With open marker inputs, or out-of-range marker values, display of markers is suppressed.

Determination of horizontal or vertical bargraph is via configuration (**Type**). Determination if value displays are visible, changeable or switched off.

Configuration of start values **Bar1_mid** or **Bar2_mid** determines, if the bargraph points only in one direction (from top or bottom) or in 2 directions from the middle. The values applied to the inputs are displayed.

A value which is adjustable via the front panel is output at the relevant analog output. Changing these values from the operating level can be suppressed. Parameters **Y_1/Y_2** are the initial values for Power-On.

The output value is displayed only with the output fed back to the relevant input, or if the display for this value is in the adjustment mode. With a positive edge at the **store** input, the values applied to the signal inputs are stored as parameters **Y_1** and **Y_2**, i.e. as output values.

With digital input **lock** set, no values can be changed. With digital input **hide** set, the operating page cannot be displayed during operation.

A 19-digit text for the display header can be adjusted user-specifically via the engineering tool. The same is applicable for further texts for identification of value and unit.



Caution!

Too frequent storing operations may lead to the destruction of the EEPROM!

Therefore

Values of the used analog inputs are stored as parameter values when detecting a positive edge at the **store**- input. This input should be activated only with relevant changes of the input values.

| In-/Outputs | | |
|-------------|-------|--|
| Name | Type | Description |
| X_1 | Float | Valid process value to be displayed as a value. Upper left display field (default = 0). |
| X_2 | Float | Valid process value to be displayed as a value. Upper right display field (default = 0). |
| Bar1 | Float | Process value to be displayed as a bargraph. Top or left bargraph (default = 0) |
| Bar2 | Float | Process value to be displayed as a bargraph. Bottom or right bargraph (default = 0) |
| Mark11 | Float | Marking at top left of the first bar. If the marker input is open or out of the permitted range, display of the marker is suppressed. |
| Mark12 | Float | Marking at top left of the first bar. If the marker input is open or out of the permitted range, display of the marker is suppressed. |
| Mark21 | Float | Marking at bottom right of the second bar. If the marker input is open or out of the permitted range, display of the marker is suppressed. |
| Mark22 | Float | Marking at bottom right of the second bar. If the marker input is open or out of the permitted range, display of the marker is suppressed. |
| hide | Bool | Display suppression, 1 = the page is not displayed in the operation. |
| lock | Bool | Adjustment blocking, 1 = the parameters on the bargraph page are not active. |
| store | Bool | With positive flank: The values applied to the signal inputs are stored as parameters Y_1 and Y_2, i.e. as output values. This input should be activated only with relevant input value changes. |
| Name | Type | Description |
| Y_1 | Float | Valid process value on display line 1. The output value is displayed only, when the output is connected with the related input, or if the display of this value is adjustable. |
| Y_2 | Float | Valid process value on display line 2. The output value is displayed only, when the output is connected with the related input, or if the display of this value is adjustable. |

| | | |
|-------|-------|------------------|
| Bl_no | Float | Own block number |
|-------|-------|------------------|

Parameter

| ID | Name | Type | Description | Access | Default | Range | Off |
|---------|--------------|-------|--|--------|---------|-------|-----|
| Y_1 | Y 1 | Float | Start value after power-on for analog output 1 | r/w | 0.0 | | |
| Y_1_min | Min. value 1 | Float | Minimum limit for adjustment of display line 1 | r/w | MIN | | |
| Y_1_max | Max. value 1 | Float | Maximum limit for adjustment of display line 1 | r/w | MAX | | |
| Y_2 | Y 2 | Float | Start value after power-on for analog output 2 | r/w | 0.0 | | |
| Y_2_min | Min. Value 1 | Float | Minimum limit for adjustment of display line 2 | r/w | MIN | | |
| Y_2_max | Max. value 2 | Float | Maximum limit for adjustment of display line 2 | r/w | MAX | | |

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|--------|----------------|-----------|--|--------|---------|---------|-----|
| Disp_1 | Display mode 1 | Enum | The line can be a pure display, remain empty, or indicate and alter a value. | r/w | 1 | | |
| | | Alterable | Value (left or top) is displayed and adjustable. | | 0 | | |
| | | Display | Value is displayed only (left or top). | | 1 | | |
| | | Empty | No display, display of value is suppressed (left or top). | | 2 | | |
| Disp_2 | Display mode 2 | Enum | The line can be a pure display, remain empty, or indicate and alter a value. | r/w | 1 | | |
| | | Alterable | Value (right or bottom) is displayed and adjustable. | | 0 | | |
| | | Display | Value is displayed only (left or top). | | 1 | | |
| | | Empty | No display, display of value is suppressed (right or bottom). | | 2 | | |
| Dp_1 | Decimals bar 1 | Int | Digits behind decimal point in value display 1 | r/w | 0 | 0 ... 3 | |
| Dp_2 | Decimals bar 2 | Int | Digits behind decimal point in value display 2 | r/w | 0 | 0 ... 3 | |

| | | | | | | | |
|----------|------------------|------------|--|-----|--------|--|--|
| Typ | Bargraph type | Enum | Bargraph situation: horizontal or vertical | r/w | 0 | | |
| | | Horizontal | Both bargraphs horizontal | | 0 | | |
| | | Vertical | Both bargraphs vertical | | 1 | | |
| Bar1_0 | Scale start bar1 | Float | Display scaling of bargraph 1; 0% (left or lower end) | r/w | 0.0 | | |
| Bar1_100 | Scale end bar1 | Float | Display scaling of bargraph 1; 100% (right or upper end) | r/w | 100.0 | | |
| Bar1_mid | bar 1 mid | Float | Display scaling of bargraph 1, start value (origin). Determine, if the bargraph indicates only in one direction (e.g. from top or from bottom) or in both directions from the central value. | r/w | 0.0 | | |
| Bar2_0 | Scale start bar2 | Float | Display scaling of bargraph 2; 0% (left or lower end) | r/w | 0.0 | | |
| Bar2_100 | Scale end bar2 | Float | Display scaling of bargraph 2; 100% (right or upper end) | r/w | 100.0 | | |
| Bar2_mid | Bar 2 mid | Float | Display scaling of bargraph 2, start value (origin). Determine, if the bargraph indicates only in one direction (e.g. from top or from bottom) or in both directions from the central value. | r/w | 0.0 | | |
| Name_1 | Name bar1 | Text | Name of value display 1 | r/w | Name_1 | | |
| Einh_1 | Unit 1 | Text | Unit of value display 1 | r/w | Unit_1 | | |
| Name_2 | Name bar 2 | Text | Name of value display 2 | r/w | Name_2 | | |
| Einh_2 | Unit 2 | Text | Unit of value display 2 | r/w | Unit_2 | | |

V_BAR operating page

V_BAR has an operating page, which can be selected in the operating page menu with the input **hide** not set. Changing the texts displayed in the unit is only possible in the engineering tool!

Max.19 characters can be entered in each text parameter.

A value configured as display cannot be changed.

Overview

- 1 Title
- 2 Name of the value displayed in the bargraph
- 3 Scale value of the bargraph
- 4 Origin of the bargraph
- 5 Unit of the displayed value
- 6 Bargraph
- 7 Button "Leave Operating Page"
- 8 Button "Alarm"
- 9 Display and input field for the value
- 10 Limit value marks for the bargraph

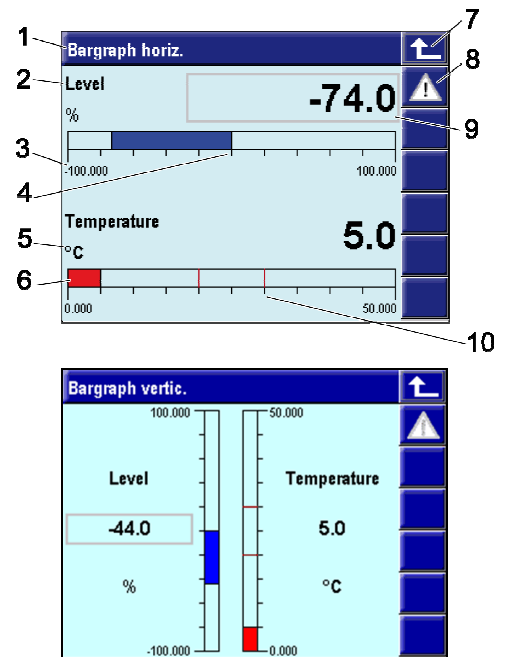
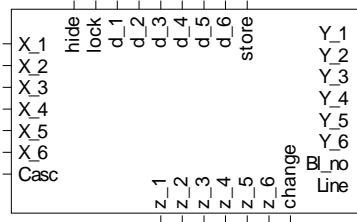


Fig. 482: Bargraph (horizontal and vertical)

III-10.3 V_Display (display / definition of process values - (No. 96))



V_DISPLAY

Abb. 483

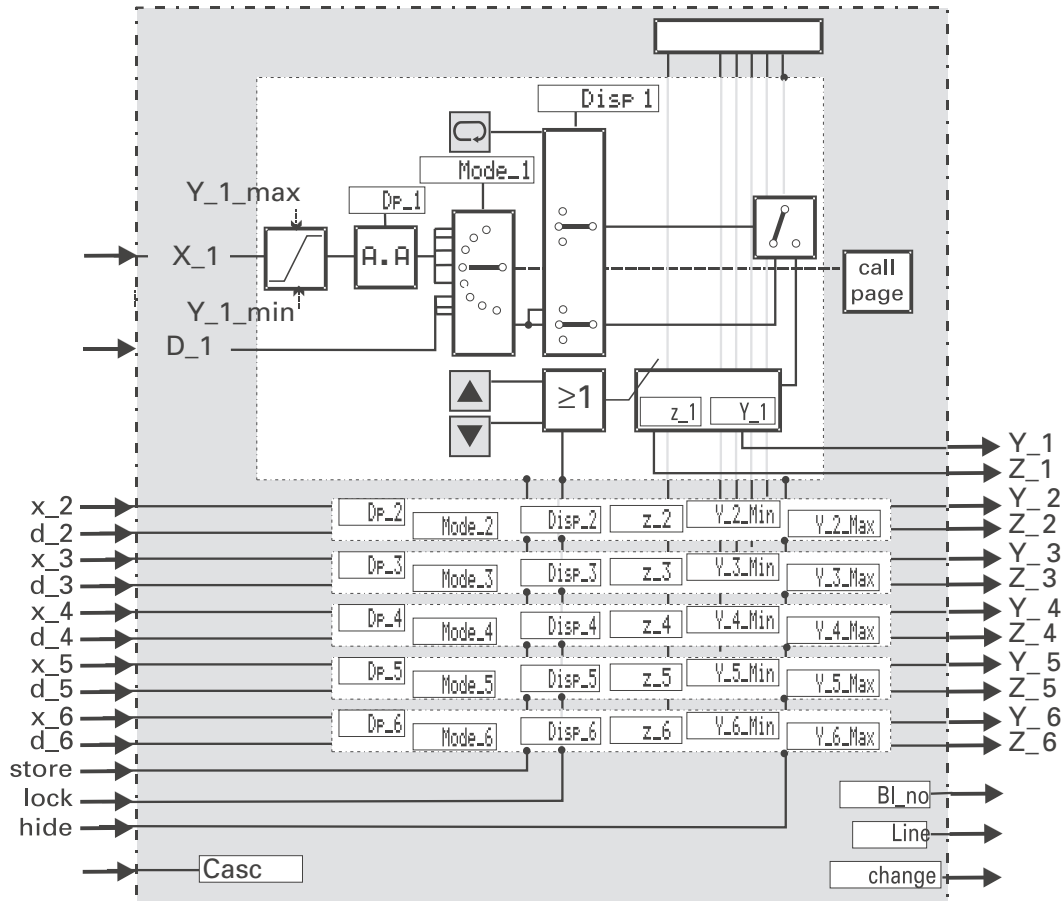


Abb. 484

General

Function block **V_DISPLAY** opens an information page which can be designed according to process and user requirements.

This page can be used for display and pre-setting of 6 analog or digital process values on 6 display lines. Controls such as "Switch" and selection of a defined operating page can be set on the display line as well.

These values can be changed also via the communication interface of **KS108** rather than only via operation. The function block can be cascaded, whereby a scroll field with more than 6 lines is possible on the operating page.

Value changes are stored as parameters **z_1 ... z_6** or **Y_1 ... Y_6**. When digital input **lock** is set, no values can be changed.

With digital input **hide** set, the operating page is not displayed.

A text (max. 19 characters) can be configured as a display header in the engineering. The same applies to further texts for identification of the value and the unit, or for digital statuses.



CAUTION

Excessively frequent saving can cause the destruction of the EEPROM.

For this reason:

This input should be activated only with relevant changes of the input values, for values of the used analog inputs are taken over as parameter values when a positive flank is detected at the **store** input.

In-/Outputs

| Name | Type | Description |
|-------------|-------|---|
| X_1 ... X_6 | Float | Analog process value 1 to be displayed |
| Casc | Float | By wiring a Casc input with the BI_no output of another V_DISPLAY, cascades can be set up |
| hide | Bool | Display suppression, 1 = the page is not displayed in the operation. |
| lock | Bool | Adjustment locking; 1 = the parameters of the V_DISPLAY page are not active. |
| d_1 ... d_6 | Bool | Digital process status 1 to be displayed |
| store | Bool | With a positive flank (0 to 1) the input values are stored in EEPROM and used as output values. |

| Name | Type | Description |
|-------------|-------|--|
| Y_1 ... Y_6 | Float | Output analog process value 1 |
| BI_no | Float | Own block number |
| Line | Float | If a value is changed during operation, the line output is set to the value (1 - 6) which was changed. The line output remains set for the duration of one calculation cycle of the V_DISPLAY block. |
| z_1 ... z_6 | Bool | Value output at digital output d_1 |
| change | Bool | If a value is changed during operation, the change output is set to 1 for the duration of one calculation cycle of the V_DISPLAY block. |

Parameter

| ID | Name | Type | Description | Access | Default | Range | Off |
|-------------|------|-------|--|--------|---------|-------|-----|
| z_1 ... z_6 | z 1 | Int | Start value for the digital output d_1 after power-on. | r/w | 0 | 0 1 | |
| Y_1 ... Y_6 | Y 1 | Float | Start value for the analog output X_1 after power-on. | r/w | 0.0 | | |
| Y_1_Min ... | Min. | Float | Minimum adjustment limit for analog | r/w | MIN | | |

| | | | | | | | |
|------------------------|-----------------|-------|--|-----|-----|--|--|
| Y_6_Min | value 1 | | value 1 | | | | |
| Y_1_Max ... Y_6_Max | Max. value 1 | Float | Maximum adjustment limit for analog value 1 | r/w | MAX | | |

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|----------------------|-------------------|-----------------|--|--------|---------|-------|-----|
| Disp_1 ... Disp_6 | Display mode 1 | Enum | The line can be a pure display, remain empty, or indicate and alter a value. | r/w | 1 | | |
| | | Alterable | The value in this line is displayed and adjustable. | | 0 | | |
| | | Display | The value in this line is displayed only. | | 1 | | |
| | | Empty | No display, display of the value is suppressed. | | 2 | | |
| Mode_1 ... Mode_6 | Display type 1 | Enum | Analog or digital process values, but also controls such as "switches", radio-buttons or invocation of a defined operating page can be put into the display line. | r/w | 0 | | |
| | | Analog | The value connected at the analog input and 2 static texts (6 characters), one before and one behind the analog value. | | 0 | | |
| | | Digital | The value at the digital input switches over between "0" text (Name_n) and "1" text (Unit_n). Static text output (e.g. heading) is possible. | | 1 | | |
| | | Time | Times in HH:MM:SS or HH:MM. The least significant decimal digit indicates the full minutes, the digits behind the decimal point indicate the seconds. | | 2 | | |
| | | Radio button | For switching over combined selection fields. Radiobutton in successive order form a common group. | | 3 | | |
| | | Switch | Switch-on/off (toggle) function. | | 4 | | |
| | | Tip switch | Short switch-on/off functions (hold). As long as the field is pressed, the output is switched on, when releasing the "key", the | | 5 | | |

| | | | | | | | |
|----------------------|------------------|-------|--|-----|--------|---------|--|
| | | | output is switched off. | | | | |
| | | Text | Indexed texts for analog integer signals (index). Adjustment of an analog value according to assigned texts, also in several languages. | | 6 | | |
| | | Menue | Change to the operating page the block number of which is applied to the analog input (one-level menu, no linking is possible). | | 7 | | |
| Dp_1 ... Dp_6 | Decimals 1 | Int | Digits behind decimal point on line 1 (only for analog values) | r/w | 0 | 0 ... 3 | |
| Text1a ... Text6a | Line 1 Text 1 | Text | 1st text on display line 1: E.g. the parameter name if values are displayed. With digital values (radio-buttons, switches, push-buttons): the first displayed text. | r/w | Name_1 | | |
| Text1b ... Text6b | Line 1 Text 2 | Text | 2nd text on display line 1: E.g. the unit of the displayed parameter if values are displayed. With digital values (radio-buttons, switches, push-buttons): the text for switch-over. | r/w | Unit_1 | | |

Overview and operation

- 1 Title
- 2 Switch "Toggle": Switch over one logical value with one click.
- 3 Display "Toggle":
- 4 Value display
- 5 Input field: One click on the field opens the numerical value editor.
- 6 Button "Leave Operating Page"
- 7 Button "Alarm"
- 8 Button "Call Operating Page"
- 9 Button "Previous page": The previous screen page will be called.
- 10 Button "Next page": The next screen page will be called.
- 11 Radio buttons: Select a value with a single click on the desired value. Only one element can be selected.

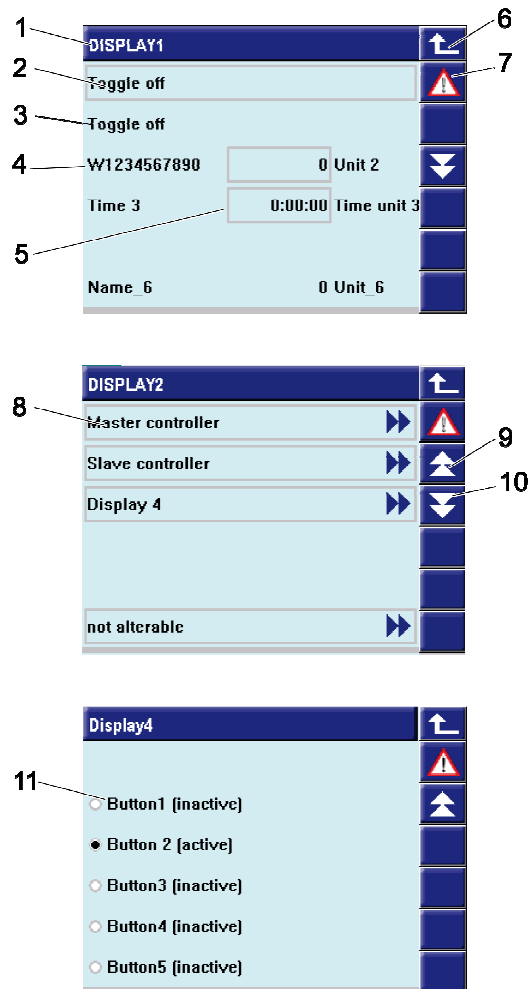


Fig. 485: User display

V_DISPLAY operating page

V_DISPLAY has an operating page, which can be selected in the operating page menu with the input **hide** not set. If Input **hide** is selected the page is invisible.

- Determination if the display line has digital or analog functions, or if it is switched off is made via configurations (generation of an empty line in the display).
- Possible display functions are: analog, digital, text, menu, push-button, switch and radio button.
- Normally, the values applied to the inputs are displayed.
- A value adjustable on the operating pages (provided that the related line was configured as "variable") is output at the relevant function output.
- Only adjustable lines are selectable.
- The change of these values from the operating level can be switched off (lock).
- Parameters z1 ... z6 or y1 ... y6 are used as initial value for the outputs at power-on.

- The output value is displayed only, if the output is fed back to the relevant input, or if the display for this value is in the adjustment mode.
- With a positive flank at the store input, the values applied to the remaining inputs are stored in parameters z1 ... z6 and y1 ... y6 and thus used as output values.

For information on the operation, see section Operating pages.

Entry and display of texts

Changing the texts displayed in the unit is only possible in the engineering tool!
Max. 19 characters can be entered into each text parameter.

Dependent on whether a line was configured as an analog, digital, radio, switch, push-button or menu line, all 19 characters (e.g. Mode i= digital) are shown in the device.

With **Mode i** = analog, a field for 10 digits in the middle of the line is reserved. The name (left) and the unit (right) are limited. The number of characters which can be displayed is dependent on the type (size) of characters.

With digital displays (digital, switch, push-button and radio):

signal = 0: dependent of line of 'Text1 a ... Text6 a'

signal = 1: dependent of line of 'Text1 b ... Text6 b'

Further detailed information on the various display types is given at the end of the section.



NOTE

If a line is configured as a display, the value of this line cannot be changed.



NOTE!

The operation of line modes radio, switch and push-button is described in section "Adjusting values".



NOTE!

This operation must be described separately in a description for operation of the system.

V_DISPLAY block cascade connection

For linking several **V_DISPLAY** operating pages, the BI-no output of a further **V_DISPLAY** must be wired with the Casc input of the calling **V_DISPLAY**. Thereby, the last page which must be connected can be linked also to the start page (ring structure).

Cascade connection of a **V_DISPLAY** block is indicated by arrows ▲▼ on the display page.

A previous block (BI-no output wiring) above the first line and a next block (Casc input wiring) below the last line are marked, otherwise, these arrows are omitted.

When touching one of these arrows and pressing, a change to the relevant **V_DISPLAY** page occurs. With standard exit from the called up **V_DISPLAY** page a change to the selection list of the operating pages is made.

Selectable display modes in detail

Data type "analog"

The line contains 2 static texts (6 characters) and the analog value connected to X1...X6. Changing the value is done as described above, if changeability is configured.

Example: Value with limits:

Apart from its maximum number of digits behind the decimal point, each value can have its own adjustment limits, which are determined by parameter values `Y_1_Min` and `Y_1_Max`.

Data type "digital"

Dependent digital input bit value of the relevant line, text "0" (Name_i) or text "1" (Unit_i) is displayed. With a static input value, static text output can be generated (e.g. headline).

Data type "time" (analog output)

Data type "time" can be used to display or to adjust times in HH:MM:SS or HH:MM.

The adjustment range is within 00:00:00 - 30 000:00:00 hours. Due to the limited resolution of a float value, adjustment is possible only in steps of 6 seconds from value 16:40:00.

Data type "radio" (radio button; digital output)

Data type "radio button" can be used for changing over between combined selection fields. After selecting, changing is done directly.

"Radio button" arranged successively in one `V_DISPLAY` form a common group. Only one element of this group is activated. By pressing the desired radio button it is activated.

A new group starts, if another data type is defined between 2 radio buttons. Unless a radio button is activated during data transmission to `V_DISPLAY`, all radio buttons remain inactive. If more than 1 button is active, the 1st one of the group is activated, the other ones are inactive.

Data type "switch" (digital output)

Data type "switch" can be used to implement switch-on/switch-off (toggle) functions.

After selection, adjusting is done directly. Pressing the function key will activate a de-activated switch and de-activate an activated switch.

Data type "push-button" (digital output)

Data type "push-button" can be used to realize short switch-on/switch-off (hold) functions.

After selection, adjusting is done directly. As long as the function key is pressed, the output is activated. When releasing the key, the output is de-activated.

Data type "text" (analog output, see also: Function block TEXT)

Data type "text" can be used for display of indexed texts for analog integer signals. Moreover, an analog value can be allocated to a text when adjusting.

The corresponding input must be connected with the index output of a text block.

The number of the text to be selected (`V_DISPLAY` output Y1...Y6) is applied to the index input of the first (next to `V_DISPLAY`) text block.

The text blocks cascaded by wiring the index output of another text block with the Casc input of the relevant text block. Text selection is always via the index input of the text block next to **V_DISPLAY**.

Via the **UsrLan** input, text blocks of different language can be appended. Language switchover (language index) is defined by the value at the **UsrLan** input of status block 98. Unless an appropriate text block for the language is available (e.g. language index too high), the corresponding text is output in the last language block found.

When selecting a text in the **V_DISPLAY** to be displayed, the number of selectable texts is limited by the number of connected text blocks.

If the index for text selection comes from a different origin, no text is displayed if the index is beyond the possible text selection (0 or >max). **V_DISPLAY** marks the line with "_____".

With text selection at **VWERT**, the initial value (parameter Y1...Y6) should be set > 0 to avoid a start value of "_____".

Data type "menu"

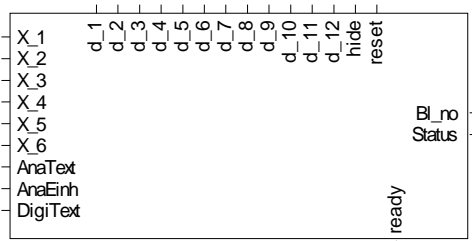
Data type "menu " can be used for changing to other operating pages (menu without sub-menu items, linking not possible). The value applied to the corresponding input is interpreted as block number of the operating page to which changing is required.

Changing to the specified page is done by pressing the Enter key. Unless the page is accessible, a change to the operating page selection list is made. This list contains all the blocks which are available for selection . If a page is not accessible, the reasons may be:

1. Block number not defined
2. Block number does not have an operating page
3. Block cannot be displayed instantaneously because hide = 1 .

When making a standard exit from the operating page, return is to the calling **V_DISPLAY** page. When using this procedure for changing to a **V_DISPLAY** operating page which also contains a menu type line, no further change will occur.

III-10.4 V_TREND (trend display (No. 99))



V_TREND

Abb. 486

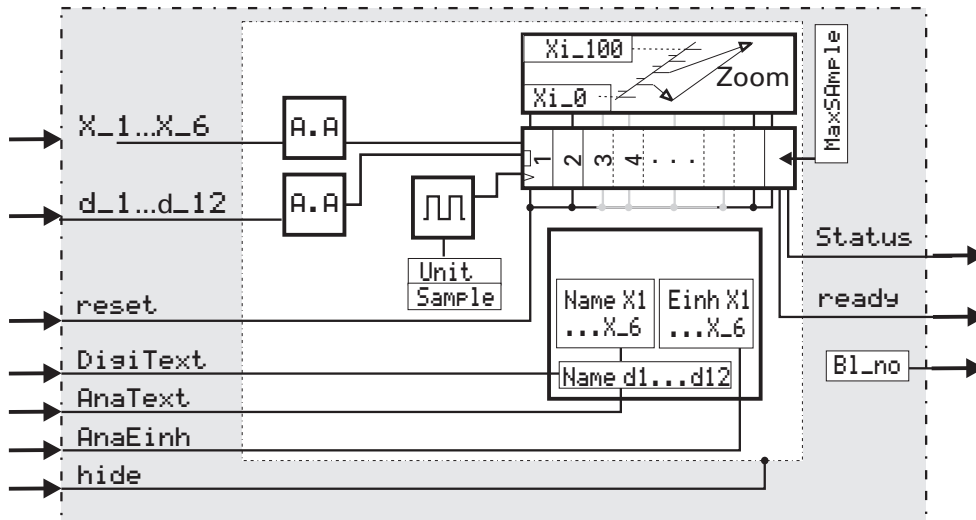


Abb. 487

General

Block **V_TREND** has 12 digital status bits (Mode_D1...D12) and 6 analog outputs (Mode_1...6). The only function of the trend block is collecting values for output on the display. The saved values cannot be read out. For this purpose, block **V_LOGGING** is provided (see next section).

Data rate

The **V_TREND** function block saves all connected signals at a fixed sampling interval. The number of recorded values is determined by configuration **MaxSamp** (max.: 12.000, min.:300, def.:1200). The minimum interval is 0.2 sec.

Analog outputs and digital status bits

For each analog value, there is a scaling ($X_{i_0} \dots X_{i_{100}}$) which can be changed at any time. Additionally, there is a parameter (**Mode 1...6**) which determines the display mode. 4 settings for this parameter are possible: **off, single, mean, min/max**.



NOTE

Parameter *STEP* indicates how many values along the X-axis should be grouped for display of one measured value. The Y-axis is reduced by the digital bits, unless display of the digital bits is suppressed via *MODE_Di*.

The start-up time segment can be specified in parameter **Step**. This parameter determines how many values should be grouped into one value.

With **Step** higher than 1:

with Mode = **single**, only a single value is displayed (every xth value)

with Mode = **mean**, the mean value of all related values is used

with Mode = **min/max**, a line from the minimum to the maximum value is plotted

with Mode = **off**, no trend is displayed .

Zooming of the displayed values is possible on the operating page. This changes the value in **Step**.



NOTE

The display range in which the analog values are displayed remains unchanged also when all analog values are switched off.

Inscription of the Y-axis

A trend curve can be selected for inscription of the analog axes.

The values for 0% and 100% on the Y-axis are displayed with max. 7 characters. Consequently, only values smaller than $9,9 \cdot 10^9$ are possible. Values close to 0 and values = 0 are shown as 0.000.

Inscription of the X-axis

The time which has passed since the points were recorded is used as inscription of the X-axis.

3 times are indicated:

1. Time on the left of the trend curve
2. Time in the cursor position
3. Time on the right of the trend curve (current time)

Display

The related value is shown below the time of the cursor. The colour of values is dependent on the colour of the selected trend curve. The cursor can be moved using two arrow keys (◀ ▶). When moving the cursor beyond the border, the entire curve is displaced by the half display range.

Inputs/outputs

The trend block provides the following outputs:

Own block number, error status, memory overflow (ready).

Presently, the status indicates only, if the memory space for the block is insufficient.

In this condition, the trend block does not operate. Ready is set, if the trend is complete.



NOTE

V_Trend has three configurations which are effective only when starting up. When they are changed in the running program, trend recording is not affected.

The V_TREND function collects the input values in a buffer memory and permits display as a trend. When the buffer is full, a new value overwrites the earliest value. The number of values in the buffer is configurable.

Data recording is done cyclically using the configured sampling interval (value + unit). The function block V_TREND has the following properties:

1. The Y-axis of KS 108 has a resolution of 180 pixels (less with existing digital status bits). The resolution of the X-axis is 240 pixels.
2. The resolution of the Y-axis can be scaled; the scale of the selected trend is used for display. Each trend curve has its own Y scaling determined by parameters $X_{i_0} - X_{i_100}$.
3. The arrow keys ◀▶ can be used to move a cursor in the trend. In this case, the value at the marker rather than the current value of the selected trend curve is shown in the middle below the trend. The value of the X-axis for the marker is also displayed (otherwise 0.00.00).
4. The lower adjustment limit is 0.2 (independent of the selected unit).
5. Output **Bl_no** provides the block number of the operating page



NOTE

*The trend function serves only for display. For reading out data from the instrument, the data logging function (**V_LOGGING**) must be used.*

In-/Outputs

| Name | Type | Description |
|--------------|-------|--|
| X_1 ... X_6 | Float | Analog input for trend curve 1 |
| AnaText | Float | Input to connect a text block (or chain of text blocks) with units for analog signals. |
| AnaEinh | Float | Input to connect a text block (or chain of text blocks) with units for analog signals. |
| d_1 ... d_12 | Bool | Digital input for digital bit 1 |
| DigiText | Float | Input to connect a text block (or chain of text blocks) with names for digital signals. |
| hide | Bool | Display suppression, 1 = the page is not displayed in the operation. |
| reset | Bool | All data of the function block are deleted |
| Name | Type | Description |
| Bl_no | Float | Own block number |
| Status | Float | (Error) status. Presently only, when the memory space is not sufficient for the block. In this condition, the trend block does not work. |
| ready | Bool | Is set when all values in the memory are valid. |

Parameter

| ID | Name | Type | Description | Access | Default | Range | Off |
|---------------|----------------|-------|---|--------|---------|----------|-----|
| X1_0 ... X6_0 | Scale start X1 | Float | Display scaling 0% for trend curve 1 (display on the y axis), i.e. smallest displayed value of the trend curve. | r/w | 0.0 | | |
| Step | Step size | Int | Selection of values on the X axis for display. No value, a single value, or the mean value can be selected, or a line | r/w | 1 | 1 ... 50 | |

| | | | | | | | |
|-------------------------|-------------------|--------------|--|-----|-------|----------------|--|
| | | | from the minimum to the maximum value is drawn. | | | | |
| X1_100 ... X6_100 | Scale end X1 | Float | Display scaling 100% for trend curve 1 (display on the y axis), i.e. highest displayed value of the trend curve. | r/w | 100.0 | | |
| Start | Start delay | Int | Start delay | r/w | 0 | 0 ... 12000 | |
| AktuTrend | Actu Trend | Int | Selection of analog trend curve whose values are to be displayed: axis scale, name and value. | r/w | 1 | 1 ... 6 | |
| Mode_1 ... Mode_6 | Display Mode 1 | Enum | Display of individual values of channels | r/w | 0 | | |
| | | Off | No display, display of value is suppressed. | | 0 | | |
| | | Single | Only a single point is displayed. | | 1 | | |
| | | Mean | The mean value of all recorded values is formed. | | 2 | | |
| | | Min / Max | A line from the minimum to the maximum value is drawn. | | 3 | | |
| Mode_D1 ... Mode_D12 | Display Mode 1 | Enum | Digital input 1 is visible | r/w | 0 | | |
| | | Off | No display, display of value is suppressed. | | 0 | | |
| | | On | The value is displayed, the digital bit is switched on. | | 1 | | |

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|---------|----------------|--------|--|--------|---------|------------------|-----|
| Unit | Unit | Enum | Time unit of sampling interval in hours, minutes or seconds. This configuration must be done before program start; a change while the program is running is without effect on trend recording. | r/w | 0 | | |
| | | Second | The time unit of the sampling interval is one second. | | 0 | | |
| | | Minute | The time unit of the sampling interval is one minute. | | 1 | | |
| | | Hour | The time unit of the sampling interval is one hour. | | 2 | | |
| MaxSamp | Max. Sample | Int | Number of recorded samples. This item must be configured before program start; a change while the program is | r/w | 1200 | 300 ... 12000 | |

| | | | | | | | |
|--------|----------|-------|--|-----|------|-------------------|--|
| | | | running is without effect on trend recording. | | | | |
| Sample | Interval | Float | Sampling interval value. This configuration must be done before program start; a change while the program is running is without effect on trend recording. | r/w | 1.0 | 0.2 ... 3600.0 | |
| Einh_ | Unit | Text | Displayed unit | r/w | Unit | | |

V_TREND operating page

The **V_TREND** block has an operating page which can be selected in the operating page menu. When input 'hide' = "1", display of the operating page is suppressed. The operating page serves for display of the trend data. The input fields change only the view of the saved data, but not the data themselves.

For further information on the operation: see chapter Operating Instructions.

Text entries: names and units

The **V_TREND** block can indicate a name for each status bit and additionally a unit for each analog output. Some special characters (ö, ü, ä) and the characters for square (²), cubic (³) and degrees (for °C) are available. When using only one unit for all analog outputs in common, this unit can be entered as parameter **Einh_**. In this case, input **AnaEinh** must not be wired.

Names and units for the bits and outputs are entered via text blocks which are attached to a text input of **V_TREND**. The names for the analog outputs are entered into the text blocks the first one of which is connected to input **AnaText**. Text blocks for the units are connected to input **AnaEinh**, text blocks for the names of the digital status bits are connected to input **DigiText**. The text blocks are extended to the required number of texts by cascading and can be used also in several languages (see text block description).

Overview

- 1 Title
- 2 Analog trend curves (the curve is read from right to left)
- 3 Cursor
- 4 Time specification (beginning, at cursor position, end). The specification is made in hours/minutes/seconds.
- 5 Value specification for the selected measured value at the cursor position
- 6 Button "Leave Operating Page"
- 7 Button "Alarm"
- 8 Value range of the selected measured value on the Y-axis
- 9 Button "Call parameter page"
- 10 Button "Cursor to the left"
- 11 Button "Cursor to the right"
- 12 Button xxx
- 13 Button xxx
- 14 Digital tracks

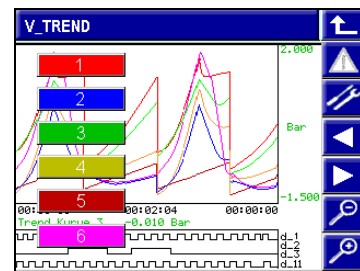
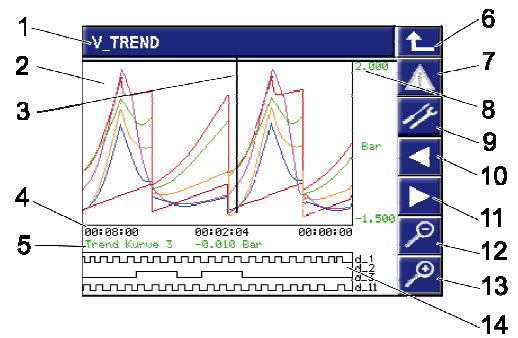
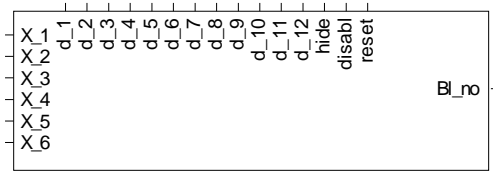


Fig. 488: Trend (Trend curves and measured value selection)

III-10.5 V_LOGGING (logging-function (No. 140))



V_LOGGING

Abb. 489

General

This function block can be used for recording 12 digital status bits and 6 analog outputs. Data is saved as files in csv format on the SD card (CSV: "Comma Separated Values"). A colon (";") is used as a separator. The file can be edited e.g. using *Microsoft Excel*.

The beginning of a file could have e.g. the following appearance:

```
"Comment:----;
2006-11-23 12:02:40
Date time;NAME X1;NAME X2;NAME X3
2006-11-23 12:02:42;55;0,224804;58,614
2006-11-23 12:02:44;55;1,16574;60,1412
2006-11-23 12:02:46;55;2,71857;59,8721
..."
```

Converted into a table, e.g. by : *Microsoft Excel*.

| | | | |
|------------------|---------|----------|---------|
| "Comment:----; | | | |
| 23.11.2006 12:02 | | | |
| Date Time | NAME X1 | NAME X2 | NAME X3 |
| 23.11.2006 12:02 | 55 | 0,224804 | 58,614 |
| 23.11.2006 12:02 | 55 | 1,16574 | 60,1412 |
| 23.11.2006 12:02 | 55 | 2,71857 | 59,8721 |

The function block has a visualization page. This page can be used for entry of file name and comment or for saving the data from the SD card on the USB stick.

In-/Outputs

| Name | Type | Description |
|--------------|-------|---|
| X_1 ... X_6 | Float | Analog input 1 to be stored (default = 0) |
| d_1 ... d_12 | Bool | Digital input 1 to be stored (default = 0) |
| hide | Bool | Display suppression, 1 = the page is not displayed in the operation. |
| disabl | Bool | disabl = 1: recording stop. |
| reset | Bool | Recording stop/start; Start starts a new file. If the reset input of the function block is soft-wired, the start/stop key is not visible on the operating page. |
| Name | Type | Description |

| | | |
|-------|-------|------------------|
| Bl_no | Float | Own block number |
|-------|-------|------------------|

| Configuration | | | | | | | |
|---------------|------------|-------------|---|--------|---------|----------------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Unit | Unit | Enum | Unit for the sampling interval: seconds / minutes / hours | r/w | 0 | | |
| | | Second | The time unit of the sampling interval is one second. | | 0 | | |
| | | Minute | The time unit of the sampling interval is one minute. | | 1 | | |
| | | Hour | The time unit of the sampling interval is one hour. | | 2 | | |
| Sample | Interval | Float | Sampling interval for recording in seconds / minutes / hours | r/w | 1.0 | 0.2 ... 3600.0 | |
| FileSize | File size | Float | File size in kbytes | r/w | 300.0 | >1 | |
| FileCount | File count | Float | Maximum number of existing files | r/w | 5.0 | 1 ... 100 | |
| DecPoint | Decimals | Enum | Comma or point can be selected as a separator for the digits behind the decimal point. | r/w | 0 | | |
| | | Comma | A comma is used as a separator for the digits behind the decimal point. | | 0 | | |
| | | Point | A point is used as a separator for the digits behind the decimal point. | | 1 | | |
| EndMod | End mode | Enum | Storage mode: Infinite or EndStop. when the function block has filled the maximum number of files with data, either data collection is finished (EndStop), or the earliest file is overwritten (infinite data collection = Infinite). | r/w | 0 | | |
| | | Infinite | The earliest file is always overwritten (infinite data collection), when the maximum number of files are filled with data. Data are always updated. | | 0 | | |
| | | Stop at end | Finish data collection (EndStop), when the maximum number of files are filled with data. To log data of a procedure. | | 1 | | |
| NAME_X1 ... | Name X1 | Text | Name of analog channel at input | r/w | NAME X1 | | |

| | | | | | | | |
|-------------|---------|------|-----------------------|-----|---------|--|--|
| NAME_X6 | | | 1 | | | | |
| NAME_d1 ... | Name d1 | Text | Name of control bit 1 | r/w | NAME d1 | | |
| NAME_d12 | | | | | | | |

Configurations of the Text type can be entered only via the engineering.

V_LOGGING operating page

The **V_LOGGING** block has an operating page which can be selected in the operating page menu. When input 'hide' is connected, this page is hidden.

For further information on the operation, see chapter Operating Instructions.

Overview

- 1 Title
- 2 Name of the output file
- 3 Current size of the output file
- 4 Remaining free storage space on the SD card
- 5 Remaining free storage space on the USB stick (this is only displayed if a USB stick is connected to the device)
- 6 Button "Leave Operating Page"
- 7 Button "Alarm"
- 8 Call button "Start/stop recording". Depending on the application development this button may not be visible.
- 9 Button "Copy"
- 10 Status display ("run": Datalogger is running, "off": Data logger stopped)
- 11 Input field "File name"
- 12 Input field "Header"
- 13 Status message

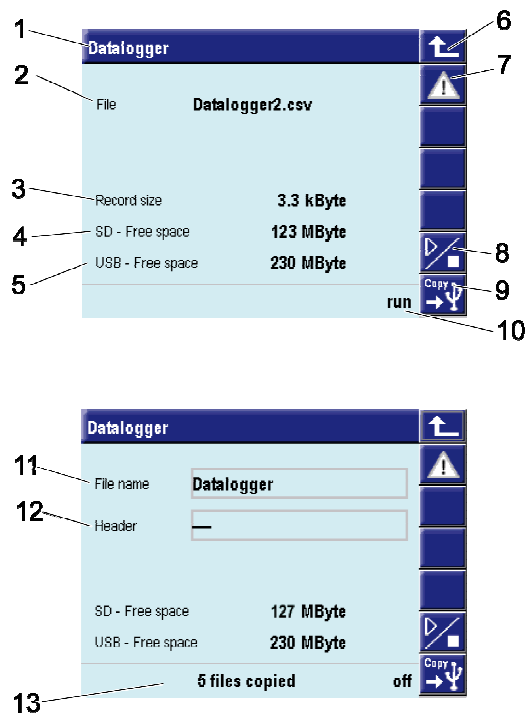
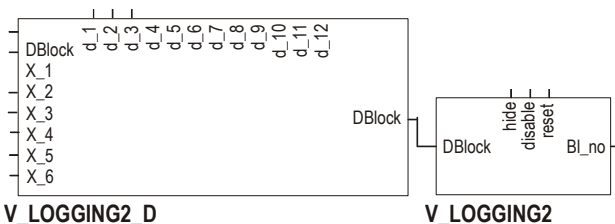


Fig. 490: Data logger (overview, input file name, copy)

III-10.6 V_LOGGING2 (scalable logging-function (No. 141)) / V_LOGGING2_D (connector for signals of V_LOGGING2 (No. 142))



V_LOGGING2_D

V_LOGGING2

Abb. 491

General

These function blocks can be used for recording digital status bits and analog outputs. A main block V_LOGGING2 is placed, to which at least one or a line of data blocks V_LOGGING2_D is connected. Via combination of V_LOGGING2 and V_LOGGING2_D the set of 12 digital and 6 analog tracks of a V_LOGGING can be cascaded to any extend.

Data is saved as files in csv format on the SD card (CSV: "Comma Separated Values"). A colon (";") is used as a separator. The file can be edited e.g. using *Microsoft Excel*.

The beginning of a file could have e.g. the following appearance:

```
"Comment:----;
2006-11-23 12:02:40
Date time;NAME X1;NAME X2;NAME X3
2006-11-23 12:02:42;55;0,224804;58,614
2006-11-23 12:02:44;55;1,16574;60,1412
2006-11-23 12:02:46;55;2,71857;59,8721
..."
```

Converted into a table, e.g. by: *Microsoft Excel*.

| | | | |
|------------------|---------|----------|---------|
| "Comment:----; | | | |
| 23.11.2006 12:02 | | | |
| Date Time | NAME X1 | NAME X2 | NAME X3 |
| 23.11.2006 12:02 | 55 | 0,224804 | 58,614 |
| 23.11.2006 12:02 | 55 | 1,16574 | 60,1412 |
| 23.11.2006 12:02 | 55 | 2,71857 | 59,8721 |

The function block V_LOGGING2 has a visualization page. This page can be used for entry of file name and comment or for saving the data from the SD card on the USB stick.

Inputs V_LOGGING2

| Name | Type | Description |
|--------|-------|--|
| DBlock | Float | Connection of the first data block V_LOGGING2_D. |
| hide | Bool | Display suppression, 1 = the page is not displayed in the operation. |
| disabl | Bool | disabl = 1: recording stop. |

| | | |
|-------|------|---|
| reset | Bool | Recording stop/start; Start starts a new file. If the reset input of the function block is soft-wired, the start/stop key is not visible on the operating page. |
|-------|------|---|

Inputs V_LOGGING2_D

| Name | Type | Description |
|--------------|-------|--|
| DBlock | Float | Block number of cascaded data function V_LOGGING2_D. |
| X_1 ... X_6 | Float | Analog input 1 to be stored (default = 0) |
| d_1 ... d_12 | Bool | Digital input 1 to be stored (default = 0) |

Outputs V_LOGGING2

| Name | Type | Description |
|-------|-------|------------------|
| Bl_no | Float | Own block number |

Outputs V_LOGGING2_D

| Name | Type | Description |
|--------|-------|------------------|
| DBlock | Float | Own block number |

Configuration V_LOGGING2

| ID | Name | Type | Description | Access | Default | Range | Off |
|-----------|------------|--------|--|--------|---------|----------------|-----|
| Unit | Unit | Enum | Unit for the sampling interval: seconds / minutes / hours | r/w | 0 | | |
| | | Second | The time unit of the sampling interval is one second. | | 0 | | |
| | | Minute | The time unit of the sampling interval is one minute. | | 1 | | |
| | | Hour | The time unit of the sampling interval is one hour. | | 2 | | |
| Sample | Interval | Float | Sampling interval for recording in seconds / minutes / hours | r/w | 1.0 | 0.2 ... 3600.0 | |
| FileSize | File size | Float | File size in kbytes | r/w | 300.0 | >1 | |
| FileCount | File count | Float | Maximum number of existing files | r/w | 5.0 | 1 ... 100 | |
| DecPoint | Decimals | Enum | Comma or point can be selected as a separator for the digits behind the decimal point. | r/w | 0 | | |
| | | Comma | A comma is used as a separator for the digits behind the decimal point. | | 0 | | |
| | | Point | A point is used as a separator for the digits behind the decimal point. | | 1 | | |

| | | | | | | | |
|--------|----------|-------------|---|-----|---|--|--|
| EndMod | End mode | Enum | Storage mode: Infinite or EndStop. when the function block has filled the maximum number of files with data, either data collection is finished (EndStop), or the earliest file is overwritten (infinite data collection = Infinite). | r/w | 0 | | |
| | | Infinite | The earliest file is always overwritten (infinite data collection), when the maximum number of files are filled with data. Data are always updated. | | 0 | | |
| | | Stop at end | Finish data collection (EndStop), when the maximum number of files are filled with data. To log data of a procedure. | | 1 | | |

Configuration V_LOGGING2_D

| ID | Name | Type | Description | Access | Default | Range | Offset |
|----------------------|---------|------|-----------------------------------|--------|---------|-------|--------|
| NAME_X1 ... NAME_X6 | Name X1 | Text | Name of analog channel at input 1 | r/w | NAME X1 | | |
| NAME_d1 ... NAME_d12 | Name d1 | Text | Name of control bit 1 | r/w | NAME d1 | | |

Configurations of the Text type can be entered only via the engineering.

V_LOGGING2 operating page

The **V_LOGGING2** block has an operating page which can be selected in the operating page menu. When input 'hide' is connected, this page is hidden.

For further information on the operation, see chapter Operating Instructions.

Overview

- 1 Title
- 2 Name of the output file
- 3 Current size of the output file
- 4 Remaining free storage space on the SD card
- 5 Remaining free storage space on the USB stick (this is only displayed if a USB stick is connected to the device)
- 6 Button "Leave Operating Page"
- 7 Button "Alarm"
- 8 Call button "Start/stop recording". Depending on the application development this button may not be visible.
- 9 Button "Copy"
- 10 Status display ("run": Datalogger is running, "off": Data logger stopped)
- 11 Input field "File name"
- 12 Input field "Header"
- 13 Status message

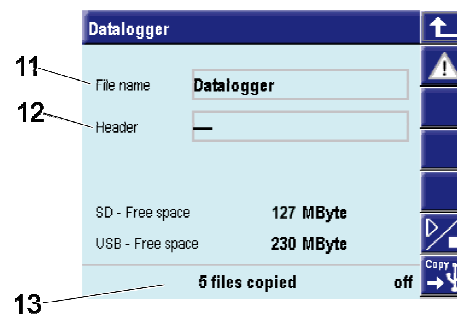
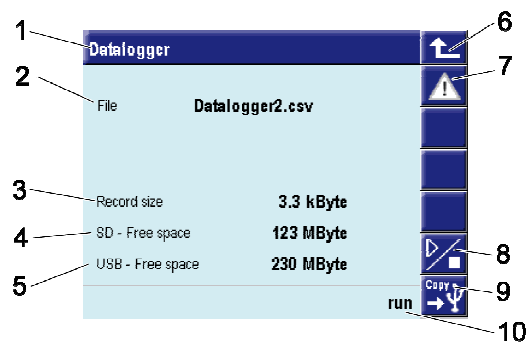
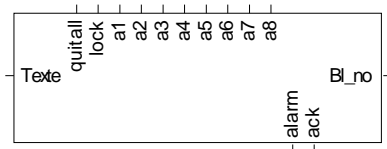


Fig. 492: Data logger (overview, input file name, copy)

III-10.7 V_ALARM (display of all alarms on one alarm-operating page (No. 109))



V_ALARM

Abb. 493

General

The V_ALARM function block handles up to 8 alarms. Alarms are displayed and can be acknowledged provided that the acknowledgement is configured. The alarms are determined by the digital inputs **a1** ... **a8** (0 alarm off, 1 alarm active). The common digital outputs indicate if an active alarm is pending and if an alarm must be acknowledged.

Text blocks with user-defined texts (also in several languages) can be connected at input **Text**. These texts are displayed automatically with the related alarm. Up to 100 alarm blocks can be positioned.

In-/Outputs

| Name | Type | Description |
|-----------|-------|--|
| Texte | Float | Connection of the first text block for definition of alarm texts |
| quitall | Bool | Acknowledgement of all alarms |
| lock | Bool | Adjustment blocking, 1 = operation of the alarm page is not active |
| a1 ... a8 | Bool | Alarm 1 input signal |
| Name | Type | Description |
| Bl_no | Float | Own block number |
| alarm | Bool | alarm = 1: min. one alarm is active. |
| ack | Bool | ack = 1: min. one alarm must be acknowledged by the operator. |

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|-------------------------|-----------------|-------------|--|--------|---------|-------|-----|
| Typ_a1 ... Typ_a8 | Alarm type 1 | Enum | Determine, if alarm 1 must be acknowledged by the operator before deleting it from the display. Unless it must be acknowledged, the alarm 1 is displayed only as long as it is actually pending. | r/w | 0 | | |
| | | Acknowledge | The alarm remains on the alarm page, until it is acknowledged. Output a2 remains set as long as there is still an alarm which must be acknowledged. | | 0 | | |

| | | | | | |
|--|----------------|---|---|--|--|
| | No acknowledge | The alarm remains on the alarm page, as long as it is pending. When the alarm is not pending any more, it is deleted from the alarm page. | 1 | | |
|--|----------------|---|---|--|--|

V_ALARM operating page

The **V_ALARM** block has an operating page which can be displayed using the alarm button provided on each operating page. The alarm button is marked by an exclamation mark in a triangle.

The alarms are displayed with the defined name in the order of occurrence. The name is defined in two text blocks which should be connected with the **V_ALARM** block. Unless the text blocks are connected, the title of the related **V_ALARM** block extended by the alarm number is displayed for the alarm.

For further information, see chapter Operating Instructions.

Overview

- 1 Title
- 2 Button "Leave Operating Page"
- 3 Button "Service"
- 4 Non-acknowledged active alarm (red) that must be acknowledged.
- 5 Active alarm (red) that does not need to be acknowledged, or that has already been acknowledged.
- 6 Alarm (black) that is no longer active which must be acknowledged.

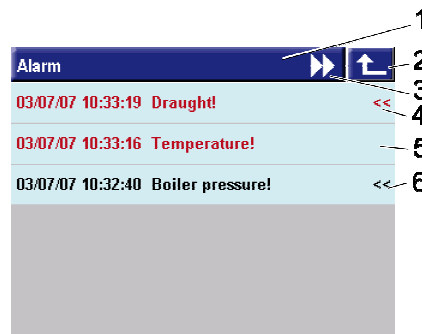
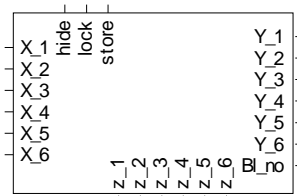


Fig. 494: Alarm page

i *NOTE!*
The characters "<<" indicate alarms that must be acknowledged.

III-10.8 V_PARA (parameter operation (No. 98))



V_PARA

Abb. 495

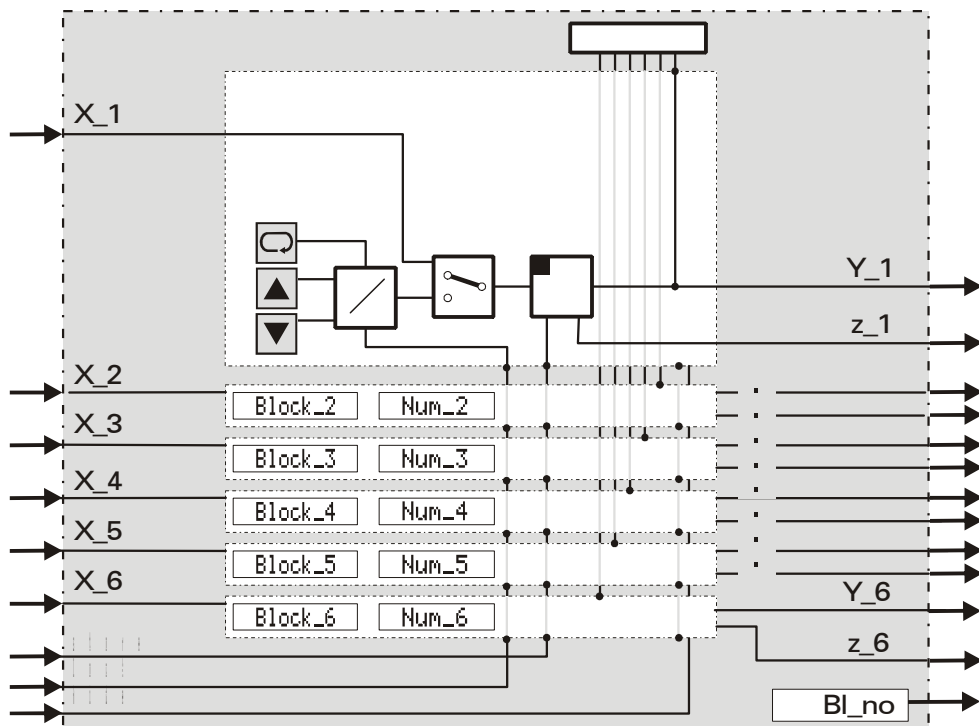


Abb. 496

General

The V_PARA function provides an operating page which can be used to change up to 6 parameters or configurations of other function blocks in the engineering at the operating level.

Each parameter to be displayed can be selected using the engineering tool. The engineering tool supports the parameter setting by a special drop-down list window which can be opened in the parameter dialogue. Clicking on the input field related to the parameter ("??") opens the following parameter setting dialogue in which the structure of the engineering is shown. The function block can be selected from the left tree; the right side shows the available configurations and parameters of this function block. Select and confirm the required entry.

(→ see the following picture).

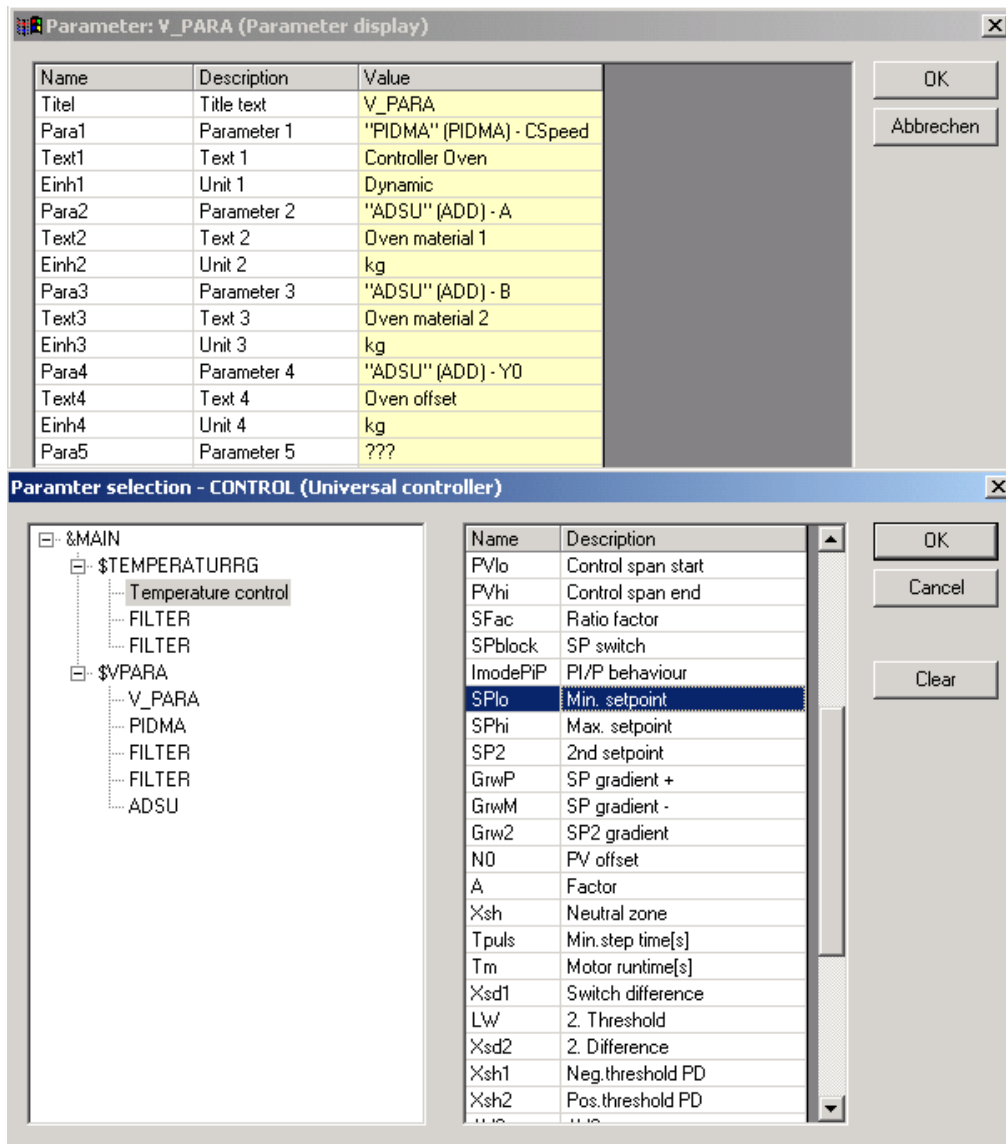


Fig. 497: Parameter selection

Additionally, an identifier and a unit text can be entered for each parameter. Values of the used analog inputs are taken over as parameter values when a positive flank is detected at the store input.

This input should be activated only with relevant changes of the input values. Excessively frequent saving can cause the destruction of the EEPROM.

In-/Outputs

| Name | Type | Description |
|-------------|-------|---|
| X_1 ... X_6 | Float | Process value to be stored as a parameter value |
| hide | Bool | Display suppression, 1 = the page is not displayed in the operation. |
| lock | Bool | Adjustment blocking, 1 = the parameters on the V_PARA page are not active. |
| store | Bool | With a positive flank (0 to 1) the input values are stored as parameter values. |
| Name | Type | Description |
| Y_1 ... Y_6 | Float | Output of the 1st parameter |

| | | |
|-------------|-------|---|
| Bl_no | Float | Own block number |
| z_1 ... z_6 | Bool | The output provides a status, which shows if the last storage of the values taken over from the inputs was successful (z1 = 0: successful). Errors may occur due to exceeded limits of the parameter value or due to non-existing parameters (z1 = 1: error). |

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|-------------------|-------------|------|--|--------|---------|-------|-----|
| Block1 ... Block6 | Parameter 1 | Text | Parameter 1, adjustable via operation page for parameters. | r/w | 0 | | |
| Text_1 ... Text_6 | Text 1 | Text | 1st text in line 1, e.g. name of parameter. | r/w | Para 1 | | |
| Einh_1 ... Einh_6 | Unit 1 | Text | Unit of parameter in line 1, e.g. °C. | r/w | Unit 1 | | |

Input and display of texts

The texts displayed in the instrument can be changed only in the engineering tool. Max. 19 characters can be entered for each text parameter. If a selected parameter is undefined (e.g. by deleting the relevant function block), "?????" is displayed by the instrument.

Assignment of parameters to display lines: Para1; Text1; Unit 1 → Line 1 ... Para6; Text6; Unit 6 → Line 6

V_PARA operating page

V_PARA is provided with an operating page which can be selected in the operating page menu with the 'hide' input not connected.

For further information on the operation: see chapter Operating Instructions.

III-11 Communication

The library **COMMUNICATION** provides functionblocks that enable an access to engineering data via MODBUS. A memory of 1000 INTEGER-values(1 word = 16 bit) each read- and write-access is available. For this access Modbus-protocol via ethernet-interface is used: Modbus via TCP/IP.

For accessing the IP-address of the device is needed. It can be found in the device → **Main menü** → **General data** → **Device data** → **IP address**.

The Modbusmessage **Read** uses the functioncodes **0x03** or **0x04**, the Modbusmessage **Write** uses the functioncodes **0x06** or **0x10**

The functionblocks summarize 10 data each. The READ-functionblocks transmit data from the inputs to the associated memory area, the WRITE-functionblocks transmit values from the memory to the outputs.

- The READ-range starts at address 1000.
- The WRITE-range starts at address 5000.

Floatvalues utilize 2 INTEGER-values, if they are defined by L1WRITE_FLOAT or L1READ_FLOAT. For the next value the address must be rised "+2". Message access occurs only with complete Floatdata. So all values must be written or read. If not, there is not output for reading, or there is no take over for writing of the last (=incomplete) value.

Bool-values are summarized to one word, they then occupy 1 address. At single transmission, adjusted via parameter **mode = bitwise**, every bool-value occupies 1 wort and 1 address.

Integer values can be transmitted as fixpoint 1 dataformat, adjustable via **mode = fixpoint**. During Reading the value is multiplied with 10, during writing divided by 10. so the first decimal is also transmitted.

Addresses

The communication-functionblocks get their start address via the **Offset**-input. The **Offset**-input of the first READ or WRITE-Block remains non wired. All remaining blocks build their address with the offset they get from their previous block: The input **Offset** of the second block gets wired with the output **Offset** of the first block...

No matter how a block is wired, he occupies the addresses for 10 data. For a float-functionblock, 20 addresses are occupied, for an integer 10, and for a bool functionblock, depending on parametering **Mode** 1 address e.g. 16 addresses are occupied.

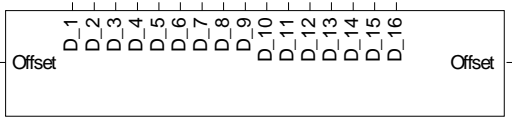
Example:

Wiring: The first block is a L1READ_FLOAT, then a L1READ_INT, as third block a L1READ_BOOL. The output **Offset** of the L1READ_FLOAT is wired to the input **Offset** of the L1READ_INT, the output **Offset** of the L1READ_INT to the input **Offset** of the L1READ_BOOL.

Access: The first floatvalue (**X_1**) is read by address 1000, the second floatvalue (**X_2**) by address 1002. The first integervalue (**X_1**) is read by address 1020, the second integervalue (**X_2**) by address 1021. The first boolvalue (**X_1**) is read by address 1030.

According to the setting of the parameter **Mode** for L1READ_BOOL the value from address 1030 comprises all 16 Bool of the functionblock (**Mode = wordwise**), and a following block starts at address 1031. With the setting **mode = bitwise** the value at address 1030 comprises only one boolvalue and a following block starts at address 1046.

III-11.1 L1READ_BOOL (Read function data bool (No. 132))



L1READ_BOOL

Abb. 498

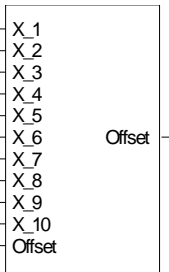
The function block writes the values in the read buffer.

| In-/Outputs | | |
|--------------|-------|--|
| Name | Type | Description |
| D_1 ... D_16 | Bool | Digital process value which can be read out via interface |
| Offset | Float | Defines the memory address for read access to the first datum. For cascading, connect the offset with output Offset of the previous block. |

| Name | Type | Description |
|--------|-------|--|
| Offset | Float | Defines the next free address in the memory for the read access. |

| Parameter | | | | | | | |
|-----------|-------------|-----------|--|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Mode | Data format | Enum | Format of the data | r/w | 0 | | |
| | | Word wise | Wordwise data format. One word is used. | | 0 | | |
| | | Bit wise | Bitwise data format. Each bit uses one word. | | 1 | | |

III-11.2 L1READ_INT (Read function data integer (No. 130))



L1READ_INT

Abb. 499

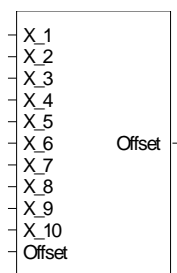
The function block writes the values in the read buffer. To this array data are written in integer format.

| In-/Outputs | | |
|-------------|------|-------------|
| Name | Type | Description |

| | | |
|-------------|-------------|--|
| X_1 ... X10 | Float | Analog process value which can be read out via interface |
| Offset | Float | Defines the memory address for read access to the first datum. For cascading, connect the offset with output Offset of the previous block. |
| Name | Type | Description |
| Offset | Float | Defines the next free address in the memory for the read access. |

| Parameter | | | | | | | |
|-----------|-------------|----------|--|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Mode | Data format | Enum | Format of the data | r/w | 0 | | |
| | | Integer | Integer (1 word) | | 0 | | |
| | | Fixpoint | Fixpoint 1 data format. Value is multiplied by 10. | | 1 | | |

III-11.3 L1READ_FLOAT (Read function data float (No. 100))



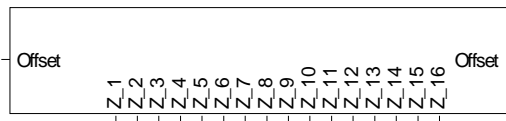
L1READ_FLOAT

Abb. 500

The function block writes the values in the read buffer. Data input into the read buffer is in float format. Each value uses 2 words in the read buffer.

| In-/Outputs | | |
|--------------|-------|--|
| Name | Type | Description |
| X_1 ... X_10 | Float | Analog process value which can be read out via interface |
| Offset | Float | Defines the memory address for read access to the first datum. For cascading, connect the offset with output Offset of the previous block. |
| Name | Type | Description |
| Offset | Float | Defines the next free address in the memory for the read access. |

III-11.4 L1WRITE_BOOL (Write function data bool (No. 133))



L1WRITE_BOOL

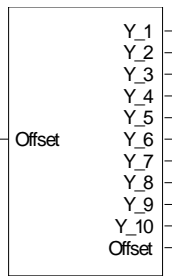
Abb. 501

The function block writes the bool data in the write buffer.

| In-/Outputs | | |
|--------------|-------|---|
| Name | Type | Description |
| Offset | Float | Defines the memory address for write access to the first datum. For cascading, connect the offset with output Offset of the previous block. |
| Z_1 ... Z_16 | Bool | Digital process value which can be written via the interface |
| Offset | Float | Defines the next free address in the memory for the write access. |

| Parameter | | | | | | | |
|-----------|-------------|-----------|--|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Mode | Data format | Enum | Format of the data | r/w | 0 | | |
| | | Word wise | Wordwise data format. One word is used. | | 0 | | |
| | | Bit wise | Bitwise data format. Each bit uses one word. | | 1 | | |

III-11.5 L1WRITE_INT (Write function data integer (No. 131))



L1WRITE_INT

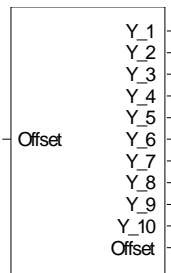
Abb. 502

The function block writes the values in the read buffer. Data are written in integer format.

| In-/Outputs | | |
|--------------|-------|---|
| Name | Type | Description |
| Offset | Float | Defines the memory address for write access to the first datum. For cascading, connect the offset with output Offset of the previous block. |
| Name | Type | Description |
| Y_1 ... Y_10 | Float | Analog process value which can be written via the interface |
| Offset | Float | Defines the next free address in the memory for the write access. |

| Parameter | | | | | | | |
|-----------|-------------|----------|------------------------|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Mode | Data format | Enum | Format of the data | r/w | 0 | | |
| | | Integer | Integer (1 word) | | 0 | | |
| | | Fixpoint | Fixpoint 1 data format | | 1 | | |

III-11.6 L1WRITE_FLOAT (Write function data float (No. 101))



L1WRITE_FLOAT

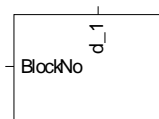
Abb. 503

The function block provides mapping in the array for the write access. Data input into this array is in float format. I.e. data are stored in 2 positions each.

| In-/Outputs | | |
|--------------|-------|---|
| Name | Type | Description |
| Offset | Float | Defines the memory address for write access to the first datum. For cascading, connect the offset with output Offset of the previous block. |
| Name | Type | Description |
| Y_1 ... Y_10 | Float | Analog process value which can be written via the interface |
| Offset | Float | Defines the next free address in the memory for the write access. |

III-12 Supplementary functions

III-12.1 CALLPG (Function for calling up an operating page (No. 127))



CALLPG

Abb. 504

With function block **CALLPG** event-triggered call-up of a particular operating page is possible, unless an operation is being carried out on this page (waiting time 5 s). The required operating page is determined by the number of its function block.

The block number is applied to input **BlockNo** of **CALLPG**. Main menu is selected with no "0". Switch-over is with the positive edge of the logic signal at digital input **d_1** of **CALLPG**. It permits e.g. changing to a particular operating page in case of an exceeded limit value.

Exceptions: Switching over is omitted with:

- active operation by the operator. Page changing is delayed and occurs only 5 seconds after the last key pressure.
- a wrong page number, or if the page is blocked at activation time.

Unless the page which should be activated is available the page survey is displayed. When leaving the operating page called up via **CALLPG**, the previously active operating page is displayed again.

Following functions blocks have an operating page **A_PROG**, **D_PROG**, **CONTROL**, **CONTROLP**, **PIDMA**, **V_DISPLAY**, **V_BAR**, **V_TREND**, **V_LOGGING**.



HINT!

With activation by **CALLPG** from an already selected page, this page is not called up again. I. e. if a sub-page was selected, the multifunction unit remains on this page.



HINT!

With multiple page changing by activation of **CALLPG**, the sequence is not buffered. After leaving the page(s) activated by **CALLPG** the initial menu page is displayed again.



HINT!

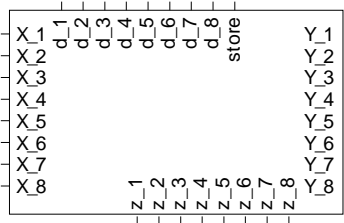
In case of **CALLPG** whilst the multifunction unit is not at operating level (main menu:Parameter, , Miscellaneous), **CALLPG** call remains active in the background. Changing to the operating page activated by **CALLPG** is done when selecting the operating level for the next time.

In-/Outputs

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------|-------|--|
| BlockNo | Float | Number of the operating page which should be displayed |
| d_1 | Bool | Positive edge causes change to the operating page set at BI_no |

III-12.2 SAFE (safety function (No. 94))



SAFE

Abb. 505

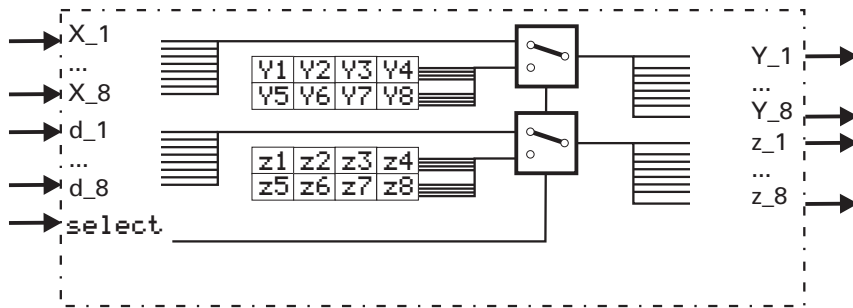


Abb. 506

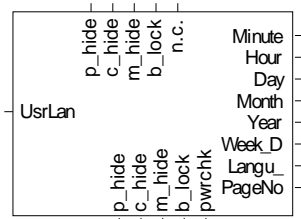
Function **SAFE** is used for generation of defined analog output values and digital statuses dependent of digital input **select**. In the normal case **select** = 0, the values applied to the inputs are switched through to the outputs without change. For **select** = 1, configured data **z_1...z_8** and **Y_1...Y_8** are switched through to the outputs.

| In-/Outputs | | |
|---------------|-------|---|
| Name | Type | Description |
| X_1 ... X_8 | Float | Analog input signal 1, is passed on to output Y_1 when input select = 0. |
| d_1 ... d_8 | Bool | Digital input signal 1, is passed on to output z_1 when input select = 0. |
| select | Bool | store = 0: The inputs are passed on to the outputs. store = 1: The parameters are passed on to the outputs. |
| Name | Type | Description |
| Y_1 ... Y_8 | Float | Analog output signal 1, coming from input (select = 0) or from parameter (select = 1). |
| z_1 ... z_8 | Bool | Digital output signal 1, coming from input (select = 0) or from parameter (select = 1). |

| Parameter | | | | | | | |
|-----------|------|------|-------------|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |

| | | | | | | | |
|-------------|-----|-------|--|-----|-----|-----|--|
| z_1 ... z_8 | z 1 | Int | Digital parameter 1, which is passed on to the output when select = 1. | r/w | 0 | 011 | |
| Y_1 ... Y_8 | Y 1 | Float | Analog parameter 1, which is passed on to the output when select = 1. | r/w | 0.0 | | |

III-12.3 STATUS (Status function (No. 125))



STATUS

Abb. 507

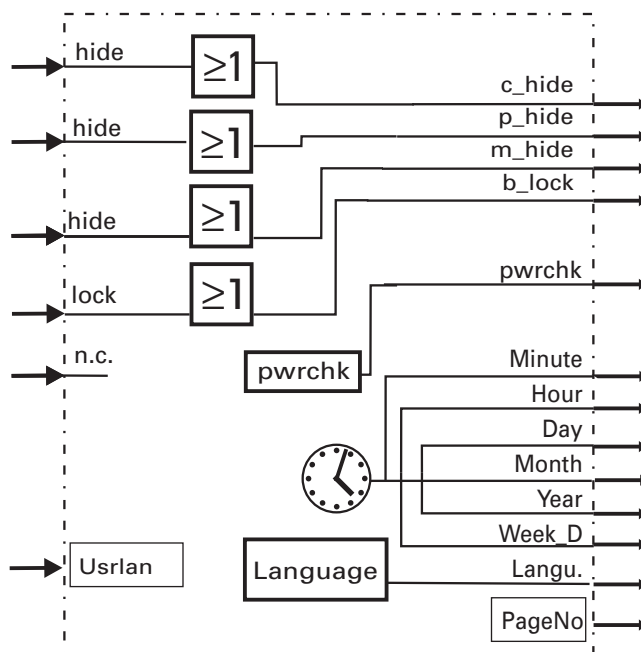


Abb. 508

This function outputs information from KS 108 status.



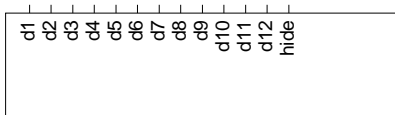
HINT!

This function block can be used only once!

| In-/Outputs | | |
|-------------|-------|--|
| Name | Type | Description |
| UsrLan | Float | Change to user language. Switch-over between text blocks connected via the language input. |

| p_hide | Bool | p_hide = 1: parameters/configurations via operation disabled |
|--------|-------|---|
| c_hide | Bool | c_hide = 1: a configuration change via operation is disabled. |
| m_hide | Bool | m_hide = 1: The main menu is not displayed, operating pages are displayed only during on-line mode. |
| b_lock | Bool | b_lock = 1: The access via the bus interface is blocked. |
| n.c. | Bool | No function |
| Name | Type | Description |
| Minute | Float | Minute of the real-time clock 0...59 |
| Hour | Float | Hour of the real-time clock 0...23 |
| Day | Float | Day of the real-time clock 0...31 |
| Month | Float | Month of the real-time clock 1...12 |
| Year | Float | Year of the real-time clock 1970...2069. Values 70 ... 99 (correspond to 1970 ... 1999) ... 0 ... 69 (correspond to 2000 ... 2069). |
| Week_D | Float | Weekday of the real-time clock 0...6 = Su...Sa |
| Langu_ | Float | Language German = 0; language English = 1; Language selection is under 'Device data' in 'Miscellaneous'. |
| PageNo | Float | Output of the number of the function block the operating page of which is displayed instantaneously. "0" means that no operating page is displayed. |
| p_hide | Bool | p_hide = 1: parameters/configurations disabled |
| c_hide | Bool | c_hide = 11: configuration change disabled |
| m_hide | Bool | m_hide = 1: The main menu is not displayed, the operating pages are displayed only during online mode. |
| b_lock | Bool | b_lock = 1: The use of the bus interface is blocked. |
| pwrchk | Bool | Power-fail check. After power-on, this value is activated (1) for 1 second. This permits detection of a temporary power failure. |

III-12.4 INFO (Information function (No. 124))



INFO

Abb. 509

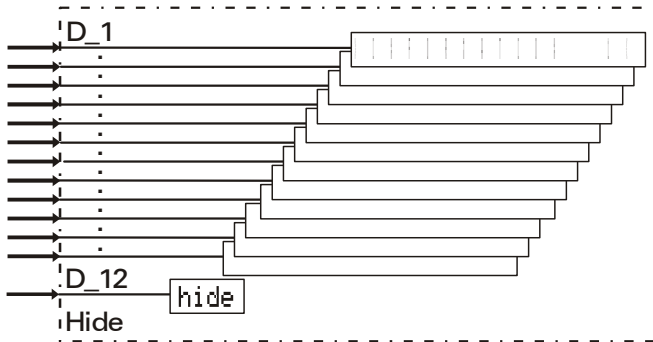


Abb. 510

General

This function can be used for display of 12 user texts with max. 19 characters each by setting the relevant input d1...d12. The information is displayed in the "header" of all pages (yellow font) in alternation with the description of the called up operating page. If several texts are available simultaneously, they are displayed successively.



HINT!

Function block INFO can be used only once!

- The user texts are displayed on all pages.
- Display of all INFO texts can be suppressed by setting the Hide signal.

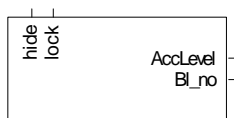
Ein-/Ausgänge

| Name | Type | Description |
|------------|------|---|
| d1 ... d12 | Bool | Display of info text 1, 1 = displayed, 0 = not displayed |
| hide | Bool | Display suppression, 1 = the info texts are not displayed in the operation. |

Konfiguration

| ID | Name | Type | Description | Access | Default | Range | Off |
|------------------|--------|------|--------------------------------|--------|-----------------|-------|-----|
| Text1 ... Text12 | Text 1 | Text | Info text 1, displayed via d_1 | r/w | >INFORMATION 1< | | |

III-12.5 PASSWORD (password function (Nr. 126))



PASSWORD

Abb. 511

Password protection

Via engineering parameters, configuration and main menu can be locked basically (via function block STATUS). For some function blocks, as controller and programmers, there are defined additionally users, who have differently limited access. The function block PASSWORD brings a operating page in which the user changes the levels.

In the levels the functions are enabled for operation according to the access level, the higher levels including the access rights of the lower levels (table: see section operating page).



NOTE!

The two highest levels (3 and 4) have the same access rights, but enable to define different users (or user groups).

In-/Outputs

| Name | Type | Description |
|------|------|---|
| hide | Bool | Display suppression. hide = 1: the page is not displayed in the operation. |
| lock | Bool | Value adjustment blocking via operation. lock = 0: adjustment enabled, lock = 1: adjustment disabled. |

| Name | Type | Description |
|----------|-------|--|
| AccLevel | Float | Access level (0: locked, 1: level 1, 2: level 2, 3: level 3, 4: level 4) |
| Bl_no | Float | Own block number |

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|------------|---------------------|-------------|---------------------------|--------|---------|-------|-----|
| NoAccess | No access | Enum | Enable for write blockage | r/w | 0 | | |
| | | allowed | | | 0 | | |
| | | not allowed | | | 1 | | |
| L1Password | Level 1 password | Float | Level 1 password | r/w | off | >0 | ja |
| L2Password | Level 2 password | Float | Level 2 password | r/w | off | >0 | ja |
| L3Password | Level 3 password | Float | Level 3 password | r/w | off | >0 | ja |
| L4Password | Level 4 password | Float | Level 4 password | r/w | off | >0 | ja |
| L1Name | Name access level 1 | Text | Name of access level 1 | r/w | Ebene 1 | | |
| L2Name | Name access level 2 | Text | Name of access level 2 | r/w | Ebene 2 | | |
| L3Name | Name access level 3 | Text | Name of access level 3 | r/w | Ebene 3 | | |
| L4Name | Name access level 4 | Text | Name of access level 4 | r/w | Ebene 4 | | |

In the levels the following functions are enabled for operation according to the access level, the higher levels including the access rights of the lower levels:

| Access level | User (example) | Operation |
|--------------|----------------|--|
| 0 | (locked) | Information only (no changes) |
| 1 | operator | Operating pages enabled: change setpoints. For programmers: select program receive , start/stop/reset/preset/search. For function block PROGRAMMER: Change of running program (without saving) |
| 2 | engineer | (Permanent) changes of program receipes and parameters |
| 3 | commissioner | Changes of basic functions, configuration |
| 4 | chief engineer | Changes of basic functions, configuration |

Overview

- 1 Title
- 2 Lowest "access level": only display, no operation.
- 3 Access level 1 ("operator"): access to operating pages
- 4 Access level 2 ("engineer"): access level 1 and additionally parameters and (permanent) changes of programs of programmers
- 5 Access level 3/4 ("commissioner", "chief engineer"): access level 2 and additionally configurations
- 6 Button "Leave Operating Page"
- 7 Button "Alarm"
- 8 Marker of active access level (here: access level 2 "engineer")

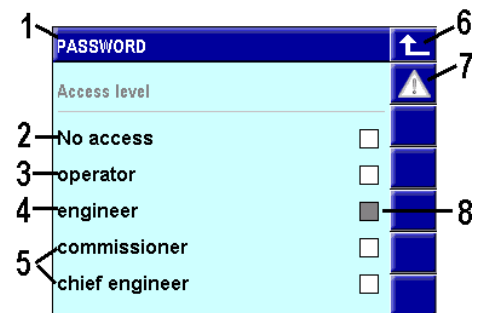
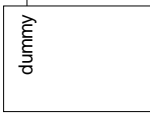


Fig. 512: PASSWORD (with all 4 access levels)

III-13 General device functions

III-13.1 GENERAL_EASY (General device functions (No. 200))



GENERAL_EASY

Abb. 513 Function of device : Use of menu "device data"

Use only one function block **GENERAL_EASY**.

Function block **GENERAL_EASY** implements menu "device data" into the main menu. With it comes the access to the sub menus "date/time", "info" and menu "Screen cleaning".

| Input | | |
|-------|------|-------------|
| Name | Type | Description |
| dummy | Bool | Not used. |


| Configuration | | | | | | | |
|---------------|----------|----------|--|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Language | Language | Enum | Setting of the language for the device, including operating pages and pages for parameters, configuration, I/O data and device data ("Miscellaneous"). Number of this setting is output at status block output "Langu_"; soft-wire to text function blocks to switch their language texts. | r/w | 0 | | |
| | | Deutsch | Language of menu is German. | | 0 | | |
| | | English | Language of menu is English. | | 1 | | |
| | | Francais | Language of menu is French. | | 2 | | |

| | | | | | | | |
|------------|------------|---------------|---|-----|---------------|----------------|--|
| LinkMode | LinkMode | Enum | Ethernet standard of the applied network. Set purposefully, if connection does not work properly with setting "Auto". Hint: This setting becomes active after a restart only. | r/w | 0 | | |
| | | Auto | The communication parameters will be determined automatically. | | 0 | | |
| | | 100BaseTx-FD | 100 MBit/s, full duplex. | | 1 | | |
| | | 100BaseTx-HD | 100 MBit/s, half duplex. | | 2 | | |
| | | 10BaseT-FD | 10 MBit/s, full duplex. | | 3 | | |
| | | 10BaseT-HD | 10 MBit/s, half duplex. | | 4 | | |
| DhcpMode | DHCP Mode | Enum | Setting, whether the entered static IP address is used or whether the device should use an IP address assigned dynamically via a DHCP server. Hint: This setting becomes active after a restart only. | r/w | 0 | | |
| | | DHCP Disabled | KS108 uses a fixed IP address. | | 0 | | |
| | | DHCP Enabled | KS108 uses an IP address dynamically assigned by a DHCP server. | | 1 | | |
| Contrast | Contrast | Int | Contrast of display. | r/w | 135 | 120 ... 150 | |
| Brightness | Brightness | Int | Brightness of display. | r/w | 135 | 120 ... 150 | |
| IpAddress | IP address | Text | Setting of the IP address of KS108, format | r/w | 192.168.0.108 | | |

| | | | | | | | |
|---------------|---------|------|---|-----|---------------|--|--|
| | | | xxx.xxx.xxx.xxx . Address must be unique in the network. Hint: This setting becomes active after a restart only. | | | | |
| NetMask | NetMask | Text | NetMask (or subnet mask) is used in networks with IP protocol to restrict (255) or allow (0) the access to IP addresses in the network. Hint: This setting becomes active after a restart only. | r/w | 255.255.224.0 | | |
| GatewayIpAddr | Gateway | Text | IP address of the gateway in use. The Gateway enables the communication between networks. Hint: This setting becomes active after a restart only. | r/w | 10.0.0.0 | | |
| HwCode | HW code | Text | Code number of hardware. | r | | | |

III-13.2 Using the "General Data" menu

In the "General Data" menu you can make fundamental system settings. For example you can set the date, the time, or the IP address.

 **NOTE!**
*The structure of the "General Data" menu and the possibility to activate the menu depend on the application development. You can only access the "General Data" menu or access the sub-menu commands, if these possibilities have been provided by the application technician.
 This means: Only if function block GENERAL_EASY is in the engineering the menu "Miscellaneous" is provided.*

The following submenu commands are available as maximum:

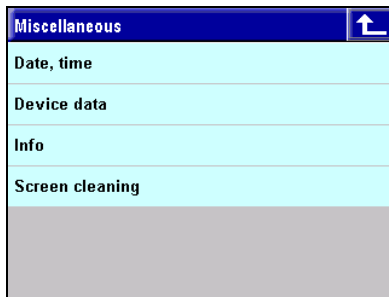


Fig. 514: Menu "General Data"

- **Date, Time** Set the system time
- **Device Data:** Select the language used by the PMA library. German and English can be selected.
- **Info:** Information on the software release version. This information can be useful when requesting service.
- **Clean screen:** An empty screen is displayed so that commands are not executed when the screen is being cleaned.

Date, Time

Set the system date in the *Date, Time* menu.



Fig. 515: Menu "Date, Time"

- Tap on a menu option to change a setting. Use the numerical value editor or the list selection editor (for month) to make the selection.

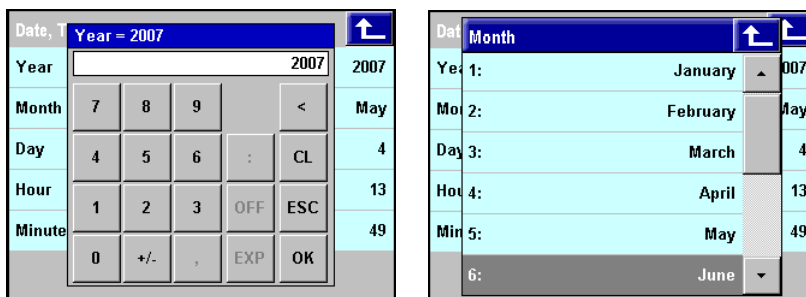


Fig. 516: Year and month selection

Device Data

In the *Device Data* menu configure the network and change device settings (language and screen settings)

| Device data | |
|-------------|--------------|
| Language | English |
| IP address | 85.86.52.18 |
| NetMask | 120.86.52.18 |
| Gateway | 120.86.52.18 |
| LinkMode | 100BaseTx-FD |
| DHCP Mode | DHCP Enabled |

Fig. 517: Menu "Device Data"

- **Language:** Tap on the menu option *Language* to change the device language. Input is executed with the list selection editor.
- **Brightness/contrast:** Tap on the menu options "Brightness" or "Contrast" to change the screen settings. Input is executed with the numerical value editor.

**NOTE!**

Information on changing the network settings is provided in section "**Error! Reference source not found. Configure network**".

Info

In the *Info* menu you will find information about the software release version that can be useful when requesting service. A software release differing in error corrections and / or added functions from the prior release is marked with a different version number. Mind these version numbers when transferring an engineering and therefor also when doing software updates.

| Info | |
|-------------------|--------------------|
| Operating version | 1 |
| Firmware version | 1.4.000 |
| HW code | K108-130-00000-000 |

Fig. 518: Menu "Info"

- **Op-Version:** Version of the PMA library on the *KS 108*.
- **FW-Version:** Version of the firmware.
- **HW-Code:** Hardware code of the device. The hardware code is a unique device designation.

III-13.2.1 Configure network

If you are using an Ethernet connection for communication with the *KS 108* then you must configure network settings. To do this, proceed as follows:

**NOTE!**

Contact your system administrator to determine the type of your network or the type of network connection of the *KS 108*.

Only make changes to the network configuration if you are aware of all necessary network parameters. An incorrect device network setting suppresses device communication and can cause general network malfunctions (e.g. double assignment of IP-Addresses).

Enter IP-Address

With the device you can either assign a permanent IP-Address, or you can have an address assigned dynamically via a DHCP server. Additional information on the "DHCP" option is available below. A fixed IP-Address is assigned as follows:

1. **Call the "General Data" menu:** Call the submenu "General Data".
2. **Call the "General Data" menu:** Call the "Device Data" submenu.
3. **Select the "IP-Address menu command:** Tap on the menu command "IP-Address". The current IP-Address is displayed on the right side of the button.
4. **Enter the IP-Address:** Enter the new IP-Address with the numerical value editor. Tap on the "OK " button to save your entry.

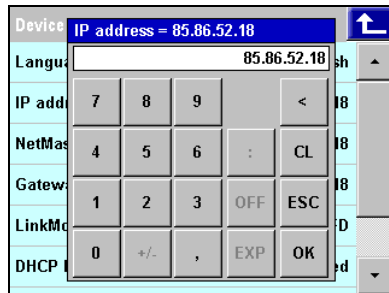


Fig. 519: Enter IP-Address

Enter the network mask

1. **Call the "General Data" menu:** Call the submenu "General Data".
2. **Call the "General Data" menu:** Call the "Device Data" submenu.
3. **Select the "NetMask" menu command:** Tap on the menu command "NetMask". The current IP-Address is displayed on the right side of the button.
4. **Enter the netmask:** Enter the new netmask with the numerical value editor. Tap on the "OK " button to save your entry.

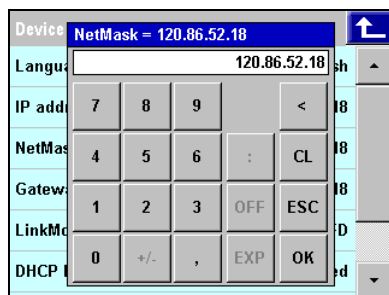


Fig. 520: Enter NetMask

Enter the gateway IP-Address:

1. **Call the "General Data" menu:** Call the submenu "General Data".
2. **Call the "General Data" menu:** Call the "Device Data" submenu.
3. **Select the "Gateway" menu command:** Tap on the "Gateway" menu command. The current gateway is displayed on the right side of the button.
4. **NetMask input:** Enter the new gateway with the numerical value editor. Tap on the "OK " button to save your entry.

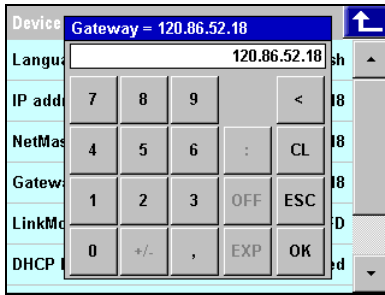


Fig. 521: Input gateway

Configuring the network connection type

Use the "LinkMode" setting to specify the Ethernet standard that your network works with. Normally you can use the "Auto" setting. However if problems should occur you can also explicitly specify the type of network connection.

The following options are available:

| Option | Explanation |
|---------------|--|
| Auto | The communication parameters will be determined automatically. |
| 100base-Tx FD | 100 Mbit/s, full duplex |
| 100base-Tx HD | 100 Mbit/s, half duplex |
| 10base-T FD | 10 Mbit/s, full duplex |
| 10base-T HD | 10 Mbit/s, half duplex |

Proceed as follows to configure the network connection type:

1. **Call the "General Data" menu:** Call the submenu "General Data".
2. **Call the "General Data" menu:** Call the "Device Data" submenu.
3. **Select the "LinkMode" menu command:** Tap on the "LinkMode" menu command. The current network connection type is displayed on the right side of the button.
4. **Select LinkMode:** Select the new network connection type in the list selection editor.

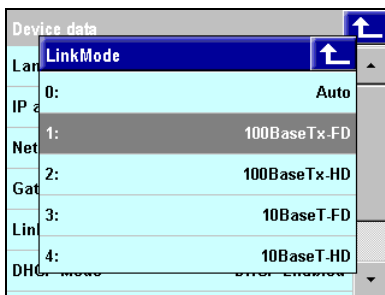


Fig. 522: Specify LinkMode

Configure DHCP mode

If the KS 108 will not be working with a fixed IP-Address, but rather with an IP-Address that is dynamically assigned by a DHCP server, then the steps below are necessary:

1. **Call the "General Data" menu:** Call the submenu "General Data".
2. **Call the "General Data" menu:** Call the "Device Data" submenu.
3. **Select the "DhcpMode" menu command:** Tap on the "DhcpMode" menu command. The current status is displayed on the right side of the button.
4. **Select DhcpMode:** Select the desired mode in the list selection editor.

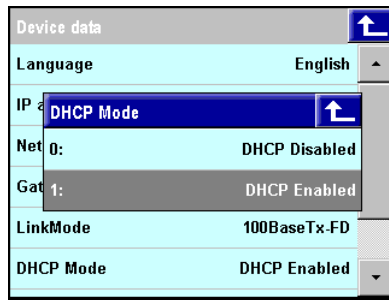
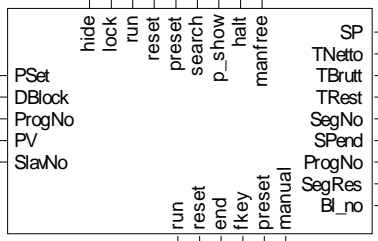


Fig. 523: Specify DhcpMode

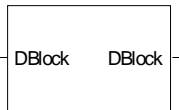
III-14 Programmer

III-14.1 A_PROG (analog programmer (No. 24))/ A_PROG_D (APROG-Data (No. 26))



A_PROG

Abb. 524



A_PROG_D

Abb. 525

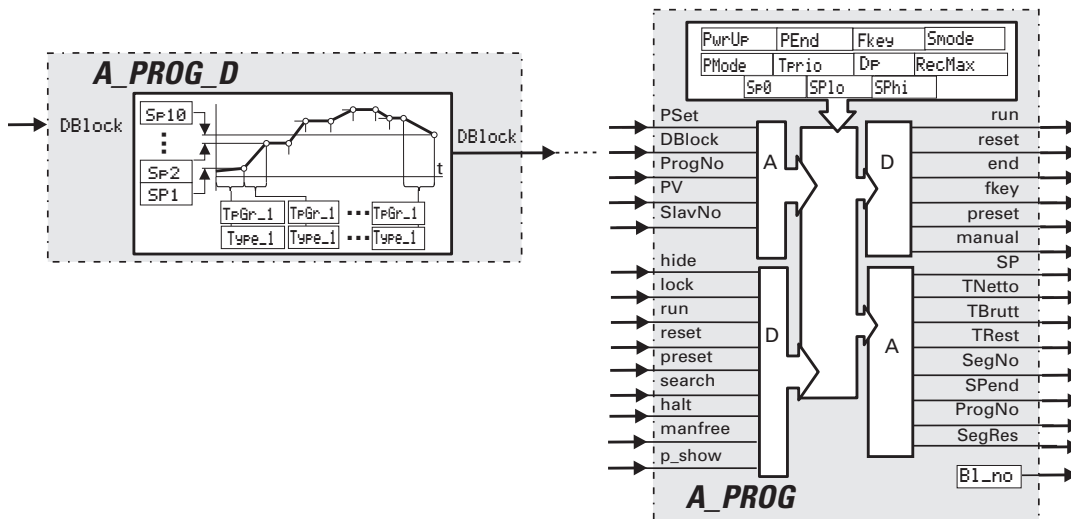


Abb. 526

General

An analog programmer comprises a programmer (**A_PROG**) and min. one data block (**A_PROG_D**), whereby output **DBlock** of the **A_PROG_D** is connected with input **DBlock** of the **A_PROG**.

By connection of several of these cascadable functions (each with 10 segments), a programmer with any number of recipes with any number of segments can be realized. Limiting is only in the number of available block numbers and in the calculation time.

The data block (**A_PROG_D**) has an analog output, at which the own block number is made available. This information is read-in by the programmer and used for segment data addressing.

If an error with segment data addressing is detected, the reset value is output (status display on operating page: 'Error'). After an engineering download, **Segment = 0** is output (Reset). If **run** is not connected, **stop** is used.

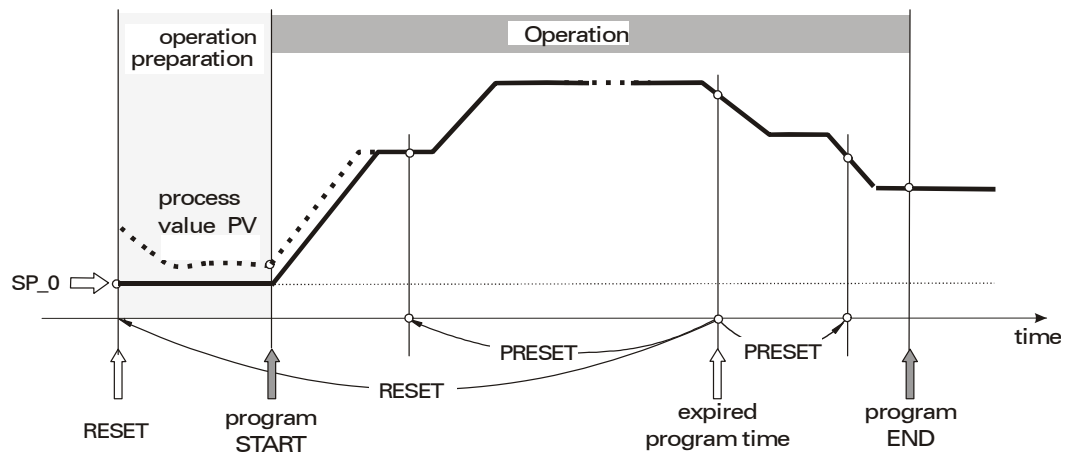


Abb. 527

| In-/Outputs A_PROG | | |
|--------------------|-------|--|
| Name | Type | Description |
| PSet | Float | Preset value for program |
| DBlock | Float | Connection of the first data block A_PROG_D. |
| ProgNo | Float | Required program number (recipe). The program number (setpoint) determines which program should be started next. Running programs are not influenced. The selected program is activated only after the next reset or re-start. |
| PV | Float | Process value for search run |
| SlavNo | Float | Block number of a connected slave output (for coupling master and slave A_PROG or D_PROG outputs or output bits) |
| hide | Bool | Display suppression. hide = 1: the page is not displayed in the operation. |
| lock | Bool | Value adjustment blocking via operation. lock = 0: adjustment enabled, lock = 1: adjustment disabled. |
| run | Bool | Program stop/run. run = 0: stop, run = 1: run |
| reset | Bool | Continue/reset program. reset = 0: continue, reset = 1: reset |
| preset | Bool | Program preset, 1 = preset |
| search | Bool | Start programm search run, 1 = search run |
| p_show | Bool | Enable program editing. Display and adjustment of all segment parameters pertaining to an effective recipe in a scroll window. Calling up this particular parameter page is done from the operating page. |
| halt | Bool | Program interruption (e.g. due to an exceeded bandwidth detected outside the programmer). Only via this digital input. Output "run" remains active. halt = 0: program is not halted. halt = 1: program is halted. |
| manfree | Bool | Disable manual mode. manfree = 0: switch-over to manual mode is not permitted, manfree = 1: switch-over to manual mode is permitted. |

| Name | Type | Description |
|--------|-------|--|
| SP | Float | Programmer setpoint |
| TNetto | Float | Net program time. Elapsed program time without halt/stop times. |
| TBrutt | Float | Gross program time. Elapsed program time inclusive of halt/stop times. |
| TRest | Float | Remaining programmer time |
| SegNo | Float | Current segment number |
| SPend | Float | End value of current segment |
| ProgNo | Float | Current program number (recipe) |
| SegRes | Float | Remaining segment time |
| Bl_no | Float | Own block number (e.g. for coupling master and slave blocks) |
| run | Bool | Program stop/run status; 0 = program stop, 1 = program run |
| reset | Bool | Program reset status, 1 = program reset |
| end | Bool | Program end status, 1 = program end reached |
| fkey | Bool | F key status / 'fkey' interface function (pressing the F key causes switch-over). |
| preset | Bool | Single preset command: a pulse is output for the duration of one cycle (dependent on the programmer cycle time). Continuous preset command: the output is always active. preset = 0: no preset status, preset = 1: A_PROG stands in preset status. |
| manual | Bool | Indication of manual mode; manual = 0: A_PROG works in automatic mode, manual = 1: A_PROG works in manual mode. |

| Parameter | | | | | | | |
|-----------|----------------|----------|--|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| PMode | Preset mode | Enum | Preset mode for input PSet or interface: Preset to segment or to time. | r/w | 1 | | |
| | | Segment | Preset to segment. With preset to segment, display of the remaining segment time is suppressed (e.g. with digital slave output bits). Used e.g. by the master to force slave outputs to the same segment number. | | 0 | | |
| | | Time | Preset to time. With preset to time, the net program time is adjustable. Used e.g. by the master to force slave outputs to the same time. | | 1 | | |
| TPrio | Start priority | Enum | Start mode in search run determines the higher priority of gradient or segment/time. | r/w | 0 | | |
| | | Gradient | Start mode in search run: Gradient has priority. When starting the search run, setpoint SP is set to the value of input PV and goes to the segment end value | | 0 | | |

| | | | | | | | |
|--------|-----------------|----------|---|-----|-----|----------|--|
| | | | with the current gradient. | | | | |
| | | Time | Start mode in search run: Time has priority. When starting the search run, setpoint SP is set to the value of input PV and approach to the segment end value is in the current remaining segment time. (The search run remains limited to the current segment, i.e. the setpoint goes from the current process value to the segment end value in the current remaining segment time.) | | 1 | | |
| Dp | Decimals | Int | Number of digits behind the decimal point for display of the setpoint | r/w | 3 | 0 ... 3 | |
| RecMax | Max. recipe no. | Int | Maximum number of recipes | r/w | 99 | 1 ... 99 | |
| SMode | Search mode | Enum | Perform search run either in the segment, in the program or programm section, or don't perform search run at all. | r/w | 0 | | |
| | | Segment | Search run in the segment. When starting the search run, setpoint SP is set to the value of input PV and approach to the segment end value is with the current gradient (TPrio = Grad.Prio) or in the current remaining segment time (TPrio). Latest start time is the end of segment, if the search value is out of the current segment. | | 0 | | |
| | | Programm | Search run in the program or program section. Search only in segments the gradients of which have the same sign. (Hold segment is neutral). As search cycle times may be very long, several cycles are made: search is only in one segment per cycle. | | 1 | | |
| | | Off | Search run switched off | | 2 | | |
| SP_0 | Reset setpoint | Float | Program setpoint after reset, rest condition. Each program begins with a start position SP_0. This is the position after reset, or after initial programmer set-up. | r/w | 0.0 | | |
| SPlo | Min. setpoint | Float | Min. setpoint limit | r/w | MIN | | |
| SPhi | Max.setpoint | Float | Max. setpoint limit | r/w | MAX | | |

| Configuration | | | | | | | |
|---------------|-------------------|-------------------|--|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| PwrUp | On mains recovery | Enum | Behaviour after mains recovery | r/w | 0 | | |
| | | Continue | Continue program at the point it was stopped by power failure. | | 0 | | |
| | | Search | Search run after mains recovery: forward and backward search from the point of failure to the next gradient change. | | 1 | | |
| | | Continue at time | Continue at actual time. Search run after mains recovery: forward and backward search from the program time in which the program would be without power failure until the next gradient change Hint: If the program time from power failure to power recovery includes min. one segment with wait status at the end, the search run in the segment in which the program would be without power failure is omitted and the program stops at the first wait status without search run. | | 2 | | |
| PEnd | On Program end | Enum | Behaviour at program end: either the last setpoint remains unchanged, or rest condition SP_0 (automatic re-start), or continuous rest condition SP_0 with reset and stop. | r/w | 0 | | |
| | | Stop | After program end: stop (the setpoint of the last segment remains unchanged). | | 0 | | |
| | | Reset | After program end: reset (goes to rest condition SP_0. The program restarts automatically, when the run condition has remained unchanged.) | | 1 | | |
| | | Reset + Stop | After program end: reset+stop (continuous rest condition SP_0 with reset und stop). | | 2 | | |
| FKey | FKey | Enum | FKey (function of the F key) can switch output Fkey, or provide a pulse at output Fkey, or control the programmer. | r/w | 0 | | |
| | | Switch (fkey) | The F key provides switch-over of the status at the fkey output (previous function). | | 0 | | |
| | | Tip switch (fkey) | The F key generates a pulse at the fkey output (pulse length = 1 cycle). | | 1 | | |
| | | Program control | The F key controls the programmer (fkey output generates a pulse when pressing the key, pulse length = 1 cycle). | | 2 | | |
| Unit | Unit | Text | Unit of the setpoint | r/w | Unit | | |

In-/Outputs A_PROG_D

| Name | Type | Description |
|--------|-------|--|
| DBlock | Float | Block number of cascaded data function A_PROGD |
| Name | Type | Description |
| DBlock | Float | Own block number |

Parameter

| ID | Name | Type | Description | Access | Default | Range | Off |
|------------|----------------|-------------|--|--------|---------|-------|-----|
| Type_1..10 | Segment type 1 | Enum | Setpoint behaviour in segment 1. The setpoint can be held, or altered using a ramp or a step change. Switch-over to the next segment is automatic or manual ("wait" for operation; configurable). | r/w | 8 | | |
| | | Time | Time to setpoint: The setpoint changes linearly from the start value to the target setpoint (Sp) of the segment during time TpGr (segment duration). The gradient is a function of this change (start value = end value of the previous segment). | | 0 | | |
| | | Rate | Rate to setpoint: The setpoint changes linearly from the start value to the target setpoint (Sp) of the segment with the adjusted gradient TpGr. The duration of the segment is a function of this change (start value = end value of the previous segment). | | 1 | | |
| | | Dwell | Dwell: The end setpoint of the previous segment is kept constant for the time TpGr. | | 2 | | |
| | | Step | Step to setpoint: The setpoint goes to the adjusted target setpoint (Sp) immediately. | | 3 | | |
| | | Time + wait | Time to setpoint and wait: The setpoint changes linearly from the start value to the target setpoint (Sp) of the segment in time TpGr. The programmer goes | | 4 | | |

| | | | | | | | |
|------------|-----------------|--------------|---|-----|-----|-----------------|--|
| | | | to stop condition at the end of segment. | | | | |
| | | Rate + wait | Rate to segment and wait: The setpoint changes linearly from the start value to the target setpoint (Sp) of the segment with the adjusted gradient TpGr. The duration of segment is a function of this change (start value = end value of the previous segment). At the segment end, the programmer goes to the stop condition. | | 5 | | |
| | | Dwell + wait | Dwell and wait: Dwell segment: The end setpoint of the previous segment is kept constant during time TpGr. At the end of the segment, the programmer waits. | | 6 | | |
| | | Step + wait | Step and wait: Step segment: The programmer goes to the adjusted target setpoint (Sp) immediately and waits at the segment end. | | 7 | | |
| | | End | The last segment in a program is the end segment. When reaching the end segment, the setpoint output last is held. | | 8 | | |
| TpGr_1..10 | Time/Gradient 1 | Float | Time or gradient for segment 1. The duration of a segment can be determined directly, or as a gradient and setpoint difference SP - segment start setpoint. Whether segment time or gradient are concerned is determined in parameter Segment type (Type). | r/w | 0.0 | 0.0 ... 1800000 | |
| SP_1..10 | Setpoint 1 | Float | End value for segment 1. Target value at the end of the first segment. Approach to this value is from the last valid setpoint (from the process value at the beginning of the 1st segment). After elapse of the program, the controller continues controlling using the target setpoint adjusted last. | r/w | 0.0 | | |

Cascading

Cascading **A_PROG_D** function blocks permits realization of a programmer with any number of segments. The segment sequence is dependent of the **A_PROG_D** function block wiring (→ see below).

The block numbers are without signification related to the order. Segment parameters from right to left in the data blocks

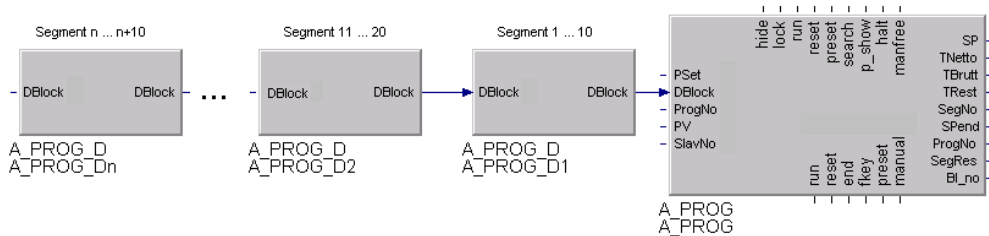


Abb. 528 Example of an analog programmer with n segments

Recipes

Analog output '**ProgNo**', at which the actual recipe number is output, and one or several **SEL_V** function blocks can be used to select a recipe.

The block number of which is switched to the **A_PROG** input (→ see below).

Selection of the required recipe is possible via analog input '**ProgNo**' or recipe number, which can be entered via operation/interface.

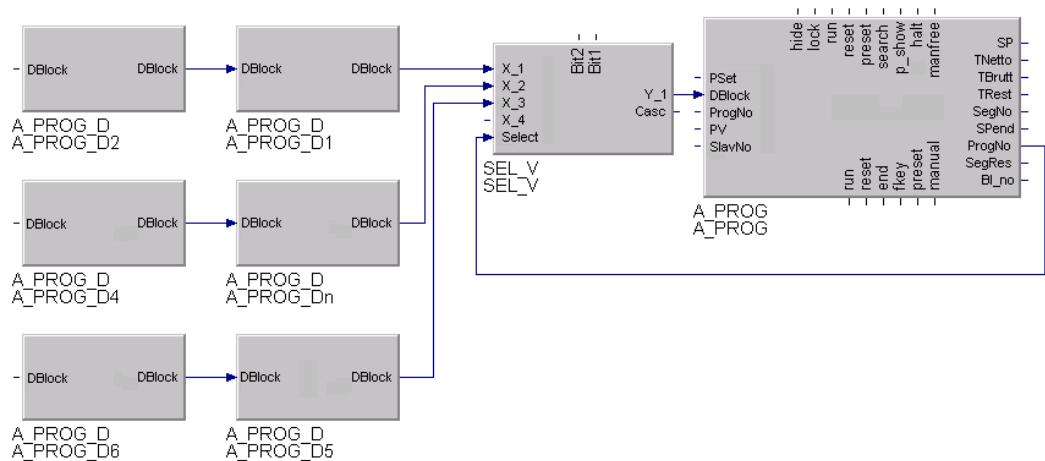


Abb. 529 Example of an analog programmer with 3 recipes à 20 segments



NOTE!

After switching over, the new recipe can be activated by the **A_PROG** only after ten **A_PROG** cycles.

Recipe change- program selection

Whilst a program is active, switching over to a different recipe on the programmer operating page is not possible. Recipe changing is possible only in reset status!

Recipe names

By linking **TEXT** blocks to the **ProgNo** input, display of recipe names rather than of recipe numbers is possible

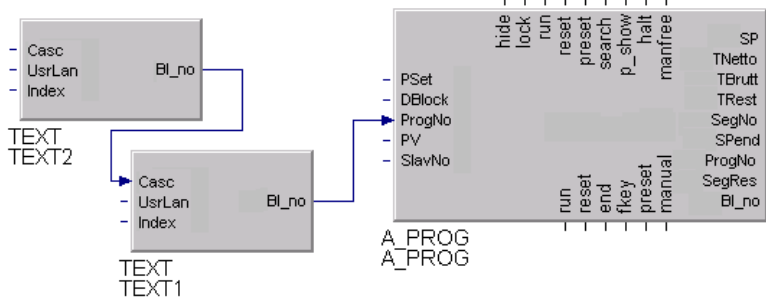


Abb. 530

This procedure can be used for both internal and external recipe selection.

With external recipe selection, the required recipe number must be applied to the index input of the **TEXT** block next to the **A_PROG** block (block 102 in this example).

This recipe number is fed to the **ProgNo** input of the programmer. With internal recipe selection (via operation or level 1 interface data), the index input of the text block need not be connected.

Operation preparation and end position

Each program starts at an initial position **SP_0**, which is used and maintained with reset or first programmer set-up

With program start from rest position, the first programmer segment runs from the instantaneous process value at the time of start command, if the corresponding process value was soft-wired at **PV** of the **A_PROG** and **search run** was configured.

With step change mode, the setpoint of the first segment is activated immediately.

At program end, either

- 0=Stop: the setpoint of the last segment is maintained

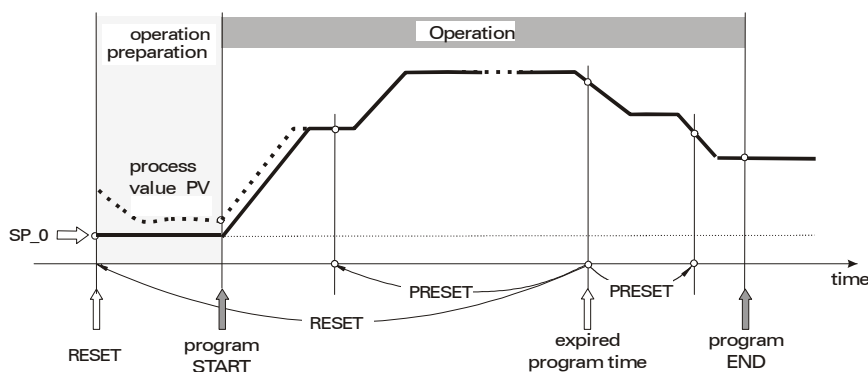


Abb. 531 Profile, with the end position kept unchanged

- 1 = Reset: the programmer goes to rest position **SP_0** and restarts automatically, if run is still valid.

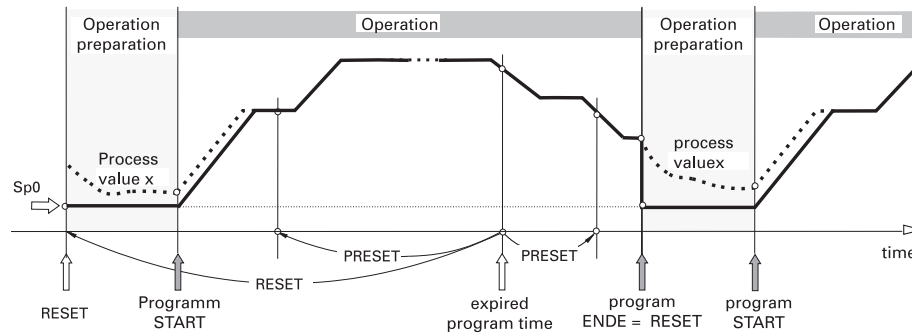


Abb. 532

- 2 = Reset + Stop: Programmer goes to rest condition **SP_0**

At program end, the number of the last segment increased by 1 is output as active segment number (**SegNo** output of operating page and interface). This is required to bring the slave block to the end status safely with a segment preset

Start setpoint

The programmer uses a common start **SP_0** for all programs. However, using an individual startvalue for a recipe is possible as follows:

Use the setpoint of the 1st segment of each program as start value.

Set the relevant segment time (**TpGr**) to 0.

Search run parameter **SMode** must be set to 'Search run in program section'. Now the search run is not limited only to the 1st segment and program start at the process value in the 2nd segment is possible.

If each recipe without search run should have an own reset setpoint (**SP_0**), the **SP_0** must be adapted as well when switching over recipes.

Halt status

Used e.g. for bandwidth monitoring

The **halt** status can be switched on and off only via the **halt** control input. Unlike the **stop** status, the **run** status remains unchanged (**run** output remains active) in the **halt** status.


Status display is "**halt**".

Automatic/manual operation




The programmer can operate both in automatic or in manual mode:



automatic: The effective setpoint is determined by the programmer.

manual: The effective can be altered via programmer operating page or via interface. However, the program continues running and can be influenced via control inputs and operation/interface as during automatic mode (run/stop/reset/preset/search).

- When switching over from automatic \Rightarrow manual, the effective setpoint remains at the last value which was effective before switching over.
- Manual \Rightarrow automatic switchover: The effective setpoint changes from the manual-setpoint to the actual programmer setpoint.
- Switching over can be done via the programmer operating page ( -key) or via interface.
- The automatic/manual mode is displayed only via the **do_manual** output.
 - 0 = automatic
 - 1 = manual
- The "**di_manfree**" control input can be used to enable switchover.
 - 0 = switchover to manual is disabled
 - 1 = switchover to manual is enabled

Programmer control via -key

Programmer control is possible by means of digital function block inputs, status changing on the operating page,  (selection key), interface and also using the -key. For selection of the -key functionality, a configuration parameter is offered:

- FKey: 0 = toggle bit changes at each key pressure at output fkey
 1 = -key function with pulse at output fkey
 2 = -key controls the programmer (fkey output generates a pulse when a key is actuated)
 Hereby, the rule that the statuses at connected control inputs have priority over the operation is applicable. The following diagram describes the status sequence dependent on the actions:

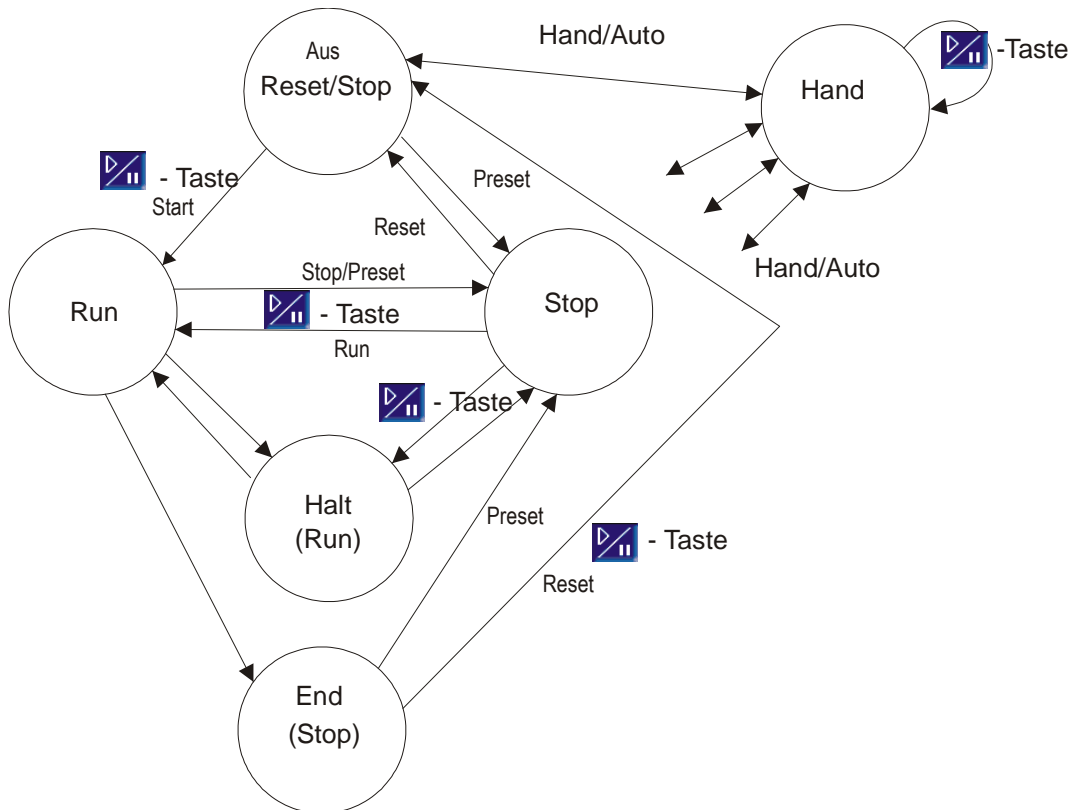


Abb. 533

Analog programmer operating page

Analog programmer **A_PROG** has an operating page, which can be selected in the operating page menu with input 'hide' not connected.

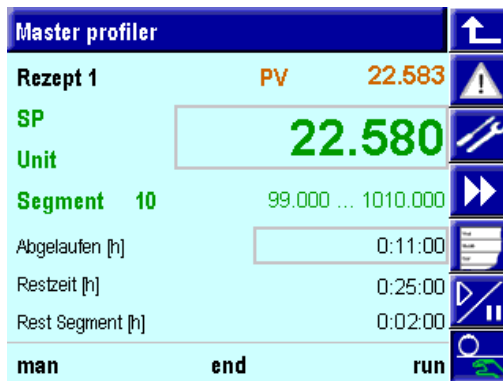
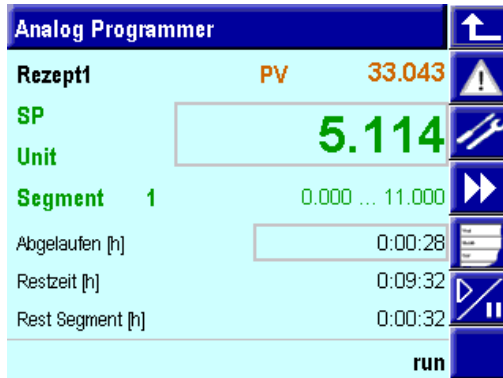


Abb. 534

Displays on left part of operating page:

Line 0: name of output

Line 1: **recipe name** / process value (if any) / measured value

Line 2: SP / setpoint

Line 3: physical unit

Line 4: segment number / **segment start and end value**

Line 5: **Elapsed program time**

Line 6: remaining program time

Line 7: segment rest time:

Separating line

Line 8: status line (display only) with

8a) Remaining segment time

8b) Halt / end status

8c) Program status: stop, run, reset, search, error

Right part of operating page, keys:

Key (line) 0: return

Key 1: display of alarm page

Key 2: display of parameter page

Key 3: Output change

Key 4: "Select field"

Key 5: programmer control (name "F key" is used in the text)

Key 6: Auto / manual key

Notes related to the features of the operating page

The identifiers marked by **bold** characters in the above picture may contain controls with variable values.

The identifiers marked by underlines in the above picture identify the items which were switched over when changing to a slave output (see section Master/slave operation). The remaining fields continue showing status and value information of the master output.

Recipe name:

Recipes can be selected in the reset status. Unless a user text (TEXT block at ProgNo input) is provided, 'Rec n' is displayed (n stands for the current recipe number).

- The process value is visible only with the process value input connected.
- The segment number is adjustable only with preset to segment.
- The setpoint can be adjusted during manual mode.
- Display of the remaining segment time is suppressed with preset to segment (e.g. with digital slave programmers).
- The program net time is adjustable with preset to time.
- 3 statuses are displayed (partly adjustable, dependent on operating status):
 - Status left: man

- Status middle: halt / end (unless one of the two statuses is active, this display is omitted)
- Status right: stop / run / reset / search / error

If the inputs (function block inputs) are used by the engineering in the following table this signal cannot be changed via the operating page (front-panel operation).

Run, reset, preset and search are concerned, see following table:

| Input field | Operation | Display | FB-input |
|--|---|---|--|
| -Key | Changing outputs | Display of information from the next slave out-put, or from the master output | - : - |
| -Key | Operating mode selection auto/manual | - / man | - : - |
| setpoint | Automatic: Programmersetpoint, Manual: Operator setting in input field | active setpoint | - : - |
| Rec | If input ProgNo is wired, entering of the desired recipe number is not possible | indicates the actual recipe number. | ProgNo |
| Seg | Entering of the desired segment number. (If control input preset is wired or PMmode is not = PresSeg, entering the desired segment number is not possible via the frontside!) | indicates the actual segment number | preset |
| tNetto | Entry of required programmer time (preset to time), If control input preset is wired or PMmode is not = PresSeg | indicates the run time total (without pause) | preset |
| Status- display (selection key)- | stop | Stop the programmer | programmer stopped |
| | run | Start the programmer | the programmer was started |
| | reset | The programmer is switched to segment 0 and 'stop' | the programmer is switched to segment 0 and 'stop' |
| | search | Start search run | Programmer makes a search run |
| -key | Program control | Changes of the status display (bottom right) | run / reset |

Program adjustment on the operating page

Direct access to parameter setting is enabled, when control input p_show = "1" at the A_PROG programmer function blocks is set. Access is via the (parameter key).

If an X_n input of the SEL_V recipe switch-over block is not connected and the relevant recipe is set nevertheless (which should be prevented by the adjustment range of the recipe number), only the recipe name (Recipe) and the start setpoint (SP_0) are indicated.

Access to parameters of inactive recipes

To enable the access to all recipes relevant for this programmer block from the programmer program edit page (including the inactive ones), the following wiring principle is compulsory:

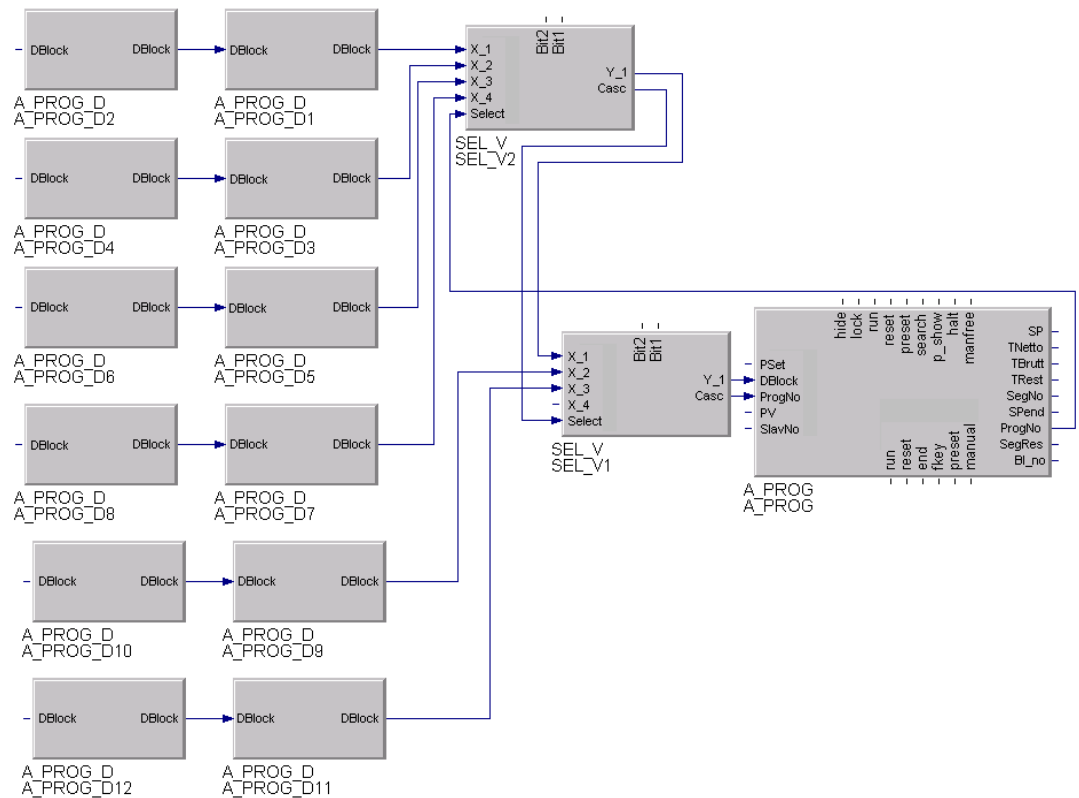


Abb. 535

The **SEL_V** block switches the parameter block number onto the programmer **Dblock** input. Via the block structure information (input select of **SEL_V**), the programmer has access to all recipes.

Unless wiring is via **SEL_V**, switchover to a different recipe on the parameter setting page is not possible, i.e. the recipe cannot be displayed.




NOTE!

For switch-over of the active recipe in reset state, however, another type of soft-wiring can be used alternatively; in this case, make sure that the block number of a new recipe is available at the **DBlock** input at the latest ten **A_PROG**-cycles after switching over.

Thereby, the order according to which the **SEL_V** block are handled plays an important part, especially if these are handled less frequently than the **A_PROG**.

Master/slave operation

Programmers frequently consist of several coupled blocks which have a common time or segment structure (e.g. master block: oven temperature, 1st slave block atmosphere/C level, 2nd slave block 1.6 digital control signals). Such a programmer is provided with an operating page extending to all blocks. On the master operating page, the slave block data can be displayed via symbol .

Wiring

Synchronization of several programmers is by slave block preset coupling. The slave blocks are forced to the same time or to the same segment number via time or segment preset by the master.

To facilitate operation of the coupled blocks, the programmer has a **SlavNo** input and a **BI-no** output. These are used by the slave block for transfer of the block number to the following block. The block the **BI-no** output of which is not connected should operate as master. Its **Tnetto** or **SegNo** output is wired to the **PSet** input of further blocks.

By linking the blocks (**BI-no > SlavNo**), a programmer with coupled blocks is created. From the master block operating page, access to the data (incl. parameters) relevant for the slave for display or adjustment is possible easily.

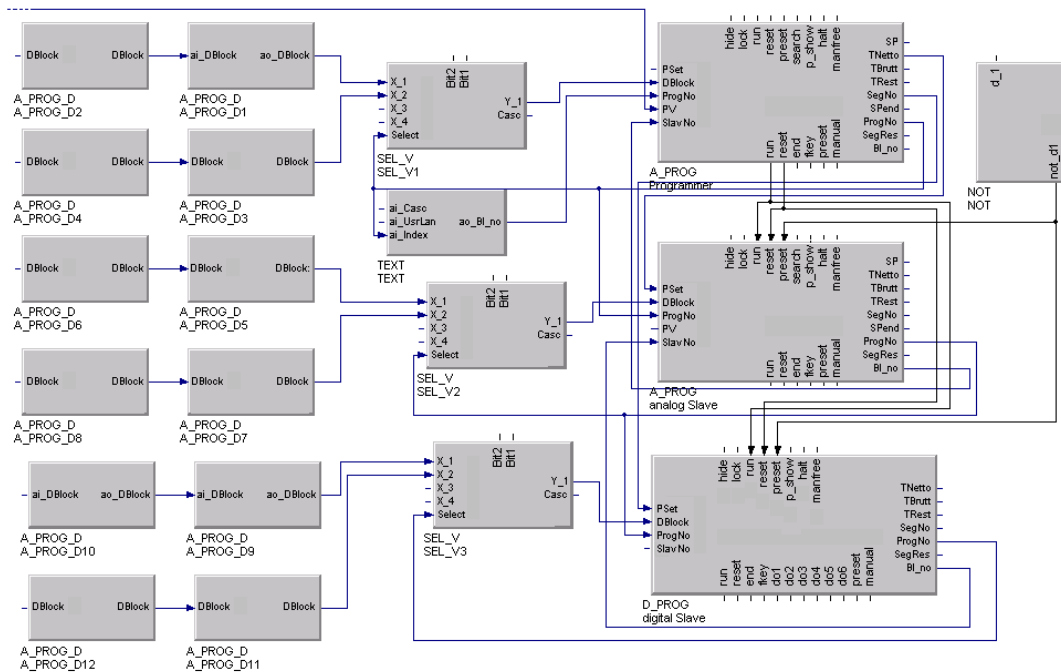


Abb. 536

Operation of a programmer with several blocks

Calling up a master block operating page via the operating page menu (page survey):

When selecting the operating page of a master programmer via the operating page menu with master/slave wiring via BI-no > SlavNo as described above, changing between the relevant programmer blocks can be done easily via the -symbol . The order is determined by the wiring order. (in the above example: Programmer → analog slave → digital slave → ...).


However, changing to the next programmer block is not complete. Only some of the values and texts (e.g. title) relevant for the next block are displayed. The remaining elements continue indicating only the master information (see analog programmer operating page).

In case a change to the operating page menu (page survey) is made in this condition of the operating page, the block selection remains unchanged. I.e. when calling up the master block again later, the data of the slave block indicated last are displayed.

Display information which is firmly assigned to the master block:


- Recipe name (can be switched over in reset state)
- Program net time (adjustable for Preset to time)
- Remaining program time
- Status display for halt/end
- Status display for stop/run/reset/search/program/quit/error (adjustable)

Display information of the actual programmer (master or slave):

- Programmer block name
- Process value
- Segment number (adjustable for preset to segment only with master)
- Actual or actual control bits (both adjustable in manual mode)
- Segment start and end value
- Remaining segment time
- Status display for automatic/ manual (switch via **auto/manual-key** 

As only the master shall decide upon the change of an active recipe, the wiring structure must be of such kind that changing is effective also for all relevant slave blocks (master ProgNo output → slave ProgNo input, see above diagram). This kind of master/slave coupling permits only a central recipe change for all blocks coupled accordingly.

Calling up a slave block operating page via the operating page menu (page survey)

When calling up the operating page of a slave programmer via the operating page menu, display of symbol  is suppressed. Easy changing to other blocks connected via the **Bl-no > SlavNo** coupling as described above is not possible. Moreover, no data of the connected master are displayed.


To prevent inadmissible adjustments (recipe selection, run/stop/reset) from being offered on a called up operating page, outputs ProgNo, run and reset of the master programmer should be wired to the relevant slave programmer inputs. For plant operation, slave programmer display is suppressed purposefully with **hide=1** when the page survey is active (PageNo at status block = 0).

Subordinated parameter page (program editing page):

On the subordinated parameter page, recipe switchover is possible at any time. But switchover acts only on the recipe parameter display on this page rather than changing to the effective recipe.

Changing directly to the parameters of the next programmer is not possible. For this, the superordinate operating page of the corresponding programmer must be used.

Programmers without coupling:

On an operating page with a function block which is not connected with other programmer blocks via **Bl-no > SlavNo** coupling, display of symbol  is suppressed.

Remaining segment time

The remaining time of the actual segment is displayed on the operating page.

It is:

- readable via interface
- available as an additional analog output signal
- always 0 with reset
- suppressed with "preset to segment"

Search Run

The term *search run* describes the behaviour of the programmer to reach the end value starting from the current process value. A search run can be performed in the current program segment or over several program segments.

A search run is used in the following cases:

- After program start: via the operator interface or via the **run** input.
- After a program restart using the **reset** command on the programmer operating page, or via the **reset** input.

- Via the **search** input.

Configuring the Programmer

In the *BlueDesign* parameter dialogue, you can determine, if a programmer can perform search runs and which type of search can be performed.

For this purpose, the *SMode* parameter offers the following options:

- Segment: search runs are made in the program segment.
- Program: search runs are made in the program section. A program section is defined as a sequence of program sections with gradient of the same direction (only upwards or only downwards). Change in direction limits a program section.
- Off: No search runs are made.

| Search mode | |
|-------------|----------|
| 0: | Segment |
| 1: | Programm |
| 2: | Off |

Abb. 537: Parameter "SMode"

Search Run in the Program Segment

With a search run, the process value (*PV*) is set as programmer set-point first. Then the process value is controlled to the segment end value either using the current gradient or using the current remaining segment time (depending on the programmer configuration).

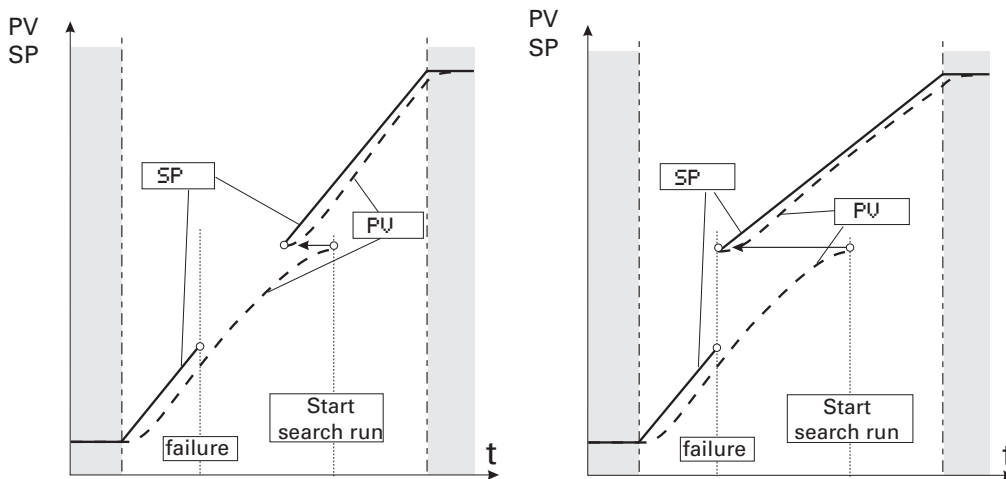


Abb. 538: Search run with the current gradient/the current remaining segment time

Configuring the programmer is done using the *TPrio* parameter in the *BlueDesign* parameter dialogue.

When making a search run, the following information must be taken into account:

- If setting "TPrio = Gradient" is used for the search run and the search value is out of the current segment, the program is continued at the segment point next to the search value.
- If the start value is equal to the end value (a segment without gradient) the program is continued at the segment start.
- With a jump segment, start is always at the beginning of the segment. The target set-point is assigned to the process value *PV*.

Search Run in the Program Section

A search run in the program section is limited to a section of several segments that have the same sign as the gradient, whereby hold segments are not considered as changes of sign. If the current program section contains hold segments, a search run is performed only, if this program section contains at least one more program segment which isn't a hold segment. If this segment is directly preceded or followed by another hold segment, the search run is realized only in the current segment.

Depending on the number of segments included in the search run, the run-time may be very long. For this reason, searching is divided into several partial runs. Each partial run checks only one segment. This is an automatic procedure which needs no interaction from operator.



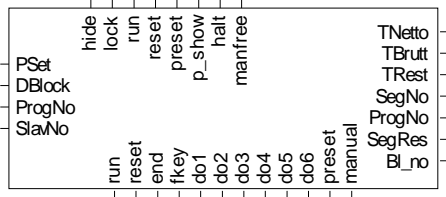
NOTE!

If the value of parameter "TPrio" is "1: time", the search run is always limited to the current program segment.

Program segments with a final waiting state (e.g. "time + wait") do not limit the search range (exception: if a search run after a power failure is concerned).

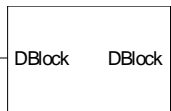
A search run can lead to termination of the program.

III-14.2 D_PROG (digital programmer (No. 27))/ D_PROG_D (D_PROG_data (No. 28))



D_PROG

Abb. 539



D_PROG_D

Abb. 540

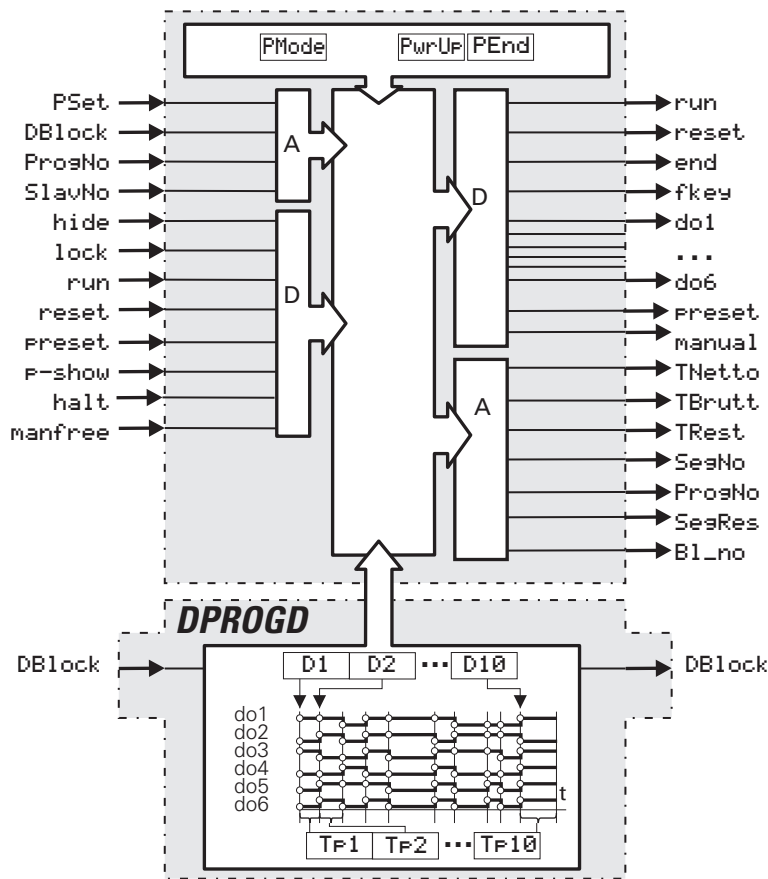


Abb. 541

General

A digital programmer comprises a programmer (D_PROG) and at least one data block (D_PROG_D), whereby output DBlock of D_PROG_D is connected with input DBlock of the D_PROG.

Connection of several of these cascable functions (each with 10 segments) permits realization of a programmer with any number of recipes and any number of segments. Limiting is only in the number of available block numbers and calculation time

The data block has an analog output, at which it provides its own block number. This information is read-in by the programmer and used for segment data addressing.

When an error in the segment data addresses is found, the reset value is output (status display on operating page: **Error**). After an engineering download, **Segment = 0** is output (Reset). If **run** is not connected, **stop** is used.

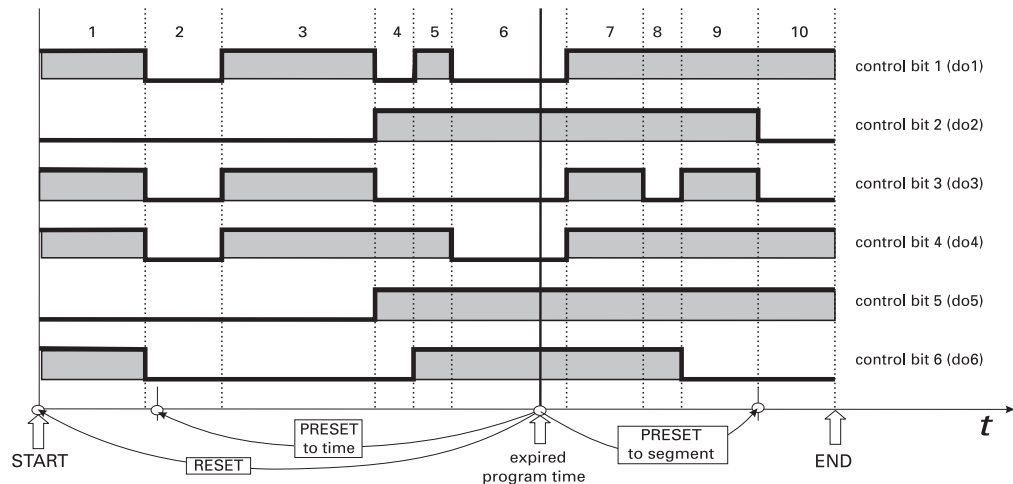


Abb. 542

D_PROG**In-/Outputs**

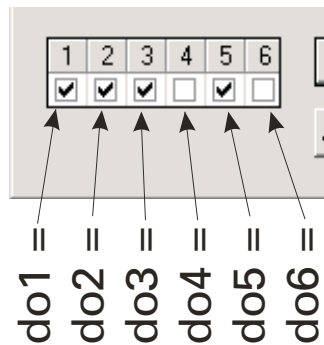
| Name | Type | Description |
|---------|-------|--|
| PSet | Float | Preset value for program |
| DBlock | Float | Connection for the first data block D_PROG_D |
| ProgNo | Float | Required program number (recipe). The program number (Soll) determines which program should be started next. Running programs are not influenced. The selected program is activated only after the next reset or re-start. |
| SlavNo | Float | Block number of a connected slave output or output bit (analog: A_PROG or digital: D_PROG) |
| hide | Bool | Display suppression, hide = 1: the page is not displayed in the operation. |
| lock | Bool | Value adjustment blocking via operation; lock = 0: adjustment enabled, lock = 1: adjustment disabled. |
| run | Bool | Program stop/run. run = 0: stop, run = 1: run |
| reset | Bool | Continue/reset program. reset = 0: continue, reset = 1: reset |
| preset | Bool | Program preset, 1 = preset |
| p_show | Bool | Program editing enabled. Display and adjustment of all segment parameters pertaining to an effective recipe in a scroll window. Calling up this particular parameter page is done from the operating page. |
| halt | Bool | Program interruption (e.g. due to an exceeded bandwidth detected outside the programmer). halt = 0: program not halted, halt = 1: program halted. |
| manfree | Bool | Disable manual mode. manfree = 0: switch-over to manual mode not permitted, manfree = 1: switch-over to manual mode permitted. |

| Name | Type | Description |
|-------------|-------|---|
| TNetto | Float | Net program time. Elapsed program time without halt/stop times |
| TBrutt | Float | Gross program time. Elapsed program time inclusive of halt/stop times |
| TRest | Float | Remaining programmer time |
| SegNo | Float | Current segment number |
| ProgNo | Float | Current program number (recipe) |
| SegRes | Float | Remaining segment time |
| Bl_no | Float | Own block number (e.g. for coupling master and slave blocks) |
| run | Bool | Program stop/run status, 0 = program stop, 1 = program run |
| reset | Bool | Continue/reset program. reset = 0: continue, reset = 1: reset |
| end | Bool | Program end status, 1 = program end reached |
| fkey | Bool | Status F key / interface function 'fkey' (pressing key F causes switch-over). |
| do1 ... do6 | Bool | Status bit 1 in the current segment |
| preset | Bool | Single preset command: a pulse is output for the duration of one cycle (dependent on the programmer cycle time). Continuous preset command: the output is always active. preset = 0: no preset status, preset = 1: D_PROG stands in preset status |
| manual | Bool | Indication of manual mode. manual = 0: D_PROG works in automatic mode, manual = 1: D_PROG works in manual mode. |

Parameter

| ID | Name | Type | Description | Access | Default | Range | Off |
|--------|--------------------|---------|--|--------|---------|------------------|-----|
| PMode | Preset mode | Enum | Preset mode for input PSet or interface: Preset to segment or to time. | r/w | 1 | | |
| | | Segment | Preset to segment. With preset to segment, display of the remaining segment time is suppressed (e.g. with digital slave output bits). Used e.g. by the master to force slave outputs to the same segment number. | | 0 | | |
| | | Time | Preset to time. With preset to time, the net program time is adjustable. Used e.g. by the master to force slave outputs to the same time. | | 1 | | |
| RecMax | Max. recipe no. | Int | Maximum number of recipes | r/w | 99 | 1 ... 99 | |
| D_0 | Reset control bits | Float | Status bits 1...6 after reset. The programmer can switch several digital signals: status bits 1...6. The reset value contains the combination of these signals, which are output after | r/w | 0.0 | 0.0 ... 111111.0 | |

reset or initial programmer set-up.



For the input of the status bits in the engineering the first number corresponds to status bit 1 (do1), the second corresponds to status bit 2 (do2), etc.

Abb. 543

| Configuration | | | | | | | |
|---------------|-------------------|-------------------|--|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| PwrUp | On mains recovery | Enum | Behaviour after mains recovery | r/w | 0 | | |
| | | Continue | | | 0 | | |
| | | Continue at time | | | 1 | | |
| PEnd | End of Program | Enum | Behaviour at program end. Either stopping, or reset, or reset with stop. | r/w | 0 | | |
| | | Stop | After end of program: stop. The value of the last segment is maintained. | | 0 | | |
| | | Reset | After end of program: reset. The programmer goes to rest position and restarts automatically, if run is still valid. | | 1 | | |
| | | Reset + Stop | After end of program: reset+stop (end condition is reset with stop). Programmer goes to rest condition. | | 2 | | |
| FKey | FKey | Enum | Fkey (function of the F key) can switch output Fkey, or provide a pulse at output Fkey, or control the programmer. | r/w | 0 | | |
| | | Switch (fkey) | The status at the fkey output (previous function) is switched over by the F key. | | 0 | | |
| | | Tip switch (fkey) | The F key generates a pulse at the fkey output (pulse length = 1 cycle). | | 1 | | |
| | | Program control | The F key controls the programmer (fkey output generates a pulse when | | 2 | | |

| | | | | | |
|--|--|--|--|--|--|
| | | the key is pressed, pulse length = 1 cycle). | | | |
|--|--|--|--|--|--|

D_PROG_D

In-/Outputs


| Name | Type | Description |
|--------|-------|---|
| DBlock | Float | Block number of cascaded data function D_PROG_D |
| Name | Type | Description |
| DBlock | Float | Own block number |

Parameter

| ID | Name | Type | Description | Access | Default | Range | Off |
|-------------------|-------------------|-------|--|--------|---------|---------------------|-----|
| Tp_1 ... Tp_10 | Segment time 1 | Float | Time for segment 1 | r/w | off | 0.0 ... 1800000 | ja |
| D_1 ... D_10 | control bits 1 | Float | Status of control bit values in segment 1 | r/w | 0.0 | 0.0 ... 111111.0 | |

D_PROG functions

All functions effective with the digital programmer are mentioned on the following list. As the procedure is almost completely identical for the analog programmer, we refer to the description in the corresponding **A_PROG** chapter

- Data blocks are cascadable
- Program selection
- Program changes during an active recipe
- Access to parameters of inactive recipes
- Programmer control via F-key 
- Halt status
- Automatic/manual mode with adjustability of control bits
- Recipe changing during reset status
- Recipe names via **TEXT** block coupling
- Program end behaviour
- Master/slave operation
- Remaining segment time on the operating page and as output signal
- Operating page elements like for the **A_PROG** (with display of control blocks and relevant block number)

Digital programmer operating page

Digital programmer **D_PROG** has an operating page, which can be selected in the operating page menu with input 'hide' not connected.

For further information on the operation: see Chapter Operating instructions.

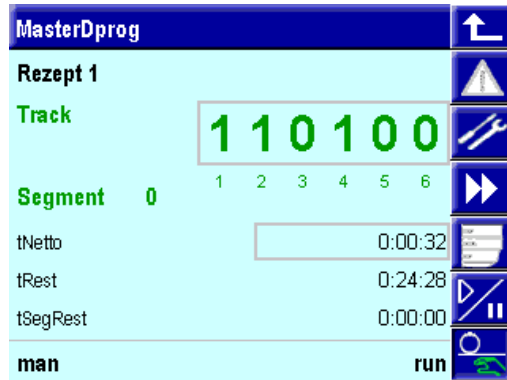


Abb. 544

Displays on left part of the operating page:

Line 0: name of programmer

Line 1: **recipe name**

Line 2: status bits (with numbers)

Line 3: < blank >

Line 4: segment number / segment start and end value

Line 5: **elapsed program time**

Line 6: remaining program time

Line 7: remaining segment time

Separating line

Line 8: status line (display only) with

8a) Remaining segment time


8b) Halt / end status

8c) Program status: stop, run, reset, search, error

Right part of operating page, keys:

Key (line) 0: return

Key 1:  display of alarm page

Key 2:  display of parameter page

Key 3:  change of bits

Key 4:  "Select field"

Key 5:  programmer control (name "Fkey" in the text)

Key 6:  auto / manual key

Notes related to the features of the operating page

The identifiers marked by **bold** characters in the above picture may contain controls with variable values.





The identifiers marked by underlines in the above picture identify the items which were switched over when changing to a slave bit (see section Master/slave operation → **A_PROG**). The remaining fields continue showing status and value information of the master bit.

*Changing bits:

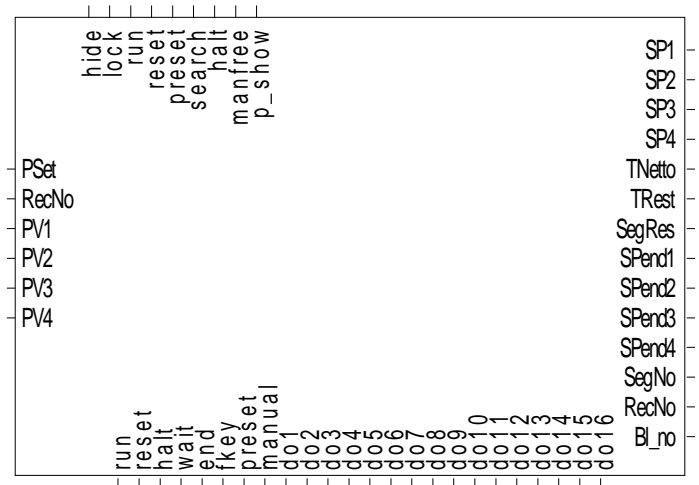
Switch-over to a different analog output or digital bit connected via programmer block coupling is possible by bit or output changing. This switch-over function is applicable only to the values marked with *. The remaining display items continue showing the values of the master bit or output.

If the inputs (function block inputs) in the following table are used by the engineering, this signal cannot be changed on the operating page (front-panel operation).

| Input fields | Operation | Display | FB input |
|--------------|-----------|---------|----------|
|--------------|-----------|---------|----------|

| | | | |
|---|--|---|--|
|  -key | Bit change | Display of data from next slave bit, or from master bit | - :- |
|  -key | auto/manual mode selection | - / man | - :- |
| Rec | Selection of required recipes is not possible, if the ProgNo input is soft-wired. | Indicates the current recipe name. | ProgNo |
| Seg | Input of required segment number (Preset to segment not adjustable via front-panel operation, if control input preset is soft-wired, or if PMode is unequal to PresSeg | Indicates the current segment number. | preset |
| tNetto | Selection of required programmer time (Preset to time), not adjustable via front-panel operation, if control input preset is soft-wired, or if PMode is unequal to PresTime . | Indicates the sum of run time (without pause times). | preset |
| Status display  (Select key) | stop | Stopping the programmer | Programmer is stopped |
| | run | Starting the programmer | Programmer is started |
| | reset | Programmer is switched to segment 0 and 'stop' . | Programmer is switched to segment 0 and 'stop' |
| | search | Start search run | Programmer makes a search run |
|  -key | Program control | Changes of status display (bottom right) | run / reset |

III-14.3 PROGRAMMER (Universal programmer (No. 29))



PROGRAMMER

Abb. 545

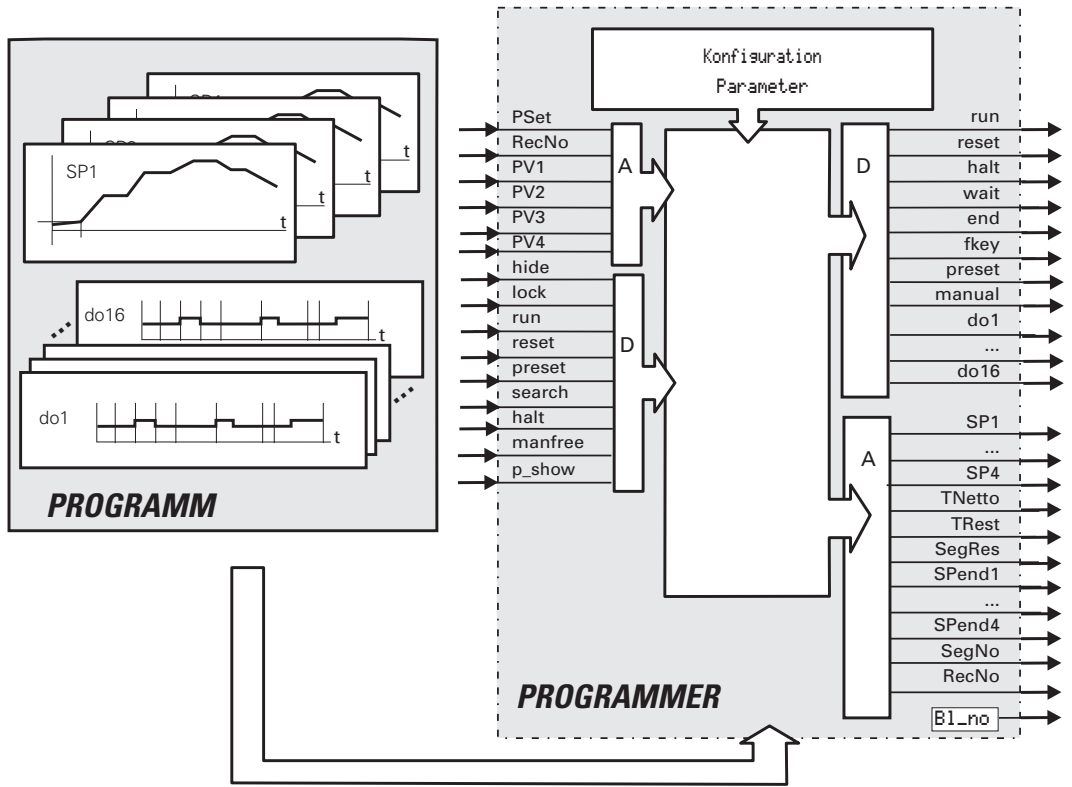


Abb. 546

General

The programmer comprises a function block **PROGRAMMER** and at least one recipe file with a program. The program files are created conveniently using the *BlueEdit* program editor, however, they can be changed in the instrument. The programmer (**PROGRAMMER**) may have up to 4 analog and up to 16 digital tracks.

Each **PROGRAMMER** file includes a recipe with a program comprising an arbitrary number of segments. Additional recipes are added as one file for each recipe. The number of segments or recipes is limited only by the available memory size.

The directory in which the recipes are stored is determined in the engineering (use an own directory for each **PROGRAMMER!**). Recipe selection is possible on the operating page or via the analog input **RecNo**.

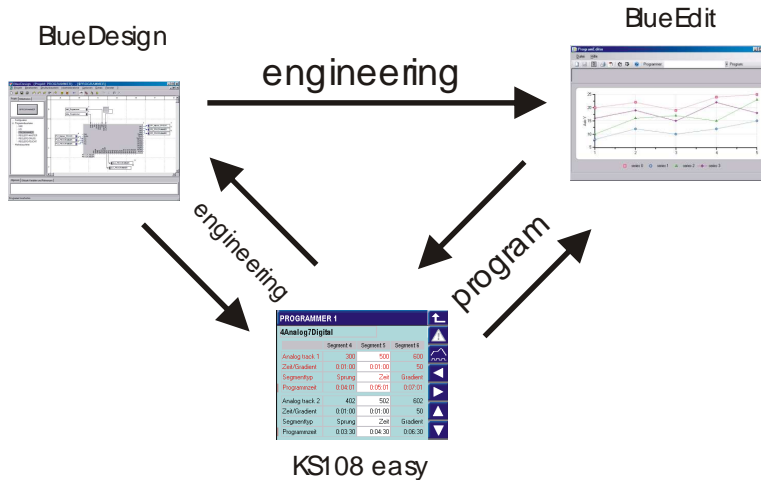


Abb. 547: Interaction of BlueDesign and KS108 easy

If an error is detected when opening a recipe file, the reset value is output (status display on operating page: 'Error'). After an engineering download, the **Reset** status is active. If input **run** isn't soft-wired, **stop** is used.

Master and slave tracks

The first analog track is always the master track. All other tracks are slave tracks.

| In-/Outputs PROGRAMMER | | |
|------------------------|-------|--|
| Name | Type | Description |
| PSet | Float | Preset value (time or segment, dependent on parameter PMode) |
| RecNo | Float | Required recipe number. The recipe number (setpoint) determines which recipe should be started next. Running recipes are not influenced. The selected recipe is activated only after the next reset. |
| PV1 | Float | Process value 1 for search run (master) |
| PV2 ... PV4 | Float | Process value 2 for search run |
| hide | Bool | Display suppression. hide = 1: the page is not displayed in the operation. |
| lock | Bool | Value adjustment blocking via operation. lock = 0: adjustment enabled, lock = 1: adjustment disabled. |
| run | Bool | Program stop/run. run = 0: stop, run= 1: run |
| reset | Bool | Continue/reset program. reset = 0: continue, reset = 1: reset |
| preset | Bool | Program preset, 1 = preset |
| search | Bool | Start programm search run, 1 = search run (operates on all analog tracks) |
| halt | Bool | Program interruption (e.g. due to an exceeded bandwidth detected outside the |

| | | |
|---------|------|---|
| | | programmer). Output "run" remains active. halt = 0: program is not halted. halt = 1: program is halted. |
| manfree | Bool | Disable manual mode. manfree = 0: switch-over to manual mode is not permitted, manfree = 1: switch-over to manual mode is permitted. |
| p_show | Bool | Enable program editing. Display and adjustment of all segment parameters pertaining to a recipe. Calling up this page is done from the main operating page. |

| Name | Type | Description |
|-------------------|-------|--|
| SP1 | Float | Programmer setpoint 1 (master) |
| SP2 ... SP4 | Float | Programmer setpoint 2 |
| TNetto | Float | Net program time. Elapsed program time without halt/stop times. (master) |
| TRest | Float | Remaining programmer time (master) |
| SegRes | Float | Remaining segment time (master) |
| SPend1 | Float | End value of current segment of analog track 1 (master) |
| SPend2 ... SPend4 | Float | End value of current segment of analog track 2 |
| SegNo | Float | Current segment number (master) |
| RecNo | Float | Current recipe number |
| Bl_no | Float | Own block number (e.g. for calling the operating page via CALLPG) |
| run | Bool | Program stop/run status; 0 = program stop, 1 = program run |
| reset | Bool | Program reset status; 1 = program reset |
| wait | Bool | Program wait status; 1 = Wait (one analog track has stopped the programmer at the end of a segment, because of a segment type with wait at the end). |
| halt | Bool | Program halt status (master); 1 = Program interruption (because of an external halt, an exceeding of the bandwidth or a wait on the other segment coupled slaves, which are still running) |
| end | Bool | Program end status (master); 1 = program end reached |
| fkey | Bool | F key status. Pressing the key causes a pulse. |
| preset | Bool | Single preset command: a pulse is output for the duration of one cycle (dependent on the programmer cycle time). Continuous preset command: the output is always active. preset = 0: no preset status, preset = 1: PROGRAMMER stands in preset status. |
| manual | Bool | Indication of manual mode, all tracks; manual = 0: none of the tracks of PROGRAMMER works in automatic mode, manual = 1: at least one track of PROGRAMMER works in manual mode. |
| do1 ... do16 | Bool | Status bit 1 |

| Parameter | | | | | | | |
|-----------|-------------|------|--|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| PMode | Preset mode | Enum | Preset mode for input PSet or interface: | r/w | 1 | | |

| | | | | | | | |
|-----------------------|-----------------|----------|---|-----|--------|---------|--|
| | | | Preset to segment or to time. | | | | |
| | | Segment | Preset to segment. Value at PSet input is used as target segment during a preset command (activated via preset input). | | 0 | | |
| | | Time | Preset to time. Value at PSet input is used as target program time during a preset command (activated via preset input). | | 1 | | |
| SMode | Search mode | Enum | Behaviour when activating search via search input or at program start | r/w | 0 | | |
| | | Segment | Search run in the segment. When starting the search run, setpoint SP is set to the value of input PV and approach to the segment end value is with the current gradient (TPrio = Grad.Prio) or in the current remaining segment time (TPrio). | | 0 | | |
| | | Programm | Search run in the program or program section. Search only in segments the gradients of which have the same sign. (Hold segment is neutral). The search cycle time may be longer than one processing cycle. | | 1 | | |
| | | Off | Search run switched off | | 2 | | |
| TPrio | Start priority | Enum | Start mode in search run determines the higher priority of gradient or segment/time. | r/w | 0 | | |
| | | Gradient | Start mode in search run: Gradient has priority. When starting the search run, setpoint SP is set to the value of input PV and goes to the segment end value with the current gradient. | | 0 | | |
| | | Time | Start mode in search run: Time has priority. When starting the search run, setpoint SP is set to the value of input PV and approach to the segment end value is in the current remaining segment time. | | 1 | | |
| Dp1 ... Dp4 | Decimals 1 | Int | Number of digits behind the decimal point for display of the setpoint 1 | r/w | 3 | 0 ... 3 | |
| SPIo1 ... SPIo4 | Min. setpoint 1 | Float | Lower limit of set-point 1 | r/w | 0.0 | | |
| SPhi1 ... SPhi4 | Max.setpoint 1 | Float | Upper limit of set-point 1 | r/w | 1000.0 | | |

| Configuration | | | | | | | |
|---------------|---------------------|------------------|--|--------|---------|----------|------|
| ID | Name | Type | Description | Access | Default | Range | Of f |
| ATracks | Analog track number | Int | Number of the used analog tracks | r/w | 1 | 1 ... 4 | |
| DTracks | Contr.output number | Int | Number of the used control outputs | r/w | 6 | 0 ... 16 | |
| PwrUp | On mains recovery | Enum | Behaviour after mains recovery | r/w | 0 | | |
| | | Continue | Continue program at the point it was stopped by power failure. | | 0 | | |
| | | Search | Search run after mains recovery: forward and backward search from the point of failure, search function depends on parameter SMode. | | 1 | | |
| | | Continue at time | Continue at actual time. Search run after mains recovery: forward and backward search from the program time in which the program would be without power failure, search function depends on parameter SMode. Hint: If the program time from power failure to power recovery includes min. one segment with wait status at the end, the search run in the segment in which the program would be without power failure is omitted and the program stops at the first wait status without search run. | | 2 | | |
| PEnd | On Program end | Enum | Behaviour at program end | r/w | 0 | | |
| | | Stop | After program end: stop (the setpoint of the last segment remains unchanged). | | 0 | | |
| | | Reset | After program end: reset (the start setpoint becomes active. The program restarts | | 1 | | |

| | | | | | | | |
|-------------------------------|---------------------|------------------|--|-----|------------------|----------------------|--|
| | | | automatically, when the run condition has remained unchanged.) | | | | |
| | | Reset + Stop | After program end: reset+stop (continuous reset, start setpoint with reset und stop). | | 2 | | |
| Coupling2 ... Coupling4 | Mast/slave coupl.2 | Enum | Coupling between slave track 2 and master track, slave 2 is coupled with time or segment of the master track. | r/w | 0 | | |
| | | Time coupling | Slave works on the same program time like the master (segments can be different). | | 0 | | |
| | | Segment coupling | Slave works every time in the same segment like the master (program time can be different). | | 1 | | |
| SpScale2 ... SpScale4 | Setpoint scaling 2 | Enum | Define scaling for the graph on the main page for track 2 independent from master or take over master scaling. | r/w | 0 | | |
| | | Own scaling | Own graphic scaling: SpLo ... SpHi | | 0 | | |
| | | Master scaling | Graphic scaling from master: SpLo1 ... SpHi1 | | 1 | | |
| Color1 ... Color4 | Setpoint color 1 | Enum | Color of graph on main page | r/w | 0 | | |
| | | Red | Red | | 0 | | |
| | | Blue | Blue | | 1 | | |
| | | Green | Green | | 2 | | |
| | | Yellow | Yellow | | 3 | | |
| TChart | Visible time [min] | Float | Visible time for graph on main page | r/w | 60.0 | 1.0 ... 1800000.0 | |
| A1Title ... A4Title | Name analog track 1 | Text | Name for analog track 1 | r/w | Analog track 1 | | |
| Unit1 ... Unit4 | Unit 1 | Text | Unit of setpoint 1 | r/w | Unit | | |
| D1Title ... D16Title | Name contr. output1 | Text | Name for control output 1 | r/w | Control output 1 | | |
| RecDir | Recipe directory | Text | Name of the directory, in which the recipes of the function block are stored | r/w | PROGRAMME R | | |

| Segment types | | | | | |
|---------------|--------------|--|--------|---------|-------|
| Name | Type | Description | Access | Default | Range |
| Segment type | Enum | Setpoint behaviour in the segment. The setpoint can be held or changed with a ramp or a step change. Continuation is automatic or manual ("Wait for operation"; configurable). | r/w | 8 | |
| | Time | Time segment: The setpoint changes linearly in the segment duration from the start value to the target setpoint of the relevant segment. The gradient is a function of this change (start value = end value of the previous segment) | | 0 | |
| | Rate | Rate to setpoint: The setpoint changes linearly with the adjusted gradient from the start value to the target value of the relevant segment. The segment duration is a function of this change (start value = end value of the previous segment). | | 1 | |
| | Dwell | Dwell: The end setpoint of the previous segment is kept constant for the time. | | 2 | |
| | Step | Step to setpoint: The setpoint goes to the adjusted target setpoint immediately. | | 3 | |
| | Time + wait | Time to setpoint and wait. The setpoint changes linearly from the start value to the target setpoint in the duration of the segment. The programmer goes to stop condition at the end of segment. | | 4 | |
| | Rate + wait | Rate to setpoint and wait: The setpoint changes linearly from the start value to the target setpoint with the adjusted gradient. The segment duration is a function of this change (start value = end value of the previous segment). At the segment end, the programmer goes to the stop condition. | | 5 | |
| | Dwell + wait | Dwell and wait: Dwell segment: The end setpoint of the previous segment is kept constant during the segment time. At the end of the segment, the programmer goes to the stop condition. | | 6 | |
| | Step + wait | Step and wait: Step segment: The programmer goes to the adjusted target setpoint immediately and waits at the segment end. | | 7 | |
| | End | The last segment in a program is the end segment. When reaching the end segment, the setpoint output last is held. | | 8 | |

Additional parameters that are determining for a program step:

| Name | Type | Description | Access | Default | Range |
|-----------------------------------|-------|--|--------|---------|-----------------|
| (Segment) time / gradient TpGr | Float | Time or gradient for the segment. The duration of a segment can be determined directly, or as a gradient and a setpoint difference SP - segment start setpoint. Whether segment time or gradient are concerned is determined in parameter Segment type (Type). | r/w | 0.0 | 0.0 ... 1800000 |

| Name | Type | Description | Access | Default | Range |
|------|------|-------------|--------|---------|-------|
|------|------|-------------|--------|---------|-------|

| | | | | | |
|----------|-------|--|-----|-----|--|
| Setpoint | Float | End value for the segment. Target value at the end of the first segment. Approach to this value is from the last valid setpoint (from the process value at the beginning of the 1 st segment). After elapse of the program, the controller continues controlling using the target setpoint adjusted last. | r/w | 0.0 | |
| SP | | | | | |

Synchronization

The analog slave tracks can be selected either as coupling by segment or by time to the master. A **time coupled** slave follows the master time, also if the master stops or changes its time because of search or preset. So it runs independently from the other slaves to its program end, where it is waiting until all tracks reach their end. After that the programmer executes its end function, which depends on the PEnd configuration.

If **segment coupling** is selected all tracks wait ('halt' state) for each other at the end of each segment, until all the tracks have ended this segment. This can happen, if the tracks have different segment times, if a search or a bandwidth violation has occurred or if a track is in a wait state at segment end.

All digital tracks are coupled by segment to the master. The segment times of these tracks are taken from the master, they do not have separate segment times.

Analog tracks



HINT!

The first analog track is always the master track. All other tracks are slave tracks.

Start setpoint

The start setpoint is part of the recipe: setpoint of the 1st segment (segment with segment number 0).

Start setpoint is active, if PROGRAMMER is in status **reset**. If no recipe is selected or the selected recipe is not available, e.g. the recipe does not exist or is not correct, or it does not match the configuration, the effective setpoint is set 0.

Operation preparation and end position

Each program starts at an initial position. This position is used and maintained with reset or first programmer set-up.

With program start from rest position, the first programmer segment runs. The program starts from the instantaneous process value at the time of start command, if the corresponding process value was soft-wired at **PV** of the **PROGRAMMER** and **search run** was configured.

With step change mode, the setpoint of the first segment is activated immediately.

At program end, depending on configuration (PEND), either

- 0=Stop: the setpoint of the last segment is maintained

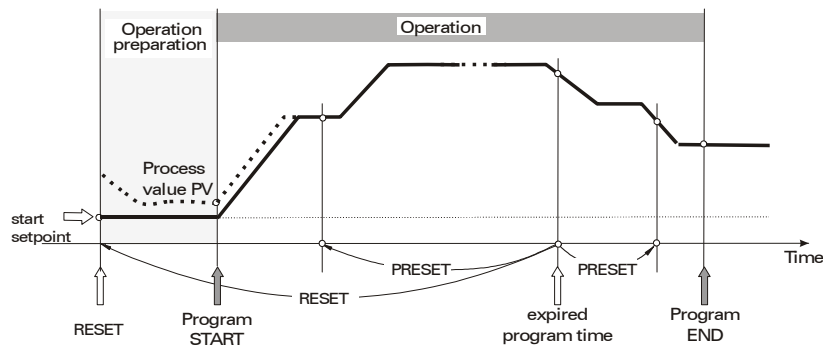


Abb. 548 Profile, with the end position kept unchanged

- 1 = Reset: the programmer goes to rest position **SP** of the first segment (segment 0). The program restarts automatically, if run is still valid.

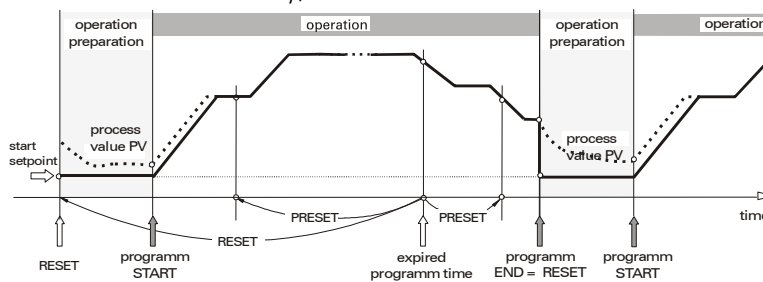


Abb. 549

- 2 = Reset + Stop: Programmer goes to rest condition **SP** of the first segment (segment 0) with reset and stops there.

At program end, the number of the last segment increased by 1 is output as active segment number (**SegNo** output). This is required to bring the slave block to the end status safely with a segment preset

Search Run

The term *search run* describes the behaviour of the programmer to reach the end value starting from the current process value. A search run can be performed in the current program segment or over several program segments.

A search run is used in the following cases:

- After program start: via the operator interface or via the **run** input.
- After a program restart using the **reset** command on the programmer operating page, or via the **reset** input.
- Via the **search** input.

Configuring the Programmer

In the *BlueDesign* parameter dialogue, you can determine, if a programmer can perform search runs and which type of search can be performed.

For this purpose, the *SMode* parameter offers the following options:

- Segment: search runs are made in the program segment.
- Program: search runs are made in the program section. A program section is defined as a sequence of program sections with gradient of the same direction (only upwards or only downwards). Change in direction limits a program section.

- Off: No search runs are made.

| Search mode | |
|-------------|----------|
| 0: | Segment |
| 1: | Programm |
| 2: | Off |

Abb. 550: Parameter "SMode"

Search Run in the Program Segment

With a search run, the process value (PV) is set as programmer set-point first. Then the process value is controlled to the segment end value either using the current gradient or using the current remaining segment time (depending on the programmer configuration).

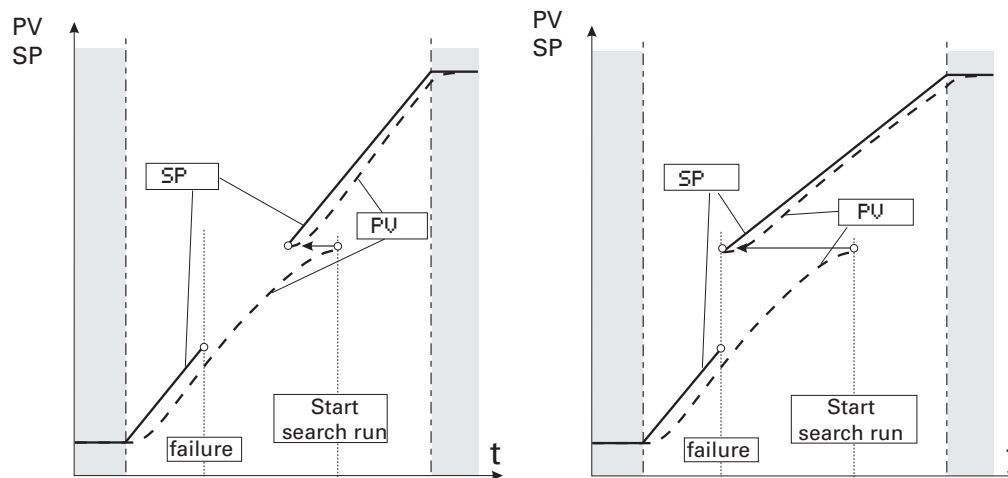


Abb. 551: Search run with the current gradient/the current remaining segment time

Configuring the programmer is done using the *TPrio* parameter in the *BlueDesign* parameter dialogue.

When making a search run, the following information must be taken into account:

- If setting "TPrio = Gradient" is used for the search run and the search value is out of the current segment, the program is continued at the segment point next to the search value.
- If the start value is equal to the end value (a segment without gradient) the program is continued at the segment start.
- With a jump segment, start is always at the beginning of the segment. The target set-point is assigned to the process value *PV*.

Search Run in the Program Section

A search run in the program section is limited to a section of several segments that have the same sign as the gradient, whereby hold segments are not considered as changes of sign. If the current program section contains hold segments, a search run is performed only, if this program section contains at least one more program segment which isn't a hold segment. If this segment is directly preceded or followed by another hold segment, the search run is realized only in the current segment.

Depending on the number of segments included in the search run, the run-time may be very long. For this reason, searching is divided into several partial runs. Each partial run checks only one segment. This is an automatic procedure which needs no interaction from operator.

**NOTE!**

If the value of parameter "TPrio" is "1: time", the search run is always limited to the current program segment.

Program segments with a final waiting state (e.g. "time + wait") do not limit the search range (exception: if a search run after a power failure is concerned).

A search run can lead to termination of the program.

Particularities of a search run with the PROGRAMMER

Dependent on master/slave coupling, a search run can have various effects:

| Master/slave coupling | Search run function | Search run of a single track | Simultaneous search run of all tracks |
|-----------------------|-----------------------|---|--|
| Time coupling | Search run in segment | Search run is permitted only for the master. As all slave tracks follow the master track time, no search run occurs with the slave tracks. | |
| | Search run in program | | |
| Segment coupling | Search run in segment | A search run is performed for the track. At the segment end, all tracks with segment coupling wait for each other before starting the next segment. (*) | |
| | | | A search run is performed for all tracks with segment coupling. At the segment end, they wait for each other before starting the next segment. (*) |
| | Search run in program | Permitted only for the master track. The master track performs a search run. All tracks with segment coupling follow the master into the new segment and perform a search run in this segment. If this segment search run isn't successful, the track continues at the segment start or end value, dependent on the setpoint (start or end) which is closer to its process value. If the process value (input PV) of a slave track with coupled segments is not soft-wired, the slave track continues at the value of the elapsed time of the master track in the segment. At the segment end, all tracks with segment coupling wait for each other before starting the next segment (*). | |

(*) With segment coupling of at least one analog track, the master track operates according to the same principle.

Dependent on parameter **SMode** (no search run / search run in the segment / search run in the program), the **search** input acts on all analog tracks simultaneously. Via the operating pages or the interface, a search run in the segment or in the program can be started independently of **SMode**. The search run for a single track can be started in the same way.

During a step or a dwell segment, the search run is without effect on this track.

If a slave track with segment coupling following the master with a search run in the program arrives at a dwell or step segment, the slave track continues at the value of the elapsed time of the master track in the segment and it waits there for the other tracks, before the next segment is started in common.

Preset

A preset is used to change to a defined time in the program.

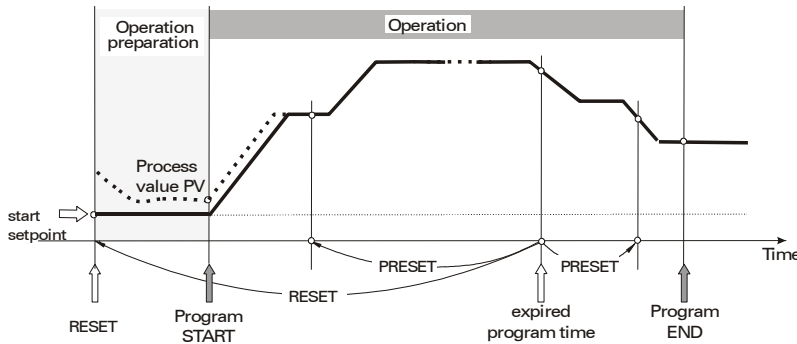


Abb. 552

For a detailed description how to call up the preset via operation, see operating page "Review profiles".

- A slave track with time coupling follows the master track.
- A slave track with segment coupling changes into the new segment of the master track and follows the time of the master track within this segment.

Bandwidth monitoring

To prevent the separation between setpoint and process value from becoming too large, bandwidth monitoring is used. This function checks, if the control deviation is within the permissible limits and stops the programmer ("Halt status"), as soon as the separation becomes too big. As soon as the control deviation has decreased sufficiently, the bandwidth monitoring terminates the **halt** status and the program is continued.

If the separation between setpoint **SP** and process value **PV** exceeds the bandwidth (**BW**) parameter, **halt** is activated:

$$HALT \leftarrow (|SP - PV| > bandwidth)$$

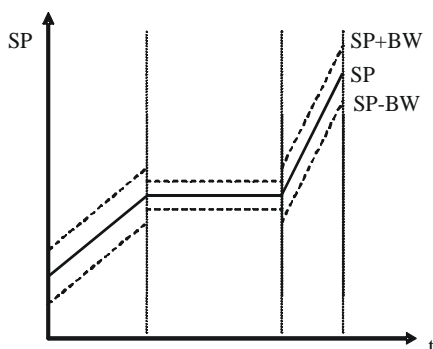


Abb. 553 Bandwidth around teh setpoint SP

- Each analog track has its own process value (PV) input and its own bandwidth.
- There is one bandwidth for each segment. Value **BW** determines the separation between process value **PV** and setpoint **SP** without triggering the **halt** status.
- If the bandwidth is exceeded, a track with time coupling stops, until **PV** is within the permissible bandwidth again. During this time, all tracks with time coupling including the master track are stopped.

A track with segment coupling shows the same bandwidth behaviour, but the other tracks continue up to their segment end. At the segment end, these tracks including the master track wait, until all tracks with coupled segments have reached the segment end.

While the bandwidth is exceeded, **halt** is signalled at the **halt** output and on the corresponding operating pages (main page and detailed page).

- Bandwidth monitoring can be switched off by setting the function to "OFF".

Control tracks (digital tracks)

Start values

The start values of the control tracks are included in the recipe: the values of the first segment (segment no. 0). The start values are active, when the **PROGRAMMER** is in the **reset** status. If no recipe is selected, or if the selected recipe is not available, e.g. because the recipe doesn't exist or isn't o.k., or because it doesn't suit the configuration, all control tracks are set to 0.

Segment time

The control tracks are coupled to the master track via the segment numbers. To reduce the division into segments, the two parameters **t+** and **t-** can be used:

- Delay **t-**: delay after which the control track reaches its adjusted value. Up to this time, the control track remains on the inverted value.
- Switch-on time **t+**: The control track remains on its adjusted value, until switch-on time **t+** has elapsed. If **t+** is set to "0", the switch-on time ends at the segment end.

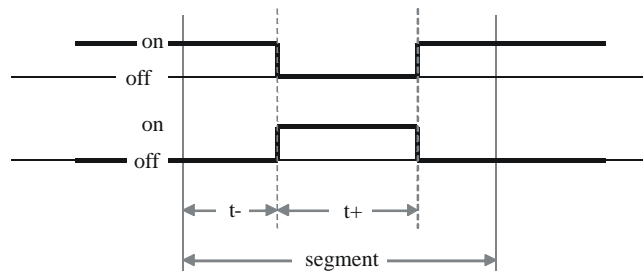


Abb. 554 Switch-on and off times within a segment

Recipes

The recipe management, i.e. creating, deleting and editing, is performed using the BlueEdit program editor. A program can be changed also using the PROGRAMMER editing page.

Fill in the basic data for the recipes in BlueDesign: Recipe directory (each PROGRAMMER must have an own directory), number of required analog tracks, number of required digital tracks.



CAUTION!

When using several PROGRAMMER function blocks, the recipes can be changed alternately.

As the PROGRAMMER function blocks start with the same recipe directory when edited in BlueDesign (default setting), they can access the same recipes.

For this reason:

- A unique recipe directory name must be used for each PROGRAMMER function block.

For creating the recipes, the engineering must be read in using BlueEdit. Subsequently, suitable recipes can be created and edited. These are uploaded into the instrument as required. Recipes in the instrument can be downloaded, edited, stored and reloaded.

Each PROGRAMMER uses its own directory specified in the **RecDir** configuration. The recipe file names have a defined structure of recipe number (001 ... 999) and recipe name: "RecNo_RecName.prf". During the start-up phase or after downloading the engineering, the PROGRAMMER generates its recipe directory, if it still doesn't exist.

**NOTE!**

A recipe directory that isn't required any more is not deleted automatically. It must be deleted using an external program (e.g. FTP tool), for instance the program editor.

Temporary recipe change

While PROGRAMMER is running, the user can change the active recipe temporarily, i.e. until the next **reset**.

**CAUTION!**

Changes of the active program made on the editing page are effective IMMEDIATELY.

This may cause unpredictable and dangerous situations in the process.

For this reason:

- Consider the effects before making changes in the current program and take measures to prevent potentially hazardous situations.

The temporary change is made on the PROGRAMMER editing page and changes will become active only during the future execution of the recipe (see editing page).

**NOTE!**

When quitting the editing page, a memory dialogue for permanent storage of recipe changes opens. If only a temporary change is required, "No" (= don't save changes permanently) must be selected.

Recipe change – program selection

The required recipe can be selected externally via the analog **RecNo** input, or internally using the recipe number adjusted via operation/interface.

**NOTE!**

During an active program sequence, switch-over to a different recipe on the programmer operating page is not possible. Recipe changing is possible only during the reset status.

Halt status

1. For example, used for external bandwidth monitoring

The **halt** status can be switched on and off using control input **halt**. Unlike the **stop** status, the **run** status remains unchanged during the **halt** status (the **run** output remains active). Status display is **halt**.

2. Internal bandwidth monitoring



The **halt** status can be triggered by internal bandwidth monitoring (see section Bandwidth monitoring). Status **halt** is displayed.


3. Synchronization of tracks with coupled segments

The **halt** status can be triggered when a track must wait for another track, because the tracks are coupled via segment coupling to the master (see section Synchronization). Status display is **halt**.

Programmer control via keys and

The programmer can be controlled via the following elements:

- the digital function block inputs,
- the interface, or
- using control keys  and .

The  key offers a choice of the following functions:

- **reset**: Selectable from any program position
- **preset**: From any program position. **preset** is used to select the review page (**preset**) to change to a different time in the program
- **segment search** : Start a search run in the segment.
- **program search** : Start a search run in the program.

The  key controls the programmer (fkey output generates a pulse when a key is activated).

For both keys, the rule saying that the states at soft-wired control inputs must be given priority over the operation. The following diagram describes the sequence of states dependent of the actions:

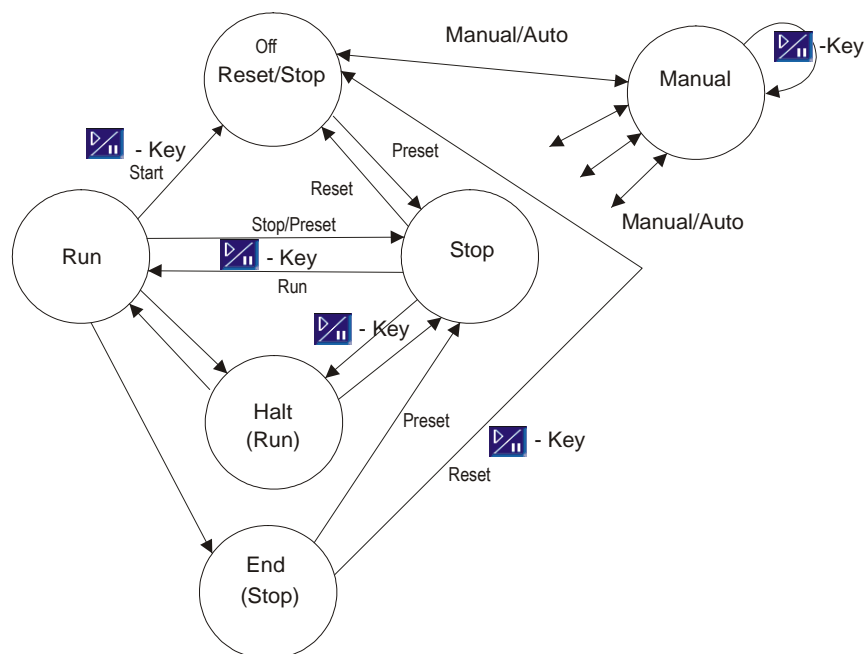




Fig. 555: Interaction between statuses of program and possible switch overs

Programmer control: General

- Recipes may be selected during the reset status.
- The process value is only shown on the detailed pages, if the process value input is soft-wired.
- The setpoint can be adjusted during manual operation (status "man").
- The 3 status displays are (depending on operating status):
 - Status left: man
 - Status middle: Estimated program end time
 - Status right: stop / run / reset / search / error / halt

NOTE!
 If the inputs (function block inputs) shown in the following table are used by the engineering, this signal cannot be changed on the operating page (front-panel operation).

This concerns inputs **run**, **reset**, **preset** and **search** (as shown in the following table):

| Input fields | Operation | Display | FB input |
|--|--|--|-------------------------------|
| Recipe name on main page | Selecting the required recipes is not possible, if the RecNo input is soft-wired. | Indicates the current recipe name. | RecNo |
|  (Selection key) | reset | Switching the programmer to segment 0. | reset |
| | preset | Select Preset via the review page | preset , Preset |
| | segment search | Starting a search run in the segment | search |
| program search | Starting a search run in the program | search | search |
|  key | Program control | Changes of the status display (bottom right) | run / reset |

Programmer operation

Operating structure

The **PROGRAMMER** has several operating pages selectable on a main page in the menu of operating pages. If the **hide** input is soft-wired, no **PROGRAMMER** operating page is displayed.

As shown in the following diagram, there is a main page for the programmer on which a branch to the detailed pages of the tracks can be made by tapping the relevant key. Changing to the parameter page is possible from the detailed pages of the tracks. The editing page is selectable only, when digital input **p_show** is soft-wired and set.

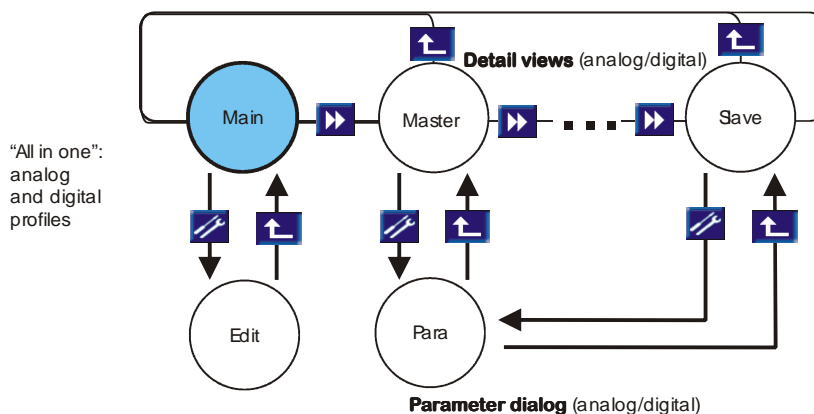


Abb. 556 Operating structure

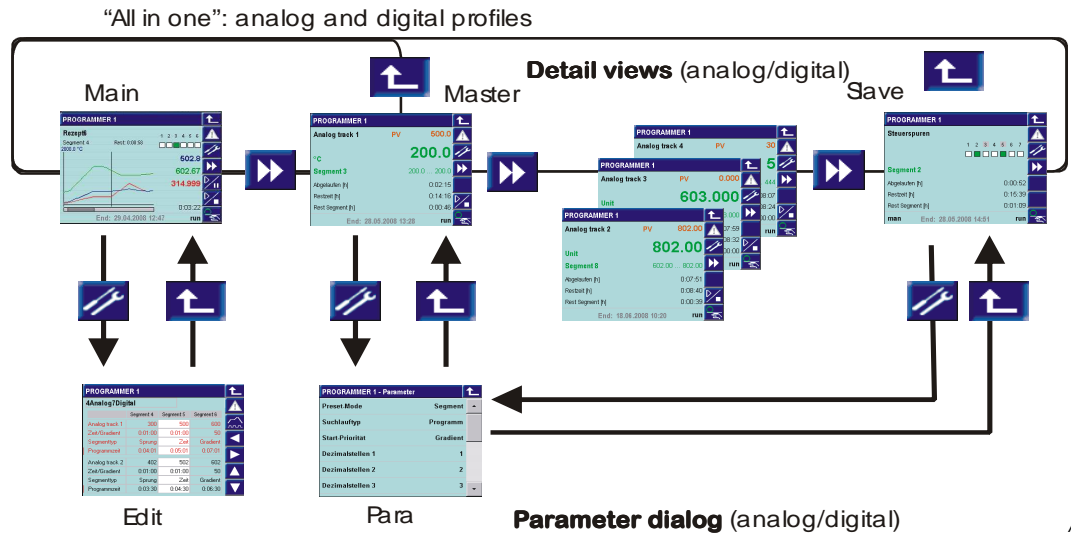
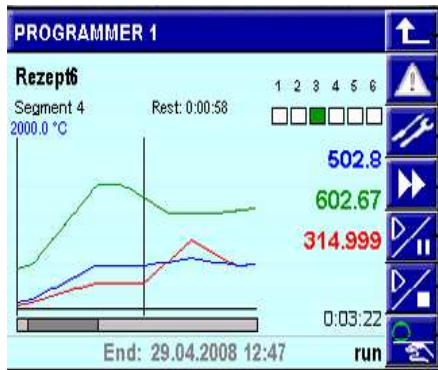


Abb.

Main operating page



- 1
- 2
- 3
- 4
- 5
- 6
- 7

Keys on the right section of the operating page:

- Key (line) 1: return
- Key 2: Access to the alarm page
- Key 3: Access to the parameter page
- Key 4: Change to the detail page
- Key 5: Programmer control
- Key 6: Programmer control selection
- Key 7: Auto / manual switch-over

Abb. 558

Display on left section of main page:

- 0: Programmer name
- 1: Active recipe**
- 2: Active segment
- 3: Remaining time of active segment
- 4: Current status of control tracks 1 to 6 (status)**
- 5 - 8: Current setpoint of analog track (up to 4)**
- 9: Elapsed program time (since start)
- 10: Maximum setpoint (SPhi1) and unit (Unit1) of master
- 11: Time stamp in the program curve
- 12: Setpoint curve in the program
- 13: Bargraph of program time (bar length to overall length as displayed time section to overall program time)
- 14: Display of manual operation "man"
- 15: Estimated program end time
- 16: Program status: run / stop / reset / search / halt / end / error

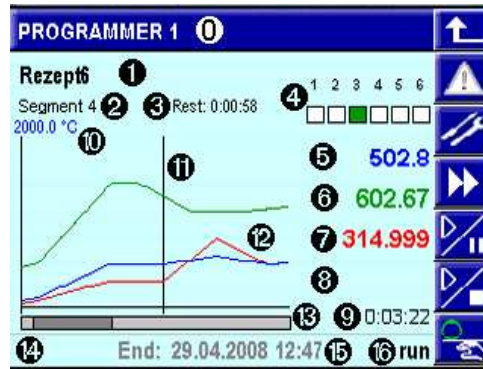


Abb. 559

The items marked in bold letters may include controls with variable values

Display on the main page

- Up to 4 setpoints with their curves are displayed; a vertical line marks the current time in the program. The visible time range is defined in configuration **TChart**. It is always updated, and the current position with the vertical time stamp remains always in the medium range.
- The Y-axis is determined by scaling of the master track with its maximum and minimum setpoint (**SPhi1**, **SPhi1**). Depending on configuration, the slave tracks are scaled like the master track, or provided with their own scaling. The scaling of individual slave tracks is not shown on the display.
- The setpoints belonging to the current time marked on the trend curve are displayed in the same colour as the setpoint curves on the right. The current program time is shown as a value below the current setpoints.
- The bar of the bargraph indicates the visible time range in relation to the overall program time.
- Items shown with a frame can be changed by tipping with your finger:
 - Recipe (only in status `reset`)
 - Setpoints (only during manual mode `man`)
 - Control tracks (only in manual mode `man`)
- An overview of the status of the first 6 control tracks is given by the "LEDs" (on / off, "LED" = Square at beginning of the line). Details and further control tracks are found on a list which can be called up by tipping on the LED field with your finger. All control tracks are listed with status and name.

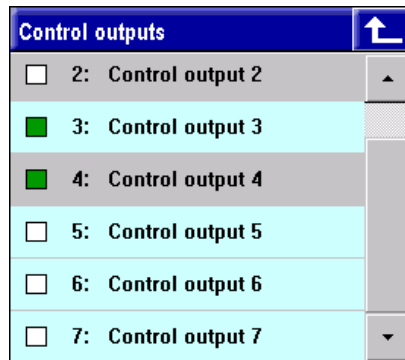



Abb. 560

Control track operation:

1. Symbol: white = off, green = on
2. Number of control track (1 ... max. 16)
3. Name of control track. The individual control tracks can be switched over: gray = manual mode, blue = automatic mode, if at least 1 control track is in manual mode.

- Tipping on a setpoint opens a window with display of track name, setpoint and unit.
- Switch-over to manual mode from the main page, marked by display "man" on the status line, puts all analog and digital tracks into manual mode. The variable items are marked in gray colour: setpoints are shown with a gray frame, control tracks are shown with a gray background.
 - Tipping on a setpoint opens a numeric pad for changing the setpoint.
 - Tipping on the LED field opens the list of control tracks. Change the status (off/on) of an individual control track by tipping on the corresponding icon. Tip on the name of the control track to switch it over between automatic and manual operation (gray background).

In status line "man" is displayed as long as at least 1 track is in manual operation. Tipping on  with "man" displayed all tracks are switched to automatic operation.

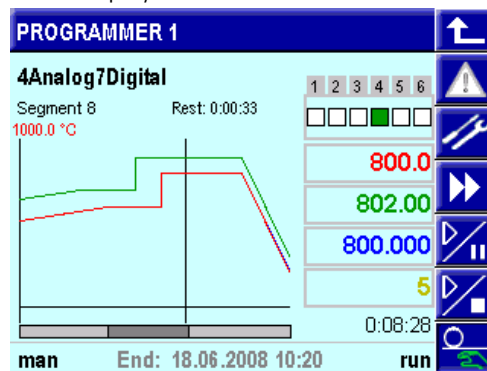






Abb. 561

Tapping  on the main page switches all tracks to manual or to automatic operation in common.

- The program can be set to a preset value. Preset to a segment or to a time in the program is possible. Preset can be activated via the profile overview page ( displays the list: "reset / preset / search", **preset** changes to the overview page). Tap the navigation keys to select the required time or segment. When leaving the page, the preset is or isn't activated ("OK" / "No"). If the preset is activated, the elapsed time is adapted and switch-over to the main operating page occurs (see operating page "Review profiles".)
- Program setting on the operating page: Direct access to the recipe parameter setting (= editing page) is enabled, when control input **p_show** = „1" is set at the programmer function block. Access is using the parameter key .
- The **run / stop / reset / preset** states are always valid for the overall PROGRAMMER, i.e. for all tracks. They can be switched over centrally on the main operating page, unless they are determined via control inputs.

Detailed pages of the analog tracks

When manual mode is activated on the operating page of a track (key ) , only this track is switched to the manual mode. All other tracks remain in the automatic mode.



NOTE!

On the status line of the main operating page, "man" is displayed, if at least 1 analog or digital track is in the manual mode.

The setpoint of a track can be adjusted only, if the track is in manual mode.

A search run can be started in the current segment. Additionally, a program search run can be started on the operating page of the master track (track 1).

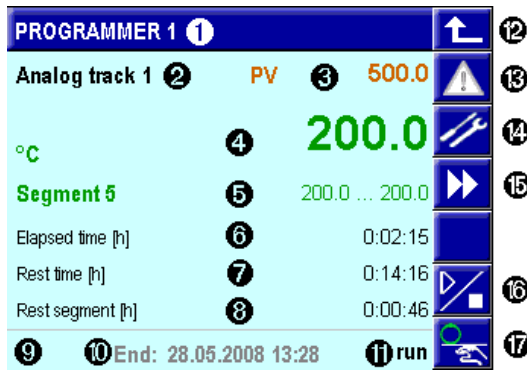
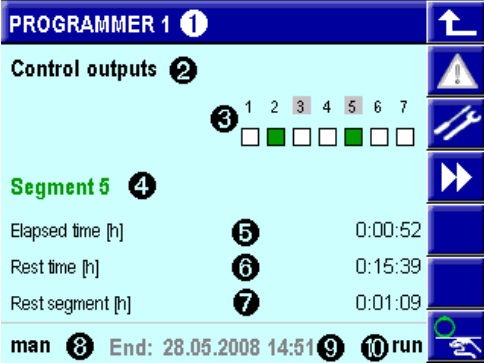


Abb. 562

- 1: PROGRAMMER name
- 2: Track name
- 3: Process value (if any) as a PV with value
- 4: Setpoint unit and **current setpoint**
- 5: Segment name/segment start and end value
- 6: Elapsed program time (master track)
- 7: Remaining program time (master track)
- 8: Remaining segment time (master track)
- 9: Display of manual mode: "man"
- 10: Estimated time for program end (master track)
- 11: Program status: stop, run, reset, search, error, halt (master track)
- 12: Return key
- 13: Key for opening the alarm page
- 14: Key for opening the parameter page
- 15: Track change key
- 16: Search run key (master track: selection dialogue)
- 17: automatic/manual switch-over key

Detailed pages of control tracks (digital tracks)



The screenshot shows a control interface for a programmer. At the top, it displays 'PROGRAMMER 1' (1) and a return key (11). Below this is the 'Control outputs' section (2), which includes a row of seven LEDs (3) numbered 1 to 7. LED 3 is green, while others are white. A gray background is visible behind LED 3, indicating manual mode. Below the LEDs is the 'Segment 5' (4) section, which shows 'Elapsed time [h]' (5) as 0:00:52, 'Rest time [h]' (6) as 0:15:39, and 'Rest segment [h]' (7) as 0:01:09. At the bottom, the status line (8) shows 'man' (8), 'End: 28.05.2008 14:51' (9), and 'run' (10). A key for opening the alarm page (12), a key for opening the parameter page (13), a key for track changing (14), and an auto/manual switch-over key (15) are also visible.




1: PROGRAMMER name
 2: Display: Control track operating page
 3: Display of control tracks:
 gray marking = control track is in manual mode
 green = control track on, white = control track off
 4: segment name
 5: Elapsed program time
 6: Remaining program time
 7: Remaining segment time
 8: Status line (only display) with
 9: Estimated time for program end
 10: Program status: stop, run, reset, search, error
 11: Return key
 12: Key for opening the alarm page
 13: Key for opening the parameter page
 14: Key for track changing
 15:  auto/manual switch-over key

Fig. 563

- The control tracks are displayed as LEDs.
- The name of the segment in which the master track works instead of the current segment number is displayed.
- The time estimated for program end is displayed on the status line.
- When tipping on the LED field, the control track list is opened. Each defined control track is displayed with number, name and a symbol for the status (off/on).
- Tipping on key , the **automatic/manual** key, also opens the list of control tracks. Tip on the name of the control track to switch it over between automatic and manual operation. On the list, the line is shown with a gray background during manual mode.
- When activating the manual mode on the operating page of control tracks () , only the control track selected on the list is switched to manual operation. All other tracks remain in the automatic mode.
- The status of a track can be changed only when it is in manual mode.
- Manual mode of the control tracks is shown by a gray field around the number of the control track. During manual operation, change the off/on status of the individual control track by tipping on the corresponding icon. Additionally, "man" is displayed on the status line, when at least 1 control track is in the manual mode.

Programmer parameters

The parameter page can be opened from the detailed pages of the tracks. On these pages, basic functions are listed, for example, the **search run type**, and can be changed. Some parameters are valid for the overall PROGRAMMER (e.g. **Preset-Mode**), others are valid for single tracks (e.g. setpoint ranges).

| PROGRAMMER 1 - Parameter | | ↑ |
|--------------------------|----------|---|
| Preset mode | Time | ▲ |
| Search mode | Segment | |
| Start priority | Gradient | |
| Decimals 1 | 1 | |
| Decimals 2 | 2 | |
| Decimals 3 | 3 | ▼ |

See table: Parameters

Abb. 564

Editing page

All recipe parameters (analog and digital tracks) are edited on a **single** scrollable page. When opening the page, the active recipe is displayed. Tap on the recipe name to switch over to other recipes.

The upper part always shows the master track in the colour configured with **Color1**. In the lower part, another track is displayed. For track paging, tap keys  and .

The following items can be changed:

- Recipe name (i.e. file selection)
- Setpoint/control track
- Time(s) or gradient
- Bandwidth



WARNING!

Changes on the editing page in the active recipe are effective immediately.

Changes in the program sequence may cause potentially hazardous conditions in the process.

For this reason:

- Consider the effects of the changes and take appropriate measures.

When leaving the editing page, a (memory) dialogue is opened. In this dialogue, the operator decides, if the changes are or aren't stored permanently. Changes in the current program sequence are taken into account in the further program sequence and deleted subsequently, unless they are saved.

Changes in another programm are not taken into account and are deleted, unless they are saved.

If the program is reset via the digital **reset** input or via the interface during editing on the editing page while the memory dialogue is still active, all changes are cancelled and the dialog is closed.

| | Segment 4 | Segment 5 | Segment 6 |
|----------------|-----------|-----------|-----------|
| Analog track 1 | 300 | 500 | 600 |
| Time/Gradient | 0:01:00 | 0:01:00 | 50 |
| Segment type | Step | Time | Rate |
| Program time | 0:04:01 | 0:05:01 | 0:07:01 |
| Analog track 2 | 402 | 502 | 602 |
| Time/Gradient | 0:01:00 | 0:01:00 | 50 |
| Segment type | Step | Time | Rate |
| Program time | 0:03:30 | 0:04:30 | 0:06:30 |

Abb. 565

For description of **segment types**:

see the Segment table.

Bandwidth:

If the separation between setpoint **SP** and process value **PV** exceeds the bandwidth parameter (**BW**), **halt** is activated (for description, the section Bandwidth monitoring).

- 1: PROGRAMMER name
- 2: Recipe (file name)
- 3 to 6: **Master track** (M=Master, always visible):
 - 3: Track name
 - 4: Depending on segment type: time/gradient
 - 5: Segment type (cannot be changed via operation)
 - 6: Change field, marked by a line. Switch-over by tipping:
 - a) Elapsed program time or
 - b) Bandwidth: Adjust the symmetric bandwidth per segment on the line.
- 7 to 10: one of the slave tracks (S=slave, 4 lines, paging with up/down keys), lines as 3 to 6:
 - 7: Track name
 - 8: depending on segment type: time/gradient
 - 9: Segment type (cannot be changed via operation)
 - 10: Program time/band width change field
- 11: Current segment (white background)
- 12: View and operation of 3 segments, scrolling by the up/down key
- 13: Return key
- 14: Key for opening the alarm page
- 15: Key for opening the overview page
- 16: Key for "Segment paging" right/left
- 17: Key for paging up/down slave tracks

Digital tracks on the editing page:

| | Segment 4 | Segment 5 | Segment 6 |
|------------------|-----------|-----------|-----------|
| Analog track 1 | 300 | 500 | 600 |
| Time/Gradient | 0:01:00 | 0:01:00 | 50 |
| Segment type | Step | Time | Rate |
| Bandwidth | OFF | 2 | 3 |
| Control output 1 | off | on | off |
| Delay t- | 0:00:00 | 0:00:00 | 0:00:00 |
| On time t+ | 0:00:00 | 0:00:00 | 0:00:00 |
| Program time | 0:04:01 | 0:05:01 | 0:07:01 |

Abb. 566

Same picture as above, but

- 1) **Bandwidth** on the master display
- 2) **Control track** on the slave display.



Switch-on delay t- : delay after which the control track goes to its adjusted value.

Switch-on time t+ : The control track remains on its adjusted value during switch-on time **t+** .

For a description of parameters, see section Segment time under Control tracks.

on = control track is on, **off** = control track is off.

Programmer page Review profiles

To provide a program overview, the "Review profiles" programmer page is opened. Tap  to select it on the editing page (tap  to open the editing page in the main menu). On this page, you can check the

program, for example, after creation or after changes. The page shows the curves of the analog tracks with setpoints in a graph. Tap the 4 arrow keys to scroll within the segments (◀▶) or per segment (◀▶) throughout the program time. Apart from the program time as a reference, the segments with segment names, the analog tracks with the setpoints and the first 6 digital tracks with states are displayed.

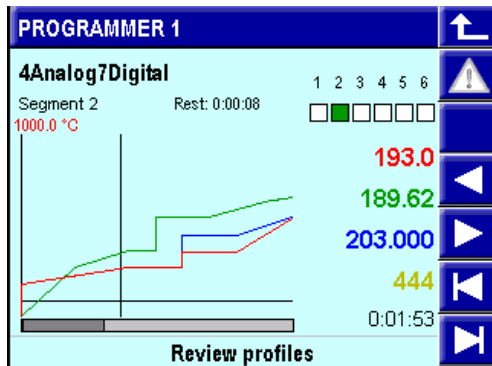





Abb. 567

The overview page is also used to make a preset for jumping to a defined program point. When tapping key  to select **preset** on the main page of the PROGRAMMER, the overview page is opened automatically. Tap on the arrow keys to navigate to the required position in the program and on the return key to leave the page. With the following prompt if a jump to the position is required, the preset can be activated or cancelled.

Note: paging with   is done by 1 pixel at a time (the time for each pixel is a function of the 80-pixel width for the configured visible time range).

III-15 Controller

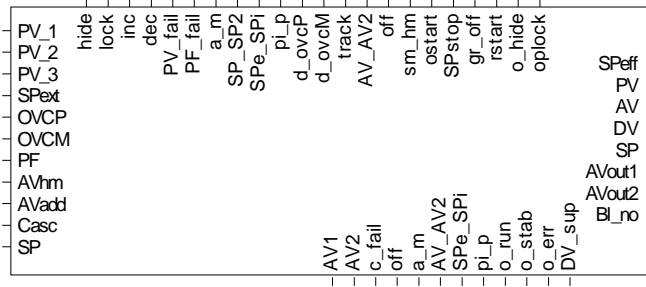
General:

Function blocks **CONTROL** and **CONTROLP** and **PIDMA** provide a complex control function. Unlike **CONTROL**, **CONTROLP** includes six selectable control parameter sets. However, **PIDMA** provides a special control algorithm and different self-tuning.

In the following sections, the main features of these functionblocks are described. Then the common controltechnical application ranges are explained.

III-15.1 CONTROL (Controlfunction with one parameter set (No. 90))

The CONTR block provides a PID controller with numerous functions such as ramp, internal/external/SP2 switchover, /process value tracking, self-tuning, override control, feed-forward control, positioning signal forcing, ratio and three-element control in 12 different controller types (continuous/2-point/3-point/3-point stepping/ ...).



CONTROL

Abb. 568

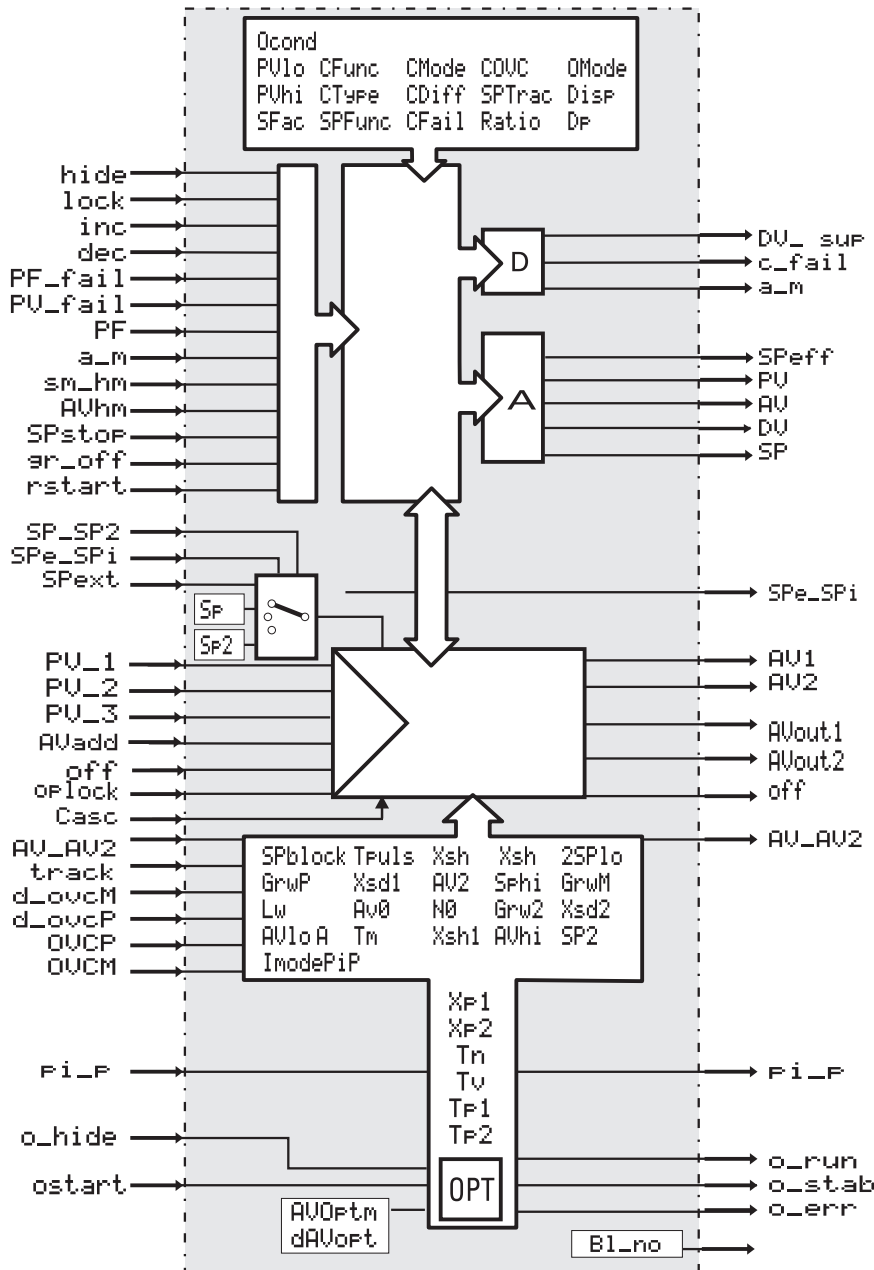


Abb. 569

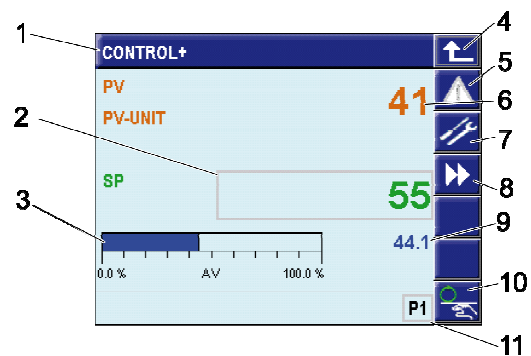
Operating page CONTROL

Function block **CONTROL** has an operating page selectable from the operating pages menu. With input **hide** set to "1" controller operating page is not displayed.

For further information on operation, see chapter 1: operation.

Overview

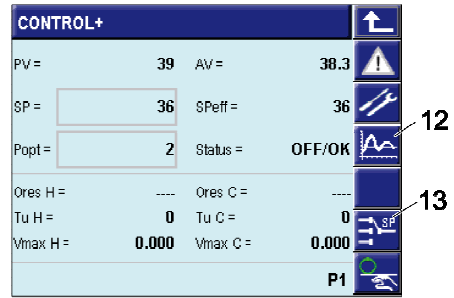
- 1 Title
- 2 Display and input field for the setpoint ("SP" = Setpoint)
- 3 Bargraph: Controller output ("AV" = "Actuating Variable"), deviation variable ("DV" = "Deviation Variable") or process value ("PV" = "Process Variable")
- 4 Button "Leave Operating Page"
- 5 Button "Alarm"
- 6 Display process value ("PV" = "Process Variable")
- 7 Button "Call parameter page"
- 8 Button "Call self-optimization"
- 9 Display controller output
- 10 Button "Switch over automatic mode/manual mode"
- 11 Controller parameter set selection (only for Control+)



Page Self-optimization

The "Self-optimization" page shows a new button compared with the main page of the controllers:

- 12 Button "Start/stop self-optimization"
- 13 Button "Setpoint switch over" (this button is also available on the main page, if switch over via parameter assignment is permitted).



| Field | Description |
|---------------|--|
| PV | "Process Variable"/process value |
| SP | "Setpoint"/ internal setpoint |
| Popt | Parameter set that will be optimized # |
| T | Optimization time* |
| AV | "Actuating Variable"/controller output |
| SPeff | ("Setpoint effective")/effective setpoint |
| Status | Current status of self-optimization Note: More detailed information on the status display is available in the function block reference. |
| Ores H | Optimization result - heating |
| Tu/ Vmax H | Process properties - heating |
| Ores C | Optimization result - cooling |
| Tu/ Vmax C | Process properties - cooling |
| Ores | Optimization result - heating/cooling* |
| Kp/Tn/Tv | Control parameters* |
| Orun | Display of "ORun", if the optimization is running, and display of "OErr" is the optimization is defective. |

* Only for controller PIDMA

Only for controller CONTROL+

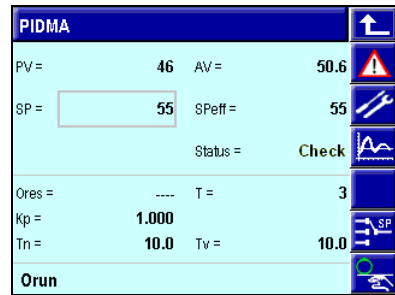


Fig. 570: Controllers "Control"/"Control+" and controller "PIDMA" (controller overview and self-optimization)

In-/Outputs

| Name | Type | Description |
|-------|-------|---|
| PV_1 | Float | Main variable PV1 |
| PV_2 | Float | Auxiliary variable PV2 e.g. for ratio control |
| PV_3 | Float | Auxiliary variable PV3 e.g. for 3-element control |
| SPext | Float | External set-point |
| OVCP | Float | Max. output limiting is to this value (not higher than AVhi). |
| OVCm | Float | Min. output limiting is to this value (not lower than AVlo). |
| PF | Float | Position feedback (without effect on the PID behaviour) |

| | | |
|-------------|-------------|---|
| AVhm | Float | Fixed positioning value [%] output after switch-over to hard manual |
| AVadd | Float | Feed-forward control. Is directly added to the controller output signal, thus avoiding the controller time behaviour. |
| Casc | Float | Cascade input for controller cascade (slave); the block number output of the master should be soft-wired to this input. |
| SP | Float | Change internal set-point |
| hide | Bool | Display suppression, 1 = page is not displayed during operation. |
| lock | Bool | Adjustment locking, 1 = the controls on the controller page are not active. |
| inc | Bool | Increment for manual adjustment |
| dec | Bool | Decrement for manual adjustment |
| PV_fail | Bool | Sensor error PV1 ... PV3 |
| PF_fail | Bool | Position feedback sensor error |
| a_m | Bool | Signal for taking the controller into manual operation. a_m = 0: controller in automatic mode, a_m = 1: controller in manual mode. |
| SP_SP2 | Bool | Switch-over to the second setpoint, SP/SP2 = 0: int./ext. setpoint, SP/SP2 = 1: SP2. |
| SPe_SPi | Bool | Switch-over from external to internal setpoint, SPe/SP = 0: external setpoint SPe, SPe/SP = 1: internal setpoint SP. |
| pi_p | Bool | PI/P switch-over, 0 = PI action; 1 = P action |
| d_ovcP | Bool | Blocking of output AV1 (open) with 3-point stepping controller. 1 = blocked. |
| d_ovcM | Bool | Blocking of output AV2 (closed) with 3-point stepping controller. 1 = blocked. |
| track | Bool | (Setpoint tracking), 0 = tracking function off; 1 = tracking function on |
| AV_AV2 | Bool | Switch-over to fixed output value AV2. 0 = controller output AV. 1 = fixed output value AV2 |
| off | Bool | Controller switch-off; 0 = controller switched on, 1 = controller switched off |
| sm_hm | Bool | (0 = soft manual), 1 = hard manual or safe positioning value AV_hm. The controller changes to manual mode directly. The value comes from the analog input AV_hm (not adjustable via operation). Transition to automatic mode is bumpless. |
| ostart | Bool | Self-tuning start; 0 = stop, 1 = start |
| SPstop | Bool | Effective setpoint freeze (can be used e.g. for bandwidth monitoring), SPstop = 1: frozen |
| gr_off | Bool | Set-point gradient suppression; 1 = gradient ineffective |
| rstart | Bool | 1 = start the set-point ramp, i.e. the set-point makes a step change towards the process value and goes to the adjusted setpoint according to the gradient. The rising flank (0 to 1) is evaluated. |
| o_hide | Bool | Self-tuning page display suppression, 1 = self-tuning page is not displayed in the operation. |
| oplock | Bool | Blockage of the auto/manual key; 1 = switch-over to manual by means of automatic/manual key is not possible. |
| Name | Type | Description |
| SPeff | Float | Effective set-point. Value after setpoint processing, after taking into account SP2, external setpoint, gradients, limitings. Control deviation and control response are a |

| | | |
|---------|-------|--|
| | | result of comparison with the effective process value. |
| PV | Float | Effective process value. Value at the end of process value processing, after taken into account calculations due to ratio and 3-element control. |
| AV | Float | Displayed output value (continuous) |
| DV | Float | Control deviation, i.e. PV - SP or process value - setpoint |
| SP | Float | Internal set-point |
| AVout1 | Float | Output value AVout1 (heating), continuous output |
| AVout2 | Float | Correcting variable AVout2 (cooling; only with continuous controller with split-range behaviour, i.e. Cfunc = splitRange) |
| Bl_no | Float | Own block number |
| AV1 | Bool | Status of switching output AV1; 0 = off, 1 = on |
| AV2 | Bool | Status of switching output AV2; 0 = off, 1 = on |
| c_fail | Bool | 1 = controller is in error handling |
| off | Bool | 0 = controller switched on; 1 = controller switched off |
| a_m | Bool | 0 = automatic; 1 = manual |
| AV_AV2 | Bool | 0 = controller output AV; 1 = fixed output value AV2 |
| SPe_SPi | Bool | Effective setpoint: 0 = external; 1 = internal set-point |
| pi_p | Bool | 0 = PI behaviour activated, 1 = P behaviour activated. |
| o_run | Bool | 1 = self-tuning running |
| o_stab | Bool | 1 = "process-at-rest" condition (for self-tuning) met |
| o_err | Bool | 1 = self-tuning error |
| DV_sup | Bool | Signal for alarm suppression after set-point change via stop input of ALARM function blocks, 1 = alarm is suppressed. |

| Parameter | | | | | | | |
|-----------|-----------|-----------|---|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| SPblock | SP switch | Enum | Blocks setpoint switch-over. In order to prevent unauthorized, accidental process interventions, switch-over can be blocked via front-panel operation. | r/w | 0 | | |
| | | Block All | Switch-over to another setpoint is not possible on the operating page. Switching over to internal, external or second setpoint is disabled for operation. | | 0 | | |
| | | Block SPe | Switching over between internal and external setpoint (in both directions) by operation is disabled, switching over to the second setpoint SP2 and | | 1 | | |

| | | | | | | | |
|----------|----------------|-----------|--|-----|-------|------------------|----|
| | | | vice versa is possible. | | | | |
| | | Block SP2 | Switching over to the second setpoint SP2 and vice versa is disabled, switching over between the internal and the external setpoint (both directions) by operation is possible. | | 2 | | |
| | | Enabled | No blockage is effective. Switching over to the external, internal or second setpoint via operation is possible. | | 3 | | |
| ImodePiP | PI/P behaviour | Enum | The behaviour of the integrator after PI/P switch-over can be adjusted: when switching over, the integral action is deleted, frozen or decays with integral action time T_n . | r/w | 0 | | |
| | | Off | Integrator switch-off. The correcting variable undergoes a step change of the amount of the integrator before switch-over. | | 0 | | |
| | | Freeze | Stop / freeze the integrator. No step change due to the integrator action, but the correcting variable can change with the P and D action. | | 1 | | |
| | | Pull down | Integrator decay to zero. The integrator action decays to zero with the integration time constant. | | 2 | | |
| SPlo | Min. setpoint | Float | Min. setpoint limit for Speff, specified in units of the process value. | r/w | 0.0 | | |
| SPhi | Max. setpoint | Float | Upper setpoint limit for Speff, specified in units of the process value | r/w | 100.0 | >0.0 | |
| SP2 | 2nd setpoint | Float | The second setpoint SP2 has priority over the other setpoints. Whether SP2 has safety functions, or whether it is only a pre-defined starting position in defined process conditions is determined only by the type of utilization and inclusion into an automation concept. | r/w | 100.0 | 0.0 ... 100.0 | |
| GrwP | SP gradient + | Float | Positive setpoint gradient [+K/min]. (Not valid for SP2). When this gradient is effective, an adjusted | r/w | off | >0.001 | ja |

| | | | | | | | |
|-------|-------------------|-------|--|-----|------|-------------------|----|
| | | | higher setpoint is effective by a ramp rather than by a step change. | | | | |
| GrwM | SP gradient - | Float | Negative setpoint gradient [-K/min]. (Not valid for SP2). When this gradient is effective, an adjusted lower setpoint is effective by a ramp rather than by a step change. | r/w | off | >0.001 | ja |
| Grw2 | SP2 gradient | Float | Gradient for the second setpoint SP2 [K/min]. With effective gradient, a setpoint change of SP2 or a switch-over to SP2 is effective by a ramp rather than by a step change. | r/w | off | >0.001 | ja |
| N0 | PV offset | Float | Zero offset (only ratio controller) | r/w | 0.0 | | |
| A | Factor | Float | Factor a with 3-element control and setpoint ramps | r/w | 1.0 | -9.99 ... 99.9 | |
| Xsh | Neutral zone | Float | Trigger point separation (stepping controller). Optimization of switching frequency (wear of actuator) and control sensitivity is recommended. | r/w | 0.2 | >0.2 | |
| Tpuls | Min.step time[s] | Float | Minimum positioning step time [s] (stepping controller). With 3-point stepping controllers: the minimum duty cycle. | r/w | 0.3 | 0.1 ... 2.0 | |
| Tm | Motor runtime[s] | Float | Actuator response time. Important for 3-point stepping controllers. | r/w | 30.0 | >5.0 | |
| Xsd1 | Switch difference | Float | With signaller, the trigger point is symmetrical to the setpoint; the switching difference can be adjusted as a hysteresis. | r/w | 1.0 | >0.1 | |
| LW | 2. Threshold | Float | Trigger point separation of additional contact for signaller with two outputs, separation from setpoint including sign. | r/w | off | | ja |
| Xsd2 | 2. Difference | Float | Switching difference for additional contact (signaller) | r/w | 1.0 | >0.1 | |
| Xsh1 | Neg.threshold PD | Float | Trigger point separation (three-point controller) for SP-PV < 0. Between the negative and the positive threshold, the control deviation is set to zero for computation of the controller reaction. | r/w | 0.0 | 0.0 ... 1000.0 | |
| Xsh2 | Pos.threshold PD | Float | Trigger point separation (PD) | r/w | 0.0 | 0.0 ... | |

| | | | | | | | |
|--------|--------------------|-------|---|-----|-------|---------------------|--|
| | | | (three-point controller) for SP-PV > 0. Between the negative and the positive threshold, the control deviation is set to zero for computation of the controller reaction. | | | 1000.0 | |
| AV2 | AV2 | Float | (Not with stepping controllers). Manual operation with the second correcting value activated. Caution: Don't confuse parameter AV2 with the second controller output. | r/w | 0.0 | -105.0 ... 105.0 | |
| AVlo | Min.actuator value | Float | Lower correcting variable limit [%] (not with stepping controllers). | r/w | 0.0 | 0.0 ... 100.0 | |
| AVhi | Max.actuator value | Float | Upper correcting variable limit [%] (not with stepping controllers). | r/w | 100.0 | 0.0 ... 105.0 | |
| AV0 | Working point | Float | Controller working point [%] (not with stepping controllers) | r/w | 0.0 | -105.0 ... 105.0 | |
| AVOptm | Start value(Tune) | Float | Start correcting value [%] for self-tuning in automatic mode, i.e. the correcting variable output at self-tuning start, in order to generate the 'process-at-rest' condition. | r/w | 0.0 | -105.0 ... 105.0 | |
| dAVopt | Step size(Tune) | Float | Self-tuning step height [%]. This is the value by which the correcting variable must change beginning from the start value, in order to cause the process reaction required for the controller. | r/w | 100.0 | 5.0 ... 100.0 | |
| Xp1 | P-band H | Float | Proportional band heating in % of (PVhi - PVlo) (control range). Determines the ratio of correcting variable and control deviation. The smaller the proportional band, the stronger is the control action at a defined control deviation. | r/w | 100.0 | 0.1 ... 999.9 | |
| Xp2 | P-band C | Float | Proportional band cooling in % of (PVhi - PVlo) (control range). Determines the ratio of correcting variable and control deviation. The smaller the proportional band, the stronger is the control action at a defined control deviation. | r/w | 100.0 | 0.1 ... 999.9 | |

| | | | | | | | |
|-----|--------------------|-------|--|-----|------|---------------|--|
| Tn | Integral time[s] | Float | Time constant [s] of the I action. The reaction of the I action is the faster the lower the adjusted integral time. With integral time = 0, the integral action is switched off for control and self-tuning. | r/w | 10.0 | >0.0 | |
| Tv | Derivative time[s] | Float | Time constant [s] of the D action. The reaction of the D action is the stronger the higher the adjusted derivative time. With derivative time = 0, the D action is switched off for control and self-tuning. | r/w | 10.0 | >0.0 | |
| Tp1 | Cycle time H | Float | Cycle time heating [s] (2 and 3-point controller). Too high: The controller has a tendency of oscillating. The cycle time corresponds to the minimum cycle time (time in seconds) at 50 % duty cycle. | r/w | 5.0 | 0.0 ... 999.9 | |
| Tp2 | Cycle time C | Float | Cycle time cooling [s] (3-point controller). Too high: The controller has a tendency of oscillating. The cycle time corresponds to the minimum cycle time (time in seconds) at 50 % duty cycle. | r/w | 5.0 | 0.0 ... 999.9 | |

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|-------|--------------|----------|---|--------|---------|-------|-----|
| CFunc | Control mode | Enum | Basic controller configuration: for various process requirements, e.g. heating, heating/cooling or three-point stepping, different control behaviours can be selected. | r/w | 9 | | |
| | | Signal 1 | Off controller or signaller with 1 output. The on/off controller or signaller switches over, when the process value leaves the band around the setpoint determined by the hysteresis. | | 0 | | |

| | | | | | | |
|--|--|-------------------|--|---|--|--|
| | | Signal 2 | The off controller or signaller with 2 outputs switches over, when the process value leaves the band around the setpoint determined by the hysteresis. The additional output can be used for additional power stages, or for an alarm function. | 1 | | |
| | | 2-point | PID controller, e.g. heating, with one switching output. The PID controller can respond to changes of the control deviation quickly and typically has no permanent control deviation. | 2 | | |
| | | 3-point | 2 x PID controller, e.g. heating/cooling. Two switching digital outputs. A PID controller can respond to changes of the control deviation quickly and typically has no permanent control deviation. | 3 | | |
| | | Cont. / switching | 2 x PID controller, e.g. heating/cooling. 1st output correcting (analog output, e.g. heating), 2nd output switching (digital output, e.g. cooling). A PID controller can respond to changes of the control deviation quickly and typically has no permanent control deviation. | 4 | | |
| | | Switching / cont. | 2 x PID controller, e.g. heating/cooling. 1st output switching (digital output, e.g. heating), 2nd output correcting (analog output, e.g. cooling). A PID controller can respond to changes of the control deviation quickly and typically has no permanent control deviation. | 5 | | |

| | | | | | | | |
|--|--|----------------|---|--|----|--|--|
| | | Delta-Star-OFF | D/ Y/Off, or 2-point controller with partial/full load switch-over. Two digital outputs: The Y1 is the switching output and the Y2 is the changeover contact for Star/Delta (D/Y). | | 6 | | |
| | | Stepping | 3-point stepping controller, e.g. for valves. 2 digital outputs. There are no positioning pulses in the lined out condition. Using the neutral zone, the switching frequency (actuator wear) and control sensitivity can be optimized. | | 7 | | |
| | | Stepp. + PF | 3-point stepping controller with position feedback, e.g. for valves. 2 digital outputs. There are no positioning pulses in the lined out condition. Neutral zone: for optimization of the switching frequency (actuator wear) and the control sensitivity, Position feedback PF serves for display of the actuator, but does not affect the PID action. | | 8 | | |
| | | Continuous | PID controller, e.g. heating, with one analog positioning output (continuous). The PID controller can respond to changes of the control deviation quickly and typically has no permanent control deviation. | | 9 | | |
| | | Split range | 2 x PID controller, e.g. heating/cooling. Two correcting analog outputs (continuous). A PID controller can respond to changes of the control deviation quickly and typically has no permanent control deviation. | | 10 | | |

| | | | | | | | |
|--------|------------------|-----------------|--|-----|----|--|--|
| | | Continuous + PF | Continuous controller with position feedback. PID controller, e.g. heating, with one analog positioning output. A PID controller can respond to changes of the control deviation quickly and typically has no permanent control deviation. The actually flowing positioning current can be displayed via input PF, without being included into the control action. | | 11 | | |
| CType | Controller type | Enum | Several input variables can form the process value. Ratio control e.g. for control of mixtures; 3-element control for processes in which load changes would be detected too late (disturbance variable). | r/w | 0 | | |
| | | Standard | The process variable measured via analog input PV_1 is used as a process value by the controller. | | 0 | | |
| | | Ratio | Process engineering frequently requires various components to be blended into one product. These ingredients should be mixed at a defined ratio. Process value PV applied to the controller is determined by the ratio of two input variables rather than being measured as a process variable. Suitable values for configuration of Ratio and constant NO must be adjusted. | | 1 | | |
| | | 3-element | With 3-element control, the process value is calculated according to the equation $PV_{eff} = PV_1 + A * (PV_2 / PV_3)$. | | 2 | | |
| SPfunc | SP intern/extern | Enum | Determine, if only the | r/w | 0 | | |

| | | | | | | | |
|-------|---------------------|-------------------|--|-----|---|--|--|
| | | | internal setpoint SP (setpoint) or also the external setpoint SPe (setpoint/cascade) should be used. | | | | |
| | | Set-point | With setpoint control, a fixed setpoint is determined by the internal setpoint SP. | | 0 | | |
| | | Set-point/Cascade | With setpoint/cascade control, switch-over from the external setpoint SPe to the internal setpoint SP is possible. Switch-over is via the front panel, via the digital SPe_SP input or via the interface. Unless this digital input is connected, or with a 0 signal applied to this input the external setpoint is taken over as an effective setpoint (SPeff). Unless both the digital input SPe_SPi and the analog input SPe are connected, the controller is fixed to the internal setpoint. | | 1 | | |
| CMode | Operating direction | Enum | Controller output action. An increasing process value is corrected by reducing (inverse, e.g. heating) or increasing the power (direct, e.g. cooling). | r/w | 0 | | |
| | | Inverse | Inverse reaction, e.g. heating. The correcting variable is increased with decreasing process value and reduced with increasing process value. | | 0 | | |
| | | Direct | Direct reaction, e.g. cooling. The correcting variable is increased with increasing process value and reduced with decreasing process value. | | 1 | | |
| CDiff | Differentiation | Enum | Rating of disturbance behaviour and control behaviour. PV differentiation: | r/w | 0 | | |

| | | | | | | | |
|-------|----------------|-------------|--|-----|---|--|--|
| | | | the controller disturbance behaviour is given a higher importance. DV differentiation: changes of the process value (disturbances) and of the setpoint are given the same importance. | | | | |
| | | Acts on DV | Differentiation of control deviation: Changes of the process value (disturbances) and of the setpoint are used dynamically to improve control. Thus disturbance behaviour and control behaviour are given the same importance. | | 0 | | |
| | | Acts on PV | Differentiation of the process value: Changes of the process value (disturbances) are used dynamically to improve control. Thus the disturbance behaviour is given higher importance. | | 1 | | |
| CFail | Sensor failure | Enum | The user determines the reaction to reach a safe process condition in case of sensor error. | r/w | 1 | | |
| | | Neutral | 3-point stepping: No output pulses. Otherwise: No output pulses or 0%. | | 0 | | |
| | | Minimum AV | 3-point stepping: The actuator is closed. Otherwise: Output of the minimum positioning value. | | 1 | | |
| | | Maximum AV | 3-point stepping: The actuator is opened. Otherwise: Output of the maximum positioning value | | 2 | | |
| | | AV2 | 3-point stepping: not selectable. Otherwise: fixed "second correcting variable" AV2 (also with manual mode, not adjustable). | | 3 | | |
| | | AV2 / AVman | 3-point stepping: not | | 4 | | |

| | | | | | | | |
|--------|-------------------|------------------|--|-----|---|--|--|
| | | | selectable. Otherwise: fixed "second correcting variable" AV2 adjustable during manual mode. | | | | |
| COVC | Override control | Enum | Override control: variable minimum or maximum output limiting for an analog input value. The transitions are bumpless. | r/w | 0 | | |
| | | Off | Switch-off: No limiting of the minimum (OVC-) or of the maximum (OVC+) positioning value to the value of an analog input. | | 0 | | |
| | | Maximum via OVCP | Limiting of the maximum positioning value to the value of an analog input (OVC_P). Transitions are bumpless. | | 1 | | |
| | | Minimum via OVCM | Limiting of the minimum positioning value to the value of an analog input (OVC_M). Transitions are bumpless. | | 2 | | |
| | | Max/Min via OVC | Limiting of the minimum (OVC-) or of the maximum (OVC+) positioning value to the value of the related analog input. The transitions are bumpless. | | 3 | | |
| SPtrac | Setpoint tracking | Enum | Setpoint tracking: To prevent step changes when switching over from the external to the internal setpoint. The SPe used so far is taken over as an internal setpoint SP. | r/w | 0 | | |
| | | Off | No tracking. (The standard rules are applicable for switch-over from external to internal setpoint.) | | 0 | | |
| | | SP follows SPe | With setpoint tracking, the SPe used so far is taken over as an internal setpoint SP when switching over from external to internal setpoint (SPe to SP). When | | 1 | | |

| | | | | | | | |
|-------|---------------|---------------|--|-----|---|--|--|
| | | | returning (SP to SPe), approach to the SPe is using the adjusted gradients GrwP/GrwM. Used to prevent step changes of the setpoint when switching over. | | | | |
| | | SP follows PV | With process value tracking, the process value PV is taken over as internal setpoint SP when switching over from external to internal setpoint (SPe to SP). When returning (SP to SPe), approach to the external setpoint SPe is using the adjusted gradients GrwP/GrwM. Used to prevent step changes e.g. when starting up, when the setpoint is far away from the instantaneous process value. | | 2 | | |
| Ratio | Ratio control | Enum | Process engineering: various ingredients must be blended into a product at a pre-defined ratio. A formula for calculation of the process value as an input ratio must be selected. | r/w | 0 | | |
| | | Type 1 | Ratio control in which the ratio of process value PV1 and process value PV2 is controlled according to the formula $(PV1 + N0) / PV2$. | | 0 | | |
| | | Type 2 | Ratio control in which the ratio of process value PV1 and sum of the two process values PV1 and PV2 is controlled according to the formula $(PV1 + N0) / (PV1 + PV2)$. | | 1 | | |
| | | Type 3 | Ratio control in which the ratio of the difference between process values PV1 and PV2 and the | | 2 | | |

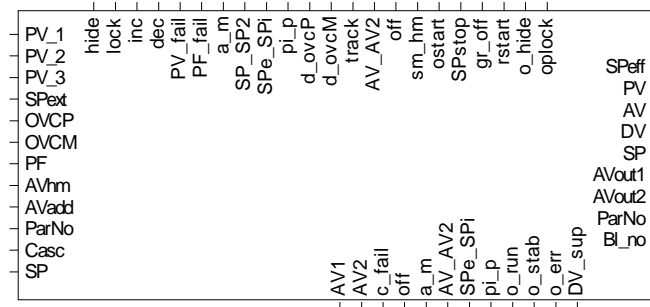
| | | | | | | | |
|-------|------------------|----------------------------------|---|-----|---|---------|--|
| | | | second process value PV2 is controlled according to the formula (PV2 - PV1 + NO) / PV2. | | | | |
| Dp | Decimals | Int | Digits behind the decimal point of the displayed process value. Also valid for display of the setpoint. | r/w | 0 | 0 ... 3 | |
| Disp | Bargraph content | Enum | Correcting variable, control deviation or process value are displayed with minimum, maximum and current value on the controller operating page. With cascade control, the bargraph has special properties. | r/w | 0 | | |
| | | Control output AV | Display of correcting variable AV (cascade slave operating page: always display of slave) in the bargraph with value and limits | | 0 | | |
| | | Deviation DV | Display of control deviation (process value - setpoint) in the bargraph with value and limits. Cascade slave operating page: Display of control deviation DV of the master with closed cascade; with open cascade ("Slave" is displayed), the slave data are displayed. | | 1 | | |
| | | Process value PVe _{eff} | Display of the effective process value PV in the bargraph with value and limits. Cascade slave operating page: Display of the effective process value PV of the master with closed cascade; with open cascade ("Slave" is displayed), the slave data are displayed. | | 2 | | |
| OMode | Tuning mode | Enum | Only standard | r/w | 0 | | |
| | | Standard | Cannot be changed. | | 0 | | |

| | | | | | | | |
|---------|--------------------|---------------------|--|-----|---------|---------------|--|
| OCond | Process in rest | Enum | Determine the "process-at-rest" condition: only with constant process value PV, or also with defined even process value changes. | r/w | 0 | | |
| | | PV = constant | "Process at rest" is detected, when process value PV is constant, i.e. the process value must be within a band $\leq 1\%$ of control range (PVhi - PVlo). | | 0 | | |
| | | PV-drift < 0 or > 0 | "Process at rest" is detected, when process value PV decreases constantly on a controller with inverse output action. Process at rest is detected, when process value PV increases constantly on a controller with direct output action. | | 1 | | |
| | | PV-drift = constant | "Process at rest" is detected, when process value PV changes constantly. In this case continuation of the constant change over the duration of identification must be ensured. | | 2 | | |
| PVlo | Control span start | Float | Lower end of control range, important for proportional band (Xp) and self-tuning | r/w | 0.0 | | |
| PVhi | Control span end | Float | Upper end of control range, important for proportional band (Xp) and self-tuning | r/w | 100.0 | >0.0 | |
| SFac | Ratio factor | Float | Factor for stoichiometric ratio, is required for process value calculation by a ratio controller | r/w | 1.0 | 0.01 ... 99.9 | |
| Unit_PV | Unit of PV | Text | Unit of process value (display only). For cascade control, special characteristics are applicable. | r/w | PV-UNIT | | |

III-15.2 CONTROLP (Controlfunction with six parametersets (No. 91))

Function block CONTROLP provides the same functionality as the CONTROL block. Additionally, it permits adaptation by adjustment.

Six parameter sets can be activated dependent on process criteria (process value, , correcting variable, control deviation), plant or batch characteristics. The parameter sets can be determined independently by self-tuning.



CONTROL+

Abb. 571

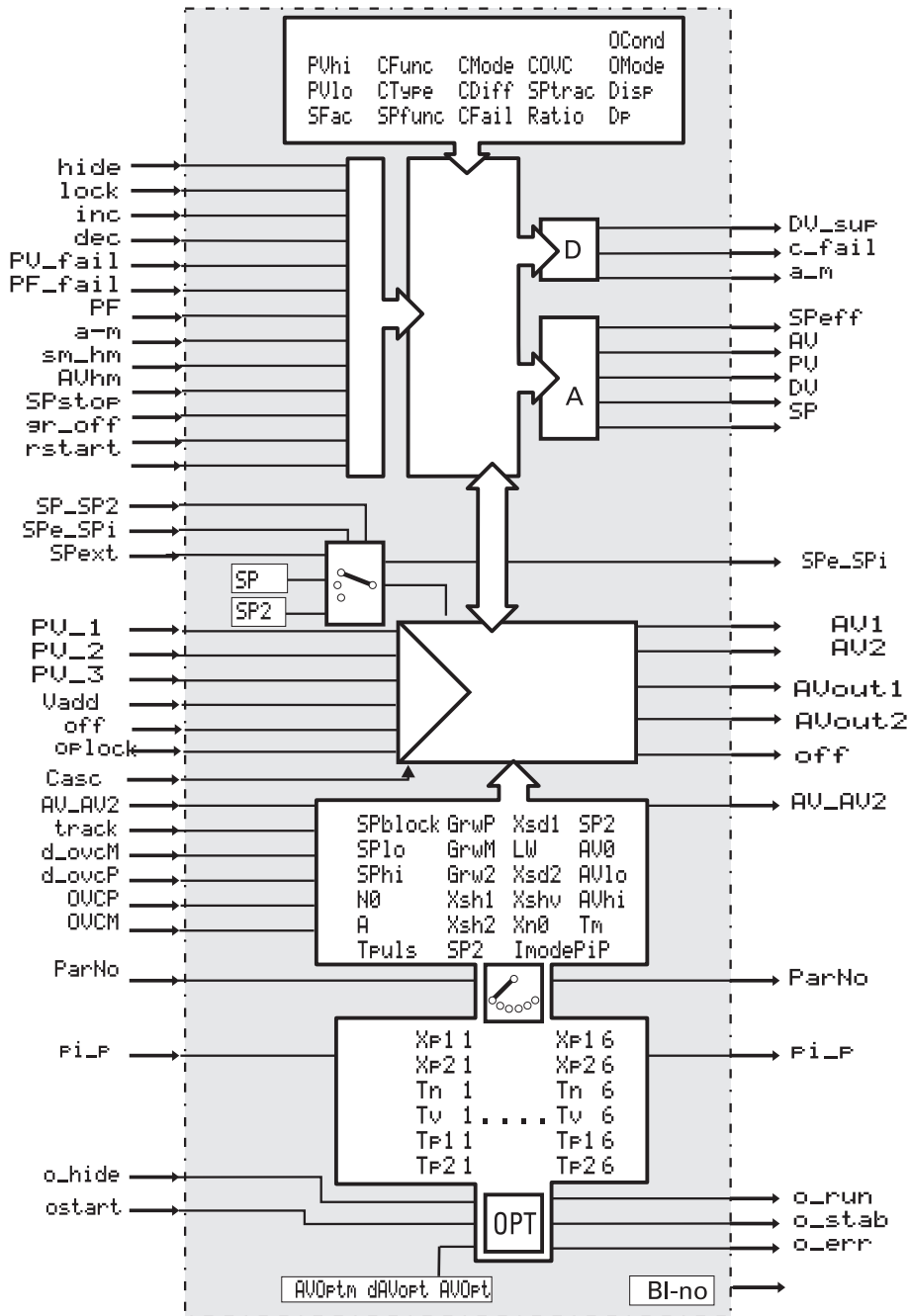


Abb. 572

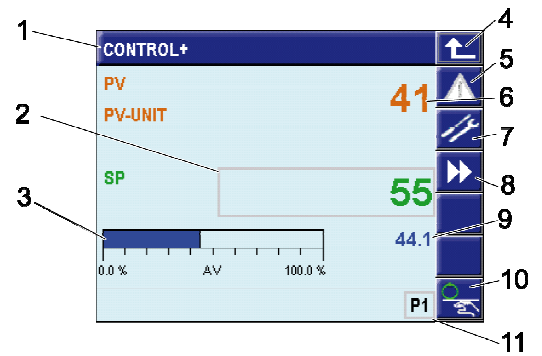
Operating page CONTROLP

Function block **CONTROLP** has an operating page selectable from the operating pages menu. With input **hide** set to "1" controller operating page is not displayed.

For further information on operation, see chapter 1: operation.

Overview

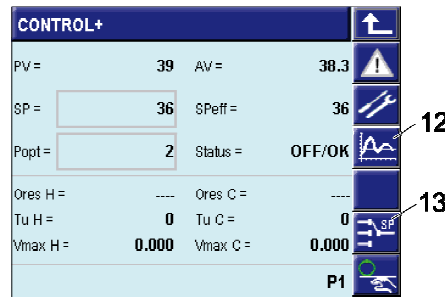
- 1 Title
- 2 Display and input field for the setpoint ("SP" = Setpoint)
- 3 Bargraph: Controller output ("AV" = "Actuating Variable"), deviation variable ("DV" = "Deviation Variable") or process value ("PV" = "Process Variable")
- 4 Button "Leave Operating Page"
- 5 Button "Alarm"
- 6 Display process value ("PV" = "Process Variable")
- 7 Button "Call parameter page"
- 8 Button "Call self-optimization"
- 9 Display controller output
- 10 Button "Switch over automatic mode/manual mode"
- 11 Controller parameter set selection (only for Control+)



Page Self-optimization

The "Self-optimization" page shows a new button compared with the main page of the controllers:

- 12 Button "Start/stop self-optimization"
- 13 Button "Setpoint switch over" (this button is also available on the main page, if switch over via parameter assignment is permitted).



| Field | Description |
|---------------|--|
| PV | "Process Variable"/process value |
| SP | "Setpoint"/ internal setpoint |
| Popt | Parameter set that will be optimized # |
| T | Optimization time* |
| AV | "Actuating Variable"/controller output |
| SPeff | ("Setpoint effective")/effective setpoint |
| Status | Current status of self-optimization Note: More detailed information on the status display is available in the function block reference. |
| Ores H | Optimization result - heating |
| Tu/ Vmax H | Process properties - heating |
| Ores C | Optimization result - cooling |
| Tu/ Vmax C | Process properties - cooling |
| Ores | Optimization result - heating/cooling* |
| Kp/Tn/Tv | Control parameters* |
| Orun | Display of "ORun", if the optimization is running, and display of "OErr" is the optimization is defective. |

* Only for controller PIDMA

Only for controller CONTROL+

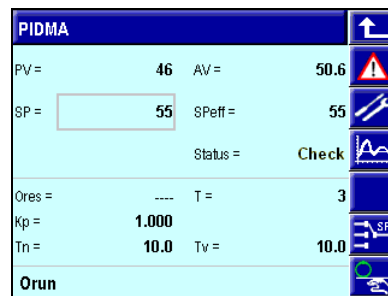


Fig. 573: Controllers "Control"/"Control+" and controller "PIDMA" (controller overview and self-optimization)

In-/Outputs

| Name | Type | Description |
|-------|-------|---|
| PV_1 | Float | Main variable PV1 |
| PV_2 | Float | Auxiliary variable PV2 e.g. for ratio control |
| PV_3 | Float | Auxiliary variable PV3 e.g. for 3-element control |
| SPext | Float | External set-point |
| OVCP | Float | Max. output limiting is to this value (not higher than AVhi). |
| OVCm | Float | Min. output limiting is to this value (not lower than than AVlo). |
| PF | Float | Position feedback (without effect on the PID behaviour) |

| | | |
|---------|-------|---|
| AVhm | Float | Fixed positioning value [%] output after switch-over to hard manual |
| AVadd | Float | Feed-forward control. Is directly added to the controller output signal, thus avoiding the controller time behaviour. |
| ParNo | Float | Effective parameter set |
| Casc | Float | Cascade input for controller cascade (slave); the block number output of the master should be soft-wired to this input. |
| SP | Float | Change internal set-point |
| hide | Bool | Display suppression, 1 = page is not displayed during operation. |
| lock | Bool | Adjustment locking, 1 = the controls on the controller page are not active. |
| inc | Bool | Increment for manual adjustment |
| dec | Bool | Decrement for manual adjustment |
| PV_fail | Bool | Sensor error PV1 ... PV3 |
| PF_fail | Bool | Position feedback sensor error |
| a_m | Bool | Signal for taking the controller into manual operation. a_m = 0: controller in automatic mode, a_m = 1: controller in manual mode. |
| SP_SP2 | Bool | Switch-over to the second setpoint, SP/SP2 = 0: int./ext. setpoint, SP/SP2 = 1: SP2. |
| SPe_SPi | Bool | Switch-over from external to internal setpoint, SPe/SP = 0: external setpoint SPe, SPe/SP = 1: internal setpoint SP. |
| pi_p | Bool | PI/P switch-over, 0 = PI action; 1 = P action |
| d_ovcP | Bool | Blocking of output AV1 (open) with 3-point stepping controller. 1 = blocked. |
| d_ovcM | Bool | Blocking of output AV2 (closed) with 3-point stepping controller. 1 = blocked. |
| track | Bool | (Setpoint tracking), 0 = tracking function off; 1 = tracking function on |
| AV_AV2 | Bool | Switch-over to fixed output value AV2. 0 = controller output AV. 1 = fixed output value AV2 |
| off | Bool | Controller switch-off; 0 = controller switched on, 1 = controller switched off |
| sm_hm | Bool | (0 = soft manual), 1 = hard manual or safe positioning value AV_hm. The controller changes to manual mode directly. The value comes from the analog input AV_hm (not adjustable via operation). Transition to automatic mode is bumpless. |
| ostart | Bool | Self-tuning start; 0 = stop, 1 = start |
| SPstop | Bool | Effective setpoint freeze (can be used e.g. for bandwidth monitoring), SPstop = 1: frozen |
| gr_off | Bool | Set-point gradient suppression; 1 = gradient ineffective |
| rstart | Bool | 1 = start the set-point ramp, i.e. the set-point makes a step change towards the process value and goes to the adjusted setpoint according to the gradient. The rising flank (0 to 1) is evaluated. |
| o_hide | Bool | Self-tuning page display suppression, 1 = self-tuning page is not displayed in the operation. |
| oplock | Bool | Blockage of the auto/manual key; 1 = switch-over to manual by means of automatic/manual key is not possible. |

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| | | |
|---------|-------|---|
| SPeff | Float | Effective set-point. Value after setpoint processing, after taking into account SP2, external setpoint, gradients, limitings. Control deviation and control response are a result of comparison with the effective process value. |
| PV | Float | Effective process value. Value at the end of process value processing, after taken into account calculations due to ratio and 3-element control. |
| AV | Float | Displayed output value (continuous) |
| DV | Float | Control deviation, i.e. PV - SP or process value - setpoint |
| SP | Float | Internal set-point |
| AVout1 | Float | Output value AVout1 (heating), continuous output |
| AVout2 | Float | Correcting variable AVout2 (cooling; only with continuous controller with split-range behaviour, i.e. Cfunc = splitRange) |
| ParNo | Float | Effective parameter set |
| Bl_no | Float | Own block number |
| AV1 | Bool | Status of switching output AV1; 0 = off, 1 = on |
| AV2 | Bool | Status of switching output AV2; 0 = off, 1 = on |
| c_fail | Bool | 1 = controller is in error handling |
| off | Bool | 0 = controller switched on; 1 = controller switched off |
| a_m | Bool | 0 = automatic; 1 = manual |
| AV_AV2 | Bool | 0 = controller output AV; 1 = fixed output value AV2 |
| SPe_SPi | Bool | Effective setpoint: 0 = external; 1 = internal set-point |
| pi_p | Bool | 0 = PI behaviour activated, 1 = P behaviour activated. |
| o_run | Bool | 1 = self-tuning running |
| o_stab | Bool | 1 = "process-at-rest" condition (for self-tuning) met |
| o_err | Bool | 1 = self-tuning error |
| DV_sup | Bool | Signal for alarm suppression after set-point change via stop input of ALARM function blocks, 1 = alarm is suppressed. |

| Paramete | | | | | | | |
|----------|-----------------|-----------|--|--------|---------|---------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| POpt | Param.set(Tune) | Int | Selection of the parameter set which must be determined by self-tuning | r/w | 1 | 1 ... 6 | |
| SPblock | SP switch | Enum | Blocks setpoint switch-over. To prevent accidental process interventions, switch-over via the front-panel operation can be disabled. | r/w | 0 | | |
| | | Block All | Switch-over to another setpoint is not possible on the operating page. Switching over to internal, external or second setpoint is | | 0 | | |

| | | | | | | | |
|----------|----------------|-----------|---|-----|-------|------------------|--|
| | | | disabled for operation. | | | | |
| | | Block SPe | Switching over between internal and external setpoint (in both directions) by operation is disabled, switching over to the second setpoint SP2 and vice versa is possible. | | 1 | | |
| | | Block SP2 | Switching over to the second setpoint SP2 and vice versa is disabled, switching over between the internal and the external setpoint (both directions) by operation is possible. | | 2 | | |
| | | Enabled | No blockage is effective. Switching over to the external, internal or second setpoint via operation is possible. | | 3 | | |
| lmodePiP | pi_p behaviour | Enum | The behaviour of the integrator after PI/P switch-over can be adjusted: when switching over, the integral action is deleted, frozen or decays with integral action time Tn. | r/w | 0 | | |
| | | Off | Integrator switch-off. The correcting variable undergoes a step change by the amount of the integrator before switch-over. | | 0 | | |
| | | Freeze | Stop / freeze the integrator. No step change due to the integrator action, but the correcting variable can change with the P and D action. | | 1 | | |
| | | Pull down | Integrator decay to zero. The integrator action decays to zero with the integration time constant. | | 2 | | |
| SPlo | Min. setpoint | Float | Lower setpoint limit for Speff, specified in units of the process value | r/w | 0.0 | | |
| SPhi | Max. setpoint | Float | Upper setpoint limit for Speff, specified in units of the process value | r/w | 100.0 | >0.0 | |
| SP2 | 2nd setpoint | Float | The second setpoint SP2 has priority over the other setpoints. Whether SP2 has safety functions, or whether it is only a pre-defined | r/w | 100.0 | 0.0 ... 100.0 | |

| | | | | | | | |
|-------|-------------------|-------|---|-----|------|-------------------|----|
| | | | start position in defined process conditions is determined only by the type of utilization and inclusion in an automation concept. | | | | |
| GrwP | SP gradient + | Float | Positive setpoint gradient [+K/min]. (Not valid for SP2). If this gradient is effective, an adjusted higher setpoint is effective by a ramp rather than by a step change. | r/w | off | >0.001 | ja |
| GrwM | SP gradient - | Float | Negative setpoint gradient [-K/min]. (Not valid for SP2). If this gradient is effective, an adjusted lower setpoint is effective by a ramp rather than by a step change. | r/w | off | >0.001 | ja |
| Grw2 | SP2 gradient | Float | Setpoint gradient for the second setpoint SP2 [K/min]. With effective gradient, an SP2 setpoint change or a switch-over to SP2 is effective by a ramp rather than by a step change. | r/w | off | >0.001 | ja |
| N0 | PV offset | Float | Zero offset (ratio controller only) | r/w | 0.0 | | |
| A | Factor | Float | Factor a with 3-element control and setpoint ramps | r/w | 1.0 | -9.99 ... 99.9 | |
| Xsh | Neutral zone | Float | Trigger point separation (stepping controller). Optimization of switching frequency (wear of actuator) and control sensitivity is recommended. | r/w | 0.2 | >0.2 | |
| Tpuls | Min.step time[s] | Float | Minimum positioning step time [s] (stepping controller). With 3-point stepping controllers: the minimum duty cycle. | r/w | 0.3 | 0.1 ... 2.0 | |
| Tm | Motor runtime[s] | Float | Actuator response time. Important for 3-point stepping controllers. | r/w | 30.0 | >5.0 | |
| Xsd1 | Switch difference | Float | With signaller, the trigger point is symmetrical to the setpoint; the switching difference can be adjusted as a hysteresis. | r/w | 1.0 | >0.1 | |
| LW | 2. Threshold | Float | Trigger point separation of additional contact for signaller with two outputs, separation | r/w | off | | ja |

| | | | | | | |
|--------------------|--------------------|-------|---|-----|-------|---------------------|
| | | | from setpoint including sign. | | | |
| Xsd2 | 2. Difference | Float | Switching difference for additional contact (signaller) | r/w | 1.0 | >0.1 |
| Xsh1 | Neg.threshold PD | Float | Trigger point separation (three-point controller) for SP-PV < 0. Between the negative and the positive threshold, the control deviation is set to zero for computation of the controller reaction. | r/w | 0.0 | 0.0 ... 1000.0 |
| Xsh2 | Pos.threshold PD | Float | Trigger point separation (PD) (three-point controller) for SP-PV > 0. Between the negative and the positive threshold, the control deviation is set to zero for computation of the controller reaction. | r/w | 0.0 | 0.0 ... 1000.0 |
| AV2 | AV2 | Float | (Not with stepping controllers). Manual operation with the second correcting value activated. Caution: Don't confuse parameter AV2 with the second controller output. | r/w | 0.0 | -105.0 ... 105.0 |
| AVlo | Min.actuator value | Float | Lower correcting variable limit [%] (not with stepping controllers). | r/w | 0.0 | 0.0 ... 100.0 |
| AVhi | Max.actuator value | Float | Upper correcting variable limit [%] (not with stepping controllers). | r/w | 100.0 | 0.0 ... 105.0 |
| AV0 | Working point | Float | Controller working point [%] (not with stepping controllers) | r/w | 0.0 | -105.0 ... 105.0 |
| AVOptm | Start value(Tune) | Float | Start correcting value [%] for self-tuning in automatic mode, i.e. the correcting variable output at self-tuning start, in order to generate the 'process-at-rest' condition. | r/w | 0.0 | -105.0 ... 105.0 |
| dAVopt | Step size(Tune) | Float | Self-tuning step height [%]. The value by which the correcting variable must change beginning from the start value, in order to cause the process reaction required for the controller. | r/w | 100.0 | 5.0 ... 100.0 |
| Xp1_1 ... Xp1_6 | P-band 1 H | Float | Proportional band heating in % of (PVhi - PVlo) (control range). Determines the ratio of correcting variable and control | r/w | 100.0 | 0.1 ... 999.9 |

| | | | | | | | |
|--------------------|---------------------|-------|---|-----|-------|------------------|--|
| | | | deviation. The smaller the proportional band, the stronger is the control action at a defined control deviation. | | | | |
| Xp2_1 ... Xp2_6 | P-band 1 K | Float | Proportional band cooling in % of (PVhi - PVlo) (control range). Determines the ratio of correcting variable and control deviation. The smaller the proportional band, the stronger is the control action at a defined control deviation. | r/w | 100.0 | 0.1 ... 999.9 | |
| Tn_1 ... Tn_6 | Integral time[s] 1 | Float | Time constant [s] of the I action. The reaction of the I action is the faster the lower the adjusted integral time. With integral time = 0, the integral action is switched off for control and self-tuning. | r/w | 10.0 | >0.0 | |
| Tv_1 ... Tv_6 | Derivative time[s]1 | Float | Time constant [s] of the D action. The reaction of the D action is the stronger the higher the adjusted derivative time. With derivative time = 0, the D action is switched off for control and self-tuning. | r/w | 10.0 | >0.0 | |
| Tp1_1 ... Tp1_6 | Cycle time1 [s] H | Float | Cycle time heating [s] (2 and 3-point controller). Too high: The controller has a tendency of oscillating. The cycle time corresponds to the minimum cycle time (time in seconds) at 50 % duty cycle. | r/w | 5.0 | 0.0 ... 999.9 | |
| Tp2_1 ... Tp2_6 | Cycle time1 [s] H | Float | Cycle time cooling [s] (3-point controller). Too high: The controller has a tendency of oscillating. The cycle time corresponds to the minimum cycle time (time in seconds) at 50 % duty cycle. | r/w | 5.0 | 0.0 ... 999.9 | |

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|-------|--------------|------|--|--------|---------|-------|-----|
| CFunc | Control mode | Enum | Basic controller configuration: for various process requirements, e.g. heating, heating/cooling or three-point | r/w | 9 | | |

| | | | | | | |
|--|--|-------------------|--|--|---|--|
| | | | stepping, different control behaviours can be selected. | | | |
| | | Signal 1 | Off controller or signaller with 1 output. The on/off controller or signaller switches over, when the process value leaves the band around the setpoint determined by the hysteresis. | | 0 | |
| | | Signal 2 | The off controller or signaller with 2 outputs switches over, when the process value leaves the band around the setpoint determined by the hysteresis. The additional output can be used for additional power stages, or for an alarm function. | | 1 | |
| | | 2-point | PID controller, e.g. heating, with one switching output. The PID controller can respond to changes of the control deviation quickly and typically has no permanent control deviation. | | 2 | |
| | | 3-point | 2 x PID controller, e.g. heating/cooling. Two switching digital outputs. A PID controller can respond to changes of the control deviation quickly and typically has no permanent control deviation. | | 3 | |
| | | Cont. / switching | 2 x PID controller, e.g. heating/cooling. 1st output correcting (analog output, e.g. heating), 2nd output switching (digital output, e.g. cooling). A PID controller can respond to changes of the control deviation quickly and typically has no permanent control deviation. | | 4 | |
| | | Switching / cont. | 2 x PID controller, e.g. heating/cooling. 1st output switching (digital output, e.g. heating), 2nd output correcting (analog output, e.g. | | 5 | |

| | | | | | | |
|--|--|----------------|---|--|----|--|
| | | | cooling). A PID controller can respond to changes of the control deviation quickly and typically has no permanent control deviation. | | | |
| | | Delta-Star-OFF | D/ Y/Off, or 2-point controller with partial/full load switch-over. Two digital outputs: The Y1 is the switching output and the Y2 is the changeover contact for Star/Delta (D/Y). | | 6 | |
| | | Stepping | 3-point stepping controller, e.g. for valves. 2 digital outputs. There are no positioning pulses in the lined out condition. Using the neutral zone, the switching frequency (actuator wear) and control sensitivity can be optimized. | | 7 | |
| | | Stepp. + PF | 3-point stepping controller with position feedback, e.g. for valves. 2 digital outputs. There are no positioning pulses in the lined out condition. Neutral zone: for optimization of the switching frequency (actuator wear) and the control sensitivity, Position feedback PF serves for display of the actuator, but does not affect the PID action. | | 8 | |
| | | Continuous | PID controller, e.g. heating, with one analog positioning output (continuous). The PID controller can respond to changes of the control deviation quickly and typically has no permanent control deviation. | | 9 | |
| | | Split range | 2 x PID controller, e.g. heating/cooling. Two correcting analog outputs (continuous). A PID controller can respond to changes of the control deviation quickly and typically has no permanent control | | 10 | |

| | | | | | | | |
|--------|------------------|-----------------|--|-----|----|--|--|
| | | | deviation. | | | | |
| | | Continuous + PF | Continuous controller with position feedback. PID controller, e.g. heating, with one analog positioning output. A PID controller can respond to changes of the control deviation quickly and typically has no permanent control deviation. The actually flowing positioning current can be displayed via input PF, without being included into the control action. | | 11 | | |
| CType | Controller type | Enum | Several input variables can form the process value. Ratio control e.g. for control of mixtures; 3-element control for processes in which load changes would be detected too late (disturbance variable). | r/w | 0 | | |
| | | Standard | The process variable measured via analog input PV_1 is used a process value by the controller. | | 0 | | |
| | | Ratio | Process engineering frequently requires various components to be blended into a product. These ingredients should be mixed at a defined ratio. Process value PV applied to the controller is determined by the ratio of two input variables rather than being measured as a process variable. Suitable values for configuration of Ratio and constant NO must be adjusted. | | 1 | | |
| | | 3-element | With 3-element control, the process value is calculated according to the equation $PV_{eff} = PV_1 + A * (PV_2 / PV_3)$. | | 2 | | |
| SPfunc | SP intern/extern | Enum | Determine, if only the internal setpoint SP (setpoint control) or also the external setpoint SPe (setpoint/cascade | r/w | 0 | | |

| | | | | | | | |
|-------|---------------------|-------------------|--|-----|---|--|--|
| | | | control) should be used. | | | | |
| | | Set-point | With setpoint control, a fixed setpoint is determined by the internal setpoint SP. | | 0 | | |
| | | Set-point/Cascade | With setpoint/cascade control, switch-over from the external setpoint SPe to the internal setpoint SP is possible. Switch-over is via the front panel, via the digital SPe_SP input or via the interface. Unless this digital input is connected, or with a 0 signal applied to this input the external setpoint is taken over as an effective setpoint (SPeff). Unless both the digital input SPe_SPi and the analog input SPe are connected, the controller is fixed to the internal setpoint. | | 1 | | |
| CMode | Operating direction | Enum | Controller output action. An increasing process value is corrected by reducing (inverse, e.g. heating) or increasing the power (direct, e.g. cooling). | r/w | 0 | | |
| | | Inverse | Inverse reaction, e.g. heating. The correcting variable is increased with decreasing process value and reduced with increasing process value. | | 0 | | |
| | | Direct | Direct reaction, e.g. cooling. The correcting variable is increased with increasing process value and reduced with decreasing process value. | | 1 | | |
| CDiff | Differentiation | Enum | Rating of disturbance behaviour and control behaviour. PV differentiation: the controller disturbance behaviour is given a higher importance. DV differentiation: changes of the process value (disturbances) and of the setpoint are given the same importance. | r/w | 0 | | |

| | | | | | | | |
|-------|-------------------|-------------|--|-----|---|--|--|
| | | Acts on DV | Differentiation of control deviation: Changes of the process value (disturbances) and of the setpoint are used dynamically to improve control. Thus disturbance behaviour and control behaviour are given the same importance. | | 0 | | |
| | | Acts on PV | Differentiation of the process value: Changes of the process value (disturbances) are used dynamically to improve control. Thus the disturbance behaviour is given higher importance. | | 1 | | |
| CFail | Sensor fail | Enum | The user determines the reaction to reach a safe process condition in case of sensor error. | r/w | 1 | | |
| | | Neutral | 3-point stepping: No output pulses. Otherwise: No output pulses or 0%. | | 0 | | |
| | | Minimum AV | 3-point stepping: The actuator is closed. Otherwise: Output of the minimum positioning value. | | 1 | | |
| | | Maximum AV | 3-point stepping: The actuator is opened. Otherwise: Output of the maximum positioning value | | 2 | | |
| | | AV2 | 3-point stepping: not selectable. Otherwise: fixed "second correcting variable" AV2 (also with manual mode, not adjustable). | | 3 | | |
| | | AV2 / AVman | 3-point stepping: not selectable. Otherwise: fixed "second correcting variable" AV2 adjustable during manual mode. | | 4 | | |
| COVC | Setpoint limiting | Enum | Override control: variable minimum or maximum output limiting for an analog input value. The transitions are bumpless. | r/w | 0 | | |
| | | Off | Switch-off: No limiting of the | | 0 | | |

| | | | | | | | |
|--------|-------------------|------------------|---|-----|---|--|--|
| | | | minimum (OVC-) or of the maximum (OVC+) positioning value to the value of an analog input. | | | | |
| | | Maximum via OVCP | Limiting of the maximum positioning value to the value of an analog input (OVC_P). The transitions are bumpless. | | 1 | | |
| | | Minimum via OVCM | Limiting of the minimum positioning value to the value of an analog input (OVC_M). The transitions are bumpless. | | 2 | | |
| | | Max/Min via OVC | Limiting of the minimum (OVC-) or of the maximum (OVC+) positioning value to the value of the related analog input. The transitions are bumpless. | | 3 | | |
| SPtrac | Setpoint tracking | Enum | Setpoint tracking: To prevent step changes when switching over from the external to the internal setpoint. The SPe used so far is taken over as an internal setpoint SP. | r/w | 0 | | |
| | | Off | No tracking. (The standard rules are applicable for switch-over from external to internal setpoint.) | | 0 | | |
| | | SP follows SPe | With setpoint tracking, the SPe used so far is taken over as an internal setpoint SP when switching over from external to internal setpoint (SPe to SP). When returning (SP to SPe), approach to the SPe is using the adjusted gradients GrwP/GrwM. Used to prevent step changes of the setpoint when switching over. | | 1 | | |
| | | SP follows PV | With process value tracking, the process value PV is taken over as internal setpoint SP when switching over from external to internal setpoint (SPe to SP). When returning (SP to SPe), approach to the external setpoint SPe is using | | 2 | | |

| | | | | | | |
|-------|------------------|-------------------|--|-----|---|---------|
| | | | the adjusted gradients GrwP/GrwM. Used to prevent step changes e.g. when starting up, when the setpoint is far away from the instantaneous process value. | | | |
| Ratio | Ratio control | Enum | Process engineering: various ingredients must be blended into a product at a pre-defined ratio. A formula for calculation of the process value as an input ratio must be selected. | r/w | 0 | |
| | | Type 1 | Ratio control in which the ratio of process value PV1 and process value PV2 is controlled according to the formula $(PV1 + NO) / PV2$. | | 0 | |
| | | Type 2 | Ratio control in which the ratio of process value PV1 and sum of the two process values PV1 and PV2 is controlled according to the formula $(PV1 + NO) / (PV1 + PV2)$. | | 1 | |
| | | Type 3 | Ratio control in which the ratio of the difference between process values PV1 and PV2 and the second process value PV2 is controlled according to the formula $(PV2 - PV1 + NO) / PV2$. | | 2 | |
| Dp | Decimals | Int | Digits behind the decimal point of the displayed process value. Also valid for display of the setpoint. | r/w | 0 | 0 ... 3 |
| Disp | Bargraph content | Enum | Correcting variable, control deviation or process value are displayed with minimum, maximum and current value on the controller operating page. With cascade control, the bargraph has special properties. | r/w | 0 | |
| | | Control output AV | Display of correcting variable AV (cascade slave operating page: always display of slave) | | 0 | |

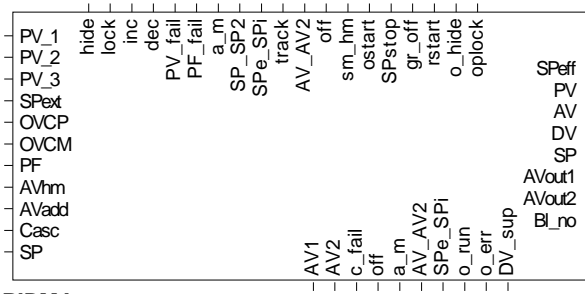
| | | | | | | | |
|-------|-----------------|---------------------|---|-----|---|--|--|
| | | | in the bargraph with value and limits | | | | |
| | | Deviation DV | Display of control deviation (process value - setpoint) in the bargraph with value and limits. Cascade slave operating page: Display of control deviation DV of the master with closed cascade; with open cascade ("Slave" is displayed), the slave data are displayed. | | 1 | | |
| | | Process value PVeff | Display of the effective process value in the bargraph with value and limits. Cascade slave operating page: Display of the effective process value PV of the master with closed cascade; with open cascade ("Slave" is displayed), the slave data are displayed. | | 2 | | |
| OMode | Tuning Mode | Enum | Only standard | r/w | 0 | | |
| | | Standard | Cannot be changed. | | 0 | | |
| OCond | Process in rest | Enum | Determine the "process-at-rest" condition: only with constant process value PV, or also with defined even process value changes. | r/w | 0 | | |
| | | PV = constant | "Process at rest" is detected, when process value PV is constant, i.e. the process value must be within a band $\leq 1\%$ of control range (PVhi - PVlo). | | 0 | | |
| | | PV-drift <0 or >0 | "Process at rest" is detected, when process value PV decreases constantly on a controller with inverse output action. Process at rest is detected, when process value PV increases constantly on a controller with direct output action. | | 1 | | |
| | | PV-drift = constant | "Process at rest" is detected, when process value PV changes constantly. In this case continuation of the constant change over the | | 2 | | |

| | | | | | | | |
|---------|--------------------|-------|--|-----|---------|---------------|--|
| | | | duration of identification must be ensured. | | | | |
| PVlo | Control span start | Float | Lower end of control range, important for proportional band (X_p) and self-tuning. | r/w | 0.0 | | |
| PVhi | Control span end | Float | Upper end of control range, important for proportional band (X_p) and self-tuning. | r/w | 100.0 | >0.0 | |
| SFac | Ratio factor | Float | Factor for stoichiometric ratio, is required for process value calculation by a ratio controller. | r/w | 1.0 | 0.01 ... 99.9 | |
| Unit_PV | Unit of PV | Text | Unit of process value (display only). For cascade control, special characteristics are applicable. | r/w | PV-UNIT | | |

III-15.3 PIDMA (Control function with particular self-tuning behaviour (No. 93))

The controller block **PIDMA** is particularly suitable for difficult processes (with delay time, or of higher order). The difference between **PIDMA** and the **CONTROL** block is only in the PID controller kernel (self-tuning and control algorithm).

The additional functions ramp, switchover, override control, feed-forward control etc. are not different.



PIDMA

Abb. 574

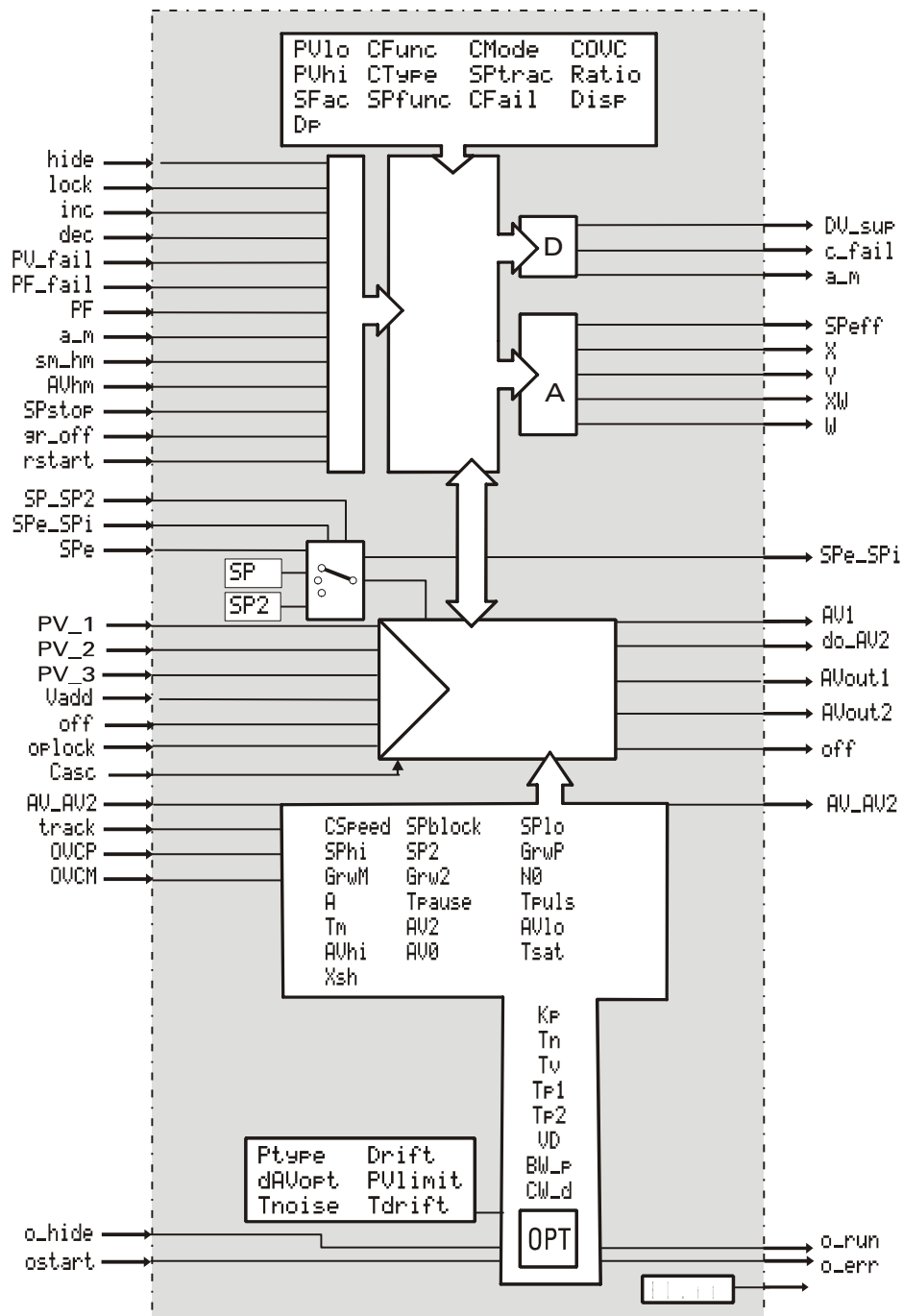


Abb. 575

The most important differences compared to controller functions CONTROL and CONTROLP are:

- Integrated, front-panel operated optimization method like PMA tune.
This function permits optimization also for difficult processes with $T_g/T_u < 3$ without engineering tool and laptop, which was not possible with previous PMA (and competitor) controllers.
- Parallel controller structure unlike all other PMA controllers with "serial structure".
- Distinction of "behaviour" and "disturbance behaviour" by adjustable factors, which can be used for individual attenuation of P (proportional action) and D (differential action) control effect on changes.
- The adjustable derivative gain VD of the D action, which is adjusted and matched to the process dynamics automatically by self-tuning. Purposeful values for VD are within 2...10, whereby all previous PMA controllers are fixed to VD=4 (empirical value for serial structure).

Using a **PIDMA** control block is purposeful where conventional PMA self-tuning methods are not satisfactory. **PIDMA** should not be used for processes where PMA self-tuning has always been and still is unrivalled:

- Processes with a ratio $T_g/T_u > 10$
- (2nd order processes; with 2 [...3] energy storing elements!).

Roughly, these are applications in the plastics processing industry (extrusion, ...), where no improvements related to quick line-out without overshoot are possible (unless a "robust" controller design with stable results also with variable process dynamics and non-linearities is required)!

With classic thermal processes (ovens of all types, dryers, ...) air conditioning, level, flow, etc., however, a considerable number of difficult cases require the expenditure of many hours spent with hotline support or even at the site, in order to make a plant operate properly.

The various types of control behaviour are not explained in detail in this section, because there are no differences compared to controller blocks **CONTROL** and **CONTROLP** (see section Control behaviour). Only the additional parameters explained at the beginning of chapter "PIDMA control characteristics" have to be taken into account.

The difference between split-range and 3-point behaviour is that **PIDMA** does not provide parameter distinction of heating and cooling.



HINT!

PIDMA does not permit the adjustment of the signaller control behaviour.

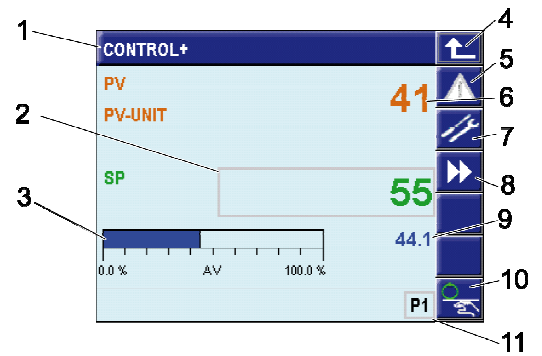
Operating page **PIDMA**

Function block **PIDMA** has an operating page selectable from the operating pages menu. With input **hide** set to "1" controller operating page is not displayed.

For further information on operation, see chapter 1: operation.

Overview

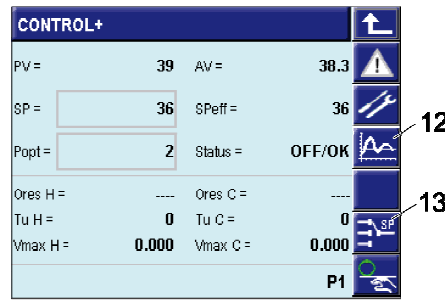
- 1 Title
- 2 Display and input field for the setpoint ("SP" = Setpoint)
- 3 Bargraph: Controller output ("AV" = "Actuating Variable"), deviation variable ("DV" = "Deviation Variable") or process value ("PV" = "Process Variable")
- 4 Button "Leave Operating Page"
- 5 Button "Alarm"
- 6 Display process value ("PV" = "Process Variable")
- 7 Button "Call parameter page"
- 8 Button "Call self-optimization"
- 9 Display controller output
- 10 Button "Switch over automatic mode/manual mode"
- 11 Controller parameter set selection (only for Control+)



Page Self-optimization

The "Self-optimization" page shows a new button compared with the main page of the controllers:

- 12 Button "Start/stop self-optimization"
- 13 Button "Setpoint switch over" (this button is also available on the main page, if switch over via parameter assignment is permitted).



| Field | Description |
|---------------|--|
| PV | "Process Variable"/process value |
| SP | "Setpoint"/ internal setpoint |
| Popt | Parameter set that will be optimized # |
| T | Optimization time* |
| AV | "Actuating Variable"/controller output |
| SPeff | ("Setpoint effective")/effective setpoint |
| Status | Current status of self-optimization Note: More detailed information on the status display is available in the function block reference. |
| Ores H | Optimization result - heating |
| Tu/ Vmax H | Process properties - heating |
| Ores C | Optimization result - cooling |
| Tu/ Vmax C | Process properties - cooling |
| Ores | Optimization result - heating/cooling* |
| Kp/Tn/Tv | Control parameters* |
| Orun | Display of "ORun", if the optimization is running, and display of "OErr" is the optimization is defective. |

* Only for controller PIDMA

Only for controller CONTROL+

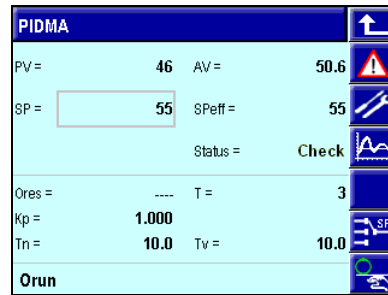


Fig. 576: Controllers "Control"/"Control+" and controller "PIDMA" (controller overview and self-optimization)

In-/Outputs

| Name | Type | Description |
|-------|-------|---|
| PV_1 | Float | Main variable PV1 |
| PV_2 | Float | Auxiliary variable PV2 e.g. for ratio control |
| PV_3 | Float | Auxiliary variable PV3 e.g. for 3-element control |
| SPext | Float | External set-point |
| OVCp | Float | Max. output limiting is to this value (not higher than AVhi). |
| OVCm | Float | Min. output limiting is to this value (not lower than AVlo). |
| PF | Float | Position feedback (with effect on the PID behaviour) |

| | | |
|---------|-------|---|
| AVhm | Float | Fixed positioning value [%] output after switch-over to hard manual |
| AVadd | Float | Feed-forward control. Is directly added to the controller output signal, thus avoiding the controller time behaviour. |
| Casc | Float | Cascade input for controller cascade (slave); the block number output of the master should be soft-wired to this input. |
| SP | Float | Change internal set-point |
| hide | Bool | Display suppression, 1 = page is not displayed during operation. |
| lock | Bool | Adjustment locking, 1 = the controls on the controller page are not active. |
| inc | Bool | Increment for manual adjustment |
| dec | Bool | Decrement for manual adjustment |
| PV_fail | Bool | Sensor error PV1 ... PV3 |
| PF_fail | Bool | Position feedback sensor error |
| a_m | Bool | Signal for taking the controller into manual operation. a_m = 0: controller in automatic mode, a_m = 1: controller in manual mode. |
| SP_SP2 | Bool | Switch-over to the second setpoint, SP/SP2 = 0: int./ext. setpoint, SP/SP2 = 1: SP2. |
| SPe_SPi | Bool | Switch-over from external to internal setpoint, SPe/SP = 0: external setpoint SPe, SPe/SP = 1: internal setpoint SP. |
| track | Bool | Setpoint tracking), 0 = tracking function off; 1 = tracking function on |
| AV_AV2 | Bool | Switch-over to fixed output value AV2. 0 = controller output AV. 1 = fixed output value AV2 |
| off | Bool | Controller switch-off; 0 = controller switched on, 1 = controller switched off |
| sm_hm | Bool | (0 = soft manual), 1 = hard manual or safe positioning value AV_hm. The controller changes to manual mode directly. The value comes from the analog input AV_hm (not adjustable via operation). Transition to automatic mode is bumpless. |
| ostart | Bool | Self-tuning start; 0 = stop, 1 = start |
| SPstop | Bool | Effective setpoint freeze (can be used e.g. for bandwidth monitoring), SPstop = 1: frozen |
| gr_off | Bool | Set-point gradient suppression; 1 = gradient ineffective |
| rstart | Bool | 1 = start the set-point ramp, i.e. the set-point makes a step change towards the process value and goes to the adjusted setpoint according to the gradient. The rising flank (0 to 1) is evaluated. |
| o_hide | Bool | Self-tuning page display suppression, 1 = self-tuning page is not displayed in the operation, |
| oplock | Bool | Blockage of the auto/manual key; 1 = switch-over to manual by means of automatic/manual key is not possible. |

| Name | Type | Description |
|-------|-------|---|
| SPeff | Float | Effective set-point. Value after setpoint processing, after taking into account SP2, external setpoint, gradients, limitings. Control deviation and control response are a result of comparison with the effective process value. |
| PV | Float | Effective process value. Value at the end of process value processing, after taken into account calculations due to ratio and 3-element control. |

| | | |
|---------|-------|---|
| AV | Float | Displayed output value (continuous) |
| DV | Float | Control deviation, i.e. PV - SP or process value - setpoint |
| SP | Float | Internal set-point |
| AVout1 | Float | Output value AVout1 (heating), continuous output |
| AVout2 | Float | Correcting variable AVout2 (cooling; only with continuous controller with split-range behaviour, i.e. Cfunc = splitRange) |
| Bl_no | Float | Own block number |
| AV1 | Bool | Status of switching output AV1; 0 = off, 1 = on |
| AV2 | Bool | Status of switching output AV2; 0 = off, 1 = on |
| c_fail | Bool | 1 = controller in error handling |
| off | Bool | 0 = controller switched on; 1 = controller switched off |
| a_m | Bool | 0 = automatic; 1 = manual |
| AV_AV2 | Bool | 0 = controller output AV; 1 = fixed output value AV2 |
| SPe_SPi | Bool | Effective setpoint: 0 = external; 1 = internal set-point |
| o_run | Bool | 1 = self-tuning running |
| o_err | Bool | 1 = self-tuning error |
| DV_sup | Bool | Signal for alarm suppression after set-point change via stop input of ALARM function blocks, 1 = alarm is suppressed. |

Parameter

| ID | Name | Type | Description | Access | Default | Range | Off |
|-------|------------------|-------------------|--|--------|---------|-------|-----|
| PType | Process type | Enum | Process type, must be determined by the user. Self-tuning is dependent on the process type: This parameter must be adjusted in a process without recovery time (integral = after an output pulse, the new process value is at a higher level). | r/w | 1 | | |
| | | With compensation | Process with recovery time, there is a process value for a positioning value. | | 1 | | |
| | | Integral | Process without recovery time, with integral behaviour. With constant correcting variable (unequal to zero), the process value changes, e.g. level. | | 2 | | |
| Drift | Drift compensat. | Enum | Process value drift | r/w | 0 | | |

| | | | | | | |
|---------|-----------------|-----------|---|-----|---|--|
| | | | compensation at self-tuning start. An even decrease or increase of the process value before self-tuning can be detected by switching on the drift monitoring function and taken into account in the next self-tuning operation. | | | |
| | | Off | No drift monitoring | | 0 | |
| | | On | A drift before self-tuning is monitored and taken into account during the following self-tuning operation. | | 1 | |
| CSpeed | Control dynamic | Enum | CSpeed can be used to determine, if the line-out to the setpoint should be fast and with slight oscillation or slow and with smooth approach to the setpoint. | r/w | 1 | |
| | | Slow | Adjust control parameters for slow and soft line-out to the setpoint. | | 1 | |
| | | Normal | Adjust control parameters normally, with optimized line-out duration and overshoot. | | 2 | |
| | | Fast | Adjust control parameters for reaching the setpoint quickly, with slight overshoot. | | 3 | |
| SPblock | SP switch | Enum | Blocks setpoint switch-over. To prevent accidental process interventions, switch-over via the front-panel operation can be blocked. | r/w | 0 | |
| | | Block All | Switch-over to another setpoint is not possible on the operating page. Switching over to internal, external or second setpoint is disabled for operation. | | 0 | |
| | | Block SPe | Switching over between internal and external setpoint (in both directions) | | 1 | |

| | | | | | | | |
|------|---------------|-----------|---|-----|-------|------------------|----|
| | | | by operation is disabled, switching over to the second setpoint SP2 and vice versa is possible. | | | | |
| | | Block SP2 | Switching over to the second setpoint SP2 and vice versa is disabled, switching over between the internal and the external setpoint (both directions) by operation is possible. | | 2 | | |
| | | Enabled | No blockage is effective. Switching over to the external, internal or second setpoint via operation is possible. | | 3 | | |
| SPlo | Min. Setpoint | Float | Lower setpoint limit for Speff, specified in units of the process value | r/w | 0.0 | | |
| SPhi | Max. Setpoint | Float | Upper setpoint limit for Speff, specified in units of the process value | r/w | 100.0 | >0.0 | |
| SP2 | 2nd setpoint | Float | The second setpoint SP2 has priority over the other setpoints. Whether SP2 has safety functions, or whether it is only a pre-defined start position in defined process conditions is determined only by the type of utilization and inclusion in an automation concept. | r/w | 100.0 | 0.0 ... 100.0 | |
| GrwP | SP-Gradient + | Float | Positive setpoint gradient [+K/min]. (Not valid for SP2). If this gradient is effective, an adjusted higher setpoint is effective by a ramp rather than by a step change. | r/w | off | >0.001 | ja |
| GrwM | SP-Gradient - | Float | Negative setpoint gradient [-K/min]. (Not valid for SP2). If this gradient is effective, an adjusted lower setpoint is effective by a ramp rather than by a | r/w | off | >0.001 | ja |

| | | | | | | | |
|--------|--------------------|-------|---|-----|-------|---------------------|----|
| | | | step change. | | | | |
| Grw2 | SP2-Gradient | Float | Setpoint gradient for the second setpoint SP2 [K/min]. With effective gradient, an SP2 setpoint change or a switch-over to SP2 is effective by a ramp rather than by a step change. | r/w | off | >0.001 | ja |
| N0 | PV-Offset | Float | Zero offset (ratio controller only) | r/w | 0.0 | | |
| A | Factor | Float | Factor a with 3-element control and setpoint ramps | r/w | 1.0 | -9.99 ... 99.9 | |
| Tpause | Min. pause time[s] | Float | Minimum positioning pause time [s] (stepping controller). In addition to minimum pulse limiting using Tpuls, Tpause permits the adjustment of the minimum pause. | r/w | 0.1 | >0.1 | |
| Tpuls | min.step time[s] | Float | Minimum positioning step time [s] (stepping controller). With 3-point stepping controllers: the minimum duty cycle. | r/w | 0.3 | 0.1 ... 2.0 | |
| Tm | Motor runtime[s] | Float | Actuator response time. Important for 3-point stepping controllers. | r/w | 30.0 | >5.0 | |
| Thron | Threshold on | Float | Not effective | r/w | 0.2 | 0.2 ... 100.0 | |
| Throff | Threshold off | Float | Not effective | r/w | 0.2 | 0.2 ... 100.0 | |
| AV2 | AV2 | Float | (Not with stepping controllers). Manual operation with the second correcting value activated. Caution: Don't confuse parameter AV2 with the second controller output. | r/w | 0.0 | -105.0 ... 105.0 | |
| AVlo | Min.actuator value | Float | Lower correcting variable limit [%] (not with stepping controllers) | r/w | 0.0 | 0.0 ... 100.0 | |
| AVhi | Max.actuator value | Float | Upper correcting variable limit [%] (not with stepping controllers) | r/w | 100.0 | 0.0 ... 105.0 | |
| AV0 | Working point | Float | Controller working point | r/w | 0.0 | -105.0 | |

| | | | | | | | |
|---------|--------------------|-------|--|-----|------|---------------------|--|
| | | | [%] (not with stepping controllers) | | | ... 105.0 | |
| dAVopt | Step size (Tune) | Float | Self-tuning step height [%]. The value by which the correcting variable must change beginning from the start value, in order to cause the process reaction required for the controller. | r/w | 20.0 | -100.0 ... 100.0 | |
| PVlimit | PV threshold(Tune) | Float | Threshold for output step change (process value change). During self-tuning, the output signal is increased, and after a subsequent process value increase by more than PVlimit, the correcting variable is reset to the original value. | r/w | 10.0 | >0.5 | |
| Tdrift | Drift test time | Float | After self-tuning start, the time for detection of a process value drift is active. The time must be long enough to permit detection of a drift independent of a disturbance and multiple "coming" and "going" of disturbances. | r/w | 30.0 | >0.0 | |
| Tnoise | Noise test | Float | After self-tuning start, the drift test time is followed by the time [s] for detection of noise on the process value. The time must be long enough to permit detection of multiple "coming" and "going" of disturbances. | r/w | 30.0 | >0.0 | |
| Kp | Control gain | Float | Controller gain. Parameter Kp valid for both heating and cooling determines the controller gain of the parallel PIDMA controller structure. | r/w | 1.0 | >0.001 | |
| Tn | Integral time[s] | Float | Time constant [s] of the I action. The reaction of the I action is the faster the lower the adjusted integral | r/w | 10.0 | >0.0 | |

| | | | | | | | |
|------|--------------------|-------|--|-----|------|---------------|--|
| | | | time. With integral time = 0, the integral action is switched off for control and self-tuning. | | | | |
| Tv | Derivative time[s] | Float | Time constant [s] of the D action. The reaction of the D action is the stronger the higher the adjusted derivative time. With derivative time = 0, the D action is switched off for control and self-tuning. | r/w | 10.0 | >0.0 | |
| Tp1 | Cycle time H | Float | Cycle time heating [s] (2 and 3-point controller). Too high: The controller has a tendency of oscillating. The cycle time corresponds to the minimum cycle time (time in seconds) at 50 % duty cycle. | r/w | 5.0 | 0.0 ... 999.9 | |
| Tp2 | Cycle time C | Float | Cycle time cooling [s] (3-point controller). Too high: The controller has a tendency of oscillating. The cycle time corresponds to the minimum cycle time (time in seconds) at 50 % duty cycle. | r/w | 5.0 | 0.0 ... 999.9 | |
| VD | Derivative gain | Float | Derivative gain of the D action, which is adjusted and matched to the process dynamics automatically during self-tuning. | r/w | 4.0 | >1.0 | |
| BW_p | SP weight P-part | Float | Proportional action setpoint rating. BW_p can be used to attenuate the controller reaction on setpoint changes (control behaviour) compared to the controller reaction on process value changes (disturbance behaviour). | r/w | 1.0 | 0.0 ... 1.0 | |
| CW_d | SP-weight D-part | Float | D action setpoint rating. CW_d can be used to attenuate the controller reaction on setpoint changes (control behaviour) compared to the controller | r/w | 0.0 | 0.0 ... 1.0 | |

| | | | | | | | |
|------|---------------------|-------|---|-----|------|------|--|
| | | | reaction on process value changes (disturbance behaviour). | | | | |
| Tsat | Integr. time sat[s] | Float | Time constant [s] for the integral action with anti-reset-wind-up. If the control algorithm temporarily determines values smaller than AVlo or higher than AVhi for the correcting variable these values are returned to the limiting values AVlo/AVhi with an accelerated integral action. | r/w | 10.0 | >1.0 | |
| Xsh | Neutral zone | Float | Neutral zone for the integral action. Influences the actuator switching frequency and fine adjustment via the control deviation in the master controller. Within this zone, the controller integral action is stopped. | r/w | 0.0 | >0.0 | |

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|-------|--------------|------------|--|--------|---------|-------|-----|
| CFunc | Control mode | Enum | Basic controller configuration: for various process requirements, e.g. heating, heating/cooling or three-point stepping, different control behaviours can be selected. | r/w | 1 | | |
| | | 2-point | PID controller, e.g. heating, with one switching output. The PID controller can respond to changes of the control deviation quickly and typically has no permanent control deviation. | | 0 | | |
| | | Continuous | PID controller, e.g. heating, with one analog positioning output (continuous). The PID controller can respond to changes of the control deviation quickly and typically has no permanent control | | 1 | | |

| | | | | | | |
|--|--|-------------------|--|--|---|--|
| | | | deviation. | | | |
| | | 3-point | 2 x PID controller, e.g. heating/cooling. Two switching digital outputs. A PID controller can respond to changes of the control deviation quickly and typically has no permanent control deviation. | | 2 | |
| | | Cont. / switching | 2 x PID controller, e.g. heating/cooling. 1st output correcting (analog output, e.g. heating), 2nd output switching (digital output, e.g. cooling). A PID controller can respond to changes of the control deviation quickly and typically has no permanent control deviation. | | 3 | |
| | | Switching / cont. | 2 x PID controller, e.g. heating/cooling. 1st output switching (digital output, e.g. heating), 2nd output correcting (analog output, e.g. cooling). A PID controller can respond to changes of the control deviation quickly and typically has no permanent control deviation. | | 4 | |
| | | Split range | 2 x PID controller, e.g. heating/cooling. Two correcting analog outputs (continuous). A PID controller can respond to changes of the control deviation quickly and typically has no permanent control deviation. | | 5 | |
| | | Stepping | 3-point stepping controller, e.g. for valves. 2 digital outputs. There are no positioning pulses in the lined out condition. Using the neutral zone, the switching frequency (actuator wear) and control sensitivity can be optimized. | | 6 | |
| | | Stepp. + PF | 3-point stepping controller with position feedback, e.g. for | | 7 | |

| | | | | | | |
|-------|-----------------|-----------------|--|-----|---|--|
| | | | valves. 2 digital outputs. There are no positioning pulses in the lined out condition. Neutral zone: for optimization of the switching frequency (actuator wear) and the control sensitivity. Position feedback PF serves for display of the actuator AND HAS INFLUENCE on the PID behaviour of the PIDMA block. | | | |
| | | Continuous + PF | Continuous controller with position feedback. PID controller, e.g. heating, with one analog positioning output. A PID controller can respond to changes of the control deviation quickly and typically has no permanent control deviation. The actually flowing positioning current can be displayed via input PF; it is included into the control action (PIDMA). | | 8 | |
| CType | Controller type | Enum | Several input variables can form the process value. Ratio control e.g. for control of mixtures; 3-element control for processes in which load changes would be detected too late (disturbance variable). | r/w | 0 | |
| | | Standard | The process variable measured via analog input PV_1 is used a process value by the controller. | | 0 | |
| | | Ratio | Process engineering frequently requires various components to be blended into a product. These ingredients should be mixed at a defined ratio. Process value PV applied to the controller is determined by the ratio of two input variables rather than being measured as a process variable. Suitable values for configuration of Ratio and constant NO must be adjusted. | | 1 | |

| | | | | | | | |
|--------|---------------------|-------------------|--|-----|---|--|--|
| | | 3-element | With 3-element control, the process value is calculated according to the equation $PV_{eff} = PV_1 + A * (PV_2 / PV_3)$. | | 2 | | |
| SPfunc | SP intern/extern | Enum | Determine, if only the internal setpoint SP (setpoint control) or also the external setpoint SPe (setpoint/cascade control) should be used. | r/w | 0 | | |
| | | Set-point | With setpoint control, a fixed setpoint is determined by the internal setpoint SP. | | 0 | | |
| | | Set-point/Cascade | With setpoint/cascade control, switch-over from the external setpoint SPe to the internal setpoint SP is possible. Switch-over is via the front panel, via the digital SPe_SP input or via the interface. Unless this digital input is connected, or with a 0 signal applied to this input the external setpoint is taken over as an effective setpoint (SPeff). Unless both the digital input SPe_SPi and the analog input SPe are connected, the controller is fixed to the internal setpoint. | | 1 | | |
| CMode | Operating direction | Enum | Controller output action. An increasing process value is corrected by reducing (inverse, e.g. heating) or increasing the power (direct, e.g. cooling). | r/w | 0 | | |
| | | Inverse | Inverse reaction, e.g. heating. The correcting variable is increased with decreasing process value and reduced with increasing process value. | | 0 | | |
| | | Direct | Direct reaction, e.g. cooling. The correcting variable is increased with increasing process value and reduced with decreasing process value. | | 1 | | |
| CFail | Sensor failure | Enum | The user determines the reaction to reach a safe | r/w | 1 | | |

| | | | | | | | |
|------|-------------------|------------------|---|-----|---|--|--|
| | | | process condition in case of sensor error. | | | | |
| | | Neutral | 3-point stepping: No output pulses. Otherwise: No output pulses or 0%. | | 0 | | |
| | | Minimum AV | 3-point stepping: The actuator is closed. Otherwise: Output of the minimum positioning value. | | 1 | | |
| | | Maximum AV | 3-point stepping: The actuator is opened. Otherwise: Output of the maximum positioning value | | 2 | | |
| | | AV2 | 3-point stepping: not selectable. Otherwise: fixed "second correcting variable" AV2 (also with manual mode, not adjustable). | | 3 | | |
| | | AV2 / AVman | 3-point stepping: not selectable. Otherwise: fixed "second correcting variable" AV2 adjustable during manual mode. | | 4 | | |
| COVC | Setpoint limiting | Enum | Override control: variable minimum and/or maximum output limiting to the value of an analog input. The transitions are bumpless. | r/w | 0 | | |
| | | Off | Switch-off: No limiting of the minimum (OVC-) or of the maximum (OVC+) positioning value to the value of an analog input. | | 0 | | |
| | | Maximum via OVCP | Limiting of the maximum positioning value to the value of an analog input (OVC_P). The transitions are bumpless. | | 1 | | |
| | | Minimum via OVCM | Limiting of the minimum positioning value to the value of an analog input (OVC_M). The transitions are bumpless. | | 2 | | |
| | | Max/Min via OVC | Limiting of the minimum (OVC-) or of the maximum (OVC+) positioning value to the value of the related analog input. The transitions are bumpless. | | 3 | | |

| | | | | | | | |
|--------|-------------------|----------------|--|-----|---|--|--|
| SPtrac | Setpoint tracking | Enum | Setpoint tracking: For prevention of step changes when switching over from the external to the internal setpoint. The SPe used so far is taken over as an internal setpoint SP. | r/w | 0 | | |
| | | Off | No tracking. (The standard rules are applicable for switch-over from external to internal setpoint.) | | 0 | | |
| | | SP follows SPe | With setpoint tracking, the SPe used so far is taken over as an internal setpoint SP when switching over from external to internal setpoint (SPe to SP). When returning (SP to SPe), approach to the SPe is using the adjusted gradients GrwP/GrwM. Used to prevent step changes of the setpoint when switching over. | | 1 | | |
| | | SP follows PV | With process value tracking, the process value PV is taken over as internal setpoint SP when switching over from external to internal setpoint (SPe to SP). When returning (SP to SPe), approach to the external setpoint SPe is using the adjusted gradients GrwP/GrwM. Used to prevent step changes e.g. when starting up, when the setpoint is far away from the instantaneous process value. | | 2 | | |
| Ratio | Ratio control | Enum | Process engineering: various ingredients must be blended into a product at a pre-defined ratio. A formula for calculation of the process value as an input ratio must be selected. | r/w | 0 | | |
| | | Type 1 | Ratio control in which the ratio of process value PV1 and process value PV2 is controlled according to the formula $(PV1 + N0) / PV2$. | | 0 | | |
| | | Type 2 | Ratio control in which the ratio | | 1 | | |

| | | | | | | | |
|------|------------------|---------------------|---|-----|---|---------|--|
| | | | of process value PV1 and sum of the two process values PV1 and PV2 is controlled according to the formula $(PV1 + NO) / (PV1 + PV2)$. | | | | |
| | | Type 3 | Ratio control in which the ratio of the difference between process values PV1 and PV2 and the second process value PV2 is controlled according to the formula $(PV2 - PV1 + NO) / PV2$. | | 2 | | |
| Dp | Decimals | Int | Digits behind the decimal point of the displayed process value. Also valid for display of the setpoint. | r/w | 0 | 0 ... 3 | |
| Disp | Bargraph content | Enum | Correcting variable, control deviation or process value are displayed with minimum, maximum and current value on the controller operating page. With cascade control, the bargraph has special properties. | r/w | 0 | | |
| | | Control output AV | Display of correcting variable AV (cascade slave operating page: always display of slave) in the bargraph with value and limits | | 0 | | |
| | | Deviation DV | Display of control deviation (process value - setpoint) in the bargraph with value and limits. Cascade slave operating page: Display of control deviation DV of the master with closed cascade; with open cascade ("Slave" is displayed), the slave data are displayed. | | 1 | | |
| | | Process value PVeff | Display of the effective process value in the bargraph with value and limits. Cascade slave operating page: Display of the effective process value PV of the master with closed cascade; with open cascade ("Slave" is displayed), the slave data are displayed. | | 2 | | |

| | | | | | | | |
|---------|--------------------|-------|--|-----|---------|---------------|--|
| PVlo | Control span start | Float | Lower end of control range | r/w | 0.0 | | |
| PVhi | Control span end | Float | Upper end of control range | r/w | 100.0 | >0.0 | |
| SFac | Ratio factor | Float | Factor for stoichiometric ratio, is required for process value calculation by a ratio controller. | r/w | 1.0 | 0.01 ... 99.9 | |
| Unit_PV | Unit of PV | Text | Unit of process value (display only). For cascade control, special characteristics are applicable. | r/w | PV-UNIT | | |

III-15.4 Controller applications:

The following chapter describes the common characteristics of controller block connection, which are independent of the **CONTROL** and **PIDMA** controller kernel, such as setpoint and correcting variable switchover and limiting as well as process value pre-processing.

Controller front-panel operation

Switchover disabling

In many applications, switchover via the front panel should be not possible.

Unwanted, accidental process interventions should be prevented by all means. For these cases, switching over via front panel operation may be disabled.

This is done by parameter **SP_Block**, which is intended for blocking individual or all switchover operations purposefully. In default setting, all switchover operations are blocked and the switchover field for front-panel operation cannot be selected.



HINT!

*Switchover to **SPe** is blocked by configuration **SPfunc** = ..*



HINT!

*When **SP** <> **SP2** switchover is blocked and **SPe** <> **SP** switchover is not possible (setpoint-control), the field is skipped when selecting.*

Cascade control operation

The cascade is one of most frequent controller structures with coupled control loops.

To facilitate construction and handling of these cascades, measures for connection and operation were taken at and in the controller blocks.

- A cascade consists of min. two controllers: a master controller, the process value of which provides the main correcting variable and a slave controller, on the process value of which the main variable is dependent.
- For building a cascade, the correcting variable output (**AVout1**) of the master may be wired to the input (**SPe**) of the slave controller via a scaling function (**SCAL**).
- The slave controller is informed on the cascade by connecting the master block number output to the slave cascade input.



HINT!

The special operating functions of a controller cascade for master and slave are grouped on the common operating page of the slave controller.

Switchover disabling on the slave by means of parameter **SP_Block** prevents cascade opening via front panel operation! This parameter can be used to influence the **SP** / **SPe** / **SP2** selection via the front panel.

Faulty wiring of a controller cascade

If an invalid cascade circuit was built up in the engineering, e.g. with the cascade input not connected to output **BI_no** of a master controller, the control function is not operable.

Error signalling is in the display field for the cascade:

Display: **C_Err**

Multiple cascade

A cascade control loop can be built up from a master controller with one or several slave controllers. Cascade operation is from the slave controller page. Display of the master operating page should be suppressed (**hide=1**).

Operator interface activation for cascade control is automatic for controllers the **Casc** input of which is connected with the **BI_no** output of another controller.

III-15.5 Setpoint functions

Terminology

| | |
|--------------|---|
| SP | Internal setpoint |
| SPe | External setpoint |
| SP2 | Second (internal) setpoint (safety setpoint) |
| SPeff | Effective setpoint |
| DV | Control deviation ($PV - SP \rightarrow$ process value - setpoint) |

General

Several possible setpoints are available. For the priorities, see the drawing shown opposite.

"Safety" **SP 2** is given priority over the other setpoints. Switchover between setpoints is possible via interface or via the digital inputs of the controller block.

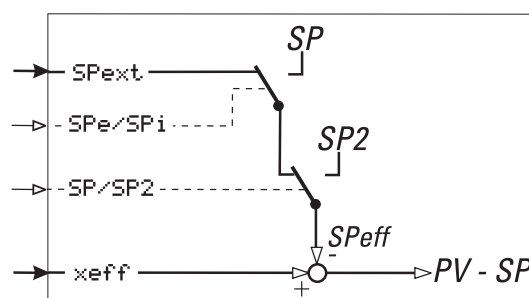


Abb. 577

If gradient control was activated, a change will be made effective continuously instead of being made effective by a step (\rightarrow gradient control). By activating digital input **SP stop**, the instantaneously effective setpoint is maintained. In this case, neither a setpoint change nor switchover to another setpoint becomes effective.

Setpoint / Setpoint/cascade

Configuration word **SPFunc** can be used to select, if switch-over to the external setpoint (/cascade) is also possible in addition to the internal setpoint.

Setpoint

(SPFunc = Setpoint) Setpoint control means that the setpoint is firmly predefined by the internal setpoint **SP**

Setpoint / cascade

(SPFunc = Setp/casc) Setpoint /cascade control permits switchover between internal setpoint **SP** and external setpoint **SPe**. Switchover is via digital input **SPe/SPI** or via the interface.

Unless this input is connected, or if a 0 signal is applied, the external setpoint is used as effective setpoint. Unless digital input **SPe/SPI** as well as analog input **SPe** are connected, the controller invariably uses the internal setpoint.

Internal setpoint SP

Internal setpoint **SP** is the setpoint that can be changed via front operation. There are different sources to change the internal setpoint: Application visualization and operation page of the controller. In both cases the setpoint is stored to become active after switch-off or power failure.

At the output **SP** the effective internal setpoint is given out as long as the signal at the input **SP** does not differ from "0". A new setpoint is taken over from the input, if its value changes and the new value is different from the active internal setpoint. The active internal setpoint is stored and is available after restart.

The following figure shows an easy application with the internal setpoint to be changed via **rectangle** in an application visualization. The sources of changes have the same priority: the last change is activated.

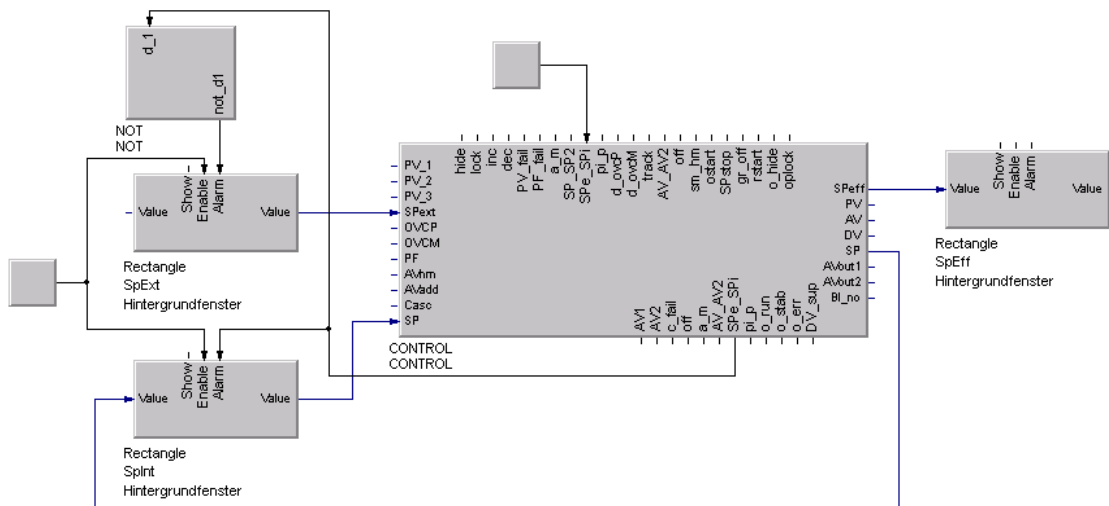


Abb. 578

SP2 – safety setpoint

The second setpoint **SP2** can be activated at any time and has highest priority. The change-over between internal setpoint and **SP2** can be triggered via interface or the digital control input **SP / SP2**. In order to make the **SP2** effective, on **SP / SP2** is a logic 1 to be attached. If the internal setpoint is to be active, a logic 0 must be given on **SPe/ SPI**.

In the past **SP2** was designated as "safety setpoint". Whether **SP2** takes over safety functions or only a predefined starting position in certain process conditions, becomes determined only by the kind of the use and integration into an automation concept.

External setpoint SPext

Switching between the internal setpoint (SP) and the external setpoint (SPe) is possible only if the parameter SPFunc is adjusted to SP/Casc. The change-over can be triggered via interface or the digital control input SPe/SP. In order to make the internal setpoint effective, on SPe/SP must be attached a logic 1. If the external setpoint is to be active, a logic 0 must be given on SPe/SP.

If the digital control input SPe/SP is not wired, the external setpoint is effective.

The internal setpoint SP is evaluated with priority. If in a place (interface or the digital control input SPe/SP) is switched to internal setpoint it is not possible to switch over to the external setpoint SPe from another source.

Gradient control - setpoint changes with gradients

Normally, setpoint changes occur stepwisely. Unless this behaviour is required, a gradient can be set-up using parameters GrwP and GrwM or Grw2.

If these parameters are set, the setpoint changes are made bumplessly. With digital input 'gr_off' not set, effective setpoint SPeff runs linearly towards the changed setpoint (target value), whereby the slope is determined by gradients GrwP and GrwM adjustable at parameter setting level. For the second setpoint SP2, an independent gradient Grw2 was introduced, which is valid for both change directions and for SP → SP2 switchover.

The gradient function is switched off, when setting GrwP and GrwM or Grw2 to "----" (engineering tool = off), or with digital input gr_off set to 1.

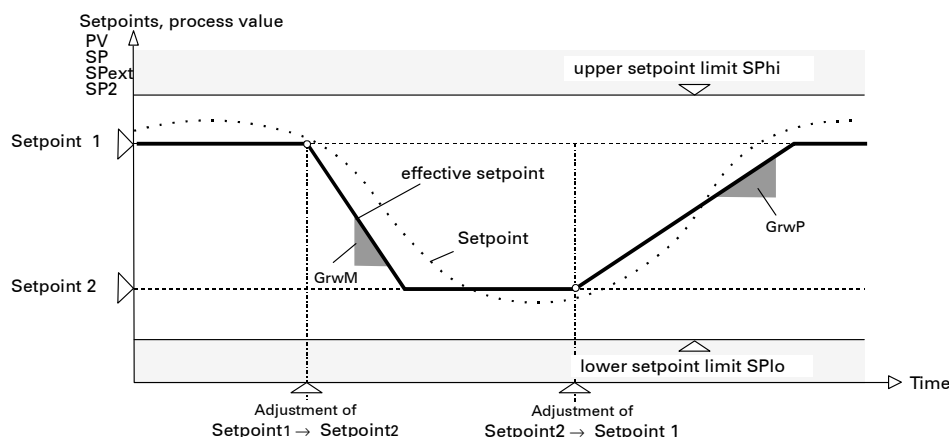


Abb. 579

Switch-over with gradients (SP→SP2, SP→SPe, controller"on")

The new setpoint is linear started outgoing from the momentary process-value. The slope of the ramp is determined related to the direction of GrwP, GrwM and/or Grw2.

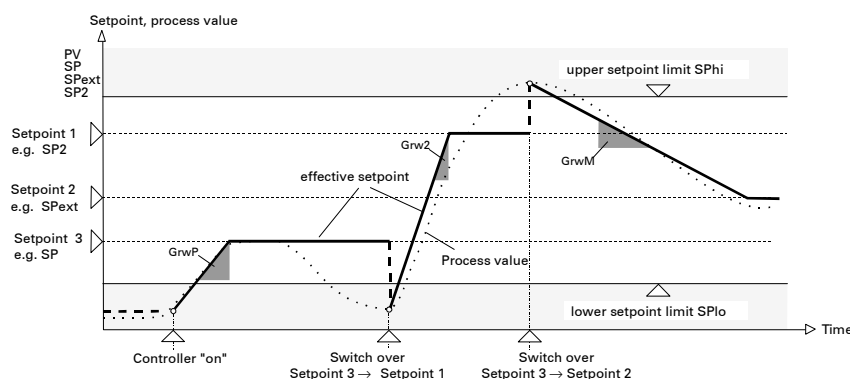


Abb. 580

i **HINT!**
 This principle applies, even if the process-value is outside the adjustable range SP_{lo}/SP_{hi} at switch-over time (e.g. when starting).

Controlling the setpoint

The digital input **rstart** reacts to a positive signal slope and sets the effective setpoint to the process value. The new goal is started on the basis of the controlled variable **SPeff**.

Such a ramp can be started only with activated gradient function (**GrwP**, **GrwM**, **Grw2** and digital input **gr_off** not set). The digital input **SP_stop** freezes the effective **SPeff**, i.e., the effective setpoint is held to the current value, even if the effective setpoint straight approaches a new goal or a new goal is selected.

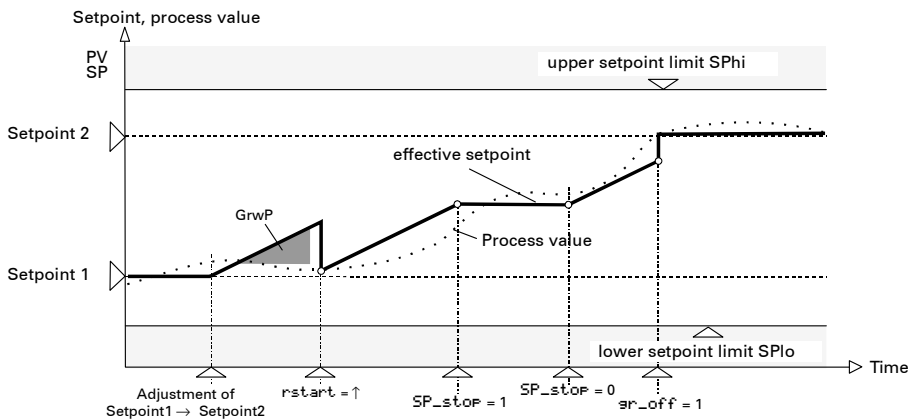


Abb. 581

Setpoint -Tracking

During the change-over of **SPe** → **SP** it can come to unwanted jumps. To avoid these jumps setpoint-tracking can be applied. Setpoint-tracking takes care that in switch-over of **SPe** → **SP**, the past **SPe** is taken over as int. **SP**.

The digital input "track" enables Tracking. When switching back (**SP** → **SPe**) **SPe** is started with the attitude of **GrwP/M**. With the configurationword is determined if the controller shall follow process value or setpoint-tracking **SPtrac**. Tracking can be activated via interface or by operating the switchover **SPe** → **SP**. Tracking is evaluated with priority.

If a source (interface or digital input) is activating setpoint-tracking, switching by operation switching **SPe** → **SP** from another source is not possible!

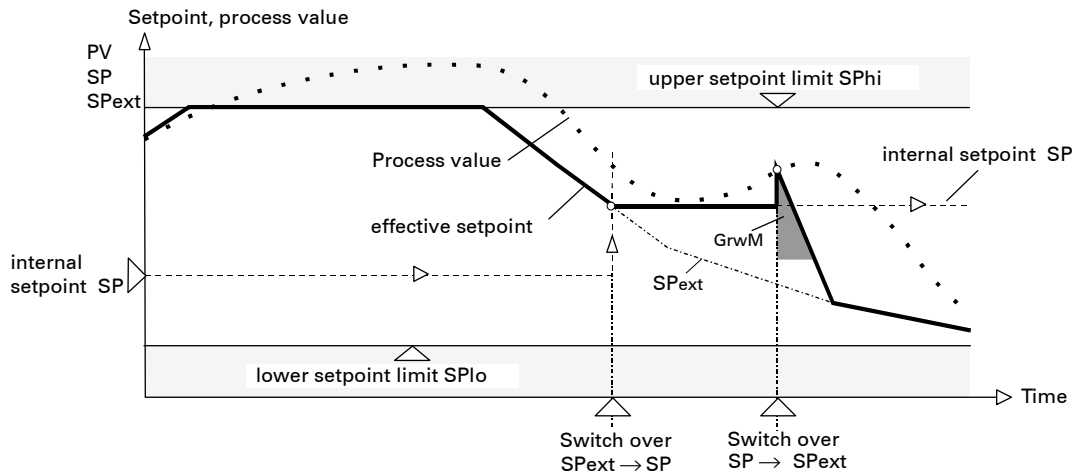


Abb. 582

Process-value-Tracking

It can occur that the is far distant from the momentary process-value (e.g. when starting a plant). In order to prevent the jump developing here, the function process-value tracking can be used.

Process-value tracking causes a take-over of the process-value on the internal setpoint , when changing over **SPe** → **SP**. When shifting (**SP** back → **SPe**) **SPe** is started with the attitude of **GrwP/M**

If the controller shall follow process value or setpoint-tracking is determined with the configurationword **SPtrac**. The digital input "track" enables Tracking. Tracking can be activated via interface or by operating the switchover **SPe** → **SP**. Tracking is evaluated with priority.



HINT!

If a signal source switches to tracking, switching from another source is not possible!

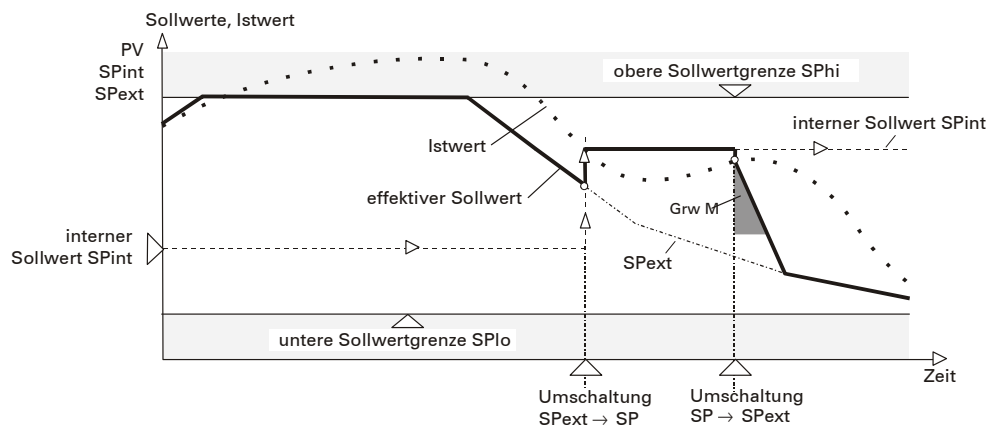


Abb. 583

Setpoint and correcting variable behaviour after setpoint switch-over

After setpoint and correcting variable switch-over, control behaviour or start-up behaviour has priority. The PID characteristic must be partly suppressed. The previous history which is important for the integral action and especially for the derivative action is largely insignificant with setpoint change due to the new target setpoint.

Switch-over operations which might affect the control behaviour are:

| | | |
|---|-----------------------------|---|
| 1 | Manual -> automatic | Switch-over from manual to automatic mode |
| 2 | Off -> start-up | Start-up after off-line (power failure/configuring) |
| 3 | SP old -> SP new | Setpoint change |
| 4 | SP -> SP2 | Switch-over to 2nd setpoint |
| 5 | SP2 -> SP | Switch-over from 2nd setpoint to normal setpoint |
| 6 | SPe -> SP, without tracking | Switch-over from external to internal setpoint without tracking |
| 7 | SP -> SPe | Switch-over from internal to external setpoint |
| 8 | SPe -> SP with tracking | Switch-over from external to internal setpoint with tracking |

The approach to a new setpoint may be affected by further parameters. Parameters **GrwP** (positive setpoint gradient), **GrwM** (negative setpoint gradient) and **Grw2** (setpoint gradient during the approach to **SP2**) can be used for gradual approach to a new target setpoint via a ramp function.

Unless a gradient is defined (**Grw = off**), approach to the new setpoint starts with a step change at the previous setpoint or at the actual process value.

To influence the correcting variable when switching over, any after-effect of the derivative action is eliminated internally or the integral action is adapted to avoid correcting variable bumps.

The following table gives a survey of the controller switch-over behaviour implemented.

Controller-internal operations during switch-over with CONTROL, CONTROLP and PIDMA

| Switch-over | Without gradient function | With gradient function |
|-------------|--|--|
| 1 | After correcting variable adaptation with deletion of a still effective derivative action, the approach to the setpoint is bumpless | The effective setpoint ramp continues running in the background during manual mode. After switching over to automatic, the correcting variable is adapted and the derivative action is deleted and the setpoint is set to the actually reached ramp setpoint (bumpless). |
| 2 | The effective setpoint is set to the process value first and after deleting a still effective derivative action, a setpoint step change to the target is made. During this step change, the PID parameters are effective. The derivative action is a result of the step change (not bumpless). | At first, the effective setpoint is set to the process value. After deleting the derivative action, the approach to the target setpoint is via a ramp. During this transition, the PID parameters are effective (bumpless starting with 0) |

| | | |
|------------|--|---|
| 3 | After deleting a still effective derivative action, a step change from the instantaneous to the target setpoint is made. During this step change, the PID parameters are effective. The derivative action is a result of the new step change (not bumpless). | After deleting the derivative action and adapting the correcting variable, changing from the old to the new target setpoint is done via a ramp (bumpless). |
| 4, 5, 6, 7 | After deleting a remaining derivative action, a setpoint step change from the instantaneous to the target setpoint is made. During this step change, the PID parameters are effective. The derivative action is only a result of the new step change (not bumpless). | The effective setpoint is set to the process value. After deleting the derivative action and adapting the correcting variable, setpoint changing from the process value to the target setpoint is done via a ramp (bumpless). |
| 8 | The internal target setpoint is set to the actual process value or to the external setpoint. Subsequently, any still effective derivative action is deleted and the correcting variable is adapted (bumpless). | The internal target setpoint is set to the actual process value or to the external setpoint. Subsequently, any still effective derivative action is deleted and the correcting variable is adapted (bumpless). |

Gentle line-out to the target setpoint with ramps

When using a setpoint ramp, a process value overshoot at the ramp end may occur. Due to the difference between setpoint and process value in the course of the ramp, an integral action is built up and must be removed after the end of the ramp.

The longer the ramp, the higher the integral action. And the more exact the process value follows the setpoint, the higher the probability that any integral action will cause an overshoot.

The target line-out function is used to adapt the integral action to the actual PD action at an adjustable distance before reaching the ramp end value, the D-dynamics is initialized and the setpoint is set to the ramp end value.

Now the controller dynamics re-starts bumplessly related to the new setpoint.

Controller parameter "a" can be used to define at which distance to the final setpoint the target orientation should be switched over to the final setpoint. The target line-out function is activated under the following conditions :

1. $SP < SP_{end}$
2. $SP > SP_{end} - 2a$
3. $PV > SP_{end} - a$

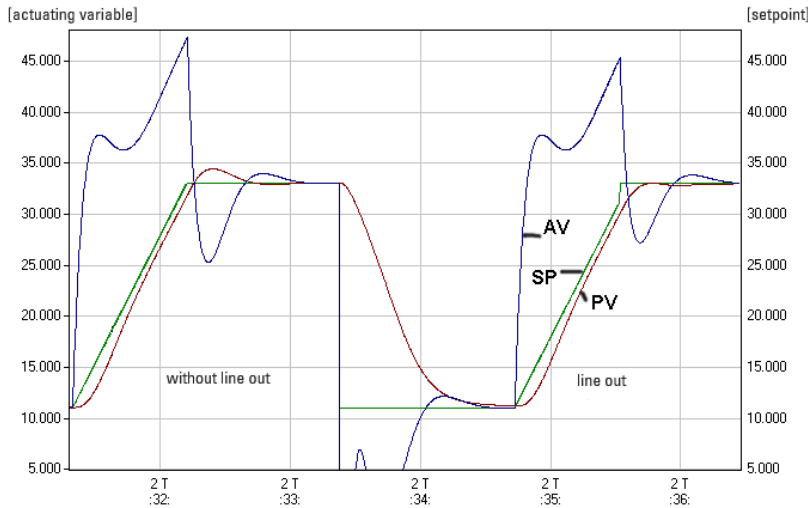


Abb. 584

Marginal conditions / restrictions:

With internal setpoint ramps, the controller knows the future target setpoint. When using external setpoints with ramp function (programmer), the ramp end value must be bound to input X3 of the controller block. When the internal ramp is active, line-out to the target setpoint is always related to the internal ramp end value, and the value at X3 is ineffective.

Target line-out is activated only, if the external ramp setpoint changes continuously. The function can be used both with differentiation of control deviation (DV) and differentiation of process value (PV).

With 3-element control, target line-out is omitted. The signification of parameter "a" is different and connection of an external end setpoint is not possible. With ratio control, target line-out is only restricted with fixed distance (1 in units of the physical quantity).

The signification of parameter a is different.

III-15.6 Process value calculation

Standard controller

The process variable measured via analog input PV_1 is used as process value by the controller.

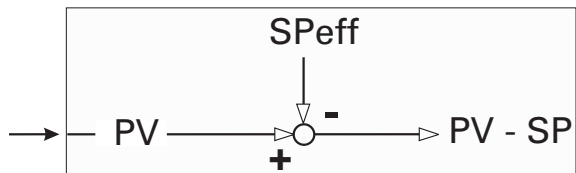


Abb. 585

Ratio controller

Process control frequently requires various components to be mixed into a product. These components must be mixed according to a given ratio.

The main component is measured and used as reference for the other components. With increasing flow of the main component, the flow of the other components will increase accordingly. This means that process

value PV used by the controller is determined by the ratio of two input variables rather than being measured as one process variable.

For optimum combustion, the fuel-air ratio must be controlled. With stoichiometric combustion, the ratio is selected so that there are no inflammable residues in the waste gas. In this case, the relative rather than the physical ratio is displayed as process value and adjusted as setpoint.

If the transmitters used by the controller are designed with a stoichiometric ratio, $\lambda = 1$ is met exactly with restless combustion.

With a process value display of 1,05, the instantaneous air excess is clearly 5%. The amount of air required for atomizing is taken into account by constant 'NO'. For selecting a ratio controller, **CType = Ratio** must be selected. Moreover, configuration word '**Ratio**' must be taken into account.



ATTENTION!

With ratio controller, note that parameters **PVlo** and **PVhi** must be set to the input range of connector **PV_1**.

Example of standard ratio control:

Standard ratio control at the example of stoichiometric combustion. An analog input is configured to 4...20 mA with physical unit m³/h (air). Values 0 and 100 are allocated to input variables 4 mA (**PVlo**) and 20 mA (**PVhi**). Atomizing air **NO** is added to this input.

E.g. an input is selected as second ratio input. This input is also configured for 4...20 mA and m³/h (gas). **PVlo** and **PVhi** values 0 and 100 are allocated to the input variables.

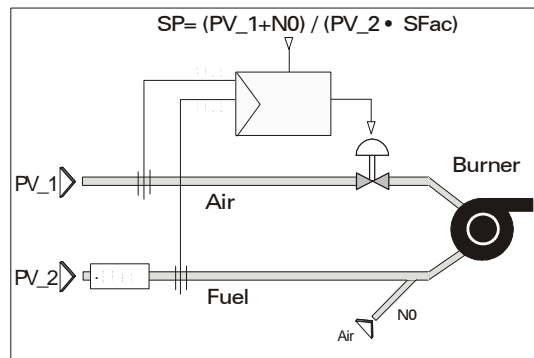


Abb. 586

Setpoint **SPeff** effective as relative ratio is multiplied by stoichiometric factor **SFac** (e.g. **SFac** = 10), i.e. a „stoichiometric“ flow ratio can be used for calculation of the control deviation.

The instantaneous (controlled) process value is calculated from the physical ratio, multiplied by $1/\text{SFac}$ and displayed as relative value.

Material batching and mixing

The following examples are intended to show that various control possibilities can be used. This is necessary, since the materials to be mixed (e.g. paste) are not always directly measurable due to their consistency. Other cases may require a component to be controlled in relation to a total rather than to another component.

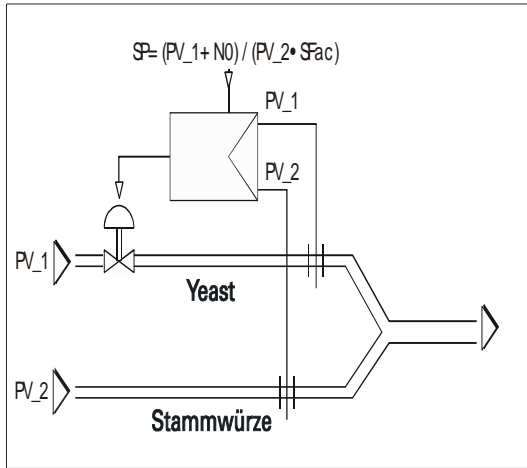


Abb. 587

Ratio = Type 1 $SP = \frac{PV1 + NO}{PV2 \cdot SFac}$

The first case is obvious. Almost everybody knows what happens during brewing. Yeast (**PV_1**) must be batched in relation to the original wort (**PV_2**). The setpoint is adjusted in '%yeast', e.g. **SP**= 3%. The ratio inputs are scaled in equal units.

The control deviation is multiplied by '**SFac** = 0,01' and calculated according to equation **DV** = (**PV_1** + **NO**) - 0,03 **SP** **PV_2**, so that exactly 3% of yeast are batched with **DV** = 0. Process value display is also in %. Constant **NO** is without importance (**NO** = 0)

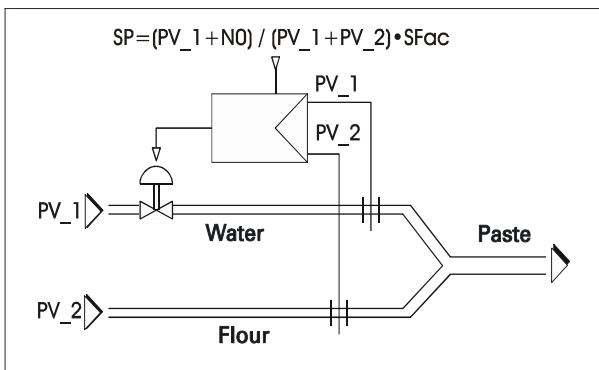


Abb. 588

Ratio = Type 2 $SP = \frac{PV1 + NO}{(PV1 + PV2) \cdot SFac}$

In this example, water (**PV_1**) must be batched as a percentage of the total (paste; **PV_1** + **PV_2**). As the paste quantity is not available directly as a measurement signal, the total is calculated internally from **PV_1** and **PV_2**. **NO** = 0 must also be adjusted in this case.

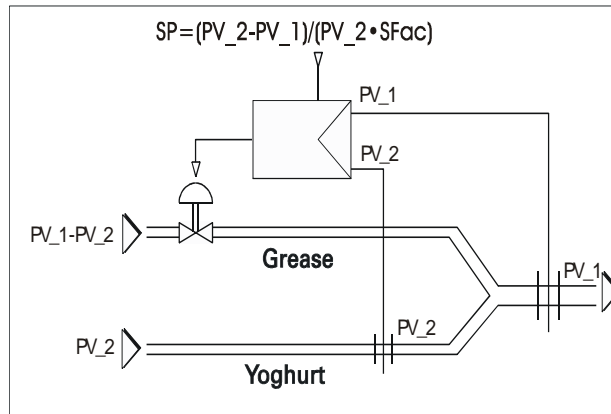


Abb. 589

Ratio = Type 3 $SP = \frac{PV_2 - PV_1 + N0}{PV_2 \cdot SFac}$

Unlike the previous examples, yoghurt (PV_2) and the final product (PV_1) are measured in this case.

Three-element control

With three-element control, process value calculation is according to equation

$$pv_{eff} = PV_1 + a \cdot (PV_2 / PV_3)$$

whereby term (PV_2/PV_3) is the difference between the steam and water flow rates.

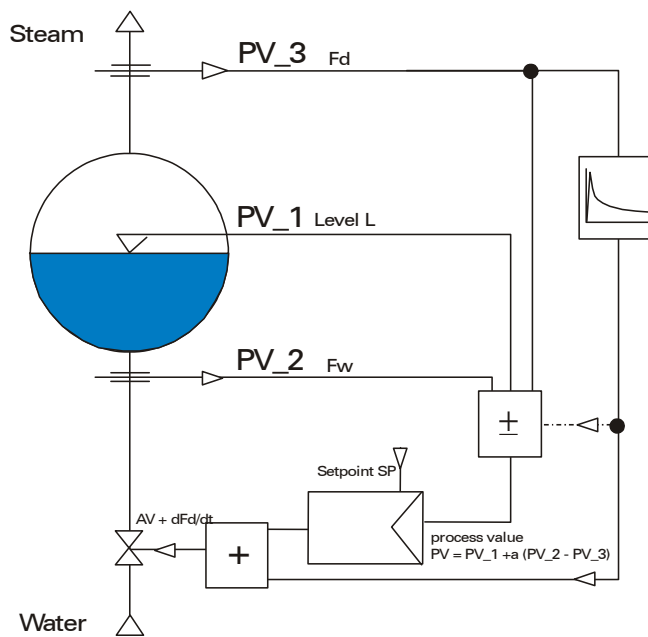


Abb. 590

For selecting a three-element controller, 'CFunc = 3-elem.' must be entered in the configuration.

III-15.7 Correcting variable processing

The following considerations related to correcting variable processing are applicable to continuous controllers, two-point, three-point and three-point stepping controllers with position feedback. The diagram opposite shows the functions and interactions of correcting variable processing.

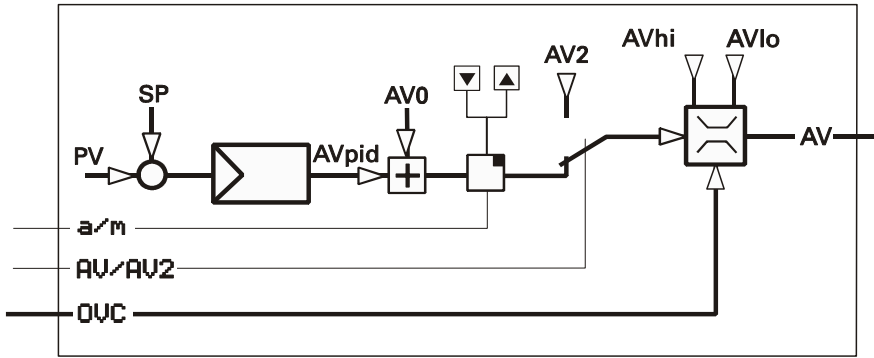


Abb. 591 Steps of the output signal processing

Second correcting value

Similar to processing, switch-over to a second preset correcting value **AV2** is possible. Switching over is done via digital input **AV/AV2**. Whether **AV2** has safety functions, or whether it is only a pre-defined start position in defined process conditions is determined only by the use and integration into an automation concept.



ATTENTION!

Second correcting value **AV2** is evaluated with priority. When switching over to **AV2** is done at one point (interface or digital control input 'AV/AV2'), switching over at the other point is not possible.

Correcting variable limits

Parameters **AVlo** and **AVhi** determine the limits of the correcting variable range within 0...100 %. With three-point and continuous controller „split range“, the correcting variable limits are within -100 ... +100 %. Parameters **AVlo** and **AVhi** are used to specify fixed correcting variable limits.

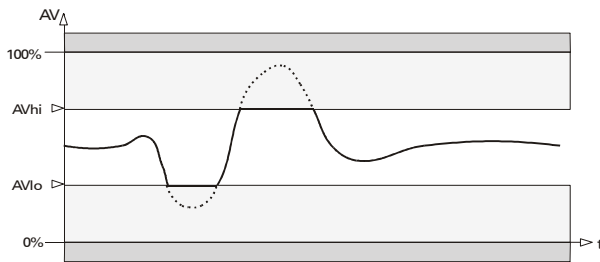


Abb. 592 Fixed correcting variable limits

External correcting variable limiting (override control)

Dependent of 'COVC' setting, the lowest (**OVC-**), the highest (**OVC+**) or lowest and highest correcting value (**OVC+/-OVC-**) can be limited by analog input signals.

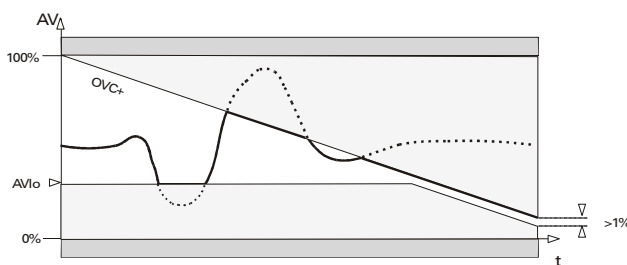


Abb. 593 Fixed correcting variable limit

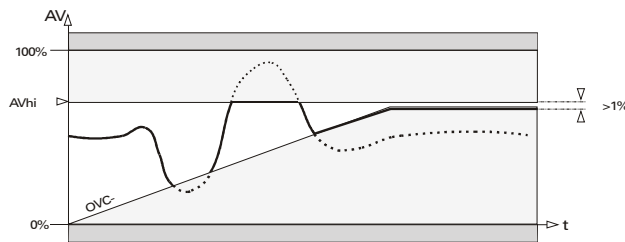


Abb. 594 Maximum value limiting

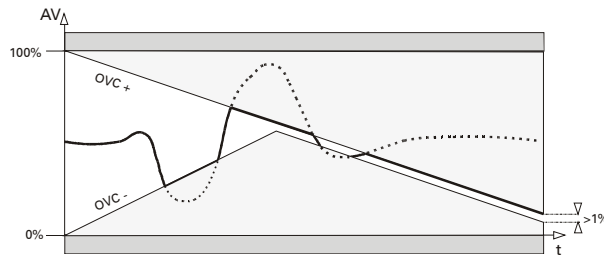


Abb. 595 Minimum value limiting

Override control is used where bumpless switch-over to another controller when reaching defined process conditions and mainly according to other criteria is required. The basic principle is that two controllers act on the same motor actuator.

Limiting control

Limiting with continuous output. Limiting control with three-point stepping output can be realized by using a continuous controller with the OVC function. A position controller (three-point stepping) provides override control.




Override (limiting) control using a three-point stepping output

Override control is also possible by means of a classical three-point stepping controller.

Which one of the two controllers influences the process is decided in the slave controller logic. The first "close" pulse coming from the limiting controller switches over to override control. The limited controller automatically retrieves the positioning authority, when it first tries to close the actuator further.

Bumpless auto/manual switch-over

Sudden process interventions by control mode switch-over are usually not desired. Excepted is purposeful switch-over $AV \rightarrow AV2$.

 switch-over is always bumpless; the last correcting value is frozen and can be changed manually.  \rightarrow  switch-over is different. Correcting value differences are compensated as follows: when switching over, the controller integral action is set to correcting value AV_M output last plus correcting variable portions of the controller P and D action running in the background ($AV_M = AV_p$).

Now, only the integrator, which adapts the correcting variable slowly to the stationary value according to the actual control deviation, is active.

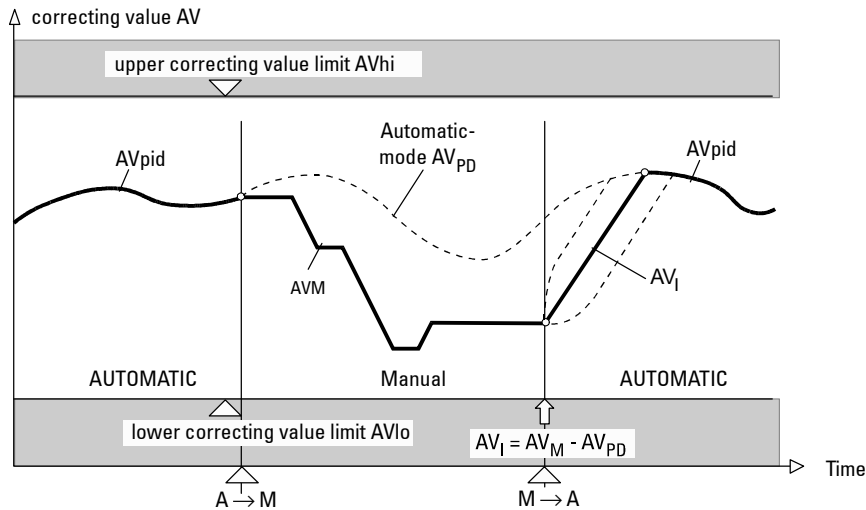


Abb. 596

III-15.8 Small controller-ABC

Some operating principles, which are realized in the controller or which are possible by means of an additional engineering are explained in the following section.

- **Anti-Reset-Wind-Up**
Measure which prevents the controller integrator from saturation.
- **Working point (AV₀)**
The working point of a P or a PD controller indicates the value output to the process with process value = . Although this value is only important for P and PD controller, it can also be of interest for controllers with integrator (automatic working point).
- **Automatic operation**
Normal controller operation. The controller controls the process by means of the adjusted control parameters. Automatic operation is effective with a/m set to 0 (automatic) **and** automatic selected **and** sm/hm set to 0 (soft manual). Contrary: manual operation.
- **Cutback**
Reset of the integral action shortly before reaching the end setpoint with setpoint ramps.
- **Cycle time**
The duration of a switching cycle (pulse and pause) at 50 % power control of a 2-point controller.
- **Line-out to the target**
By early setpoint switch-over to the ramp end setpoint, the controller is given a new target orientation for smooth line-out to the target.
- **Bandwidth control**

With program control or gradient control, there may be a considerable control deviation if the process is slow. This can be prevented by monitoring the control deviation for an adjusted tolerance band by means of additional function blocks. With out-of-tolerance, the change is stopped (**SPstop** with controller or stop with program controller).

■ Three-element control

Particularly suitable for processes in which load changes would be detected too late (e.g. level control for steam boilers). In this case, a disturbance variable is used at which the mass balance (steam removal, feed water) is evaluated, subtracted and added to the control variable (after differentiation, if necessary).

■ Feed-forward control




Especially suitable for processes with long delay time, e.g. pH-control. A disturbance variable is used, at which the evaluated, differentiated or delayed value of an analog input (**AVadd**) is added directly to the controller output for avoiding the controller time behaviour.

■ Gradient control

Particularly suitable for processes in which energy shocks or quick setpoint changes must be avoided. Setpoint changes are bumpless in both directions, since the effective setpoint always runs towards the changed setpoint (destination) by means of gradients **GrwP** or **GrwM**. For the second **SP2**, gradient **Grw2** acts in both directions, also with **SP** → **SP2** switch-over.

■ Manual operation

When switching over to manual operation, the automatic sequence in the control loop is interrupted. Modes soft manual and hard manual are available. Switch-over automatic → manual and vice versa are bumpless. Manual operation is effective with a/m set to 1 (manual) **or** manual selected **or** sm/hm set to 1 (hard manual). Contrary: automatic.

 **HINT!**
 If automatic remains selected via key , the controller changes to automatic after omission of the a/m signal. With manual selected additionally via key , the controller remains in manual mode after omission of the a/m signal.

■ Hard-Manual (sm/hm)

Safety output value **AV_hm**. The controller output goes to the preset value immediately, when hard manual is active (the controller is switched to manual mode directly). Safety output value is delivered by analog input **AV_hm** and can't be adjusted via display. Switch-over to automatic mode is bumpless.

■ Cascade control

Particularly suitable for temperature control in e.g. steam boilers. A continuous master controller (load controller) provides its output signal as an external setpoint to a slave controller, which varies the output value.

■ Override-Control (OVC)

Limiting of the min. (OVCM) or max. (OVCP) output value to the value of an analog input. Limitation by override control can be used e.g. with control continued by a different controller dependent of different conditions when reaching defined process statuses. The transitions from unlimited → limited output value and vice versa are bumpless.

■ Program control

The effective setpoint follows the profile of a programmer (A_PROG with A_PROG_D). It is connected to input **SPe**; the controller must be set to **SPfunc= setp/casc** and the digital input **SPe/SP** must be set to 0.

■ Process at rest

For a clear optimization attempt during self-tuning, the control variable must be in a stable position. Various rest conditions can be selected:

| | | |
|--|---------------------|--------------------------------|
| Process behaviour with constant output value | Recommended setting | Stability PIR_H is reached, if |
|--|---------------------|--------------------------------|

| | | |
|---|-----------|--|
| A constant process value is reached in relatively short time (standard process). | grad=0 | the process value is constant during 1 minute. |
| After a relatively long time, a constant process value is reached (slow process). | grad<0/>0 | the process value decreases constantly during 1 minute (controller inverse) or increases constantly during 1 minute (controller direct). |
| The process is affected from the outside. | grad<>0 | the process change is constant during 1 minute. The output action is unimportant. |

■ Ramp function

Changes in ramps rather than in steps. See gradient control.

■ Control parameters

For controller optimization, the controller must be matched to the process characteristics. The effective parameters are **SP1**, **Tn**, **Tv** and **AV0**. Dependent of controller operating principle, the following additional parameters are possible: **TF1** (with 2-point/3-point controllers), **SP2** and **TF2** (with 3-point controllers), **SPsh** and **TFuls** and **Tm** (with 3-point stepping controllers).

■ Control behaviour

Generally, fast line-out to the without overshoot is required. Dependent of process, various control behaviours are desirable for this process:

- easily controllable processes ($k < 10\%$) can be controlled with PD controllers,
- processes with medium controllability ($k 10\ldots 22\%$) using PID controllers and
- badly controllable processes ($k > 22\%$) with PI controllers.

■ Controller OFF (off)

With input **off** =1, there are no pulses at the switching output and the continuous outputs are 0%.

■ Self-tuning

For optimum process control, the controller must be matched to the process requirements. The time required for this purpose can be reduced considerably by self-tuning. During self-tuning, the controller makes one adaptation attempt during which the control parameters are determined automatically from the process characteristics for fast line-out to the without overshoot.

■ Soft-Manual

with automatic → manual change-over, the last output value remains active and can be adjusted via keys ▼▲. Transitions automatic → manual and vice versa are bumpless.

■ Switch-over

In principle, the following setpoints are possible: Internal **SP**, second internal setpoint **SP2** and external **SPe**. With program control, external setpoint **SPe** must be selected. The analog comes from A_PROG and is applied to input **SPe**.

■ AV feedback control

Particularly suitable for processes in which load changes lead to process value drops. A load-dependent change to setpoint (preferable) or process value is made. The evaluated and filtered output value is added to the setpoint in a separate function block. Use the **SPe** input and set the controller to **SPe**.

■ PI/P switch-over

When optimizing slow processes, e.g. big furnaces, the controller I action can cause problems: if starting up was optimized, line-out can be slow; with optimization of the disturbance behaviour, there may be an important overshoot. This effect is prevented by switching off the I action during start-up or with high control deviations (e.g. by applying a limit contact to the control deviation) and switching it on again only when the process value approaches setpoint. To prevent permanent control deviations, the limit contact must be further away from setpoint than the permanent control deviations.

- With parameter **ImodePiP** the behaviour of the integrator is set for change over of structure PI <-> P. The I part of controller is resetted with the switch over (set to "0", **ImodePiP** = 0 = off), frozen (**ImodePiP** = 1 = freeze) or pulled down with integral time constant **Tn** (**ImodePiP** = 2 = pull down).

■ Tracking

During switch-over from external or program setpoint to internal setpoint, setpoint or output value step changes may occur. By means of the tracking function, the transition is bumpless.

- Process value tracking: During switch-over, the effective process value is used as internal setpoint.
- Setpoint-tracking: During switch-over, the external or program setpoint used so far is taken over as internal setpoint.

■ Behaviour with **Fail** (configuration of the controller behaviour with **PV_fail**, **Cfail**)

| Selected behaviour | Effect with 3-point stepping controllers | Effect with other controllers |
|--------------------|--|--|
| Neutral | No output pulses | No output pulses or 0% |
| AVlo | Actuator is closed | AVlo (= limiting) |
| AVhi | Actuator is opened | AVhi (= limiting) |
| AV2 | Not selectable | AV2 fixed, also with manual operation |
| AV2/AVman | Not selectable | AV2, adjustable in manual mode with ▲▼ |

■ Ratio control

Particularly suitable for controlling mixtures, e.g. fuel-air mixture for ideal or stoichiometric combustion. For taking e.g. the atomizer air into account, zero offset NO can be added.

■ PV/DV- differentiation

Dynamic changes of process value or setpoint affect control differently.

- PV-differentiation: Process value changes (disturbances) are used dynamically to permit better control results. Disturbances are valued stronger.
- DV-differentiation: Changes of process value (disturbances) and setpoint are used dynamically to permit a better control result. In this case, the improvement is dependent of both disturbance and control behaviour.

■ Controller operating principle

The static operating principles for controllers with P or PD behaviour with adjustable working point **AV0** are shown. On controllers with I action, the working point is shifted automatically. The outputs (⊕) are described with h („heating“), c („cooling“), („open“) and („close“).

III-15.9 Control behaviour

The following chapter describes the different control behaviours adjustable with the configuration parameter CFUNC and determines the parameters effective then. All available parameters can be adjusted in the engineering tool. However it is not recognizable, which of the adjusted values are really affecting the process. The following compilation shall help to enlighten, which parameters are really used, dependent from the adjusted controller type. Therefore the relevant parameters for control behaviour are accented grey in the tables

Signaller, 1 output:

The signaller is suitable for processes with small T_u and low v_{max} .

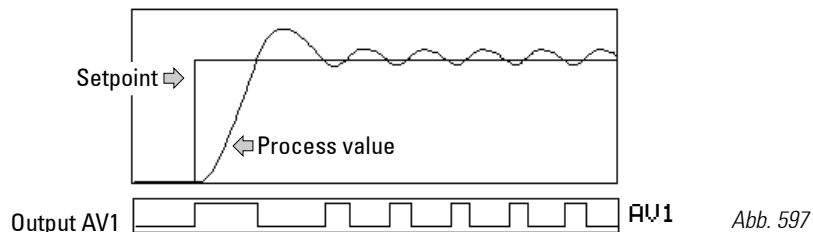


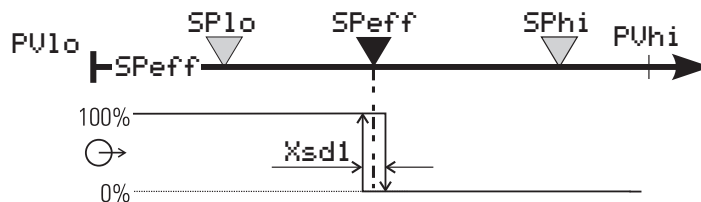
Abb. 597

The advantage is in the low switching frequency. Switch-on is always at a fixed value below the , switch-off is always at a fixed value above the setpoint.

The control variable oscillation band is determined as a result of :

$$X_0 = X_{max} \cdot \frac{T_u}{T_g} + X_{sd} = V_{max} \cdot T_u + X_{sd}$$

The signal function corresponds to limit signalling, whereby the setpoint is the limit value. The trigger point is symmetrical to the setpoint: hysteresis X_{sd} is adjustable.



Static operating principle of the signalling function of a signaller, 1 output

Abb. 598

| Configuration | Effective controller parameters of a signaller with one output | | |
|-----------------------------------|--|------------------------|-------------------------|
| CFunc = signaller, 1 output | SP1o ¹⁾ | Lower limit for SPeff | -29 999 ...999 999 |
| | SPhi ¹⁾ | Upper limit for SPeff | -29 999 ...999 999 |
| | SP2 ¹⁾ | Additional setpoint | -29 999 ...999 999 |
| | GrwF ²⁾ | Setpoint gradient plus | off / 0,001 ... 999 999 |

| | | |
|--------------------------|--|-------------------------|
| GrwM²⁾ | Setpoint gradient minus | off / 0,001 ... 999 999 |
| Grw2²⁾ | Setpoint gradient for SP2 | off / 0,001 ... 999 999 |
| NO | Zero offset (only effective with CType=ratio controller) | -29 999 ... 999 999 |
| A | Factor a (only effective with CType=3-element control) | -9,99 ... 99,99 |
| Xsd1¹⁾ | Signaller switching difference | 0,1 ... 999 999 |
| Title | Title of controller page (only display) | 16 characters |
| UnitPV | Process value unit (only display) | 6 characters |

- 1) The values are specified in the process value unit - e.g. [°C, °F, bar, %, etc.]
- 2) The rate of change must be specified in units /minute (e.g. °C/min). → see gradient control.

Signaller, 2 outputs

The signaller is suitable for processes with small T_u and low v_{max} .

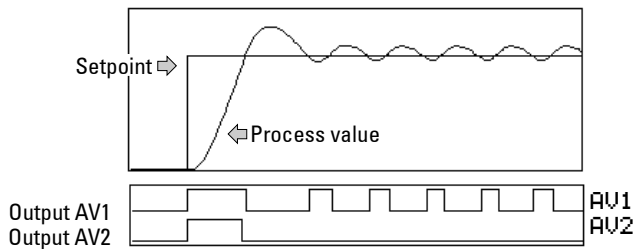


Abb. 599

The advantage is in the low switching frequency. Switch-on is always at a fixed value below setpoint, whereas switch-off is always at a fixed point above setpoint.

The control variable oscillation band is determined as a result of :

$$X_0 = X_{max} \cdot \frac{T_u}{T_g} + X_{Sd} = V_{max} \cdot T_u + X_{Sd}$$

The signalling function provides alarm signalling, whereby the setpoint is the limit value. The trigger point is symmetrical to the setpoint ; the hysteresis X_{sd1} is adjustable.

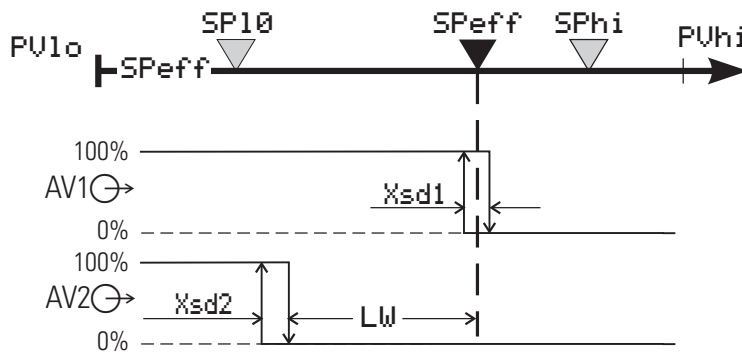


Abb. 600

Static operating principle of the signalling function
Signaller, 2 outputs

LW is shown as a negative value in the example (e.g. -20)

The signaller with two outputs has an additional "limit contact". Its difference from the is adjustable in parameter LW (including polarity sign).

| Configuration | Effective controller parameters of a signaller with two outputs | | |
|---------------------------------------|---|--|-------------------------------------|
| CFunc = signaller, 2 outputs | SP1o ¹⁾ | Lower limit for Weff | -29 999 ... 999 999 |
| | SP1i ¹⁾ | Upper limit for Weff | -29 999 ... 999 999 |
| | SP2 ¹⁾ | Additional | -29 999 ... 999 999 |
| | GrwP ²⁾ | gradient plus | off / 0,001 ... 999 999 |
| | GrwM ²⁾ | gradient minus | off / 0,001 ... 999 999 |
| | Grw2 ²⁾ | gradient for SP2 | off / 0,001 ... 999 999 |
| | NØ | Zero offset (only effective with CType=ratio controller) | -29 999 ... 999 999 |
| | A | Factor a (only effective with CType=3element control) | -9,99 ... 99,99 |
| | Xsd1 ¹⁾ | Signaller switching difference | 0,1 ... 999999 |
| | LW | Trigger point separation of additional contact OFF = the additional contact is switched off | -29 999 ... 999 999-32 000 = OFF |
| | Xsd2 ¹⁾ | Switching difference of additional contact | 0,1 ... 999 999 |
| | Title | Controller page title (only display) | 16 characters |
| | Unit.PV | Unit of the process value (only display) | 6 characters |

- 1) The values are specified in the process value unit - e.g. [°C, °F, bar, %, etc.]
- 2) The rate of change must be specified in units / minute (e.g. °C/min) → see gradient control.

Two-point controller

Switching controller with two switching statuses:

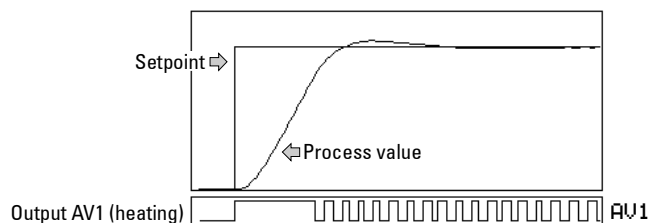
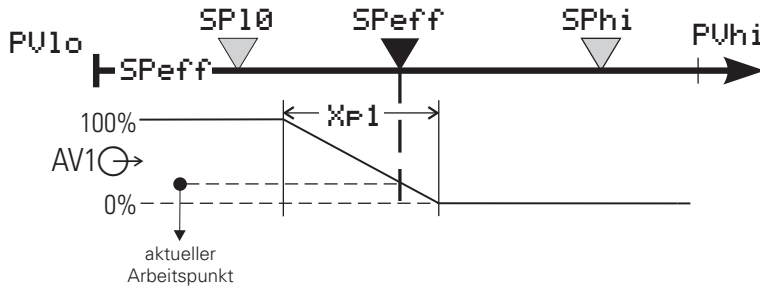


Abb. 601

1. Heating switched on; → output AV1 = 1
 2. Heating switched off; → output AV1 = 0
- E.g. for temperature control with electrical heating (inverse operation) or cooling (direct operation)

Adjust cycle time Tp1 as follows:

With higher Tp1, oscillations must be expected. Tp1 corresponds to the minimum cycle time (time in seconds) at 50 % duty cycle.



Static operating principle of a two-point controller

Abb. 602

PD action

($T_n = 0 =$ switched off $T_n = \infty$)

The working point is in the middle of proportional band X_{p1} at 50 % duty cycle. In order to keep the control variable constant, a defined quantity of energy dependent of is required. This energy causes a permanent control deviation, which increases with growing X_{p1} .

DPID action

By means of the I action, line-out is without permanent control deviation.

The static characteristic of a two-point controller is identical to the one of a continuous controller, with the difference that a duty cycle instead of a linearly variable current signal is output (relay contact, logic signal 0/20mA or control output 0/24V).

Working point AV0 and cycle time T_{p1} at 50% are adjustable.

The shortest switch-on or switch-off time is the cycle time of the function block, that is the cycle time of the task the controller runs in.

| Configuration | Effective controller parameters of a two-point controller | |
|--------------------------|---|---|
| CFunc = 2-point | PoPt | Parameter set for self-tuning (only with CONTROLP) |
| | SPlo ¹⁾ | Lower limit for SPeff |
| | SPhi ¹⁾ | Upper limit for SPeff |
| | SP2 ¹⁾ | Additional setpoint |
| | GrwP ²⁾ | Setpoint gradient plus |
| | GrwM ²⁾ | Setpoint gradient minus |
| | Grw2 ²⁾ | Setpoint gradient for SP2 |
| | N0 | Zero offset (only effective with CType=ratio controller) |
| | A | Factor a (only effective with CType=3-element control) |
| | AV2 | Additional correcting variable |
| | AVlo | Min. correcting variable limiting |
| | AVhi | Max. correcting variable limiting |
| | AV0 | Correcting variable working point(start-up correcting variable) |
| | AVOptm | Correcting variable during process at rest (not with PIDMA) |
| | dAVOpt | Self-tuning step change height |
| | Xp1(1...6) ³⁾⁴⁾ | Proportional band 1 |
| Tn1(1...6) ⁴⁾ | Integral action time | |
| Tv1(1...6) ⁴⁾ | Derivative action time | |

| | |
|--------------------------------|--|
| TF1(1...6)⁴⁾ | Cycle time heating |
| Title | Title of controller page (only display) |
| Unit PU | Unit of the process value (only display) |

- 1) The values are specified in the process value units - e.g. [°C, °F, bar, %, etc.]
- 2) Specify the rate of change in units / minute (e.g. °C/min)→ see gradient control
- 3) % specifications are related to measuring range **PV_{hi} - PV_{lo}** .
They are not related to values **SP_{lo}** and **SP_{hi}** .
- 4) (1...6) refers to the six parameter sets of CONTROLP (e.g. Xp1, Xp2, Xp3...Xp6).

Three-point controller

Switching controller with three switching statuses:

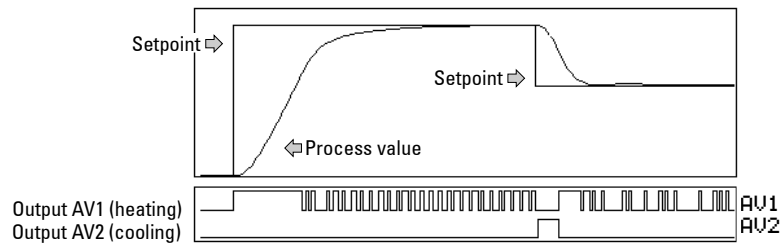


Abb. 603

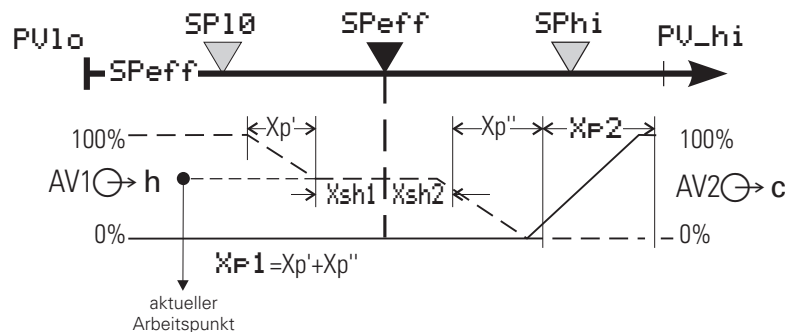
1. Heating switched on; → output AV1 = 1, AV2 = 0
2. Heating and cooling switched off; → outputs AV 1 = 0, AV 2 = 0
3. Cooling switched on; → outputs AV 1 = 0, AV 2 = 1

E.g. for temperature control with electrical heating (h) and cooling (c).

Adjust cycle time **TF1** and **TF2** as follows:

$$Tp1 \leq 0,25 \cdot Tu \text{ (h)} \quad Tp2 \leq 0,25 \cdot Tu \text{ (c)}$$

With higher **TF1/TF2**, oscillations have to be expected. Cycle times **TF1** and **TF2** are the minimum cycle times at 50% duty cycle.



Static operating principle of a three-point controller

604

Abb.

PD/PD action

($T_n = 0 =$ switched off $T_n = \hat{1}$)

The positioning range reaches from 100 % heating (AV1) to 100 % cooling (AV2). The proportional bands must be adapted to the various heating and cooling power values. In order to keep the control variable constant, a defined amount of energy dependent of is required. This causes a permanent control deviation, which increases with growing $X_p(1,2)$.

DPID/DPID action

By means of the I action, line-out without permanent control deviation is possible. Transition from trigger point 1 (heating) to trigger point 2 (cooling) is without neutral zone. The proportional bands must be adapted to the various heating and cooling power values.

Direct/inverse switchover only causes exchanging of the outputs for "heating/cooling". Expressions "heating" and "cooling" may also mean similar processes (dosing acid/lye, ...). The neutral zone is adjustable separately for the trigger points (Xsh1, Xsh2) i.e. it need not be symmetrical to the .

The type of positioning signals is selectable:

- CFunc = 3-point heating switching, cooling switching
- CFunc = cont/switch heating continuous, cooling switching
- CFunc = switch/cont heating switching, cooling continuous

Combination "heating continuous" and "cooling continuous" is covered by "splitRange - continuous controller with split-range behaviour". r see also "continuous controller".

| Configuration | Effective controller parameters with three-point controller | | |
|-----------------|---|--|------------------------|
| CFunc = 3-point | PofT | Parameter set for self-tuning (only with CONTROLP) | 1...6 |
| | SPlo ¹⁾ | Lower limit for SPeff | -29 999 ...999 999 |
| | SPhi ¹⁾ | Upper limit for SPeff | -29 999 ...999 999 |
| | SP2 ¹⁾ | Additional setpoint | -29 999 ...999 999 |
| | GrwP ²⁾ | Setpoint gradient plus | aus / 0,001 ...999 999 |
| | GrwM ²⁾ | Setpoint gradient minus | aus / 0,001 ...999 999 |
| | Grw2 ²⁾ | Setpoint gradient for SP2 | aus / 0,001 ...999 999 |
| | NØ | Zero offset (effective only with CType=ratio controller) | -29 999 ...999 999 |
| | A | Factor a (effective only with CType=3-element control) | -9,99 ... 99,99 |
| | Xsh1 ³⁾ | Neutral zone (DV > 0) | 0,0 ... 1000 [%] |
| | Xsh2 ³⁾ | Neutral zone (DV < 0) | 0,0 ... 1000 [%] |
| | AV2 | Additional positioning value | 0 ... 100 [%] |
| | AVlo | Min. output limiting | 0 ... 100 [%], >AVhi |
| | AVhi | Max. output limiting | 0 ... 100 [%], <AVlo |
| AVØ | Correcting variable working point (start-up corr. variable) | 0...100 [%] | |

| | | |
|----------------------------------|--|-------------------|
| AVOftm | Positioning value during process at rest | 0...100 [%] |
| dAVOft | Step height during self-tuning | 5...100 [%] |
| XP1(1...6)³⁾⁵⁾ | Proportional band 1 | 0,1 ... 999,9 [%] |
| XP2(1...6)³⁾⁵⁾ | Proportional band 2 | 0,1 ... 999,9 [%] |
| Tn1(1...6) | Integral action time | 0 ... 999 999 [s] |
| Tv1(1...6) | Derivative action time | 0 ... 999 999 [s] |
| TF1(1...6)⁵⁾ | Cycle time heating | 0,4 ... 999,9 [s] |
| TF2(1...6)⁵⁾ | Cycle time cooling | 0,4 ... 999,9 [s] |
| Title | Title of controller page (only display) | 16 characters |
| Unit PU | Unit of process value (only display) | 6 characters |

- 1) The values are specified in the process value unit - e.g. [°C, °F, bar, %, etc.]
- 2) The rate of change must be specified in units/minute (e.g. °C/min)→see gradient control.
- 3) % specifications are related to measuring range **PVhi - PVlo**.
There is no relation to values **SPlo** and **SPhi**.
- 4) As default, value **AVlo** is set to 0. In this case, output **AV1** cannot switch!
- 5) (1...6) refers to the six parameter sets of CONTROLP (e.g. Xp1, Xp2, Xp3...Xp6).

$\Delta / Y / off$

The principle is identical to the control behaviour of a 2-point controller with additional contact.

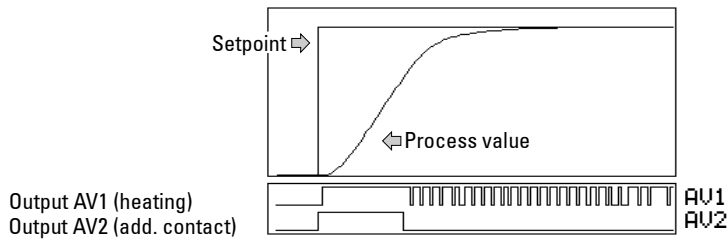
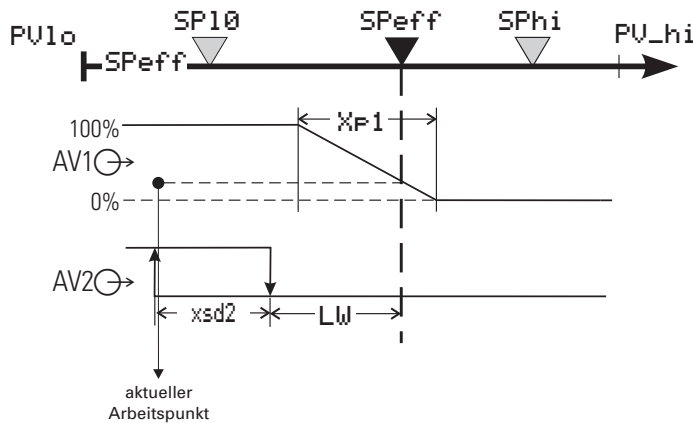


Abb. 605

Output **AV2** is used for switchover of the connected circuit between " Δ " and "Y". Output **AV1** switches the heating energy on and off.

E.g. for temperature control with electrical heating (inverse operation) or cooling (direct operation).

Cycle time T_{p1} must be adjusted as follows: With higher T_{p1} , oscillations must be expected. T_{p1} corresponds to the minimum cycle time (time in seconds) at 50 % duty cycle.



Static operating principle of the $\Delta / Y / off$ function

Abb. 606

PD action

($T_n = 0 =$ switched off $T_n = \infty$)

The working point is in the middle of the proportional band X_{p1} at 50 % duty cycle.

For keeping the control variable constant, a defined amount of energy dependent of is required. This causes a permanent control deviation, which increases with higher X_{p1} .

DPID action

By means of the I action, line-out without permanent control deviation is possible.

The static characteristic of a two-point controller is identical to the one of a continuous controller. The difference is that a duty cycle instead of a linearly variable current signal is output (relay contact, logic signal 0/20mA or control output 0/24V).

Working point $AV0$ and cycle time T_{p1} of the cycle at 50% are adjustable.

The shortest switch-on or switch-off time is the cycle time of the function block, that is the cycle time of the task the controller runs in.

Configuration | Effective controller parameters with D / Y / off controller

| | | | |
|---------------------------------|---|--|--------------------------------------|
| CFunc = 2- P+add. contact | POft | Parameter set for self-tuning (only with CONTROLP) | 1...6 |
| | SPlo³⁾ | Lower limit for SPEff | -29 999 ...999 999 |
| | SPhi¹⁾ | Upper limit for SPEff | -29 999 ...999 999 |
| | SP2¹⁾ | Additional setpoint | -29 999 ...999 999 |
| | GrwP²⁾ | Setpoint gradient plus | off / 0,001 ... 999 999 |
| | GrwM²⁾ | Setpoint gradient minus | off / 0,001 ... 999 999 |
| | Grw2²⁾ | Setpoint gradient for SP2 | off / 0,001 ... 999 999 |
| | NØ | Zero offset (effective only with CType=ratio controller) | -29 999 ...999 999 |
| | A | Factor a (effective only with CType=3-element control) | -9,99 ... 99,99 |
| | LW¹⁾ | Trigger point separation of additional contact OFF = the additional contact is switched off | -29 999 ... 999 999 -32 000 = OFF |
| | Xsd2¹⁾ | Switching difference of additional contact | 0,1 ... 999 999 |
| | AV2 | Additional positioning value | 0 ... 100 [%] |
| | AVlo | Min. output limiting | 0 ... 100 [%], >AVhi |
| | AVhi | Max. output limiting | 0 ... 100 [%], <AVlo |
| | AVØ | Working point of correcting variable (start-up correcting variable) | 0...100 [%] |
| | AVOfm | Positioning value during process at rest | 0...100 [%] |
| | dAVoft | Self-tuning step height | 5...100 [%] |
| | XP1(1...6)³⁾⁴⁾ | Proportional band 1 | 0,1 ... 999,9 [%] |
| | Tn1(1...6)⁴⁾ | Integral time | 0 ... 999 999 [s] |
| | Tv1(1...6)⁴⁾ | Derivative time | 0 ... 999 999 [s] |
| Tf1(1...6)⁴⁾ | Cycle time heating | 0,4 ... 999,9 [s] | |
| Title | Title of controller page (only display) | 16 characters | |
| Unit PV | Unit of process value (only display) | 6 characters | |

- 1) The values are specified in the process value unit - e.g. [°C, °F, bar, %, etc.]
- 2) The rate of change must be specified in units/minute (e.g. °C/min)→ see gradient control.
- 3) % specifications are related to measuring range **PVhi** - **PVlo**.
There is no relation to values **SPlo** and **SPhi**.
- 4) (1...6) refers to the six parameter sets of CONTROLP (e.g. Xp1, Xp2, Xp3...Xp6).

Three-point stepping controller

Switching controller for control of a valve (e.g. temperature control by means of motorized valve and gas-air mixture)

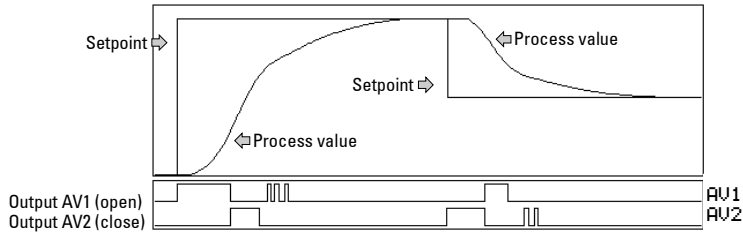
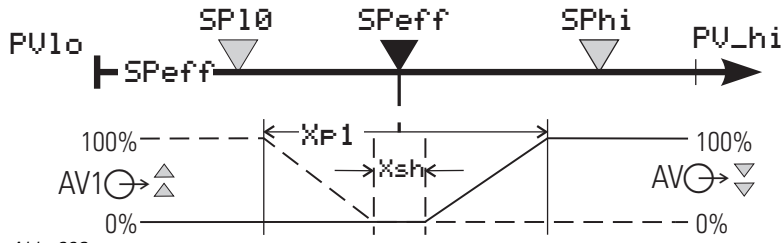


Abb. 607

- 1. Open valve; → outputs AV1 = 1, AV2 = 0
- 2. Don't move valve; → outputs AV1 = 0, AV2 = 0
- 3. Close valve; → outputs AV1 = 0, AV2 = 1

To validate the adjusted $Xp1$ for the actuator response time, response time Tm must be adjusted. The smallest positioning step is the cycle time of the function block, that is the cycle time of the task the controller runs in.

HINT!
 With controllers CONTROL and CONTROLP the position feedback has no influence on the PID-behaviour! (With controller PIDMA it has influence.)



Static operating principle of a three-point stepping controller

Abb. 608

Adjusting the neutral zone

Neutral zone X_{sh} can be increased in case of excessive output switching. However, note that an increase the neutral zone will reduce the control sensitivity.

For this reason, we recommend optimizing switching frequency (actuator wear) and control sensitivity.

Three-point stepping controllers can be operated with or without position feedback PF .

- Stepping** 3-point stepping controller
- Stepp.+PF** 3-point stepping controller with position feedback

Whereby PF is not used for control in CONTROL an CONTROLP (in PIDMA it is used). The static characteristic of a three-point stepping controller is shown in the figure above.

The hysteresis shown in this diagram is practically unimportant, but can be calculated from the adjustable min. pulse length $T_{puls} \approx 100ms$.

$$X_{sh} = \left(\frac{Tpuls}{2} - 0,5 * Ts \right) \cdot \frac{X_p}{Tm}$$

**HINT!**

With **TPULS** switched off, the shortest positioning step **TPULS** is dependent of **Tm**, **Xsh** and **Xp**. By varying **Xsh**, a required minimum pulse length **TPULS** can be realized:

$$X_{sh} = 12,5 \cdot Xp \cdot \frac{Tpuls}{Tm} - 0,75$$

| Configuration | Effective controller parameters with three-point stepping controller | | |
|--------------------------------|--|--|-------------------------|
| CFunc = stepp stepp + PV | Popt | Parameter set for self-tuning (only with CONTROLP) | 1...6 |
| | SPlo ¹⁾ | Lower limit for SPeff | -29 999 ...999 999 |
| | SPhi ¹⁾ | Upper limit for SPeff | -29 999 ...999 999 |
| | SP2 ¹⁾ | Additional setpoint | -29 999 ...999 999 |
| | GrwP ²⁾ | Setpoint gradient plus | aus / 0,001 ... 999 999 |
| | GrwM ²⁾ | Setpoint gradient minus | aus / 0,001 ... 999 999 |
| | Grw2 ²⁾ | Setpoint gradient for SP2 | aus / 0,001 ... 999 999 |
| | N0 | Zero offset (effective only with CType=ratio controller) | -29 999 ...999 999 |
| | A | Factor a (effective only with CType=3-element control) | -9,99 ... 99,99 |
| | Xsh ³⁾ | Trigger point separation | 0,2 ... 20 [%] |
| | TPuls | Min. positioning step time | 0,1 ... 2 [s] |
| | Tm | Actuator response time | 5 ... 999 999 [s] |
| | AV2 | Additional positioning value (only with step PF →with position feedback) | 0 ... 100 [%] |
| | AVOftm | Positioning value during process at rest | 0...100 [%] |
| | dAVOft | Self-tuning step height | 5...100 [%] |
| | Xp1(1...6) ³⁾⁴⁾ | Proportional band 1 | 0,1 ... 999,9 [%] |
| | Tn1(1...6) ⁴⁾ | Integral action time | 0 ... 999 999 [s] |
| Tv1(1...6) ⁴⁾ | Derivative action time | 0 ... 999 999 [s] | |
| Title | Title of controller page (only display) | 16 characters | |
| Unit PV | Unit of the process value (only display) | 6 characters | |

- 1) The values must be specified in the process value unit - e.g. [°C, °F, bar, %, etc.]
- 2) The rate of change must be specified in units/minute (e.g. °C/min).
r see gradient control page .
- 3) % specifications are related to measuring range **PVhi - PVlo** .
There is no relationship to values **SPlo** and **SPhi** .
- 4) (1...6) refers to the six parameter sets of CONTROLP (e.g. Xp1, Xp2, Xp3...Xp6).

Continuous controller / Split range

Continuous controller

An analog value is provided as correcting variable by output **AVout1**, e.g. temperature control with electrical heating and thyristor power regulator.

A continuous controller in 'split-range' operation is comparable with a three-point controller. The neutral zone is also separately adjustable.

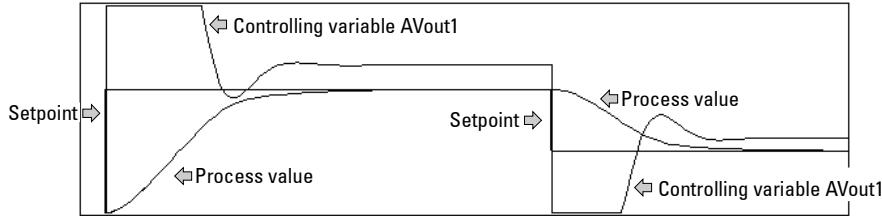
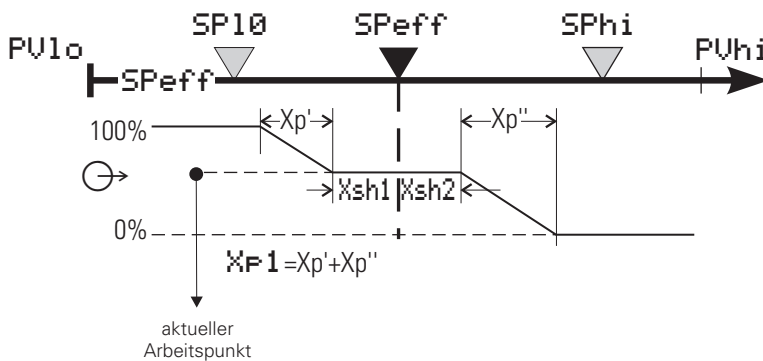


Abb. 609

Within limits Xsh1 and Xsh2, the control deviation for calculation of the controller reaction is set to zero. A pure P controller does not change the correcting variable within these limits. A PID controller has a dynamic behaviour which has not always decayed, also when reaching "control deviation = 0". Both D and I action can still have an effect according to the characteristic determined by Tv due to a preceding disturbance or a setpoint step change. This effect can be strong enough to cause the control deviation to leave range Xsh1/Xsh2, i.e. the P action is activated again for reaching the neutral zone.



Operating principle of the proportional part of the continuous controller

Abb. 610

Selection from the following continuous controllers is possible:

1. **CFunc** = continuous → continuous controller.
2. **CFunc** = splitRang → continuous controller with split-range operation. The continuous output is split on outputs AVout1 and AVout2 .
3. **CFunc** = continuous PF → continuous controller with position feedback PF. The actually flowing positioning current can be displayed via input PF. PF is not included in the control operation for CONTROL and CONTROLP (for PIDMA it is included).

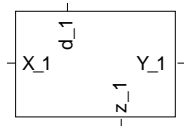
| Configuration | Effective controller parameters of a continuous controller | | |
|-----------------------------------|--|--|--------------------|
| CFunc = Cont. SplitRange | Port | Parameter set for self-tuning (only with CONTROLP) | 1...6 |
| | SPlo ¹⁾ | Lower limit for SPeff | -29 999 ...999 999 |
| | SPHi ¹⁾ | Upper limit for SPeff | -29 999 ...999 999 |
| | SP2 ¹⁾ | Additional setpoint | -29 999 ...999 999 |

| | | |
|----------------------------------|---|--|
| GrwP²⁾ | Setpoint gradient plus | aus / 0,001 ... 999 999 |
| GrwM²⁾ | Setpoint gradient minus | aus / 0,001 ... 999 999 |
| Grw2²⁾ | Setpoint gradient for SP2 | aus / 0,001 ... 999 999 |
| N0 | Zero offset (effective only with CType=ratio controller) | -29 999 ...999 999 |
| A | Factor a (effective only with CType=3-element control) | -9,99 ... 99,99 |
| Xsh1³⁾ | Neutral zone (DV > 0) | 0,0 ... 1000 [%] |
| Xsh2³⁾ | Neutral zone (DV < 0) | 0,0 ... 1000 [%] |
| AV2 | Additional positioning value | 0 ... 100 [%] |
| AVlo | Min. output limiting | (-100) 0 ... 100 [%], < AVhi |
| AVhi | Max. output limiting | (-100) 0 ... 100 [%] > AVlo |
| AV0 | Correcting variable working point (start-up correcting variable) | -100...100 [%] |
| AVOptm | Positioning value during process at rest | 0...100 [%] |
| dAVopt | Self-tuning step height | 5...100 [%] |
| XP1(1...6)³⁾⁴⁾ | Proportional band 1 | 0,1 ... 999,9 [%] |
| XP2(1...6)³⁾⁴⁾ | Proportional band 2 (only with continuous controller split range) | 0,1 ... 999,9 [%] |
| Tn1(1...6)⁴⁾ | Integral action time | 0 ... 999 999 [s] |
| Tv1(1...6)⁴⁾ | Derivative action time | 0 ... 999 999 [s] |
| Title | Title of controller page (only display) | 16 characters |
| Unit.PV | Process value unit (only display) | 6 characters |

- 1) The values must be specified in the process value unit, e.g. [°C, °F, bar, %, etc.]
- 2) The rate of change must be specified in units/minute e.g. °C/min).→ see gradient control.
- 3) % specifications are related to measuring range **PVhi** - **PVlo** .
There is no relationship to values **SPlo** and **SPhi** .
- 4) (1..6) refers to the six parameter sets of CONTROLP (e.g. Xp1, Xp2, Xp3...Xp6).

III-16 Process output

III-16.1 OUT (process output (No. 116))



OUT

Abb. 611

Function **OUT** is used to process the output signal. Switching signals are converted into continuous signals with range 0 ... 100%. Continuous signals are converted into switching signals.

Connect output Bl_no to the function block which does the communication to the hardware. That is e.g. **KS108_A0_1_SF** or **KS108_A0_2_U_BP**.

In-/Outputs

| Name | Type | Description |
|------|-------|--|
| X_1 | Float | Analog source signal to be output |
| d_1 | Bool | Digital source 0 > 0 and 1 > 100 signal to be output |
| Name | Type | Description |
| Y_1 | Float | Determined analog output signal in % |
| z_1 | Bool | Determined digitized analog output signal 0 = 49.5 %; 1 = 50.5 % |

Configuration:

| ID | Name | Type | Description | Access | Default | Range | Off |
|------|------------------|---------------------|---|--------|---------|-------|-----|
| Src | Signal source | Enum | Signal source to be used | r/w | 0 | | |
| | | Digital | The input signal comes from digital input d1. | | 0 | | |
| | | Analog | The input signal comes from analog input X1. | | 1 | | |
| Mode | Output direction | Enum | Actuator output action | r/w | 0 | | |
| | | Direct | Direct / normally open. For an increased input signal, the output signal must be increased as well. | | 0 | | |
| | | Inverse | Inverse / normally closed. For an increased input signal, the output signal is reduced. | | 1 | | |
| Type | Behaviour | Enum | Function of continuous output | r/w | 0 | | |
| | | Continuous 0...100% | Continuous 0 - 100%. The output is varied continuously between 0% and | | 0 | | |

| | | | | | | | |
|------|-------------|-------------------|--|-----|-------|--|--|
| | | | 100%. | | | | |
| | | Logic 0 / 100% | Logic 0/100%. The output is switched between the 0% and 100% values. | | 1 | | |
| X0 | Scale start | Float | Analog input value for 0 % | r/w | 0.0 | | |
| X100 | Scale end | Float | Analog input value for 100 % | r/w | 100.0 | | |

III-17 Vario I/O modules

A vario I/O system comprises a bus coupler and one or several vario I/O modules. A vario I/O system with the required internal electrical connections is set up by connecting the vario I/O modules to the bus coupler.

In the engineering, the vario I/O system is shown in the form of function blocks according to the hardware.

All Vario I/O function blocks have a **Slot** output that must be soft-wired to a **Slot** input of the bus coupler function block. The position number of the related hardware module determines the number of the bus coupler input.

Survey of vario I/O modules and related function blocks:

| Function block | Hardware |
|---|-----------------|
| AI_2_SF | VARIO AI 2/SF |
| AI_8_SF | VARIO AI 8/SF |
| AO_1_SF | VARIO AO 1/SF |
| AO_2_U_BP | VARIO AO 2/4/BP |
| DI_2 | VARIO DI 2/24 |
| DI_4 | VARIO DI 4/24 |
| DI_8 | VARIO DI 8/24 |
| DI_16 | VARIO DI 16/24 |
| DO_2 | VARIO DO 2/24 |
| DO_4 | VARIO DO 4/24 |
| DO_8 | VARIO DO 8/24 |
| DO_16 | VARIO DO 16/24 |
| DO_1_230 | VARIO DO 1/230 |
| DO_4_230 | VARIO DO 4/230 |
| RTD_2 | VARIO RTD 2 |
| RTD_6 | VARIO RTD 6-DO6 |
| RTD_6_HC (when using the heating current functionality) | VARIO RTD 6-DO6 |
| UTH_2 | VARIO UTH 2 |
| UTH_4 | VARIO UTH 4-DO8 |
| UTH_4_HC (when using the heating current functionality) | VARIO UTH 4-DO8 |
| UTH_8 | VARIO UTH 8-DO8 |
| UTH_8_HC (when using the heating current functionality) | VARIO UTH 8-DO8 |
| VARIO_BK_ETH | VARIO BK ETH |

III-17.1 Short-form instructions for building up a vario I/O system: hardware



DANGER

The VARIO station can work with voltage ranges up to 230V. Touching these live contacts can cause harm to human health and even life-threatening injury.

Failure to take the VARIO station into operation correctly may cause injury to persons, and material damage.

For this reason:

- Follow the safety hints and the detailed instructions for the vario I/O system !!!



CAUTION

Don't change modules under voltage.

Removing a module from the station, or inserting a module into the station without disconnecting the supply voltage may cause the destruction of the module.

For this reason:

- Before removing a module from the station, or before inserting a module into the station, disconnect the overall station from the supply voltage. Switch on the supply voltage only after building up the complete station.

1. Installing the Vario I/O system on the top-hat rail:

Shift the modules of the vario I/O system together snap them in position on the top-hat rail.

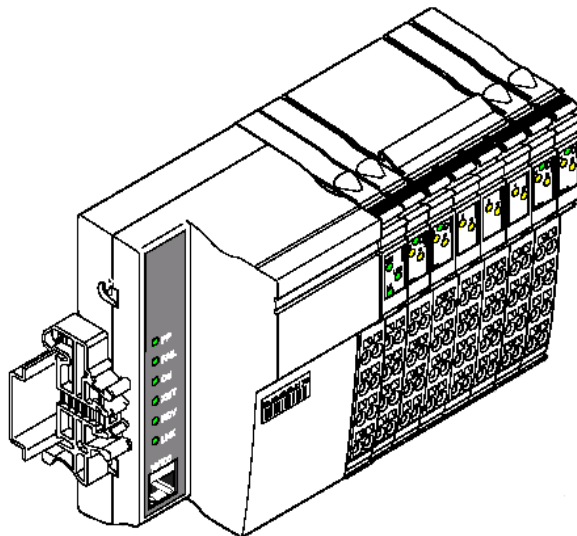


Abb. 612: vario I/O system with I/O modules on a top-hat rail

To install a module, proceed as follows:

- First of all, fit the electronics sockets required for composing the station on the rail in vertical position (Fig. A). Make sure that all grooves and tongues of adjacent modules are joined perfectly (Fig. B).
- Then, plug the connectors into the corresponding sockets. Insert the front catch of the connector into the front locking mechanism (Fig. C) and press the connector towards the socket, until it is firmly seated in the back locking mechanism (Fig. D). The connector is not provided with grooves like the electronics socket. In order to snap an electronics socket into position, the space on its left side must be free. If there is a connector on the left side, it must be removed.

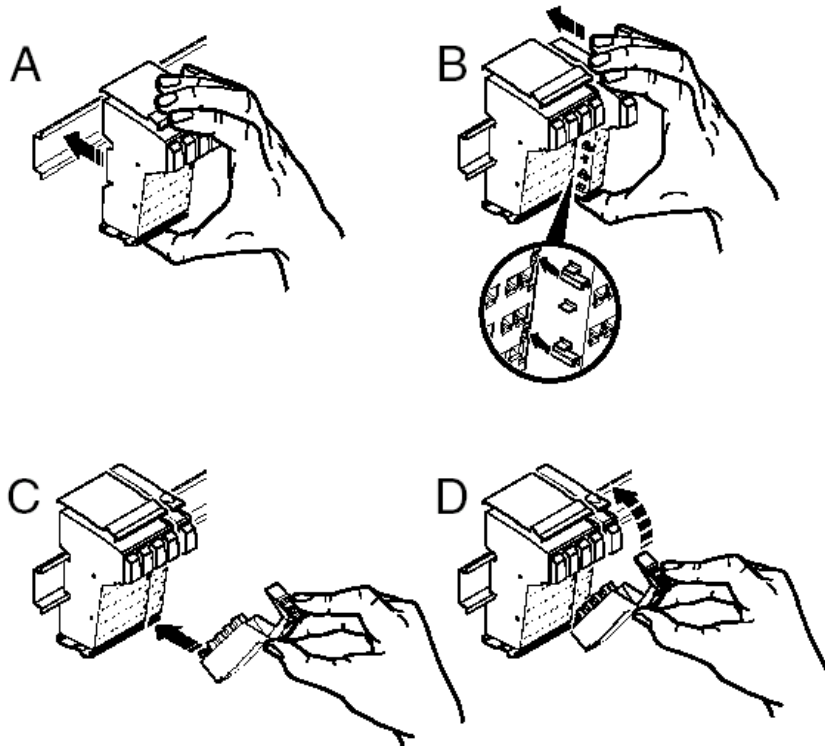


Abb. 613: Mounting the vario I/O system

2. Completing the system.

Fit end stops on both sides of the VARIO station. The end stops ensure correct locking of a VARIO station on the top-hat rail and serve as side terminations.



DANGER

An Ethernet VARIO station must be terminated mechanically by a plate!

Hazardous contact voltages may cause considerable injury to persons, and material damage.

For this reason:

- Fit a plate as a mechanical termination of an Ethernet Vario station.

3. Wiring procedure:

Connect the protective earth. Connect the screening. Connect the supply voltage. Connect the bus.

Recommendation: Take the VARIO station into operation on the bus only, when the program for determination of the IP address is prepared (see below: VarioConfigurator).

For additional information : see VARIO Ethernet projecting manual.

III-17.2 Short-form instructions for building up a VARIO I/O system: HW configuration



NOTE

Follow the safety hints and the detailed instructions for the VARIO I/O system.

For additional information, see the "VARIO Ethernet projecting manual".

To enable KS108 to access the VARIO station, the VARIO station IP address must be specified in the engineering. Its IP address must be known and independent of a restart. After delivery, the Vario station does

not have an IP address. The IP address can be defined automatically via a DHCP server or entered manually, e.g. via a direct Ethernet connection.

DHCP server: How to use a DHCP server is not described in detail in this operating manual. Due to the differences between the systems and programs, we recommend consulting the network administrator in case of doubt. Make sure that the VARIO station has always the same IP address. This setting can be made in the DHCP server defining the fixed IP address dependent on the MAC address.

Direct Ethernet connection:

4. Building up a direct Ethernet connection (IP address setting without DHCP server):

Preparing the PC:

If the PC is used in a network with DHCP server, i.e. it normally receives its IP address automatically, the settings for the LAN connection must be changed. The LAN access must be given a different network mask and changed to a fixed IP address.

Example:

Under Windows 2000, the LAN is accessible under **"Control Panel" ... "dial-up modem connection" ... "Local Area Connection"**. Window **Status** appears with a **"Properties"** button that can be clicked for access to window **Local Area Connection properties**. Under the **"General"** tab, you will find a list of the components. Here, line **"Internet protocol (TCP/IP)"** is important. NOTE THE SETTINGS, in order to be able to reproduce them subsequently. For the direct Ethernet connection to the VARIO station, option **"Internet protocol (TCP/IP)"** must be checked and selected. The **"Properties"** button can be clicked to display window **Internet protocol (TCP/IP) Properties**. In this window, select radio button **"use the following IP address"**. Now, a valid **IP address** for the PC, a **subnet mask** and the gateway can be filled in, if required.

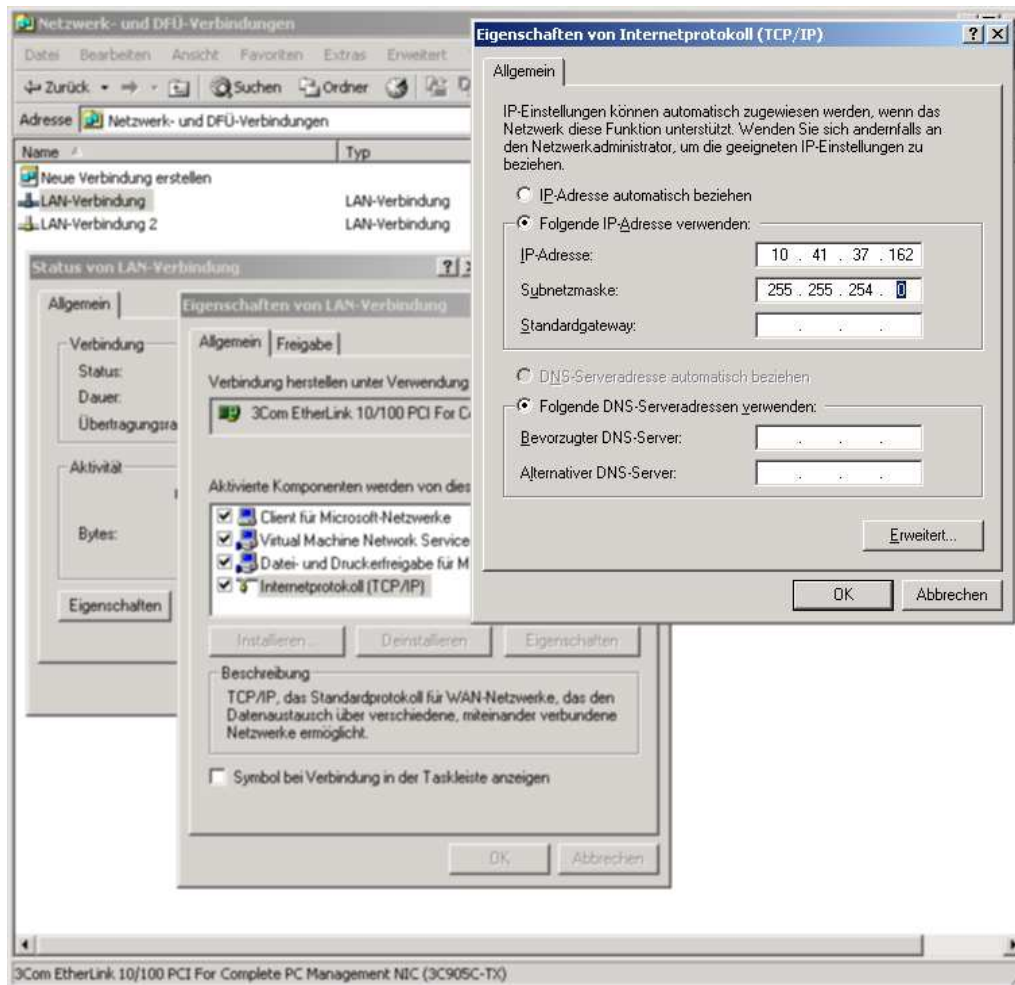


Abb. 614: Change-over of the PC to direct ethernet-connection to the VARIO-station

Subsequently, connect the computer with the VARIO station using a standard Ethernet cable.



NOTE

Switch on the VARIO station only, when you have prepared settings for the IP address and the network mask in the VarioConfigurator (or in the DHCP server, if you use it).

5. Taking the VARIO station into operation: Mostly, the IP address is determined via the PC. For this purpose, the VarioConfigurator program can be used, which is always required for configuration of the VARIO station. Alternatively, the IP address can be determined also via any DHCP server.

- After delivery, the VARIO Ethernet bus coupler has neither a valid IP address nor a network mask. This is the reason why the unit continuously sends BootP requests, until a valid IP address is received. If valid IP parameters are received, they are saved as configuration data by the instrument.
- (Subsequent starting: If the instrument has already valid configuration data, it sends only three BootP requests when taken into operation again with activated **Bootp**. When it receives a BootP reply, the new parameters are stored. If the instrument does not receive a reply, it starts with the last configuration. If Bootp is switched off, the instrument starts directly with the entered address.)

**NOTE**

If there is a DHCP server in the network, the right address must be assigned either via the DHCP server, or BOOTP must be de-activated! Otherwise, the VARIO station uses the (random) IP address assigned by the DHCP server and can be addressed only using this address. (See below: VarioConfigurator).

6. Setting the IP address using the VarioConfigurator, the program for configuring the VARIO station: If the PC is connected directly with the VARIO station and the VarioConfigurator program is started the IP address can be assigned. This setting is required to access the VARIO station in a network.

**NOTE**

The VarioConfigurator program is supplied with a help function in which the required steps are described in clear, concise form.

1. Select the bus coupler from the right column in the program and click to adopt it.
2. Select "Tools" ... "IP Manager" to open the **UDP Bootp Server** window. Select tab "first IP setting". Specify the MAC address of the VARIO station. It is printed on a label on the top of the VARIO bus coupler. Enter this **MAC address** into the corresponding field, then fill in the selected **IP address** of the VARIO station (must be unique!) and the **Subnet mask**. Dependent on network, a gateway may have to be specified. Select option "**transmitt IP address**" to transfer the settings, as soon as the MAC address sends a Bootp reply.
3. Switch on the VARIO station or the bus coupler. Now the VARIO station opens window **UDP Bootp Server** and adopts the new IP address.

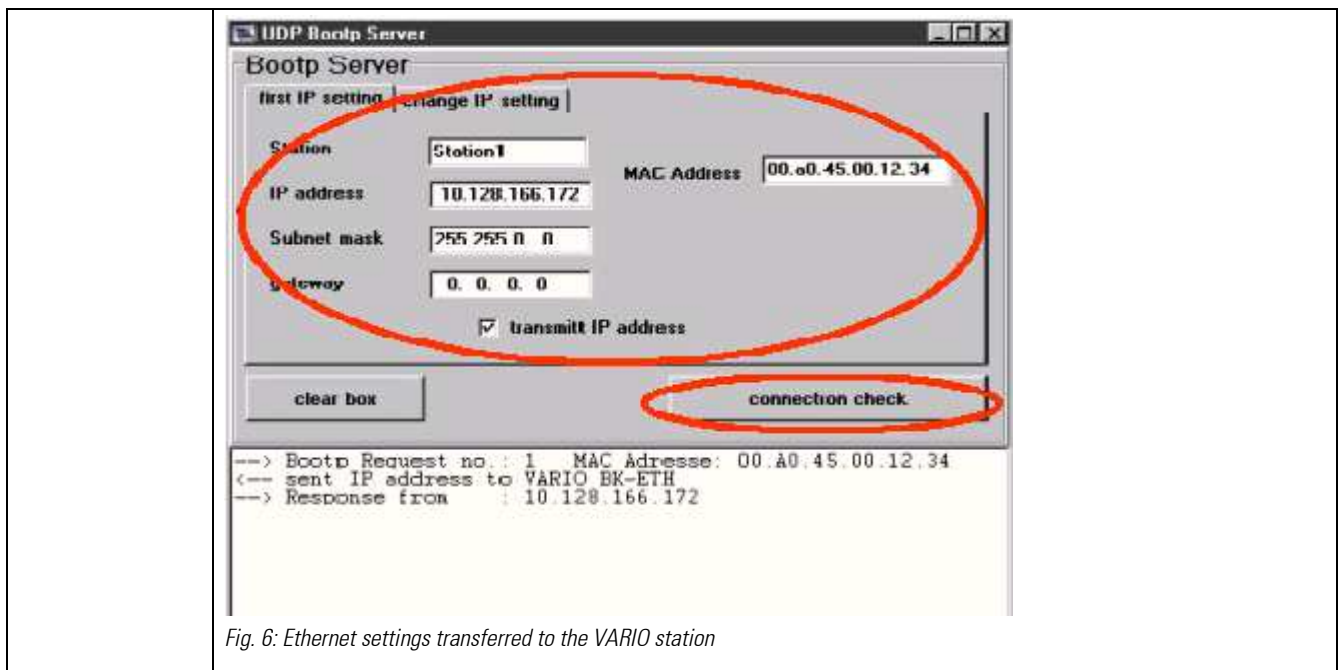


Fig. 6: Ethernet settings transferred to the VARIO station

4. Now, switch off the BootP under "Tools" ... "BK-ETH diagnosis" : Click "**BootP off**". Select "Online" ... "check config" to write the setting into the VARIO station. Explanation: When BootP is activated, an IP address can be assigned to the VARIO station. As DHCP or BootP servers are used frequently for management of the Ethernet network, the IP settings of the VARIO station could be changed automatically.
5. When the VARIO station with the right IP address has replied and if you have de-activated Bootp, you can cut the direct connection to the VARIO station and re-establish the LAN connection of your computer. Now you can use the VARIO station in your network.

4. **System configuration using the VarioConfigurator:** When the VARIO station is connected via Ethernet and the VarioConfigurator program is started, the VARIO station can be configured. The VARIO station requires the module configuration for self-diagnosis.

**NOTE**

The VarioConfigurator program is supplied with a help function in which the required steps are described in clear, concise form.

1. Configuring the system using the main screen: Select the bus coupler from the right column and click to adopt the choice. Proceed analogously to add the I/O modules.
2. Double click on the bus coupler to open the **properties BK ETH** window. Fill in the IP address and the subnet mask. If necessary, specify the gateway as well.
3. Click "Online" ... "check config" to save and check the configuration. In the event of an error, adapt the modules and their order in the VarioConfigurator to the VARIO station.
4. The I/O module parameters can be set and checked (debugged) here or later, in the engineering.

III-17.3 Short-form instructions for building up a VARIO I/O system: Software and debugging

**NOTE**

Follow the safety hints and the detailed instructions of the VARIO system.

5. **Creating or displaying an engineering:** Select the relevant function block for each module and put it into the engineering. Soft-wire the "Slot" module outputs to the right "Slot_(No.)" inputs of the bus coupler function block according to the module position number in the installed system. Remaining soft-wiring procedure: Inputs, outputs, error outputs. Set the bus coupler and module parameters during the "Run" mode.

**NOTE**

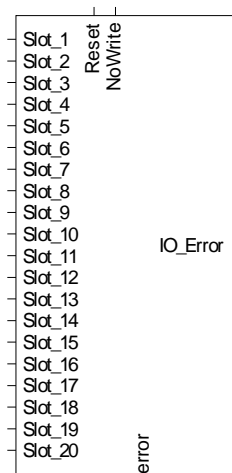
For correct functioning of the communication, the IP address in the engineering and in the bus coupler must be equal.

6. **Downloading the engineering:**
BlueDesign must be in the "Run" mode. Execute the following steps:
- "Load the user program".
 - Select the "target system": Check the settings".
 - "Connect". "OK". The user program is loaded into KS108 and activated. The on-line connection is built up.
7. **Checking the engineering functions:**
Determine the process values and signals. Set the outputs. Generate the alarms, etc.

**NOTE**

The configurations and parameters are not stored in the VARIO station. For this reason, the configurations and parameters must be downloaded after restarting.

III-17.4 VARIO_BK_ETH (Vario ethernet buscoupler (No. 150))



VARIO_BK_ETH

Abb. 615

The function block is used for communication with a Vario ethernet buscoupler module.

Up to 20 vario I/O modules can be soft-wired.

Inputs **Slot_1 - Slot_20** must be soft-wired with the **Slot** output of the used vario I/O function blocks, e.g. **AI_2_SF**, **AO_1_SF**, **RTD_6_HC** etc.

Coding of output IO_Error

| error | name | Description |
|-------|------------------|--|
| 0 | NO_ERROR | No error |
| 1 | NO_SOCKET | Initialisation error |
| 2 | NO_RESPONSE | No answer received from vario system during initialisation |
| 3 | FRAME_SIZE | Length of received answer is not correct. |
| 4 | NOT_ACKNOWLEDGED | For every message an ID is transmitted - the correct ID was not received in an answer message. |
| 5 | DATA_SIZE | The length of the received cyclic message was different from the specified length. |
| 6 | DATA_ERROR | Data loss occurred during normal communication |
| 7 | CONFIG_ERROR | Number or types of the modules found at the buscoupler is different from the KS1x8 project. |
| 8 | NO_CONFIG | In KS1x8 no modules are defined for the buscoupler . |
| 9 | BK_ERR | The from buskoppler received message cannot be interpreted |
| 700 | CONFIG_ERROR | The number of modules connected to the bus coupler does not match the engineering (bigger or smaller). |
| 7xy | CONFIG_ERROR | The module number xy connected to the bus coupler is different from the module in the engineering. |



NOTE

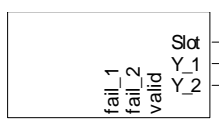
For additional information, see the operating instructions of the VARIO ethernet buscoupler module.

| In-/Outputs | | |
|--------------------|-------|-------------------------------|
| Name | Type | Description |
| Slot_1 ... Slot_20 | Float | 1st module on the bus system |
| Reset | Bool | Communication and error reset |
| NoWrite | Bool | Data are not written |

| Name | Type | Description |
|----------|-------|--|
| IO_Error | Float | Type of error of communication to the VARIO I/O system |
| error | Bool | Error of communication to the VARIO I/O system |

| Parameter | | | | | | | |
|-----------|------------|------|--|--------|---------|--------------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Timeout | Timeout | Int | Time-out for detection of failure of the vario system [ms] | r/w | 1000 | 10 ... 30000 | |
| IP_Adr | IP address | Text | IP address | r/w | 0.0.0.0 | | |

III-17.5 AI_2_SF (I/O-module with 2 analog inputs (No. 165))



AI_2_SF

Abb. 616

The function block is used for configuration and parameter setting of a module with 2 analog inputs for current or voltage. The sensors can be connected in 2 or 3-wire technology.

The conditioned measured values and the measured value status signals are available at the outputs of the function block.

Output **Slot** must be soft-wired with the input of a **KS108_VARIO_BK_ETH** type function block.



NOTE!

For additional information, see the operating instructions of the VARIO I/O module AI 2/SF.

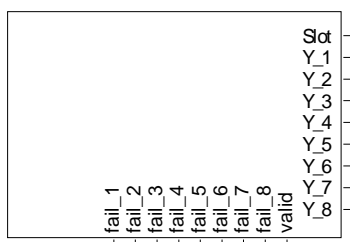
| In-/Outputs | | |
|-------------|-------|---|
| Name | Type | Description |
| Slot | Float | Own block number |
| Y_1 | Float | Conditioned measured value 1 |
| Y_2 | Float | Conditioned measured value 2 |
| fail_1 | Bool | Signals a sensor error at the input (short-circuit, wrong polarity, ..) |
| fail_2 | Bool | Signals a sensor error at the input (short-circuit, wrong polarity, ..) |
| valid | Bool | The communication to the VARIO I/O system (hardware) is established. Measured values are valid. |

| Parameter | | | | | | | |
|-----------|-------------------|-------|--|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| X0_1 | Scale start 1 | Float | Physical value at 0% | r/w | 0.0 | | |
| X100_1 | Scale end 1 | Float | Physical value at 100% | r/w | 100.0 | | |
| Xfail_1 | Substitute value1 | Float | Substitute value with sensor fail | r/w | 9999.0 | | |
| Tfm_1 | Filter time 1 [s] | Float | Filter time constant of signal processing in s | r/w | 0.5 | >0.0 | |
| X0_2 | Scale start 2 | Float | Physical value at 0% | r/w | 0.0 | | |
| X100_2 | Scale end 2 | Float | Physical value at 100% | r/w | 100.0 | | |
| Xfail_2 | Substitute value2 | Float | Substitute value with sensor fail | r/w | 9999.0 | | |
| Tfm_2 | Filter time 2 [s] | Float | Filter time constant of signal processing in s | r/w | 0.5 | >0.0 | |

| Configuration | | | | | | | |
|---------------|---------------|--------------|---|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Typ_1 | Sensor type 1 | Enum | Sensor type | r/w | 0 | | |
| | | Not used | The input is not used. | | 0 | | |
| | | 0...20 mA | Current : 0...20 mA | | 30 | | |
| | | -20...+20 mA | Current : -20...20 mA | | 31 | | |
| | | 4...20 mA | Current : 4...20 mA | | 32 | | |
| | | 0...10 V | Voltage : 0...10 V | | 42 | | |
| | | -10...+10V | Voltage : -10...10 V | | 43 | | |
| Fail_1 | Fail | Enum | Signal response on sensor fail | r/w | 1 | | |
| | | Disabled | Switched off, Inp = 0. With fail, the input is set to zero. | | 0 | | |
| | | Substitute | Substitute value Inp = Xfail. With | | 1 | | |

| | | | | | | | |
|--------|---------------|------------------|--|-----|----|--|--|
| | | value | fail, the substitute value is used. | | | | |
| Typ_2 | Sensor type 2 | Enum | Sensor type | r/w | 0 | | |
| | | Not used | The input is not used. | | 0 | | |
| | | 0...20 mA | Current : 0...20 mA | | 30 | | |
| | | -20...+20 mA | Current : -20...20 mA | | 31 | | |
| | | 4...20 mA | Current : 4...20 mA | | 32 | | |
| | | 0...10 V | Voltage : 0...10 V | | 42 | | |
| | | -10...+10V | Voltage : -10...10 V | | 43 | | |
| Fail_2 | Fail | Enum | Signal response on sensor fail | r/w | 1 | | |
| | | Disabled | Switched off, Inp = 0. With fail, the input is set to zero. | | 0 | | |
| | | Substitute value | Substitute value Inp = Xfail. With fail, the substitute value is used. | | 1 | | |

III-17.6 AI_8_SF (I/O-module with 8 analog inputs (No. 166))




AI_8_SF

Abb. 617

The function block is used for configuration and parameter setting of a module with 8 analog inputs for current or voltage. The sensors can be connected in 2 or 3-wire technology.

The conditioned measured values and the measured value status signals are available at the outputs of the function block.

Output **Slot** must be soft-wired with the input of a **KS108_VARIO_BK_ETH** type function block.

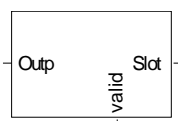
 **NOTE**
For additional information, see the operating instructions of the VARIO I/O module AI 8/SF.

| In-/Outputs | | |
|-------------------|-------|---|
| Name | Type | Description |
| Slot | Float | Own block number |
| Y_1 ... Y_8 | Float | Conditioned measured value 1 |
| fail_1 ... fail_8 | Bool | Signals a sensor error at the input (short-circuit, wrong polarity, ..) |
| valid | Bool | The communication to VARIO I/O system (hardware) is established. Measured values are valid. |

| Parameter | | | | | | | |
|---------------------|-------------------|-------|--|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| X0_1 ... X0_8 | Scale start 1 | Float | Physical value at 0% | r/w | 0.0 | | |
| X100_1 ... X100_8 | Scale end 1 | Float | Physical value at 100% | r/w | 100.0 | | |
| Xfail_1 ... Xfail_8 | Substitute value1 | Float | Substitute value with sensor fail | r/w | 9999.0 | | |
| Tfm_1 ... Tfm_8 | Filter time 1 [s] | Float | Filter time constant of signal processing in s | r/w | 0.5 | >0.0 | |

| Configuration | | | | | | | |
|-------------------|---------------|------------------|--|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Typ_1 ... Typ_8 | Sensor type 1 | Enum | Sensor type | r/w | 0 | | |
| | | Not used | The input is not used. | | 0 | | |
| | | 0...20 mA | Current : 0...20 mA | | 30 | | |
| | | -20...+20 mA | Current : -20...20 mA | | 31 | | |
| | | 4...20 mA | Current : 4...20 mA | | 32 | | |
| | | 0...40 mA | Current : 0...40 mA | | 33 | | |
| | | -40...+40 mA | Current : -40...40 mA | | 34 | | |
| | | 0...5 V | Voltage : 0...5 V | | 40 | | |
| | | -5...+5 V | Voltage : -5...5 V | | 41 | | |
| | | 0...10 V | Voltage : 0...10 V | | 42 | | |
| | | -10...+10V | Voltage : -10...10 V | | 43 | | |
| | | 0...25 V | Voltage : 0...25 V | | 44 | | |
| | | -25...+25 V | Voltage : -25...25 V | | 45 | | |
| | | 0...50 V | Voltage : 0...50 V | | 46 | | |
| Fail_1 ... Fail_8 | Fail | Enum | Signal response on sensor fail | r/w | 1 | | |
| | | Disabled | Switched off, Inp = 0. With fail, the input is set to zero. | | 0 | | |
| | | Substitute value | Substitute value Inp = Xfail. With fail, the substitute value is used. | | 1 | | |

III-17.7 AO_1_SF (I/O-module with 1 analog output (No. 155))



AO_1_SF

Abb. 618

The function block is used for configuration and parameter setting of a module with 1 analog output for optional connection of a voltage or current signal with connection of the actuator in 2-wire technology with screening.

The output value must be applied to input **Outp**. The input of 0...100 [%] is converted to the output. All output signals are set, the electrical wiring on the module defines the signal used (current 0...20 mA, current 4...20 mA, voltage 0...10V).

Output **Slot** must be soft-wired with the input of a **KS108_VARIO_BK_ETH** type function block.



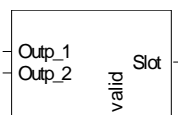
NOTE

For additional information, see the operating instructions of the VARIO AO 1/SF module.

In-/Outputs

| Name | Type | Description |
|-------|-------|--|
| Outp | Float | 0 ... 100% output value |
| Slot | Float | Own block number |
| valid | Bool | The communication to VARIO I/O system (hardware) is established. |

III-17.8 AO_2_U_BP (I/O-module with 2 analog outputs (No.156))



AO_2_U_BP

Abb. 619

The function block is used for configuration and parameter setting of a module with 2 analog outputs for voltage signals. The sensors can be connected in 2 or 3-wire technology.

The output values must be applied to inputs **outp_1** and **outp_2**, the input scale is 0...100 [%].

Output **Slot** must be soft-wired with the input of a **KS108_VARIO_BK_ETH** type function block.

**NOTE**

For additional information, see the operating instructions of the VARIO AO 2/U/BP module.

In-/Outputs

| Name | Type | Description |
|--------|-------|-------------------------|
| Outp_1 | Float | 0 ... 100% output value |
| Outp_2 | Float | 0 ... 100% output value |

| Name | Type | Description |
|-------|-------|--|
| Slot | Float | Own block number |
| valid | Bool | The communication to VARIO I/O system (hardware) is established. |

Parameter

| ID | Name | Type | Description | Access | Default | Range | Off |
|--------|-------------|-----------|--|--------|---------|-------|-----|
| Type_1 | Behaviour 1 | Enum | Function of continuous output | r/w | 0 | | |
| | | 0...10V | The continuous output is varied between 0 and 10 V. | | 0 | | |
| | | -10...10V | The continuous output is varied between negative -10V and positive +10V. | | 1 | | |
| Type_2 | Behaviour 2 | Enum | Function of continuous output | r/w | 0 | | |
| | | 0...10V | The continuous output is varied between 0 and 10 V. | | 0 | | |
| | | -10...10V | The continuous output is varied between negative -10V and positive +10V. | | 1 | | |

III-17.9 DI_2 (I/O-module with 2digital inputs (No.181))

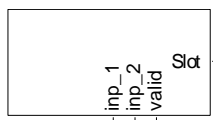
**DI_2**

Abb. 620

The function block is used for configuration and parameter setting of a module with 2 digital inputs. The sensors can be connected in 2, 3 or 4-wire technology. The digital input values are available at the outputs.

Output **Slot** must be soft-wired with the input of a **KS108_VARIO_BK_ETH** type function block.

**NOTE**

For additional information, see the operating instructions of the VARIO DI 2/24 module.

| In-/Outputs | | |
|-------------|-------|---|
| Name | Type | Description |
| inp_1 | Bool | Signal at input 1 of the DI module |
| inp_2 | Bool | Signal at input 2 of the DI module |
| valid | Bool | The communication to VARIO I/O system (hardware) is established. Measured values are valid. |
| Slot | Float | Own block number |

III-17.10 DI_4 (I/O-module with 4 digital inputs (No.182))

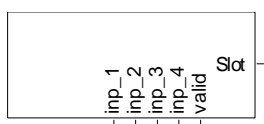
**DI_4**

Abb. 621

The function block is used for configuration and parameter setting of a module with 4 digital inputs. The sensors can be connected in 2 or 3-wire technology.

The digital input values are available at the outputs.

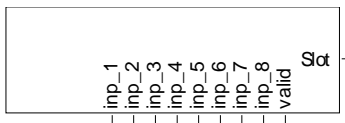
Output **Slot** must be soft-wired with the input of a **KS108_VARIO_BK_ETH** type function block.

**NOTE**

For additional information, see the operating instructions of the VARIO DI 4/24 module.

| In-/Outputs | | |
|-----------------|-------|---|
| Name | Type | Description |
| inp_1 ... inp_4 | Bool | Signal at input 1 of the DI module |
| valid | Bool | The communication to VARIO I/O system (hardware) is established. Measured values are valid. |
| Slot | Float | Own block number |

III-17.11 DI_8 (I/O-module with 8 digital inputs (No.183))



DI_8

Abb. 622

The function block is used for configuration and parameter setting of a module with 8 digital inputs. The sensors can be connected in 2 or 3-wire technology.

The digital input values are available at the outputs.

Output **Slot** must be soft-wired with the input of a **KS108_VARIO_BK_ETH** type function block.



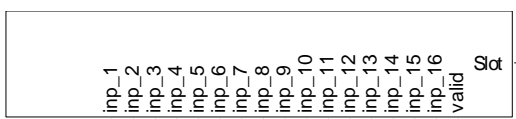
NOTE

For additional information, see the operating instructions of the VARIO DI 8/24 module.

In-/Outputs

| Name | Type | Description |
|-----------------|-------|---|
| inp_1 ... inp_8 | Bool | Signal at input 1 of the DI module |
| valid | Bool | The communication to VARIO I/O system (hardware) is established. Measured values are valid. |
| Slot | Float | Own block number |

III-17.12 DI_16 (I/O-module with 16 digital inputs (No.184))



DI_16

Abb. 623

The function block is used for configuration and parameter setting of a module with 16 digital inputs. The sensors can be connected in 2 or 3-wire technology. The digital input values are available at the outputs.

Output **Slot** must be soft-wired with the input of a **KS108_VARIO_BK_ETH** type function block.



NOTE

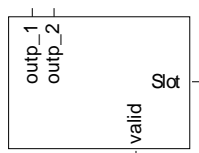
For additional information, see the operating instructions of the VARIO DI 16/24 module.

In-/Outputs

| Name | Type | Description |
|------------------|------|------------------------------------|
| inp_1 ... inp_16 | Bool | Signal at input 1 of the DI module |

| | | |
|-------|-------|---|
| valid | Bool | The communication to VARIO I/O system (hardware) is established. Measured values are valid. |
| Slot | Float | Own block number |

III-17.13 DO_2 (I/O-module with 2 digital outputs (No.170))



DO_2

Abb. 624

The function block is used for configuration and parameter setting of a module with 2 digital outputs. The actuators are connected in 2, 3 and 4 wire technology. The outputs are protected against short circuit and overload.

The output values must be applied to inputs **outp_1** and **outp_2**.

Output **Slot** must be soft-wired with the input of a **KS108_VARIO_BK_ETH** type function block.

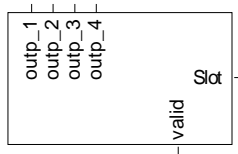


NOTE

For additional information, see the operating instructions of the VARIO DO 2/24 module.

| In-/Outputs | | |
|-------------|-------|--|
| Name | Type | Description |
| outp_1 | Bool | Signal of output 1 of the DO module |
| outp_2 | Bool | Signal of output 2 of the DO module |
| Name | Type | Description |
| valid | Bool | The communication to VARIO I/O system (hardware) is established. |
| Slot | Float | Own block number |

III-17.14 DO_4 (I/O-module with 4 digital outputs (No.171))



DO_4

Abb. 625

The function block is used for configuration and parameter setting of a module with 4 digital outputs. The actuators are connected in 2, 3 and 4 wire technology. The outputs are protected against short circuit and overload.

The output values must be applied to inputs **outp_1** ... **outp_4**.

Output **Slot** must be soft-wired with the input of a **KS108_VARIO_BK_ETH** type function block.



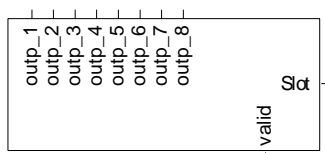
NOTE

For additional information, see the operating instructions of the VARIO DO 4/24 module.

In-/Outputs

| Name | Type | Description |
|-------------------|-------|--|
| outp_1 ... outp_4 | Bool | Signal of output 1 of the DO module |
| Name | Type | Description |
| valid | Bool | The communication to VARIO I/O system (hardware) is established. |
| Slot | Float | Own block number |

III-17.15 DO_8 (I/O-module with 8 digital outputs (No.172))



DO_8

Abb. 626

The function block is used for configuration and parameter setting of a module with 8 digital outputs. The actuators are connected in 2, 3 and 4 wire technology. The outputs are protected against short circuit and overload. The output values must be applied to inputs **outp_1** ... **outp_8**.

Output **Slot** must be soft-wired with the input of a **KS108_VARIO_BK_ETH** type function block.

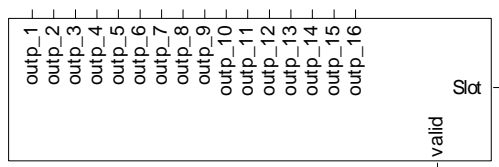


NOTE

For additional information, see the operating instructions of the VARIO DO 8/24 module.

| In-/Outputs | | |
|-------------------|-------|--|
| Name | Type | Description |
| outp_1 ... outp_8 | Bool | Signal of output 1 of the DO module |
| Name | Type | Description |
| valid | Bool | The communication to VARIO I/O system (hardware) is established. |
| Slot | Float | Own block number |

III-17.16 DO_16 (I/O-module with 16 digital outputs (No.173))



DO_16

Abb. 627

The function block is used for configuration and parameter setting of a module with 16 digital outputs. The actuators are connected in 2, 3 and 4 wire technology. The outputs are protected against short circuit and overload.

The output values must be applied to inputs **outp_1 ... outp_16**.

Output **Slot** must be soft-wired with the input of a **KS108_VARIO_BK_ETH** type function block.

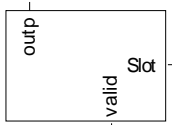


NOTE

For additional information, see the operating instructions of the VARIO DO 16/24 module.

| In-/Outputs | | |
|--------------------|-------|--|
| Name | Type | Description |
| outp_1 ... outp_16 | Bool | Signal of output 1 of the DO module |
| Name | Type | Description |
| valid | Bool | The communication to VARIO I/O system (hardware) is established. |
| Slot | Float | Own block number |

III-17.17 DO_1_230 (I/O-module with 1 digital relay output (No.174))



DO_1_230

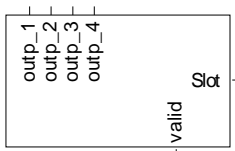
Abb. 628

The function block is used for configuration and parameter setting of a module with 1 digital relay output. The output value must be applied to input **outp**.

Output **Slot** must be soft-wired with the input of a **KS108_VARIO_BK_ETH** type function block.

| In-/Outputs | | |
|-------------|-------|--|
| Name | Type | Description |
| outp | Bool | Signal for output by the DO_1_230 module |
| valid | Bool | The communication to VARIO I/O system (hardware) is established. |
| Slot | Float | Own block number |

III-17.18 DO_4_230 (I/O-module with 4 digital relay outputs (No.175))



DO_4_230

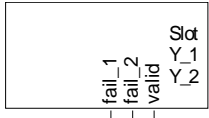
Abb. 629

The function block is used for configuration and parameter setting of a module with 4 digital relay outputs. The output values must be applied to inputs **outp_1** ... **outp_4**.

Output **Slot** must be soft-wired with the input of a **KS108_VARIO_BK_ETH** type function block.

| In-/Outputs | | |
|-------------------|-------|--|
| Name | Type | Description |
| outp_1 ... outp_4 | Bool | Signal of output 1 of the DO module |
| valid | Bool | The communication to VARIO I/O system (hardware) is established. |
| Slot | Float | Own block number |

III-17.19 RTD_2 (I/O-module with 2 analog inputs (No. 163))



RTD_2

Abb. 630

The function block is used for configuration and parameter setting of a module with 2 analog inputs for Temperature measurement resistors. The module supports platinum and nickel sensors acc. to DIN standard and SAMA guideline. Additionally, the CU10, CU50, CU53 as well as KTY81 and KTY84 sensors are supported. Sensor connection is in 2, 3 and 4- wire technology. The sensors are connected in 2, 3 and 4 wire technology.



NOTE!

During projecting, note that no isolating voltage between the analog inputs and the BUS is specified. For thermistor measurement, for instance, this means that the user may have to provide signals with **safe isolation**.

The conditioned measured values and the measured value status signals are available at its outputs. Output **Slot** must be soft-wired with the input of a **KS108_VARIO_BK_ETH** type function block.



NOTE

For additional information, see the operating instructions of the VARIO RTD 2 I/O module.

Unit

Unit

°C, °F or without unit can be selected. Process values are converted automatically when changing units. All other values (set-points, limit values...) must be converted manually.

In-/Outputs

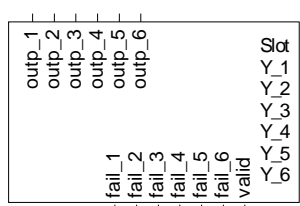
| Name | Type | Description |
|--------|-------|---|
| Slot | Float | Own block number |
| Y_1 | Float | Conditioned measured value 1 |
| Y_2 | Float | Conditioned measured value 2 |
| fail_1 | Bool | Signals a sensor error at the input (short-circuit, wrong polarity, ..) |
| fail_2 | Bool | Signals a sensor error at the input (short-circuit, wrong polarity, ..) |
| valid | Bool | The communication to VARIO I/O system (hardware) is established. Measured values are valid. |

| Parameter | | | | | | | |
|-----------|-------------------|-------|---|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| X0_1 | Scale start 1 | Float | Physical value at 0% | r/w | 0.0 | | |
| X100_1 | Scale end 1 | Float | Physical value at 100% | r/w | 100.0 | | |
| Xfail_1 | Substitute value1 | Float | Substitute value with sensor fail | r/w | 9999.0 | | |
| Tfm_1 | Filter time 1 [s] | Float | Filter time constant of signal processing [s] | r/w | 0.5 | >0.0 | |
| X0_2 | Scale start 2 | Float | Physical value at 0% | r/w | 0.0 | | |
| X100_2 | Scale end 2 | Float | Physical value at 100% | r/w | 100.0 | | |
| Xfail_2 | Substitute value2 | Float | Substitute value with sensor fail | r/w | 9999.0 | | |
| Tfm_2 | Filter time 2 [s] | Float | Filter time constant of signal processing [s] | r/w | 0.5 | >0.0 | |

| Configuration | | | | | | | |
|---------------|---------------|------------------|--|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Unit | Unit | Enum | Engin. unit of measured value | r/w | 0 | | |
| | | °C | Unit = °C, degrees Celsius | | 0 | | |
| | | °F | Unit = °F, degrees Fahrenheit | | 1 | | |
| Typ_1 | Sensor type 1 | Enum | Sensor type | r/w | 0 | | |
| | | Not used | The input is not used. | | 0 | | |
| | | Pt100 | Pt100 (-200...850°C, -328...1562°F) | | 20 | | |
| | | Pt1000 | Pt1000 (-200 ... 850°C, -140...1562°F) | | 21 | | |
| | | Ni100 | Ni100 (-60...180°C, -76...356°F) | | 22 | | |
| | | Ni1000 | Ni1000 (-60...180°C, -76...356°F) | | 23 | | |
| | | KTY81-110 | KTY81-110 (-55...150°C, -67...302°F) | | 24 | | |
| | | KTY84 | KTY84 (-40...300°C, -104...572°F) | | 25 | | |
| | | 400 Ohm | Resistance (0...400 Ohm) | | 26 | | |
| | | 4000 Ohm | Resistance (0...4000 Ohm) | | 28 | | |
| Fail_1 | Fail | Enum | Signal response on sensor fail | r/w | 1 | | |
| | | Disabled | Switched off, Inp = 0. With fail, the input is set to zero. | | 0 | | |
| | | Substitute value | Substitute value Inp = Xfail. With fail, the substitute value is used. | | 1 | | |
| Typ_2 | Sensor type 2 | Enum | Sensor type | r/w | 0 | | |
| | | Not used | The input is not used. | | 0 | | |

| | | | | | | | |
|--------|------|------------------|--|-----|----|--|--|
| | | Pt100 | Pt100 (-200...850°C, -328...1562°F) | | 20 | | |
| | | Pt1000 | Pt1000 (-200 ... 850°C, -140...1562°F) | | 21 | | |
| | | Ni100 | Ni100 (-60...180°C, -76...356°F) | | 22 | | |
| | | Ni1000 | Ni1000 (-60...180°C, -76...356°F) | | 23 | | |
| | | KTY81-110 | KTY81-110 (-55...150°C, -67...302°F) | | 24 | | |
| | | KTY84 | KTY84 (-40...300°C, -104...572°F) | | 25 | | |
| | | 400 Ohm | Resistance (0..400 Ohm) | | 26 | | |
| | | 4000 Ohm | Resistance (0..4000 Ohm) | | 28 | | |
| Fail_2 | Fail | Enum | Signal response on sensor fail | r/w | 1 | | |
| | | Disabled | Switched off, Inp = 0. With fail, the input is set to zero. | | 0 | | |
| | | Substitute value | Substitute value Inp = Xfail. With fail, the substitute value is used. | | 1 | | |

III-17.20 RTD_6 (I/O-module with 6 analog in- and outputs (No. 169))



RTD_6

Abb. 631

The function block is used for configuration and parameter setting of a module with 6 analog inputs for Temperature measurement resistors PT100 and 6 analog outputs. Connection is in 2 and 3-wire technology with screening. Moreover, the module has 1 heating current input for totalizing current converter (see note!). The inputs and outputs are galvanically isolated.

i **NOTE!**
 To use the heating current input for totalizing current converter, the RTD_6_HC function block must be used.

The conditioned measured values and the measured value status signals are available at its outputs. The output values must be applied to the inputs.

Output **Slot** must be soft-wired with the input of a **KS108_VARIO_BK_ETH** type function block.

Unit

Unit

°C, °F or without unit can be selected. Process values are converted automatically when changing units. All other values (set-points, limit values...) must be converted manually.

Scaling

Analog input 0 % (X0_i) and 100 % (X100_i) scaling

These two values are used for scaling of a digital input. Process value corrections, which are e.g. required due to unfavourable sensor installation, or to correct the tolerances of several sensors, can be made using the 2-point correction method. If the sensor is a transmitter with standard signal output (e.g. weight -> 0...10V), scaling of the physical quantity can be determined for this input.

In this case, the output span start value **X0_i**, i.e. the span start in units of the physical quantity (e.g. 0 t) is assigned to the transmitter output span start (electrical signal span start, e.g. 4 mA).

The output span end **X100_i**, i.e. the span end in units of the physical quantity (e.g. 10 t) is assigned to the transmitter output span end (electrical signal span end, e.g. 20 mA).

In-/Outputs

| Name | Type | Description |
|-------------------|------|----------------------------------|
| outp_1 ... outp_6 | Bool | Signal at output 1 of the module |

| Name | Type | Description |
|-------------------|-------|---|
| Slot | Float | Own block number |
| Y_1 ... Y_6 | Float | Conditioned measured value 1 |
| fail_1 ... fail_6 | Bool | Signals a sensor error at the input (short-circuit, wrong polarity, ..) |
| valid | Bool | The communication to VARIO I/O system (hardware) is established. Measured values are valid. |

Parameter

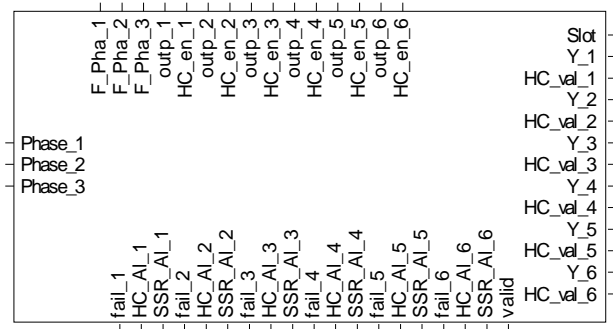
| ID | Name | Type | Description | Access | Default | Range | Off |
|---------------------|-------------------|-------|---|--------|---------|-------|-----|
| X0_1 ... X0_6 | Scale start 1 | Float | Physical value at 0% | r/w | 0.0 | | |
| X100_1 ... X100_6 | Scale end 1 | Float | Physical value at 100% | r/w | 100.0 | | |
| Xfail_1 ... Xfail_6 | Substitute value1 | Float | Substitute value with sensor fail | r/w | 9999.0 | | |
| Tfm_1 ... Tfm_6 | Filter time 1 [s] | Float | Filter time constant of signal processing [s] | r/w | 0.5 | >0.0 | |

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|-----------------|---------------|------|-------------------------------|--------|---------|-------|-----|
| Unit | Unit | Enum | Engin. unit of measured value | r/w | 0 | | |
| | | °C | Unit = °C, degrees Celsius | | 0 | | |
| | | °F | Unit = °F, degrees Fahrenheit | | 1 | | |
| Typ_1 ... Typ_6 | Sensor type 1 | Enum | Sensor type | r/w | 0 | | |

| | | | | | | | |
|----------------------|------|------------------|--|-----|----|--|--|
| | | Not used | The input is not used. | | 0 | | |
| | | Pt100 | Pt100 (-200...850°C, -328...1562°F) | | 20 | | |
| | | 450 Ohm | Resistance (0...450 Ohm) | | 27 | | |
| Fail_1 ... Fail_6 | Fail | Enum | Signal response on sensor fail | r/w | 1 | | |
| | | Disabled | Switched off, Inp = 0. With fail, the input is set to zero. | | 0 | | |
| | | Substitute value | Substitute value Inp = Xfail. With fail, the substitute value is used. | | 1 | | |


III-17.21 RTD_6_HC (I/O-module with 6 analog in- and outputs and heating current (No. 162))



RTD_6_HC

Abb. 632

The function block is used for configuration and parameter setting of a module with 6 analog inputs for Temperature measurement resistors PT100 and 6 analog outputs. Connection is in 2 and 3-wire technology with screening. Moreover, the module has 1 heating current input for totalizing current converter (see note!). The inputs and outputs are galvanically isolated.

 **NOTE!**
If the heating current input for totalizing current converter is **not** used, the less complicated RTD_6 function block can be used.

The conditioned measured values and the measured value status signals are available at its outputs. The output values must be applied to the inputs.

Output **Slot** must be soft-wired with the input of a **KS108_VARIO_BK_ETH** type function block.

Unit

Unit
°C, °F or without unit can be selected. Process values are converted automatically when changing units. All other values (set-points, limit values...) must be converted manually.

Scaling

Analog input 0 % (X0_i) and 100 % (X100_i) scaling

These two values are used for scaling of a digital input. Process value corrections, which are e.g. required due to unfavourable sensor installation, or to correct the tolerances of several sensors, can be made using the 2-point correction method. If the sensor is a transmitter with standard signal output (e.g. weight → 0...10V), scaling of the physical quantity can be determined for this input.

In this case, the output span start value **X0_i**, i.e. the span start in units of the physical quantity (e.g. 0 t) is assigned to the transmitter output span start (electrical signal span start, e.g. 4 mA).

The output span end **X100_i**, i.e. the span end in units of the physical quantity (e.g. 10 t) is assigned to the transmitter output span end (electrical signal span end, e.g. 20 mA).

Heating current monitoring

The heating current monitoring function of the VARIO modules can be used to monitor all outputs of a module by means of a current transformer. The relevant totalizing current input is provided on the I/O extension modules.

The heating currents can be monitored at an adjustable time interval. Independent of the input signals, all outputs configured accordingly except the one to be monitored are switched off and the heating current is measured. The interval for heating current monitoring is adjustable per module for all its outputs.

Heating current monitoring interval [sec] (Hc.Cy):

A monitoring measurement is made at intervals of the adjusted number of seconds. This means that each individual output is measured again after outputs + 1 (for short circuit measurement) interval. This interval should not be too short, because a heater is a relatively insensitive product and does not need frequent checking. Moreover, switching off the other outputs during measurement might impair the control quality, which is dependent on these outputs. This applies particularly to fast processes.

Duty cycle for heating current monitoring [100 ms] (Hc.Ti)

This is the time during which the output to be checked is energized. All other outputs pertaining to this module are switched off. This value can be increased in steps of 100 ms from its minimum setting of 200 ms. The actual measurement time is 100 ms and starts at the end of this adjusted time. The preceding time is the signal settling time. The value to be adjusted is dependent on:

1. **The switching element type**. Contactors and, in particular, power contactors, have a longer cycle time and require a higher number of duty cycles. For solid-state relays, this value may be neglected.
2. **The settling time of the current transformer**. On transformers with built-in transmitter, settlement of the output value takes longer.

Generally, this setting should be as low as possible, because the other values are switched off and the function (e.g. control) is disturbed during this time.

Heating current alarm (Hc.ALMode)

This parameter determines, if the heating current must be monitored for too high (overload) or too low (interruption) values. With both settings, the actuator is monitored for short circuit. Normally, the parameter is set for low load, because the resistance of heating elements tends to increase over the time, or they even burn out.

Current transformation ratio (Tr.Rat)

For heating current display with the right scaling (e.g. in A) in HC.Me, the transformation ratio of the connected current transformer must be set. Example: The transformer has a transformation ratio of 1000:1. 1000 must be set under parameter Tr.Rat . With a 50 mA AC input current flowing into the heating current input, a heating current value of 50.0 A is output.

Phase conductor (Phase1 ... Phase3)

The heating current measurement can be corrected using a value derived from the phase conductor voltage. A low supply voltage would cause a low current and trigger a heating current (fail) alarm. The (low) heating current is corrected by the per cent value of voltage reduction with the supply voltage correction switched on. This applies analogously to overvoltage.

0: - (switched off, no supply voltage correction) / 1: Phase 1 / 2: Phase 2 / 3: Phase 3.

This setting determines to which phase conductor the heating of this output is connected.

With three-phase current heating switched by an output, using all 3 phase conductors for correction is not possible. One phase conductor must be selected.

SSR alarm

This alarm is the output of an error message determined using the heating current monitoring function. In one heating current monitoring phase, all outputs are switched off and no (heating) current may flow. If there is a current flow, this means a short circuit of the switching element.

Normal voltage [V] (U.norm)

During heating current measurement, a low supply voltage may cause faulty measurements and thus faulty alarms, in particular, if the current limit values are very close to the rated current. This setting offers the possibility to activate the supply voltage correction and to determine the "normal voltage" simultaneously. A separate voltage transformer transforms the supply voltage into a DC voltage with a 100% value of 7V DC. Fill in the supply voltage **rating** to produce these 7V DC (normally 230V). Any supply voltage variation generates a secondary voltage \neq 7V. This value is used for correction of the current value.

| In-/Outputs | | |
|---------------------|-------|---|
| Name | Type | Description |
| Phase_1 | Float | Voltage of phase 1 |
| Phase_2 | Float | Voltage of phase 2 |
| Phase_3 | Float | Voltage of phase 3 |
| F_Pha_1 | Bool | Fail voltage phase 1 |
| F_Pha_2 | Bool | Fail voltage phase 2 |
| F_Pha_3 | Bool | Fail voltage phase 3 |
| outp_1 ... outp_6 | Bool | Signal at output 1 of the module |
| HC_en_1 ... HC_en_6 | Bool | Enable heating current monitoring at output 1 |

| Name | Type | Description |
|-----------------------|-------|---|
| Slot | Float | Own block number |
| Y_1 ... Y_6 | Float | Conditioned measured value 1 |
| HC_val_1 ... HC_val_6 | Float | Heating current measured value |
| fail_1 ... fail_6 | Bool | Signals a sensor error at the input (short-circuit, wrong polarity, ..) |

| | | |
|-----------------------|------|--|
| HC_AI_1 ... HC_AI_6 | Bool | Heating current alarm |
| SSR_AI_1 ... SSR_AI_6 | Bool | Short-circuit alarm |
| valid | Bool | The communication to VARIO I/O system (hardware) is established. Measured values are valid. |

Parameter

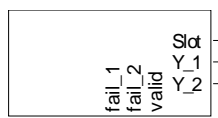
| ID | Name | Type | Description | Access | Default | Range | Off |
|--------------------------|--------------------|-------|--|--------|---------|---------------|-----|
| X0_1 ... X0_6 | Scale start 1 | Float | Physical value at 0% | r/w | 0.0 | | |
| X100_1 ... X100_6 | Scale end 1 | Float | Physical value at 100% | r/w | 100.0 | | |
| Xfail_1 ... Xfail_6 | Substitute value 1 | Float | Substitute value with sensor fail | r/w | 9999.0 | | |
| Tfm_1 ... Tfm_6 | Filter time 1 [s] | Float | Filter time constant of signal processing [s] | r/w | 0.5 | >0.0 | |
| HC_Lim_1 ... HC_Lim_6 | HC Limit 1 | Float | Heating current limit [A] (with switch-off value – 32000) | r/w | off | 1.5 ... 50 | ja |

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|----------------------|---------------------|------------------|--|--------|---------|----------------|-----|
| Unit | Unit | Enum | Engin. unit of measured value | r/w | 0 | | |
| | | °C | Unit = °C, degrees Celsius | | 0 | | |
| | | °F | Unit = °F, degrees Fahrenheit | | 1 | | |
| Hc_Cy | HC Cycle time [s] | Int | Cycle time of heating current monitoring [s] | r/w | 10 | 1 ... 3000 | |
| Hc_Ti | HC test [n*100ms] | Int | Switch-on cycles for heating current monitoring [100 ms] | r/w | 2 | 2 ... 255 | |
| U_norm | Standard voltage[V] | Int | Standard voltage [V] | r/w | 230 | 1 ... 32000 | |
| Typ_1 ... Typ_6 | Sensor type 1 | Enum | Sensor type | r/w | 0 | | |
| | | Not used | The input is not used. | | 0 | | |
| | | Pt100 | Pt100 (-200...850°C, -328...1562°F) | | 20 | | |
| | | 450 Ohm | Resistance (0...450 Ohm) | | 27 | | |
| Fail_1 ... Fail_6 | Fail | Enum | Signal response on sensor fail | r/w | 1 | | |
| | | Disabled | Switched off, Inp = 0. With fail, the input is set to zero. | | 0 | | |
| | | Substitute value | Substitute value Inp = Xfail. With fail, the substitute value is used. | | 1 | | |

| | | | | | | | |
|--------------------------|----------------------|---------------------------|---|-----|------|----------------|--|
| Hc_AiM_1 ... Hc_Aim_6 | HC alarm mode 1 | Enum | Type of heating current alarm | r/w | 1 | | |
| | | Max + short circuit | Activate the overload and short-circuit monitoring. Overload occurs when current I exceeds the heating current limit value. | | 0 | | |
| | | Min + short circuit | Activate the break and short circuit monitoring. Break occurs when current I is lower than the heating current limit. | | 1 | | |
| Phase_1 ... Phase_6 | Voltage phase 1 | Enum | Used external conductor | r/w | 0 | | |
| | | ---- | Not active | | 0 | | |
| | | phase 1 | External conductor 1 | | 1 | | |
| | | phase 2 | External conductor 2 | | 2 | | |
| | | phase 3 | External conductor 3 | | 3 | | |
| Tr_Rat_1 ... Tr_Rat_6 | Transform ratio 1 | Int | Transformer ratio | r/w | 1000 | 1 ... 30000 | |

III-17.22 UTH_2 (I/O-module with 2 analog inputs (No. 161))



UTH_2

Abb. 633

The function block is used for configuration and parameter setting of a module with 2 analog inputs for for measuring signals from commercially available thermocouples. 13 different thermocouple types to DIN EN 60584-1 and DIN 43710 as well as a linear voltage input of -15 mV to +85 mV are supported. Connection is in 2-wire technology.

The conditioned measured values and the measured value status signals are available at its outputs.

Output **Slot** must be soft-wired with the input of a **KS108_VARIO_BK_ETH** type function block.



NOTE

For additional information, see the operating instructions of the VARIO UTH 2 I/O module.

Unit

Unit

°C, °F or without unit can be selected. Process values are converted automatically when changing units. All other values (set-points, limit values...) must be converted manually.

| In-/Outputs | | |
|-------------|-------|---|
| Name | Type | Description |
| Slot | Float | Own block number |
| Y_1 | Float | Conditioned measured value 1 |
| Y_2 | Float | Conditioned measured value 2 |
| fail_1 | Bool | Signals a sensor error at the input (short-circuit, wrong polarity, ..) |
| fail_2 | Bool | Signals a sensor error at the input (short-circuit, wrong polarity, ..) |
| valid | Bool | The communication to VARIO I/O system (hardware) is established. Measured values are valid. |

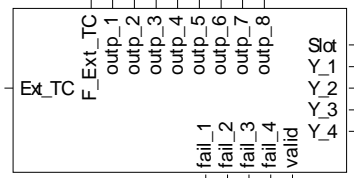
| Parameter | | | | | | | |
|-----------|-------------------|-------|---|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| X0_1 | Scale start 1 | Float | Physical value at 0% | r/w | 0.0 | | |
| X100_1 | Scale end 1 | Float | Physical value at 100% | r/w | 100.0 | | |
| Xfail_1 | Substitute value1 | Float | Substitute value with sensor fail | r/w | 9999.0 | | |
| Tfm_1 | Filter time 1 [s] | Float | Filter time constant of signal processing [s] | r/w | 0.5 | >0.0 | |
| X0_2 | Scale start 2 | Float | Physical value at 0% | r/w | 0.0 | | |
| X100_2 | Scale end 2 | Float | Physical value at 100% | r/w | 100.0 | | |
| Xfail_2 | Substitute value2 | Float | Substitute value with sensor fail | r/w | 9999.0 | | |
| Tfm_2 | Filter time 2 [s] | Float | Filter time constant of signal processing [s] | r/w | 0.5 | >0.0 | |

| Configuration | | | | | | | |
|---------------|---------------|----------|--|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Unit | Unit | Enum | Engin. unit of measured value | r/w | 0 | | |
| | | °C | Unit = °C, degrees Celsius | | 0 | | |
| | | °F | Unit = °F, degrees Fahrenheit | | 1 | | |
| Typ_1 | Sensor type 1 | Enum | Sensor type | r/w | 0 | | |
| | | Not used | The input is not used. | | 0 | | |
| | | Typ L | Typ L (-100...900°C, -148...1652°F), Fe-CuNi DIN | | 1 | | |
| | | Typ J | Typ J (-100...900°C, -148...1652°F), Fe-CuNi DIN | | 2 | | |
| | | Typ K | Typ K (-100...1350°C, -148...2462°F), NiCr-Ni | | 3 | | |
| | | Typ N | Typ N (-100...1300°C, -148...2372°F), Nicrosil- | | 4 | | |

| | | | | | | | |
|--------|---------------|------------------|--|-----|----|--|--|
| | | | Nisil | | | | |
| | | Typ S | Typ S (0...1760°C, 32...3200°F), PtRh-Pt10% | | 5 | | |
| | | Typ R | Typ R (0...1760°C, 32...3200°F), PtRh-Pt13% | | 6 | | |
| | | Typ T | Typ T (-200...400°C, -328...752°F), Cu-CuNi | | 7 | | |
| | | Typ C | Typ C (0...2315°C, 32...4199°F), W5%Re-W26%Re | | 8 | | |
| | | Typ E | Typ E (-100...1000°C, -148...1832°F), NiCr-CuNi | | 10 | | |
| | | Typ B | Typ B (0/100...1820°C, 32/172 ... 3308°F), PtRh-Pt6% | | 11 | | |
| | | Typ W | Typ W (0...2315°C, 32...4199°F) | | 12 | | |
| | | -15..85mV | Voltage : -15...+85mV | | 48 | | |
| Fail_1 | Fail | Enum | Signal response on sensor fail | r/w | 1 | | |
| | | Disabled | Switched off, Inp = 0. With fail, the input is set to zero. | | 0 | | |
| | | Substitute value | Substitute value Inp = Xfail. With fail, the substitute value is used. | | 1 | | |
| Typ_2 | Sensor type 2 | Enum | Sensor type | r/w | 0 | | |
| | | Not used | The input is not used. | | 0 | | |
| | | Typ L | Typ L (-100...900°C, -148...1652°F), Fe-CuNi DIN | | 1 | | |
| | | Typ J | Typ J (-100...900°C, -148...1652°F), Fe-CuNi DIN | | 2 | | |
| | | Typ K | Typ K (-100...1350°C, -148...2462°F), NiCr-Ni | | 3 | | |
| | | Typ N | Typ N (-100...1300°C, -148...2372°F), Nicrosil-Nisil | | 4 | | |
| | | Typ S | Typ S (0...1760°C, 32...3200°F), PtRh-Pt10% | | 5 | | |
| | | Typ R | Typ R (0...1760°C, 32...3200°F), PtRh-Pt13% | | 6 | | |
| | | Typ T | Typ T (-200...400°C, -328...752°F), Cu-CuNi | | 7 | | |
| | | Typ C | Typ C (0...2315°C, 32...4199°F), W5%Re-W26%Re | | 8 | | |
| | | Typ E | Typ E (-100...1000°C, -148...1832°F), NiCr-CuNi | | 10 | | |
| | | Typ B | Typ B (0/100...1820°C, 32/172 ... 3308°F), PtRh-Pt6% | | 11 | | |
| | | Typ W | Typ W (0...2315°C, 32...4199°F) | | 12 | | |
| | | -15..85mV | Voltage : -15...+85mV | | 48 | | |
| Fail_2 | Fail | Enum | Signal response on sensor fail | r/w | 1 | | |
| | | Disabled | Switched off, Inp = 0. With fail, the input is set to zero. | | 0 | | |

| | | | | | | |
|--|------------------|--|--|---|--|--|
| | Substitute value | Substitute value Inp = Xfail. With fail, the substitute value is used. | | 1 | | |
|--|------------------|--|--|---|--|--|

III-17.23 UTH_4 (I/O-module with 4 analog in- and 8 digital outputs (No. 168))



UTH_4

Abb. 634

The function block is used for configuration and parameter setting of a module with 4 analog inputs for thermocouples and 8 digital outputs. Connection is in 2-wire technology with screening. Moreover, the module has 1 heating current input (see note!)



NOTE

To use the heating current input for totalizing current converter, the *UTH_4_HC* function block must be used.

The conditioned measured values and the measured value status signals are available at its outputs. The output values must be applied to inputs **outp_1** ... **outp_8**.

Output **Slot** must be soft-wired with the input of a **KS108_VARIO_BK_ETH** type function block.

Unit

Unit

°C, °F or without unit can be selected. Process values are converted automatically when changing units. All other values (set-points, limit values...) must be converted manually.

Scaling

Analog input 0 % (X0_i) and 100 % (X100_i) scaling

These two values are used for scaling of a digital input. Process value corrections, which are e.g. required due to unfavourable sensor installation, or to correct the tolerances of several sensors, can be made using the 2-point correction method. If the sensor is a transmitter with standard signal output (e.g. weight -> 0...10V), scaling of the physical quantity can be determined for this input.

In this case, the output span start value **X0_i**, i.e. the span start in units of the physical quantity (e.g. 0 t) is assigned to the transmitter output span start (electrical signal span start, e.g. 4 mA).

The output span end **X100_i**, i.e. the span end in units of the physical quantity (e.g. 10 t) is assigned to the transmitter output span end (electrical signal span end, e.g. 20 mA).

Internal and external temperature compensation

Internal TC :

If the internal temperature compensation is selected, the internal (built-in) temperature compensation is active. Other settings are not required.

External TC via thermostat:

A thermostat terminal box is used. Take the compensating lead up to this box, and use copper lead between the box and the Vario station. The temperature of this thermostat is known (mostly 50 °C). The value is applied to input "Ext_TC" of the function block and used for calculation of the external TC.

| In-/Outputs | | |
|-------------------|-------|----------------------------------|
| Name | Type | Description |
| Ext_TC | Float | Temperature of the external TC |
| F_Ext_TC | Bool | External TC fail |
| outp_1 ... outp_8 | Bool | Signal at output 1 of the module |

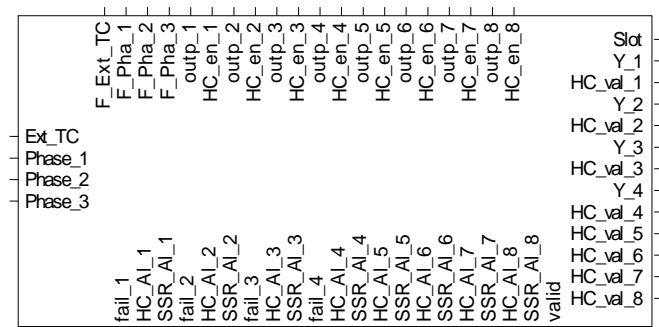
| Name | Type | Description |
|-------------------|-------|--|
| Slot | Float | Own block number |
| Y_1 ... Y_4 | Float | Conditioned measured value 1 |
| fail_1 ... fail_4 | Bool | Signals a sensor error at the input (short-circuit, wrong polarity, ..) |
| valid | Bool | The communication to VARIO I/O system (hardware) is established. Measured values are valid. |

| Parameter | | | | | | | |
|----------------------|-------------------|-------|---|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| X0_1 ... X0_4 | Scale start 1 | Float | Physical value at 0% | r/w | 0.0 | | |
| X100_1 ... X100_4 | Scale end 1 | Float | Physical value at 100% | r/w | 100.0 | | |
| Xfail_1 ... Xfail_4 | Substitute value1 | Float | Substitute value with sensor fail | r/w | 9999.0 | | |
| Tfm_1 ... Tfm_4 | Filter time 1 [s] | Float | Filter time constant of signal processing [s] | r/w | 0.5 | >0.0 | |

| Configuration | | | | | | | |
|--------------------|---------------|------|-------------------------------|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Unit | Unit | Enum | Engin. unit of measured value | r/w | 0 | | |
| | | °C | Unit = °C, degrees Celsius | | 0 | | |
| | | °F | Unit = °F, degrees Fahrenheit | | 1 | | |
| Typ_1 ... Typ_4 | Sensor type 1 | Enum | Sensor type | r/w | 0 | | |

| | | | | | | | |
|----------------------|------------------|------------------|--|-----|----|--|--|
| | | Not used | The input is not used. | | 0 | | |
| | | Typ L | Typ L (-100...900°C, -148...1652°F), Fe-CuNi DIN | | 1 | | |
| | | Typ J | Typ J (-100...900°C, -148...1652°F), Fe-CuNi DIN | | 2 | | |
| | | Typ K | Typ K (-100...1350°C, -148...2462°F), NiCr-Ni | | 3 | | |
| | | Typ N | Typ N (-100...1300°C, -148...2372°F), Nicrosil-Nisil | | 4 | | |
| | | Typ S | Typ S (0...1760°C, 32...3200°F), PtRh-Pt10% | | 5 | | |
| | | Typ R | Typ R (0...1760°C, 32...3200°F), PtRh-Pt13% | | 6 | | |
| | | Typ T | Typ T (-200...400°C, -328...752°F), Cu-CuNi | | 7 | | |
| | | Typ C | Typ C (0...2315°C, 32...4199°F), W5%Re-W26%Re | | 8 | | |
| | | Typ D | Typ D (0...2315°C, 32...4199°F), W3%Re-W25%Re | | 9 | | |
| | | Typ E | Typ E (-100...1000°C, -148...1832°F), NiCr-CuNi | | 10 | | |
| | | Typ B | Typ B (0/100...1820°C, 32/172 ... 3308°F), PtRh-Pt6% | | 11 | | |
| | | 0...70mV | Voltage : 0...70mV | | 47 | | |
| Fail_1 ... Fail_4 | Fail | Enum | Signal response on sensor fail | r/w | 1 | | |
| | | Disabled | Switched off, Inp = 0. With fail, the input is set to zero. | | 0 | | |
| | | Substitute value | Substitute value Inp = Xfail. With fail, the substitute value is used. | | 1 | | |
| STK_1 ... STK_4 | Temp. compens. 1 | Enum | Location of temperature compensation | r/w | 0 | | |
| | | Internal CJC | Internal temperature compensation | | 0 | | |
| | | External CJC | External temperature compensation | | 1 | | |

III-17.24 UTH_4_HC (I/O-module with 4 analog in-, 8 digital outputs and heating current (No. 164))



UTH_4_HC

Abb. 635

The function block is used for configuration and parameter setting of a module with 4 analog inputs for thermocouples and 8 digital outputs. Connection is in 2-wire technology with screening. Moreover, the module has 1 heating current input (see note!)



NOTE!

If the heating current input for totalizing current converter is **not** used, the less complicated UTH_4 function block can be used.

The conditioned measured values and the measured value status signals are available at its outputs. The output values must be applied to inputs **outp_1** ... **outp_8**.

Output **Slot** must be soft-wired with the input of a **KS108_VARIO_BK_ETH** type function block.

Unit

Unit

°C, °F or without unit can be selected. Process values are converted automatically when changing units. All other values (set-points, limit values...) must be converted manually.

Scaling

Analog input 0 % (X0_i) and 100 % (X100_i) scaling

These two values are used for scaling of a digital input. Process value corrections, which are e.g. required due to unfavourable sensor installation, or to correct the tolerances of several sensors, can be made using the 2-point correction method. If the sensor is a transmitter with standard signal output (e.g. weight -> 0...10V), scaling of the physical quantity can be determined for this input.

In this case, the output span start value **X0_i**, i.e. the span start in units of the physical quantity (e.g. 0 t) is assigned to the transmitter output span start (electrical signal span start, e.g. 4 mA).

The output span end **X100_i**, i.e. the span end in units of the physical quantity (e.g. 10 t) is assigned to the transmitter output span end (electrical signal span end, e.g. 20 mA).

Internal and external temperature compensation

Internal TC :

If the internal temperature compensation is selected, the internal (built-in) temperature compensation is active. Other settings are not required.

External TC via thermostat:

A thermostat terminal box is used. Take the compensating lead up to this box, and use copper lead between the box and the Vario station. The temperature of this thermostat is known (mostly 50 °C). The value is applied to input "Ext_TC" of the function block and used for calculation of the external TC.

Heating current monitoring

The heating current monitoring function of the VARIO modules can be used to monitor all outputs of a module by means of a current transformer. The relevant totalizing current input is provided on the I/O extension modules.

The heating currents can be monitored at an adjustable time interval. Independent of the input signals, all outputs configured accordingly except the one to be monitored are switched off and the heating current is measured. The interval for heating current monitoring is adjustable per module for all its outputs.

Heating current monitoring interval [sec] (Hc.Cy):

A monitoring measurement is made at intervals of the adjusted number of seconds. This means that each individual output is measured again after outputs + 1 (for short circuit measurement) interval. This interval should not be too short, because a heater is a relatively insensitive product and does not need frequent checking. Moreover, switching off the other outputs during measurement might impair the control quality, which is dependent on these outputs. This applies particularly to fast processes.

Duty cycle for heating current monitoring [100 ms] (Hc.Ti)

This is the time during which the output to be checked is energized. All other outputs pertaining to this module are switched off. This value can be increased in steps of 100 ms from its minimum setting of 200 ms. The actual measurement time is 100 ms and starts at the end of this adjusted time. The preceding time is the signal settling time. The value to be adjusted is dependent on:

1. **The switching element type** . Contactors and, in particular, power contactors, have a longer cycle time and require a higher number of duty cycles. For solid-state relays, this value may be neglected.
2. The **settling time of the current transformer**. On transformers with built-in transmitter, settlement of the output value takes longer.

Generally, this setting should be as low as possible, because the other values are switched off and the function (e.g. control) is disturbed during this time.

Heating current alarm (Hc.ALMode)

This parameter determines, if the heating current must be monitored for too high (overload) or too low (interruption) values. With both settings, the actuator is monitored for short circuit. Normally, the parameter is set for low load, because the resistance of heating elements tends to increase over the time, or they even burn out.

Current transformation ratio (Tr.Rat)

For heating current display with the right scaling (e.g. in A) in HC.Me, the transformation ratio of the connected current transformer must be set. Example: The transformer has a transformation ratio of 1000:1. 1000 must be set under parameter Tr.Rat . With a 50 mA AC input current flowing into the heating current input, a heating current value of 50.0 A is output.

Phase conductor (Phase1 ... Phase3)

The heating current measurement can be corrected using a value derived from the phase conductor voltage. A low supply voltage would cause a low current and trigger a heating current (fail) alarm. The (low) heating current is corrected by the per cent value of voltage reduction with the supply voltage correction switched on. This applies analogously to overvoltage.

0: - (switched off, no supply voltage correction) / 1: Phase 1 / 2: Phase 2 / 3: Phase 3.

This setting determines to which phase conductor the heating of this output is connected.

With three-phase current heating switched by an output, using all 3 phase conductors for correction is not possible. One phase conductor must be selected.

SSR alarm

This alarm is the output of an error message determined using the heating current monitoring function. In one heating current monitoring phase, all outputs are switched off and no (heating) current may flow. If there is a current flow, this means a short circuit of the switching element.

Normal voltage [V] (U.norm)

During heating current measurement, a low supply voltage may cause faulty measurements and thus faulty alarms, in particular, if the current limit values are very close to the rated current. This setting offers the possibility to activate the supply voltage correction and to determine the "normal voltage" simultaneously. A separate voltage transformer transforms the supply voltage into a DC voltage with a 100% value of 7V DC. Fill in the supply voltage **rating** to produce these 7V DC (normally 230V). Any supply voltage variation generates a secondary voltage \neq 7V. This value is used for correction of the current value.

| In-/Outputs | | |
|---------------------|-------|---|
| Name | Type | Description |
| Ext_TC | Float | Temperature of the external TC |
| Phase_1 | Float | Voltage of phase 1 |
| Phase_2 | Float | Voltage of phase 2 |
| Phase_3 | Float | Voltage of phase 3 |
| F_Ext_TC | Bool | External TC fail |
| F_Pha_1 | Bool | Fail voltage phase 1 |
| F_Pha_2 | Bool | Fail voltage phase 2 |
| F_Pha_3 | Bool | Fail voltage phase 3 |
| outp_1 ... outp_8 | Bool | Signal at output 1 of the module |
| HC_en_1 ... HC_en_8 | Bool | Enable heating current monitoring at output 1 |

| Name | Type | Description |
|-----------------------|-------|---|
| Slot | Float | Own block number |
| Y_1 ... Y_4 | Float | Conditioned measured value 1 |
| HC_val_1 ... HC_val_8 | Float | Heating current measured value |
| fail_1 ... fail_4 | Bool | Signals a sensor error at the input (short-circuit, wrong polarity, ..) |
| HC_AI_1 ... HC_AI_8 | Bool | Heating current alarm |

| | | |
|-----------------------|------|--|
| SSR_AI_1 ... SSR_AI_8 | Bool | Short-circuit alarm |
| valid | Bool | The communication to VARIO I/O system (hardware) is established. Measured values are valid. |

Parameter

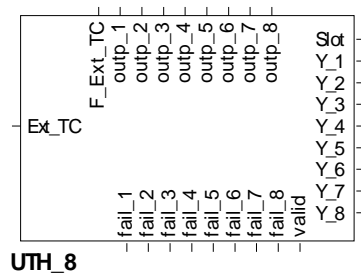
| ID | Name | Type | Description | Access | Default | Range | Off |
|-----------------------|-------------------|-------|--|--------|---------|------------|-----|
| X0_1 ... X0_4 | Scale start 1 | Float | Physical value at 0% | r/w | 0.0 | | |
| X100_1 ... X100_4 | Scale end 1 | Float | Physical value at 100% | r/w | 100.0 | | |
| Xfail_1 ... Xfail_4 | Substitute value1 | Float | Substitute value with sensor fail | r/w | 9999.0 | | |
| Tfm_1 ... Tfm_4 | Filter time 1 [s] | Float | Filter time constant of signal processing [s] | r/w | 0.5 | >0.0 | |
| HC_Lim_1 ... HC_Lim_8 | HC Limit 1 | Float | Heating current limit [A] (with switch-off value -32000) | r/w | off | 1.5 ... 50 | ja |

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|-----------------|---------------------|----------|--|--------|---------|-------------|-----|
| Unit | Unit | Enum | Engin. unit of measured value | r/w | 0 | | |
| | | °C | Unit = °C, degrees Celsius | | 0 | | |
| | | °F | Unit = °F, degrees Fahrenheit | | 1 | | |
| Hc_Cy | HC cycle time [s] | Int | Cycle time of heating current monitoring [s] | r/w | 10 | 1 ... 3000 | |
| Hc_Ti | HC test [n*100ms] | Int | Switch-on cycles of heating current monitoring [100ms] | r/w | 2 | 2 ... 255 | |
| U_norm | standard voltage[V] | Int | Standard voltage [V] | r/w | 230 | 1 ... 32000 | |
| Typ_1 ... Typ_4 | Sensor type 1 | Enum | Sensor type | r/w | 0 | | |
| | | Not used | The input is not used. | | 0 | | |
| | | Typ L | Typ L (-100...900°C, -148...1652°F), Fe-CuNi DIN | | 1 | | |
| | | Typ J | Typ J (-100...900°C, -148...1652°F), Fe-CuNi DIN | | 2 | | |
| | | Typ K | Typ K (-100...1350°C, -148...2462°F), NiCr-Ni | | 3 | | |
| | | Typ N | Typ N (-100...1300°C, -148...2372°F), Nicrosil-Nisil | | 4 | | |
| | | Typ S | Typ S (0...1760°C, 32...3200°F), PtRh-Pt10% | | 5 | | |
| | | Typ R | Typ R (0...1760°C, 32...3200°F), PtRh- | | 6 | | |

| | | | | | | | |
|--------------------------|------------------|---------------------|---|-----|------|----------------|--|
| | | | Pt13% | | | | |
| | | Typ T | Typ T (-200...400°C, -328...752°F), Cu-CuNi | | 7 | | |
| | | Typ C | Typ C (0...2315°C, 32...4199°F), W5%Re-W26%Re | | 8 | | |
| | | Typ D | Typ D (0...2315°C, 32...4199°F), W3%Re-W25%Re | | 9 | | |
| | | Typ E | Typ E (-100...1000°C, -148...1832°F), NiCr-CuNi | | 10 | | |
| | | Typ B | Typ B (0/100...1820°C, 32/172 ... 3308°F), PtRh-Pt6% | | 11 | | |
| | | 0...70mV | Voltage : 0...70mV | | 47 | | |
| Fail_1 ... Fail_4 | Fail | Enum | Signal response on sensor fail | r/w | 1 | | |
| | | Disabled | Switched off, Inp = 0. With fail, the input is set to zero. | | 0 | | |
| | | Substitute value | Substitute value Inp = Xfail. With fail, the substitute value is used. | | 1 | | |
| STK_1 ... STK_4 | Temp. compens. 1 | Enum | Location of temperature compensation | r/w | 0 | | |
| | | Internal CJC | Internal temperature compensation | | 0 | | |
| | | External CJC | External temperature compensation | | 1 | | |
| Hc_AiM_1 ... Hc_AiM_8 | HC alarm mode 1 | Enum | Type of heating current alarm | r/w | 1 | | |
| | | Max + short circuit | Activate the overload and short-circuit monitoring. Overload occurs when current I exceeds the heating current limit value. | | 0 | | |
| | | Min + short circuit | Activate the break and short circuit monitoring. Break occurs when current I is lower than the heating current limit. | | 1 | | |
| Phase_1 ... Phase_8 | Phase | Enum | Used external conductor | r/w | 0 | | |
| | | ---- | Not active | | 0 | | |
| | | phase 1 | External conductor 1 | | 1 | | |
| | | phase 2 | External conductor 2 | | 2 | | |
| | | phase 3 | External conductor 3 | | 3 | | |
| Tr_Rat_1 ... Tr_Rat_8 | Transform ratio1 | Int | Transformer ratio | r/w | 1000 | 1 ... 30000 | |

III-17.25 UTH_8 (I/O-module with 8 analog in- and 8 digital outputs (No. 167))



UTH_8

Abb. 636

The function block is used for configuration and parameter setting of a module with 4 analog inputs for thermocouples and 8 digital outputs. Connection is in 2-wire technology with screening. Moreover, the module has 1 heating current input (see note!)



NOTE

To use the heating current input for totalizing current converter, the *UTH_8_HC* function block must be used.

The conditioned measured values and the measured value status signals are available at its outputs. The output values must be applied to inputs **outp_1** ... **outp_8**.

Output **Slot** must be soft-wired with the input of a **KS108_VARIO_BK_ETH** type function block.

Unit

Unit

°C, °F or without unit can be selected. Process values are converted automatically when changing units. All other values (set-points, limit values...) must be converted manually.

Scaling

Analog input 0 % (X0_i) and 100 % (X100_i) scaling

These two values are used for scaling of a digital input. Process value corrections, which are e.g. required due to unfavourable sensor installation, or to correct the tolerances of several sensors, can be made using the 2-point correction method. If the sensor is a transmitter with standard signal output (e.g. weight -> 0...10V), scaling of the physical quantity can be determined for this input.

In this case, the output span start value **X0_i**, i.e. the span start in units of the physical quantity (e.g. 0 t) is assigned to the transmitter output span start (electrical signal span start, e.g. 4 mA).

The output span end **X100_i**, i.e. the span end in units of the physical quantity (e.g. 10 t) is assigned to the transmitter output span end (electrical signal span end, e.g. 20 mA).

Internal and external temperature compensation

Internal TC :

If the internal temperature compensation is selected, the internal (built-in) temperature compensation is active. Other settings are not required.

External TC via thermostat:

A thermostat terminal box is used. Take the compensating lead up to this box, and use copper lead between the box and the Vario station. The temperature of this thermostat is known (mostly 50 °C). The value is applied to input "Ext_TC" of the function block and used for calculation of the external TC.

| In-/Outputs | | |
|--------------------|-------------|----------------------------------|
| Name | Type | Description |
| Ext_TC | Float | Temperature of the external TC |
| F_Ext_TC | Bool | External TC fail |
| outp_1 ... outp_8 | Bool | Signal at output 1 of the module |

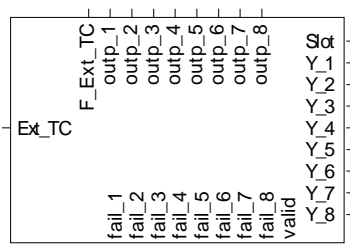
| Name | Type | Description |
|-------------------|-------------|---|
| Slot | Float | Own block number |
| Y_1 ... Y_8 | Float | Conditioned measured value 1 |
| fail_1 ... fail_8 | Bool | Signals a sensor error at the input (short-circuit, wrong polarity, ..) |
| valid | Bool | The communication to VARIO I/O system (hardware) is established. Measured values are valid. |

| Parameter | | | | | | | |
|---------------------|-------------------|-------------|---|---------------|----------------|--------------|------------|
| ID | Name | Type | Description | Access | Default | Range | Off |
| X0_1 ... X0_8 | Scale start 1 | Float | Physical value at 0% | r/w | 0.0 | | |
| X100_1 ... X100_8 | Scale end 1 | Float | Physical value at 100% | r/w | 100.0 | | |
| Xfail_1 ... Xfail_8 | Substitute value1 | Float | Substitute value with sensor fail | r/w | 9999.0 | | |
| Tfm_1 ... Tfm_8 | Filter time 1 [s] | Float | Filter time constant of signal processing [s] | r/w | 0.5 | >0.0 | |

| Configuration | | | | | | | |
|----------------------|---------------|-------------|--|---------------|----------------|--------------|------------|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Unit | Unit | Enum | Engin. unit of measured value | r/w | 0 | | |
| | | °C | Unit = °C, degrees Celsius | | 0 | | |
| | | °F | Unit = °F, degrees Fahrenheit | | 1 | | |
| Typ_1 ... Typ_8 | Sensor type 1 | Enum | Sensor type | r/w | 0 | | |
| | | Not used | The input is not used. | | 0 | | |
| | | Typ L | Typ L (-100...900°C, -148...1652°F), Fe-CuNi DIN | | 1 | | |

| | | | | | | | |
|----------------------|---------------------|---------------------|---|-----|----|--|--|
| | | Typ J | Typ J (-100...900°C, -148...1652°F), Fe-CuNi DIN | | 2 | | |
| | | Typ K | Typ K (-100...1350°C, -148...2462°F), NiCr- Ni | | 3 | | |
| | | Typ N | Typ N (-100...1300°C, -148...2372°F), Nicrosil- Nisil | | 4 | | |
| | | Typ S | Typ S (0...1760°C, 32...3200°F), PtRh- Pt10% | | 5 | | |
| | | Typ R | Typ R (0...1760°C, 32...3200°F), PtRh- Pt13% | | 6 | | |
| | | Typ T | Typ T (-200...400°C, -328...752°F), Cu- CuNi | | 7 | | |
| | | Typ C | Typ C (0...2315°C, 32...4199°F), W5%Re- W26%Re | | 8 | | |
| | | Typ D | Typ D (0...2315°C, 32...4199°F), W3%Re- W25%Re | | 9 | | |
| | | Typ E | Typ E (-100...1000°C, -148...1832°F), NiCr- CuNi | | 10 | | |
| | | Typ B | Typ B (0/100...1820°C, 32/172 ... 3308°F), PtRh-Pt6% | | 11 | | |
| | | 0...70mV | Voltage : 0...70mV | | 47 | | |
| Fail_1 ... Fail_8 | Fail | Enum | Signal response on sensor fail | r/w | 1 | | |
| | | Disabled | Switched off, Inp = 0. With fail, the input is set to zero. | | 0 | | |
| | | Substitute value | Substitute value Inp = Xfail. With fail, the substitute value is used. | | 1 | | |
| STK_1 ... STK_8 | Temp. compens. 1 | Enum | Location of temperature compensation | r/w | 0 | | |
| | | Internal CJC | Internal temperature compensation | | 0 | | |
| | | External CJC | External temperature compensation | | 1 | | |

III-17.26 UTH_8_HC (I/O-module with 8 analog in-, 8 digital outputs and heating current (No. 160))



UTH_8

Abb. 637

The function block is used for configuration and parameter setting of a module with 8 analog inputs for thermocouples and 8 digital outputs. Connection is in 2-wire technology with screening. Moreover, the module has 1 heating current input (see note!)



NOTE!

If the heating current input for totalizing current converter is **not** used, the less complicated UTH_8 function block can be used.

The conditioned measured values and the measured value status signals are available at its outputs. The output values must be applied to inputs **outp_1** ... **outp_8**.

Output **Slot** must be soft-wired with the input of a **KS108_VARIO_BK_ETH** type function block.

Unit

Unit

°C, °F or without unit can be selected. Process values are converted automatically when changing units. All other values (set-points, limit values...) must be converted manually.

Scaling

Analog input 0 % (X0_i) and 100 % (X100_i) scaling

These two values are used for scaling of a digital input. Process value corrections, which are e.g. required due to unfavourable sensor installation, or to correct the tolerances of several sensors, can be made using the 2-point correction method. If the sensor is a transmitter with standard signal output (e.g. weight -> 0...10V), scaling of the physical quantity can be determined for this input.

In this case, the output span start value **X0_i**, i.e. the span start in units of the physical quantity (e.g. 0 t) is assigned to the transmitter output span start (electrical signal span start, e.g. 4 mA).

The output span end **X100_i**, i.e. the span end in units of the physical quantity (e.g. 10 t) is assigned to the transmitter output span end (electrical signal span end, e.g. 20 mA).

Internal and external temperature compensation

Internal TC :

If the internal temperature compensation is selected, the internal (built-in) temperature compensation is active. Other settings are not required.

External TC via thermostat:

A thermostat terminal box is used. Take the compensating lead up to this box, and use copper lead between the box and the Vario station. The temperature of this thermostat is known (mostly 50 °C). The value is applied to input "Ext_TC" of the function block and used for calculation of the external TC.

Heating current monitoring

The heating current monitoring function of the VARIO modules can be used to monitor all outputs of a module by means of a current transformer. The relevant totalizing current input is provided on the I/O extension modules.

The heating currents can be monitored at an adjustable time interval. Independent of the input signals, all outputs configured accordingly except the one to be monitored are switched off and the heating current is measured. The interval for heating current monitoring is adjustable per module for all its outputs.

Heating current monitoring interval [sec] (Hc.Cy):

A monitoring measurement is made at intervals of the adjusted number of seconds. This means that each individual output is measured again after outputs + 1 (for short circuit measurement) interval. This interval should not be too short, because a heater is a relatively insensitive product and does not need frequent checking. Moreover, switching off the other outputs during measurement might impair the control quality, which is dependent on these outputs. This applies particularly to fast processes.

Duty cycle for heating current monitoring [100 ms] (Hc.Ti)

This is the time during which the output to be checked is energized. All other outputs pertaining to this module are switched off. This value can be increased in steps of 100 ms from its minimum setting of 200 ms. The actual measurement time is 100 ms and starts at the end of this adjusted time. The preceding time is the signal settling time. The value to be adjusted is dependent on:

1. **The switching element type** . Contactors and, in particular, power contactors, have a longer cycle time and require a higher number of duty cycles. For solid-state relays, this value may be neglected.
2. **The settling time of the current transformer**. On transformers with built-in transmitter, settlement of the output value takes longer.

Generally, this setting should be as low as possible, because the other values are switched off and the function (e.g. control) is disturbed during this time.

Heating current alarm (Hc.ALMode)

This parameter determines, if the heating current must be monitored for too high (overload) or too low (interruption) values. With both settings, the actuator is monitored for short circuit. Normally, the parameter is set for low load, because the resistance of heating elements tends to increase over the time, or they even burn out.

Current transformation ratio (Tr.Rat)

For heating current display with the right scaling (e.g. in A) in HC.Me, the transformation ratio of the connected current transformer must be set. Example: The transformer has a transformation ratio of 1000:1. 1000 must be set under parameter Tr.Rat . With a 50 mA AC input current flowing into the heating current input, a heating current value of 50.0 A is output.

Phase conductor (Phase1 ... Phase3)

The heating current measurement can be corrected using a value derived from the phase conductor voltage. A low supply voltage would cause a low current and trigger a heating current (fail) alarm. The (low) heating

current is corrected by the per cent value of voltage reduction with the supply voltage correction switched on. This applies analogously to overvoltage.

0: - (switched off, no supply voltage correction) / 1: Phase 1 / 2: Phase 2 / 3: Phase 3.

This setting determines to which phase conductor the heating of this output is connected.

With three-phase current heating switched by an output, using all 3 phase conductors for correction is not possible. One phase conductor must be selected.

SSR alarm

This alarm is the output of an error message determined using the heating current monitoring function. In one heating current monitoring phase, all outputs are switched off and no (heating) current may flow. If there is a current flow, this means a short circuit of the switching element.

Normal voltage [V] (U.norm)

During heating current measurement, a low supply voltage may cause faulty measurements and thus faulty alarms, in particular, if the current limit values are very close to the rated current. This setting offers the possibility to activate the supply voltage correction and to determine the "normal voltage" simultaneously. A separate voltage transformer transforms the supply voltage into a DC voltage with a 100% value of 7V DC. Fill in the supply voltage **rating** to produce these 7V DC (normally 230V). Any supply voltage variation generates a secondary voltage $\neq 7V$. This value is used for correction of the current value.

| In-/Outputs | | |
|---------------------|-------|---|
| Name | Type | Description |
| Ext_TC | Float | Temperature of the external TC |
| Phase_1 | Float | Voltage of the phase 1 |
| Phase_2 | Float | Voltage of the phase 2 |
| Phase_3 | Float | Voltage of the phase 3 |
| F_Ext_TC | Bool | External TC fail |
| F_Pha_1 | Bool | Fail voltage phase 1 |
| F_Pha_2 | Bool | Fail voltage phase 2 |
| F_Pha_3 | Bool | Fail voltage phase 3 |
| outp_1 ... outp_8 | Bool | Signal at the output 1 of the module |
| HC_en_1 ... HC_en_8 | Bool | Enable heating current monitoring at output 1 |

| Name | Type | Description |
|-----------------------|-------|---|
| Slot | Float | Own block number |
| Y_1 ... Y_8 | Float | Conditioned measured value 1 |
| HC_val_1 ... HC_val_8 | Float | Heating current measured value |
| fail_1 ... fail_8 | Bool | Signals a sensor error at the input (short-circuit, wrong polarity, ..) |
| HC_AI_1 ... HC_AI_8 | Bool | Heating current alarm |
| SSR_AI_1 ... SSR_AI_8 | Bool | Short-circuit alarm |
| valid | Bool | The communication to VARIO I/O system (hardware) is established. Measured values are valid. |

| Parameter | | | | | | | |
|--------------------------|-------------------|-------|--|--------|---------|---------------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| X0_1 ... X0_8 | Scale start 1 | Float | Physical value at 0% | r/w | 0.0 | | |
| X100_1 ... X100_8 | Scale end 1 | Float | Physical value at 100% | r/w | 100.0 | | |
| Xfail_1 ... Xfail_8 | Substitute value1 | Float | Substitute value with sensor fail | r/w | 9999.0 | | |
| Tfm_1 ... Tfm_8 | Filter time 1 [s] | Float | Filter time constant of signal processing [s] | r/w | 0.5 | >0.0 | |
| HC_Lim_1 ... HC_Lim_8 | HC Limit 1 | Float | Heating current limit [A] (with switch-off value -32000) | r/w | off | 1.5 ... 50 | ja |

| Configuration | | | | | | | |
|--------------------|---------------------|----------|--|--------|---------|----------------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Unit | Unit | Enum | Engin. unit of measured value | r/w | 0 | | |
| | | °C | Unit = °C, degrees Celsius | | 0 | | |
| | | °F | Unit = °F, degrees Fahrenheit | | 1 | | |
| Hc_Cy | HC Cycle time[s] | Int | Cycle time of heating current monitoring [s] | r/w | 10 | 1 ... 3000 | |
| Hc_Ti | HC test [n*100ms] | Int | Switch-on cycles of heating current monitoring [100ms] | r/w | 2 | 2 ... 255 | |
| U_norm | standard voltage[V] | Int | Standard voltage [V] | r/w | 230 | 1 ... 32000 | |
| Typ_1 ... Typ_8 | Sensor type 1 | Enum | Sensor type | r/w | 0 | | |
| | | Not used | The input is not used. | | 0 | | |
| | | Typ L | Typ L (-100...900°C, -148...1652°F), Fe-CuNi DIN | | 1 | | |
| | | Typ J | Typ J (-100...900°C, -148...1652°F), Fe-CuNi DIN | | 2 | | |
| | | Typ K | Typ K (-100...1350°C, -148...2462°F), NiCr-Ni | | 3 | | |
| | | Typ N | Typ N (-100...1300°C, -148...2372°F), Nicrosil-Nisil | | 4 | | |
| | | Typ S | Typ S (0...1760°C, 32...3200°F), PtRh-Pt10% | | 5 | | |
| | | Typ R | Typ R (0...1760°C, 32...3200°F), PtRh-Pt13% | | 6 | | |
| | | Typ T | Typ T (-200...400°C, -328...752°F), Cu-CuNi | | 7 | | |

| | | | | | | | |
|-----------------------------|----------------------|------------------------|--|-----|------|----------------|--|
| | | Typ C | Typ C (0...2315°C, 32...4199°F), W5%Re-W26%Re | | 8 | | |
| | | Typ D | Typ D (0...2315°C, 32...4199°F), W3%Re-W25%Re | | 9 | | |
| | | Typ E | Typ E (-100...1000°C, -148...1832°F), NiCr-CuNi | | 10 | | |
| | | Typ B | Typ B (0/100...1820°C, 32/172 ... 3308°F), PtRh-Pt6% | | 11 | | |
| | | 0...70mV | Voltage : 0...70mV | | 47 | | |
| Fail_1 ... Fail_8 | Fail | Enum | Signal response on sensor fail | r/w | 1 | | |
| | | Disabled | Switched off, Inp = 0. With fail, the input is set to zero. | | 0 | | |
| | | Substitute value | Substitute value Inp = Xfail. With fail, the substitute value is used. | | 1 | | |
| STK_1 ... STK_8 | Temp. compens. 1 | Enum | Location of temperature compensation | r/w | 0 | | |
| | | Internal CJC | Internal temperature compensation | | 0 | | |
| | | External CJC | External temperature compensation | | 1 | | |
| Hc_AiM_1 ... Hc_AiM_8 | HC alarm mode 1 | Enum | Type of the heating current alarm | r/w | 1 | | |
| | | Max + short circuit | Activate the overload and short-circuit monitoring. Overload occurs when current I exceeds the heating current limit value. | | 0 | | |
| | | Min + short circuit | Activate the break and short circuit monitoring. Break occurs when current I is lower than the heating current limit. | | 1 | | |
| Phase_1 ... Phase_8 | Phase | Enum | Used external conductor | r/w | 0 | | |
| | | ---- | Not active | | 0 | | |
| | | phase 1 | External conductor 1 | | 1 | | |
| | | phase 2 | External conductor 2 | | 2 | | |
| | | phase 3 | External conductor 3 | | 3 | | |
| Tr_Rat_1 ... Tr_Rat_8 | Transform ratio 1 | Int | Transformer ratio | r/w | 1000 | 1 ... 30000 | |

III-18 rail line I/O-Module

III-18.1 General

A **rail line** RL 400 I/O system comprises a bus coupler and one or several **rail line** RL 400 I/O modules. A vario I/O system with the required internal electrical connections is set up by connecting the **rail line** RL 400 I/O modules to the bus coupler.

In the engineering, the hardware combination of **rail line** RL 400 I/O system is shown accordingly in the form of function blocks.

All **rail line** RL 400 I/O function blocks have a **Slot** output which must be soft-wired to the relevant **Slot** input of the bus coupler function block - whereby the number of the hardware position of the module determines the number of the input of the bus coupler.

Survey of **rail line** RL 400 I/O modules and related function blocks:

| Functionsblock | Hardware | Hint |
|---|--------------|---|
| RAIL_BK_CAN_10, RAIL_BK_CAN_20, RAIL_BK_CAN_40, RAIL_BK_CAN_62 | RL40-CANopen | DeviceNet on demand |
| RL_422 | RL 422-0 | Analog inputs, 4 x I / U / TPS / Pot. Universal module, 16 Bit |
| RL_423-0 | RL 423-0 | RTD 4 x Pt100 galvanic isolated |
| RL_423-1 | RL 423-1 | RTD 4 x Pt1000 galvanic isolated |
| RL_423-2 | RL 423-2 | RTD 4 x Pt100/Pt1000 galvanic isolated |
| RL_424-0 | RL 424-0 | 2 x TC galvanic isolated |
| RL_424-1 | RL 424-1 | 2 x TC / O2 (mV) |
| RL_424-2 | RL 424-2 | 4 x TC galvanic isolation 2/2 |
| RL_442 | RL 442-0 | Digital inputs 2 x 4 x 24 VDC (pnp) |
| RL_442 | RL 442-1 | Digital inputs 2 x 4 x -24 VDC (npn) |
| RL_442 | RL 442-2 | Digital inputs 2 x 4 x contact (potential-free) |
| RL_443 | RL 443-0 | 4 x relay (115/230/400V AC) |
| RL_451 | RL 451-0 | Digital outputs 2 x 4 x 24 VDC/2A |
| RL_451 | RL 451-1 | Digital outputs 2 x 4 x 24 VDC/2A (free wheeling diode) |
| RL_452 | RL 452-0 | Relay outputs 4 x 230 VDC/5A |
| RL_461 | RL 461-0 | Combi module, 2 x AI ($\pm U / \pm I$, 16 Bit), differential inputs; 2 x AO ($\pm U / \pm I$, 12 Bit) |
| CI45 | CI45 | Transmitter UNIFLEX CI45 |
| KS45 | KS45 | Universal controller KS45 |
| TB45 | TB45 | Temperature limiter TB45 |

III-18.2 I/O - error coding

The following information is given at the output **IO Error** of any **rail line** I/O modules (except bus coupler, see RAIL_BK_CAN_xx):

| | |
|----------|--|
| IO_Error | Error status of communication with the rail line system |
| 0 | No error |
| 1 | Initialization |
| 2 | Connection failure |
| 3 | Module is not plugged into the system |
| 4 | Wrong module plugged into the system |
| 200 | The number of modules connected to the bus coupler does not match the engineering (bigger or smaller). |
| 2xy | The module number xy connected to the bus coupler is different from the module in the engineering. |

III-18.3 Short-form instructions for building up a rail line RL 400 I/O system: Hardware



NOTE

Follow the safety hints and the detailed instructions of the rail line system.

1. Installing the bus connector on the top-hat rail:

Shift the bus connectors for inserting the bus couplers and the modules of the RL 400 I/O system together and snap them into position.



Abb. 638: RL 400 system with I/O modules on a top-hat rail

2. Install the modules : Determine the system structure.
3. Make the hardware settings: CAN bus coupler: address switch, Baud rate.

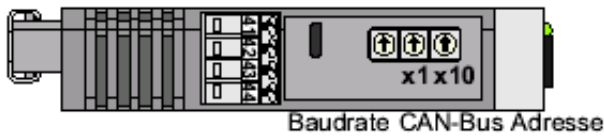
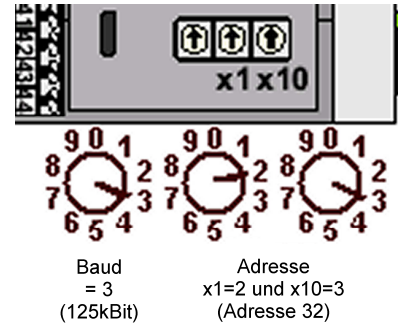


Abb. 639: Example for bus coupler:
CANrail field bus coupler- CAN bus seen from the bottom with socket for the supply voltage connector and rotary selector for Baud rate and CAN bus address.



NOTE
A terminating resistor must be fitted at the start and end points of a CAN bus topology. For this reason, a terminating resistor for the CAN bus must be used, if the CAN bus coupler is installed at the start or at the end of the CAN bus topology.

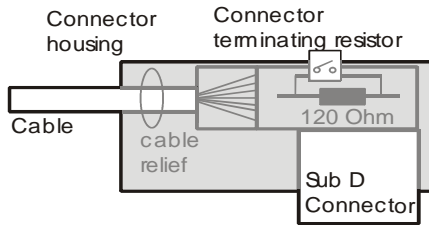


Abb. 640: CAN connector with switchable terminating resistor. There are also intermediate plugs with integrated terminating resistor for CAN.

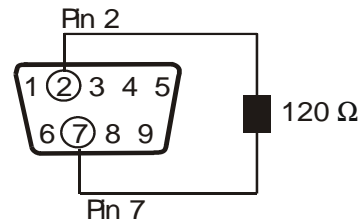


Abb. 641: Circuit diagram for CAN terminating resistor.

4. Wiring procedure:

- Connect the DC supply voltage.
- Connect the bus and install the terminating resistors: terminate the CAN bus with at least 1 (2 are better) terminating resistor (see section "Preparing the KS108 hardware for CAN bus" below).
- Switch on the supply voltage.

5. Activate system scan:

Press the "Conf" button on the bus coupler during approx. 5 seconds). You will hear an acoustic feedback. The bus coupler reads the modules, determines the module type and module position, assigns the (automatic) addresses in the RL system and sets the terminating resistor of the RL system (not the CAN terminating resistor) at the last module. The existing system configuration is adopted in the bus coupler.

- If there are any faults, remove them (e.g. wiring).
- When the bus identification is finished without detecting an error, the uppermost LED on the bus coupler is lit = "ok".

6. Preparing the KS108 hardware for CAN bus:

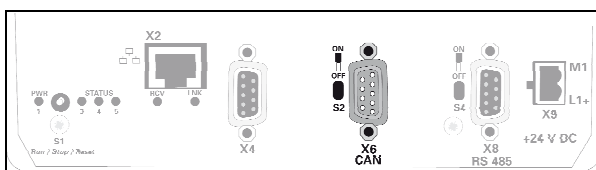


Fig. 642: CAN interface

The CAN interface corresponds to the ISO 11898 standard and can be operated to a maximum baud rate of 1Mbit/s. The interface has an additional isolating element.

**NOTE!**

A terminating resistor must be located at the beginning and end of a CAN bus topology. Therefore: If the device is at the beginning or end of the CAN bus topology then the terminating resistor must be activated with switch (Fig. 642/6) for the CAN bus. To do this move the switch up to the "ON" position.

Pinning:

| PIN | Description |
|-----|---------------------|
| 1 | NC (do not connect) |
| 2 | CAN_L |
| 3 | CAN_GND |
| 4 | NC (do not connect) |
| 7 | CAN_H |

III-18.4 Short-form instructions for building up a rail line RL 400 I/O system: Software and debugging

**NOTE!**

Follow the safety hints and the detailed instructions of the rail line system.

7. Creating or displaying an engineering:

Select the relevant function block for each module and place it in the engineering. Soft-wire the "Slot" module outputs to the right "Slot_(No.)" inputs of the bus coupler function block according to the module position number in the installed system. Remaining soft-wiring procedure: Inputs, outputs, error outputs. Set the bus coupler and module parameters in the "Run" mode.

**NOTE!**

To ensure correct functioning of the communication, the CAN settings for address and Baud rate in the engineering and in the CAN bus coupler must be identical.

8. Downloading the engineering:

For transmission of the configuration parameters (settings) to the RL400 modules, the "Config" input of the bus coupler function block must be set to "1" for downloading. BlueDesign must be in the "Run" mode.

Execute the following steps:

- "Load the user program".
- Select the "target system": Check the settings".
- "Connect". "OK". The user program is loaded into KS108 and activated. The on-line connection is built up.
- After transmission, reset the "Config" input of the bus coupler function block to "0".

9. Checking the engineering functions:

Determine the process values and signals. Set the outputs. Generate the alarms, etc.

10. Subsequent configuration change:

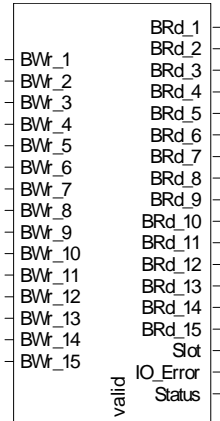
For subsequent configuration changing, a new download is required. For this download, set the "Config" input of the bus coupler function block to "1" again and reset it to "0" after transmission.



NOTE!

The settings of the individual modules (function modules CI45, KS45, TB45, and bus coupler as well as the "normal" RL I/O modules) are saved in the modules and available also after a restart. This means that download into the RL system is required only for changes of these configuration settings or after system changes, e.g. module replacement.

III-18.5 Railline CI 45 (Universal Transmitter (No. 285))



CI45

Abb. 643

The function block is used to connect a transmitter module.

UNIFLEX CI 45 transmitters are suitable for precise, low-priced signal measurement and signal conversion solutions. Each CI 45 has at least one universal input, one universal output and one relay. Optionally, the transmitter can be equipped with an additional relay. Galvanic isolation is provided between inputs, outputs, supply voltage and communication interfaces.



NOTE!

As the parameter set of this module is very extensive, setting of configurations and parameter is done with the program BlueControl®.

The data transferred to the inputs BWr_1 to BWr_15 is transmitted to the module CI45. The data 1 to 15 is defined in the engineering of CI45, which is done with BlueControl® (in "Bus data (write)", see Help of BlueControl® "bus data").

The data given at the outputs BRd_1 to BRd_15 are received of the module CI45. The data 1 to 15 is defined in the engineering of CI45, which is done with BlueControl® (in "Bus data (read)", see Help of BlueControl® "bus data").

Output **Slot** must be soft-wired with the input of a **RL_BK_CAN_(No.)** type function block, according to module position.

Coding of output IO-Error

See table in the general part of the description of RL 400 system. Only if no error message is pending at output IO_Error, data is exchanged with the module in the rail line system.

Coding of output Status

The following table describes the module status information. This value is valid only, if no error message is pending at output IO_Error, because this is the prerequisite of data exchange with the module in the rail line system. The information is bit-coded.

| Bit | Description |
|-----|--|
| 0 | Alarm 1 = 1, if error is detected at one of the measured inputs (e.g. sensor break). |
| 1 | Alarm 2 = 1, if at least one alarm is pending (e.g. out-of-limits). |
| 2 | Status 1: Internal error of the module. Various messages are grouped together. The exact error message is displayed on the instrument. |
| 3 | Wrong_output_value; indicates that an out-of-limits process value for a channel was transferred. |



NOTE!

For additional information: see operating instructions of the CI45-module.

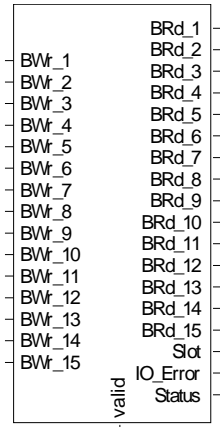
In-/Outputs

| Name | Type | Description |
|-------------------|-------|---|
| BWr_1 ... BWr_15 | Float | 1st datum to be written via bus |
| Name | Type | Description |
| BRd_1 ... BR_d_15 | Float | 1st datum read via bus |
| IO_Error | Float | Type of the communication error to the module in the I/O system |
| Slot | Float | Own block number |
| Status | Float | Status message of the module |
| valid | Bool | The communication to VARIO I/O system (hardware) is established. Measured values are valid. |

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|--------|----------------|------------|--|--------|---------|-------|-----|
| ErrBeh | Error behavior | Enum | Behaviour in error condition | r/w | 0 | | |
| | | Last value | Error behaviour: Use of the last value is continued. | | 0 | | |
| | | Zero | Error behaviour: Value zero (value is set to "0"). | | 1 | | |

III-18.6 Railline KS 45 (Universal Controller (No. 286))



KS45

Abb. 644

The function block is used to connect a universal controller module.

KS 45 universal controllers are suitable for precise, low-priced control solutions in all fields of industry. Simple on/off control, PID control and three-point stepping control can be selected. The process value signal is connected via a universal input. A second analog input can be used for heating current measurement or as an external set-point input. A KS 45 has at least one universal input and two switching outputs. Optionally, the controller may be equipped with a universal output or with optocoupler outputs. The universal output can be configured as a continuous output with current or voltage, for controlling solid-state relays or for transmitter supply. Galvanic isolation between inputs, outputs, supply voltage and communication interfaces is provided.

i *NOTE!*
 As the parameter set of this module is very extensive, setting of configurations and parameter is done with the program BlueControl®.

The data transferred to the inputs BWr_1 to BWr_15 is transmitted to the module KS45. The data 1 to 15 is defined in the engineering of KS45, which is done with BlueControl® (in "Bus data (write)", see Help of BlueControl® "bus data").

The data given at the outputs BRd_1 to BRd_15 are received of the module KS45. The data 1 to 15 is defined in the engineering of KS45, which is done with BlueControl® (in "Bus data (read)", see Help of BlueControl® "bus data").

Output **Slot** must be soft-wired with the input of a **RL_BK_CAN_(No.)** type function block.

Coding of output IO-Error

See table in the general part of the description of RL 400 system. Only if no error message is pending at output IO_Error, data is exchanged with the module in the rail line system.

Coding of output Status

The following table describes the module status information. This value is valid only, if no error message is pending at output IO_Error, because this is the prerequisite of data exchange with the module in the rail line system. The information is bit-coded.

| Bit | Description |
|-----|--|
| 0 | Alarm 1 = 1, if there is an error on one of the measured inputs (e.g. sensor break). |

| | |
|---|--|
| 1 | Alarm 2 = 1 , if at least one alarm is pending (e.g. out-of-limits). |
| 2 | Status 1: Internal error of the module. Various messages are grouped together. The exact error message is displayed on the instrument. |
| 3 | Wrong_output_value; indicates that an out-of-limits process value for a channel was transferred. |

**NOTE!**

For additional information: see operating instructions of the KS 45-module.

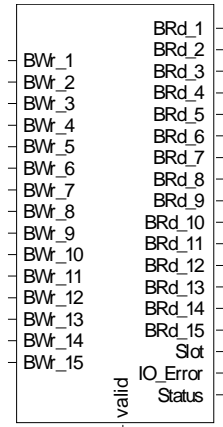
In-/Outputs

| Name | Type | Description |
|------------------|-------|--|
| BWr_1 ... BWr_15 | Float | 1st datum to be written via bus |
| Name | Type | Description |
| BRd_1 ... BRd_15 | Float | 1st datum read via the bus |
| Slot | Float | Own block number |
| IO_Error | Float | Type of the communication error to the module in the I/O system |
| Status | Float | Status message of the module |
| valid | Bool | The communication to VARIO I/O system (hardware) is established. Measured values are valid. |

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|--------|----------------|------------|--|--------|---------|-------|-----|
| ErrBeh | Error behavior | Enum | Behaviour in error condition | r/w | 0 | | |
| | | Last value | Error behaviour: Use of the last value is continued. | | 0 | | |
| | | Zero | Error behaviour: Value zero (value is set to "0"). | | 1 | | |

III-18.7 Railline TB 45 (Temperaturelimiter/ -monitor (No. 287))



TB45

Abb. 645

The function block is used to connect a temperature limiter module.

The TB 45 temperature limiter is used for process monitoring. It provides input signal measurement, out-of-limits signalling and switch-off functions. The unit may be used for heating and cooling processes. It may be configured as an electronic temperature limiter, as a temperature monitor or as a limit signaller. Each TB 45 has at least one universal input, one limit contact and one pre-alarm contact. Optionally, the controller may be equipped with a universal output and a second universal input. Galvanic isolation between inputs, outputs, supply voltage and communication interfaces is provided.

i *NOTE!*
 As the parameter set of this module is very extensive, setting of configurations and parameter is done with the program BlueControl®.

The data transferred to the inputs BWr_1 to BWr_15 is transmitted to the module TB45. The data 1 to 15 is defined in the engineering of TB45, which is done with BlueControl® (in "Bus data (write)", see Help of BlueControl® "bus data").

The data given at the outputs BRd_1 to BRd_15 are received of the module TB45. The data 1 to 15 is defined in the engineering of TB45, which is done with BlueControl® (in "Bus data (read)", see Help of BlueControl® "bus data").

Output **Slot** must be soft-wired with the input of a **RL_BK_CAN_(No.)** type function block.

Coding of output IO-Error

See table in the general part of the description of RL 400 system. Only if no error message is pending at output IO_Error, data is exchanged with the module in the rail line system.

Coding of output Status

The following table describes the module status information. This value is valid only, if no error message is pending at output IO_Error, because this is the prerequisite of data exchange with the module in the rail line system. The information is bit-coded.

| Bit | Description |
|-----|--|
| 0 | Alarm 1 = 1, if there is an error on one of the measured inputs (e.g. sensor break). |

| | |
|---|--|
| 1 | Alarm 2 = 1 , if at least one alarm is pending (e.g. out-of-limits). |
| 2 | Status 1: Internal error of the module. Various messages are grouped together. The exact error message is displayed on the instrument. |
| 3 | Wrong_output_value; indicates that an out-of-limits process value for a channel was transferred. |

**NOTE!**

For additional information: see operating instructions of the TB 45-module.

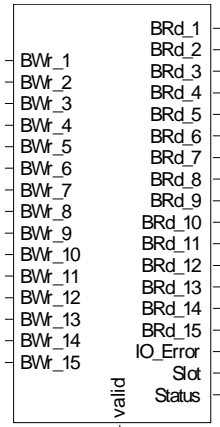
In-/Outputs

| Name | Type | Description |
|------------------|-------|---|
| BWr_1 ... BWr_15 | Float | 1st datum to be written via bus |
| Name | Type | Description |
| BRd_1 ... BRd_15 | Float | 1st datum read via bus |
| IO_Error | Float | Type of the communication error to the module in the I/O system |
| Slot | Float | Own block number |
| Status | Float | Status message of the module |
| valid | Bool | The communication to I/O system (hardware) is established. Measured values are valid. |

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|--------|----------------|------------|--|--------|---------|-------|-----|
| ErrBeh | Error behavior | Enum | Behaviour in error condition | r/w | 0 | | |
| | | Last value | Error behaviour: Use of the last value is continued. | | 0 | | |
| | | Zero | Error behaviour: Value zero (value is set to "0"). | | 1 | | |

III-18.8 Railline SG45 (Universal transmitter (No. 288))



SG45

Abb. 646

The function block is used to connect a transmitter module for load cells, strain gauges, and melt pressure sensors.

The Uniflex SG 45 is a transmitter for processing and monitoring input signals form strain gauges, load cells, melt pressure sensors, and resistance bridges. It can be used in practically all industrial applications involving the precise measurement, indication, and processing of force, weight, or melt pressure.

i *NOTE!*
 As the parameter set of this module is very extensive, setting of configurations and parameter is done with the program BlueControl®.

The data transferred to the inputs BWr_1 to BWr_15 is transmitted to the module SG45. The data 1 to 15 is defined in the engineering of SG45, which is done with BlueControl® (in "Bus data (write)", see Help of BlueControl® "bus data").

The data given at the outputs BRd_1 to BRd_15 are received of the module SG45. The data 1 to 15 is defined in the engineering of SG45, which is done with BlueControl® (in "Bus data (read)", see Help of BlueControl® "bus data").

Output **Slot** must be soft-wired with the input of a **RL_BK_CAN_(No.)** type function block.

Coding of output IO-Error

See table in the general part of the description of RL 400 system. Only if no error message is pending at output IO_Error, data is exchanged with the module in the rail line system.

Coding of output Status

The following table describes the module status information. This value is valid only, if no error message is pending at output IO_Error, because this is the prerequisite of data exchange with the module in the rail line system. The information is bit-coded.

| Bit | Description |
|-----|--|
| 0 | Alarm 1 = 1, if there is an error on one of the measured inputs (e.g. sensor break). |
| 1 | Alarm 2 = 1, if at least one alarm is pending (e.g. out-of-limits). |

| | |
|---|--|
| 2 | Status 1: Internal error of the module. Various messages are grouped together. The exact error message is displayed on the instrument. |
| 3 | Wrong_output_value; indicates that an out-of-limits process value for a channel was transferred. |

**NOTE!**

For additional information: see operating instructions of the SG 45-module.

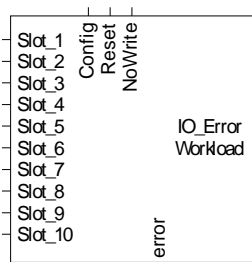
In-/Outputs

| Name | Type | Description |
|-------------------|-------|---|
| BWr_1 ... BWr_15 | Float | 1st datum to be written via bus |
| Name | Type | Description |
| BRd_1 ... BR_d_15 | Float | 1st datum read via bus |
| Slot | Float | Own block number |
| IO_Error | Float | Type of the communication error to the module in the I/O system |
| Status | Float | Status message of the module |
| valid | Bool | The communication to VARIO I/O system (hardware) is established. Measured values are valid. |

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|--------|----------------|-------------|---|--------|---------|-------|-----|
| ErrBeh | Error behavior | Enum | Behaviour in error condition | r/w | 0 | | |
| | | Last value | Error behaviour: Use of the last value is continued. | | 0 | | |
| | | Zero | Error behaviour: Value zero (value is set to "0"). | | 1 | | |
| | | Error value | Error behaviour: Error value (the pre-defined error value is used). | | 2 | | |

III-18.9 Railline BK CAN 10 (Buscoupler CAN (No. 280))



RAIL_BK_CAN_10

Abb. 647

The function block is used for configuration and parameter setting of a **rail line** RL40 CANopen bus coupler module. Up to 10 **rail line** RL400 I/O modules can be soft-wired.

Inputs **Slot_1** to **Slot_10** must be soft-wired with the **Slot** output of the used rail line I/O function blocks, e.g. **RL_442**, **RL_461**, **CI45**. The order at the inputs **Slot_1** to **Slot_10** must be the same as that of the modules plugged in the **rail line** system.

Coding of output IO_error

| Error | Description |
|-------|--|
| 0 | No error |
| 1 | Initialization error |
| 2 | RL400 and KS1x8 configuration mismatch, or no RL400 system found |
| 3 | Error reading the status of modules in the RL400 system |
| 4 | Configuration error of RL400 system CAN interface |
| 5 | Configuration error of RL400 system modules |
| 6 | Error when switching the CAN bus to process mode |



NOTE!

For additional information: see documentation of the **rail line** CANopen bus coupler **RL40-CANopen**.

In-/Outputs

| Name | Type | Description |
|--------------------|-------|------------------------------------|
| Slot_1 | Float | 1st module on bus system |
| Slot_2 ... Slot_10 | Float | 2nd module on bus system |
| Config | Bool | Configure all modules |
| Reset | Bool | Reset the communication and errors |
| NoWrite | Bool | Data will not be written |

| Name | Type | Description |
|----------|-------|---|
| IO_Error | Float | Type of the communication error to the I/O system |
| Workload | Float | Workload of the function block in %. Should not be higher than 90 % continuously. |
| error | Bool | Communication error to the I/O system |

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|------------|--------------------|-----------|--|--------|---------|--------------|-----|
| Rail.Adr | address | Int | node address of the bus coupler on the CAN bus | r/w | 32 | 1 ... 127 | |
| Baudrate | Baud rate | Enum | Baud rate on the CAN bus | r/w | 4 | | |
| | | 20 kBit | The CAN bus transfer rate is 20 kbits. | | 0 | | |
| | | 50 kBit | The CAN bus transfer rate is 50 kbits. | | 1 | | |
| | | 100 kBit | The CAN bus transfer rate is 100 kbits. | | 2 | | |
| | | 125 kBit | The transfer rate over the CAN bus is 125 kbits. | | 3 | | |
| | | 250 kBit | The transfer rate over the CAN bus is 250 kbits. | | 4 | | |
| | | 500 kBit | The transfer rate over the CAN bus is 500 kbits. | | 5 | | |
| | | 800 kBit | The transfer rate over the CAN bus is 800 kbits. | | 6 | | |
| | | 1000 kBit | The transfer rate over the CAN bus is 1000 kbits. | | 7 | | |
| CobId.RPDO | COB-ID receive PDO | Int | COB-ID of the receive PDO. If the datum is switched off, the predefined values are used for the PDO. These are: 1st receive PDO 0x200 + Can bus address 2nd receive PDO 0x300 + Can bus address 3rd receive PDO 0x400 + Can bus address 4th receive PDO 0x500 + Can bus address | r/w | off | 385 ... 1407 | ja |

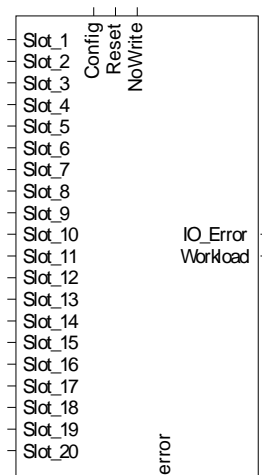
| | | | | | | | |
|--------------|---------------------|-------------------|---|-----|------|------------------|----|
| CobId.TPDO | COB-ID send PDO | Int | COB-ID of send PDO. If the datum is switched off, the predefined values are used for the PDO. These are: 1st send PDO 0x180 + Can bus address 2nd send PDO 0x280 + Can bus address 3rd send PDO 0x380 + Can bus address 4th send PDO 0x480 + Can bus address | r/w | off | 385 ... 1407 | ja |
| Typ.TPDO | Typ of the send PDO | Enum | Definition, if the send PDO should work as synchronous or asynchronous PDO | r/w | 1 | | |
| | | Synchronous TPDO | Synchronous TPDO. Settings within 1 - 240 are possible. | | 0 | | |
| | | Asynchronous TPDO | Asynchronous TPDO. The value of 255 is fixed. | | 1 | | |
| STyp.TPDO | Typ of the syn. PDO | Int | Define after how many SYNC messages the synchronous PDO should be generated. | r/w | 240 | 1 ... 240 | |
| Inhibit.TPDO | Inhibit time | Int | Define after which min. time an asynchronous PDO may be generated again. The values are multiples of 0.1ms. | r/w | 0 | 0 ... 30000 | |
| GuardTime | life time | Int | Cycle time of lifetime [ms] | r/w | 2000 | 100 ... 30000 | ja |

**NOTE!**

Only the rail line address must be set for the **RL 400 Rail Line System**; the Baud rate may have to be set, if it was changed on the hardware.

Changing further configuration settings is purposeful **only** in case of integration into an **existing CAN** system and may be done only by trained and authorized personnel.

III-18.10 Railline BK CAN 20 (Buscoupler CAN (No. 281))



RAIL_BK_CAN_20

Abb. 648

The function block is used for configuration and parameter setting of a **rail line** RL40 CANopen bus coupler module. Up to 20 **rail line** RL400 I/O modules can be soft-wired.

Inputs **Slot_1** to **Slot_20** must be soft-wired with the **Slot** output of the used rail line I/O function blocks, e.g. **RL_442**, **RL_461**, **CI45**. The order at the inputs **Slot_1** to **Slot_20** must be the same as that of the modules plugged in the **rail line** system.

Coding of output IO_error

| Error | Description |
|-------|--|
| 0 | No error |
| 1 | Initialization error |
| 2 | RL400 and KS1x8 configuration mismatch, or no RL400 system found |
| 3 | Error reading the status of modules in the RL400 system |
| 4 | Configuration error of RL400 system CAN interface |
| 5 | Configuration error of RL400 system modules |
| 6 | Error when switching the CAN bus to process mode |



NOTE!

For additional information: see documentation of the **rail line** CANopen bus coupler **RL40-CANopen**.

In-/Outputs

| Name | Type | Description |
|--------------------|-------|--------------------------|
| Slot_1 ... Slot_20 | Float | 1st module on bus system |
| Config | Bool | Configure all modules |

| Reset | Bool | Reset the communication and errors |
|----------|-------|---|
| NoWrite | Bool | Data will not be written |
| Name | Type | Description |
| IO_Error | Float | Type of the communication error to the I/O system |
| Workload | Float | Workload of the function block in %. Should be not higher than 90 % continuously. |
| error | Bool | Communication error to the I/O system |

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|-------------|--------------------|-----------|--|--------|---------|-----------------|-----|
| Rail.Adr | address | Int | node address of the bus coupler on the CAN bus | r/w | 32 | 1 ... 127 | |
| Baudrate | Baud rate | Enum | Baud rate on the CAN bus | r/w | 4 | | |
| | | 20 kBit | The CAN bus transfer rate is 20 kbits. | | 0 | | |
| | | 50 kBit | The CAN bus transfer rate is 50 kbits. | | 1 | | |
| | | 100 kBit | The CAN bus transfer rate is 100 kbits. | | 2 | | |
| | | 125 kBit | The transfer rate over the CAN bus is 125 kbits. | | 3 | | |
| | | 250 kBit | The transfer rate over the CAN bus is 250 kbits. | | 4 | | |
| | | 500 kBit | The transfer rate over the CAN bus is 500 kbits. | | 5 | | |
| | | 800 kBit | The transfer rate over the CAN bus is 800 kbits. | | 6 | | |
| | | 1000 kBit | The transfer rate over the CAN bus is 1000 kbits. | | 7 | | |
| CoblId.RPDO | COB-ID receive PDO | Int | COB-ID of the receive PDO. If the datum is switched off, the predefined values for the PDO are used. These are: 1st receive PDO 0x200 + Can bus address 2nd receive PDO 0x300 + Can bus address 3rd receive PDO 0x400 + Can bus address 4th receive PDO 0x500 + Can bus address | r/w | off | 385 ... 1407 | ja |

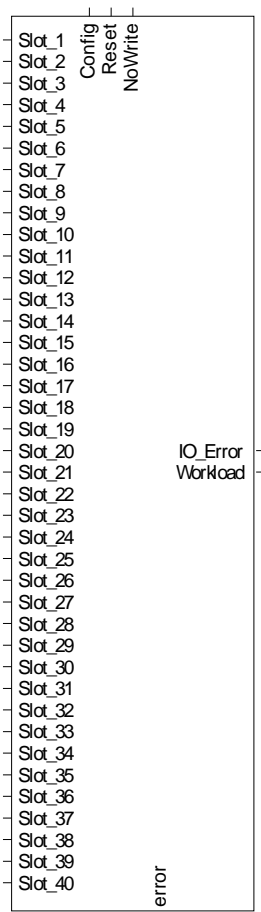
| | | | | | | | |
|--------------|---------------------------|----------------------|---|-----|------|------------------|----|
| CobId.TPDO | COB-ID send PDO | Int | COB-ID of send PDO. If the datum is switched off, the predefined values are used for the PDO. These are: 1st send PDO 0x180 + Can bus address 2nd send PDO 0x280 + Can bus address 3rd send PDO 0x380 + Can bus address 4th send PDO 0x480 + Can bus address | r/w | off | 385 ... 1407 | ja |
| Typ.TPDO | Typ of the send PDO | Enum | Definition, if the send PDO should work as synchronous or asynchronous PDO | r/w | 1 | | |
| | | Synchronous TPDO | Synchronous TPDO. Settings within 1 - 240 are possible. | | 0 | | |
| | | Asynchronous TPDO | Asynchronous TPDO. The value of 255 is fixed. | | 1 | | |
| STyp.TPDO | Typ of the syn. PDO | Int | Define after how many SYNC messages the synchronous PDO should be generated. | r/w | 240 | 1 ... 240 | |
| Inhibit.TPDO | Inhibit time | Int | Define after which min. time an asynchronous PDO may be generated again. The values are multiples of 0.1ms. | r/w | 0 | 0 ... 30000 | |
| GuardTime | life time | Int | Cycle time of lifetime [ms] | r/w | 2000 | 100 ... 30000 | ja |

**NOTE!**

Only the rail line address must be set for the **RL 400 Rail Line System**; the Baud rate may have to be set, if it was changed on the hardware.

Changing further configuration settings is purposeful **only** in case of integration into an **existing CAN system** and may be done only by trained and authorized personnel.

III-18.11 Railline BK CAN 40 (Buscoupler CAN (No. 282))



RAIL_BK_CAN_40

Abb. 649

The function block is used for configuration and parameter setting of a **rail line** RL40 CANopen bus coupler module. Up to 40 **rail line** RL400 I/O modules can be soft-wired.

Inputs **Slot_1** to **Slot_40** must be soft-wired with the **Slot** output of the used rail line I/O function blocks, e.g. **RL_442**, **RL_461**, **CI45**. The order at the inputs **Slot_1** to **Slot_40** must be the same as that of the modules plugged in the **rail line** system.

Coding of output IO_error

| Error | Description |
|-------|--|
| 0 | No error |
| 1 | Initialization error |
| 2 | RL400 and KS1x8 configuration mismatch, or no RL400 system found |
| 3 | Error reading the status of modules in the RL400 system |
| 4 | Configuration error of RL400 system CAN interface |
| 5 | Configuration error of RL400 system modules |
| 6 | Error when switching the CAN bus to process mode |

**NOTE!**

For additional information: see documentation of the **rail line CANopen bus coupler RL40-CANopen**.

In-/Outputs

| Name | Type | Description |
|--------------------|-------|------------------------------------|
| Slot_1 ... Slot_40 | Float | 1st module on bus system |
| Config | Bool | Configure all modules |
| Reset | Bool | Reset the communication and errors |
| NoWrite | Bool | Data will not be written |

| Name | Type | Description |
|----------|-------|---|
| IO_Error | Float | Type of the communication error to the I/O system |
| Workload | Float | Workload of the function block in %. Should not be more than 90 % continuously. |
| error | Bool | Communication error to the I/O system |

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|-----------|-----------|-----------|---|--------|---------|-----------|-----|
| Rail.Adr | address | Int | node address of the bus coupler on the CAN bus | r/w | 32 | 1 ... 127 | |
| Baudrate | Baud rate | Enum | Baud rate on the CAN bus | r/w | 4 | | |
| | | 20 kBit | The CAN bus transfer rate is 20 kbits. | | 0 | | |
| | | 50 kBit | The CAN bus transfer rate is 50 kbits. | | 1 | | |
| | | 100 kBit | The CAN bus transfer rate is 100 kbits. | | 2 | | |
| | | 125 kBit | The transfer rate over the CAN bus is 125 kbits. | | 3 | | |
| | | 250 kBit | The transfer rate over the CAN bus is 250 kbits. | | 4 | | |
| | | 500 kBit | The transfer rate over the CAN bus is 500 kbits. | | 5 | | |
| | | 800 kBit | The transfer rate over the CAN bus is 800 kbits. | | 6 | | |
| | | 1000 kBit | The transfer rate over the CAN bus is 1000 kbits. | | 7 | | |
| Cobl.RPDO | COB-ID | Int | COB-ID of the receive PDO. If | r/w | off | 385 ... | ja |

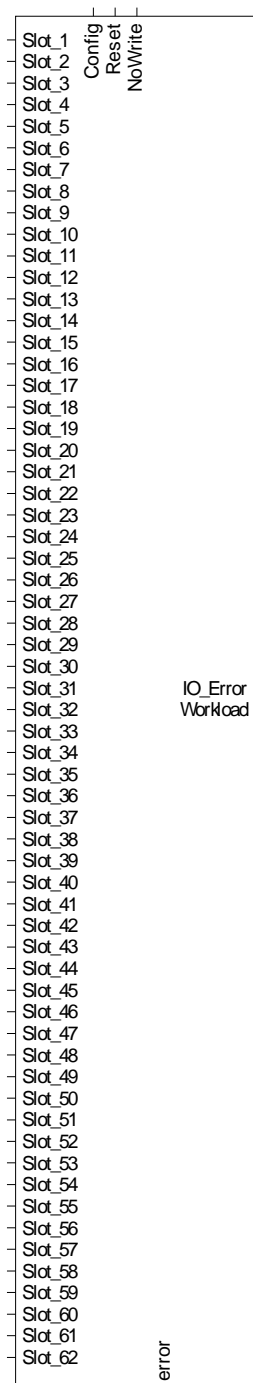
| | | | | | | | |
|--------------|---------------------|-------------------|---|-----|------|------------------|----|
| | receive PDO | | the datum is switched off, the predefined values for the PDO are used. These are: 1st receive PDO 0x200 + Can bus address 2nd receive PDO 0x300 + Can bus address 3rd receive PDO 0x400 + Can bus address 4th receive PDO 0x500 + Can bus address | | | 1407 | |
| CobId.TPDO | COB-ID send PDO | Int | COB-ID of send PDO. If the datum is switched off, the predefined values are used for the PDO. These are: 1st send PDO 0x180 + Can bus address 2nd send PDO 0x280 + Can bus address 3rd send PDO 0x380 + Can bus address 4th send PDO 0x480 + Can bus address | r/w | off | 385 ... 1407 | ja |
| Typ.TPDO | Typ of the send PDO | Enum | Definition, if the send PDO should work as synchronous or asynchronous PDO | r/w | 1 | | |
| | | Synchronous TPDO | Synchronous TPDO. Settings within 1 - 240 are possible. | | 0 | | |
| | | Asynchronous TPDO | Asynchronous TPDO. The value of 255 is fixed. | | 1 | | |
| STyp.TPDO | Typ of the syn. PDO | Int | Define after how many SYNC messages the synchronous PDO should be generated. | r/w | 240 | 1 ... 240 | |
| Inhibit.TPDO | Inhibit time | Int | Define after which min. time an asynchronous PDO may be generated again. The values are multiples of 0.1ms. | r/w | 0 | 0 ... 30000 | |
| GuardTime | life time | Int | Cycle time of lifetime [ms] | r/w | 2000 | 100 ... 30000 | ja |

**NOTE!**

*Only the rail line address must be set for the **RL 400 Rail Line System**; the Baud rate may have to be set, if it was changed on the hardware.*

*Changing further configuration settings is purposeful **only** in case of integration into an **existing** CAN system and may be done only by trained and authorized personnel.*

III-18.12 Railline BK CAN 62 (Buscoupler CAN (No. 283))



RAIL_BK_CAN_62

Abb. 650

The function block is used for configuration and parameter setting of a **rail line** RL40 CANopen bus coupler module. Up to 62 **rail line** RL400 I/O modules can be soft-wired.

Inputs **Slot_1** to **Slot_62** must be soft-wired with the **Slot** output of the used rail line I/O function blocks, e.g. **RL_442**, **RL_461**, **CI45**. The order at the inputs **Slot_1** to **Slot_62** must be the same as that of the modules plugged in the **rail line** system.

Coding of output IO_error

| Error | Description |
|-------|--|
| 0 | No error |
| 1 | Initialization error |
| 2 | RL400 and KS1x8 configuration mismatch, or no RL400 system found |
| 3 | Error reading the status of modules in the RL400 system |
| 4 | Configuration error of RL400 system CAN interface |
| 5 | Configuration error of RL400 system modules |
| 6 | Error when switching the CAN bus to process mode |

**NOTE!**

For additional information: see documentation of the *rail line* CANopen bus coupler **RL40-CANopen**.

In-/Outputs

| Name | Type | Description |
|--------------------|-------|------------------------------------|
| Slot_1 ... Slot_62 | Float | 1st module on bus system |
| Config | Bool | Configure all modules |
| Reset | Bool | Reset the communication and errors |
| NoWrite | Bool | Data will not be written |

| Name | Type | Description |
|----------|-------|---|
| IO_Error | Float | Type of the communication error to the I/O system |
| Workload | Float | Workload of the function block in %. Should be not higher than 90 % continuously. |
| error | Bool | Communication error to the I/O system |

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|----------|-----------|----------|--|--------|---------|-----------|-----|
| Rail.Adr | address | Int | node address of the bus coupler on the CAN bus | r/w | 32 | 1 ... 127 | |
| Baudrate | Baud rate | Enum | Baud rate on the CAN bus | r/w | 4 | | |
| | | 20 kBit | The CAN bus transfer rate is 20 kbits. | | 0 | | |
| | | 50 kBit | The CAN bus transfer rate is 50 kbits. | | 1 | | |
| | | 100 kBit | The CAN bus transfer rate is 100 kbits. | | 2 | | |

| | | | | | | | |
|------------|---------------------|-------------------|--|-----|-----|-----------------|----|
| | | 125 kBit | The transfer rate over the CAN bus is 125 kbits. | | 3 | | |
| | | 250 kBit | The transfer rate over the CAN bus is 250 kbits. | | 4 | | |
| | | 500 kBit | The transfer rate over the CAN bus is 500 kbits. | | 5 | | |
| | | 800 kBit | The transfer rate over the CAN bus is 800 kbits. | | 6 | | |
| | | 1000 kBit | The transfer rate over the CAN bus is 1000 kbits. | | 7 | | |
| CobId.RPDO | COB-ID receive PDO | Int | COB-ID of the receive PDO. If the datum is switched off, the predefined values for the PDO are used. These are: 1st receive PDO 0x200 + Can bus address 2nd receive PDO 0x300 + Can bus address 3rd receive PDO 0x400 + Can bus address 4th receive PDO 0x500 + Can bus address | r/w | off | 385 ... 1407 | ja |
| CobId.TPDO | COB-ID send PDO | Int | COB-ID of send PDO. If the datum is switched off, the predefined values are used for the PDO. These are: 1st send PDO 0x180 + Can bus address 2nd send PDO 0x280 + Can bus address 3rd send PDO 0x380 + Can bus address 4th send PDO 0x480 + Can bus address | r/w | off | 385 ... 1407 | ja |
| Typ.TPDO | Typ of the send PDO | Enum | Definition, if the send PDO should work as synchronous or asynchronous PDO | r/w | 1 | | |
| | | Synchronous TPDO | Synchronous TPDO. Settings within 1 - 240 are possible. | | 0 | | |
| | | Asynchronous TPDO | Asynchronous TPDO. The value of 255 is fixed. | | 1 | | |
| STyp.TPDO | Typ of the syn. PDO | Int | Define after how many SYNC messages the synchronous | r/w | 240 | 1 ... 240 | |

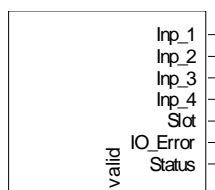
| | | | | | | | |
|--------------|--------------|-----|---|-----|------|---------------|----|
| | | | PDO should be generated. | | | | |
| Inhibit.TPDO | Inhibit time | Int | Define after which min. time an asynchronous PDO may be generated again. The values are multiples of 0.1ms. | r/w | 0 | 0 ... 30000 | |
| GuardTime | life time | Int | Cycle time of lifetime [ms] | r/w | 2000 | 100 ... 30000 | ja |

**NOTE!**

Only the rail line address must be set for the **RL 400 Rail Line System**; the Baud rate may have to be set, if it was changed on the hardware.

Changing further configuration settings is purposeful **only** in case of integration into an **existing CAN system** and may be done only by trained and authorized personnel.

III-18.13 Railline 422 (Analog input-module (No. 260))



RL_422

Abb. 651

The function block is used for configuration and parameter setting of a module with 4 analog inputs. These inputs can be current or voltage inputs, for two-wire transmitter with supply (TPS) or potentiometer inputs (voltage divider circuit). The conditioned measured values are available at its outputs.

Output **Slot** must be soft-wired with the input of a **RL_BK_CAN_(no.)** type function block according to module position.

Coding of output IO-Error

See table in the general part of the description of RL 400 system. Only if no error message is pending at output IO_Error, data is exchanged with the module in the rail line system.

Coding of output Status

| Bit | Description |
|-----|---|
| 0 | Alarm 0 = 1, when the analog input signals an error (overshoot) and the channel is active |
| 1 | Not used |
| 2 | Status 1: EEPROM error |
| 3 | Not used. |



NOTE!

For additional information: see operating note of the **rail line** I/O module RL_422.

Filter

A first order mathematic filter is built in. Time constant and band width are adjustable. The filter band width is the adjustable tolerance around the measured value within which the filter is active. Process value changes higher than the bandwidth are passed through directly (without filtering).

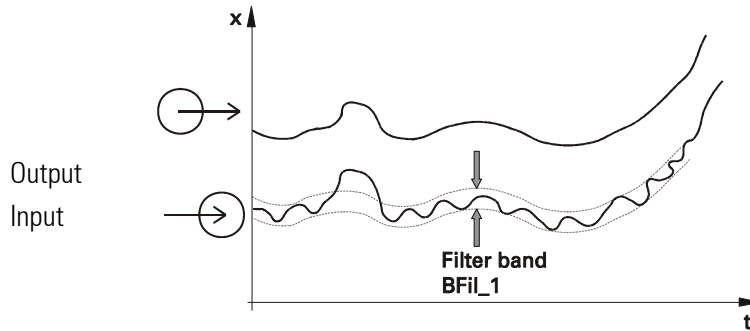
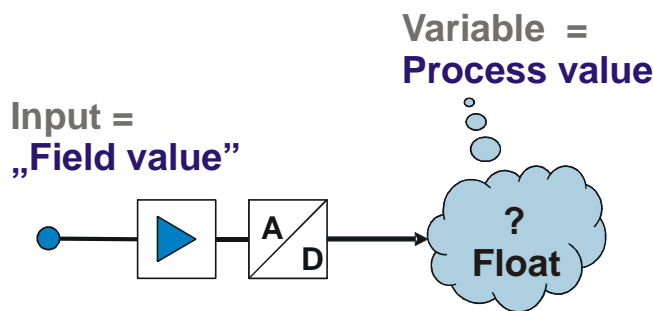


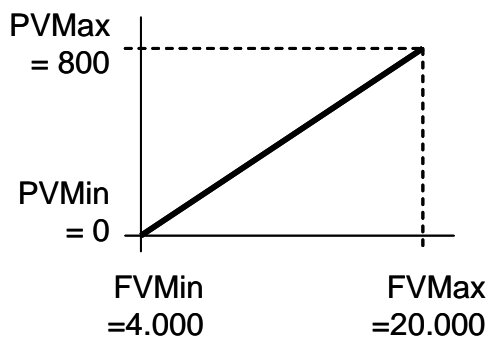
Fig. 9: Signal filtering using the filter time constant as long as it is within the bandwidth.

Standard input signal scaling

With the standard input signals, the displayed value should be adapted to the measured values. Originally, the measured values are "field values", i.e. "raw measured values" as coming from the input hardware. Normally, however, the "display value" reflecting the meaning of the physical value in the system, for example the filling level of a bowl instead of the actual input value (e.g. in millivolt) is used. For this purpose, linear 2-point scaling is possible: the input span start (= field value PV) is set to the display span start (=process value PV) and the span end value is set to the display span end.



In the engineering, the display value (PV) appears as value of the input.



Example: Standard input signal converted into level

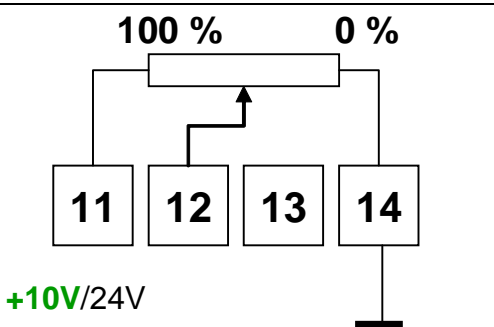
| | | | |
|--|------------------------|--------------------------|---|
| | Signal: | 4 ... 20 mA | (field value FV) |
| | Range: | 0 ... 800 m ³ | (process value PV) |
| | FVMin / FVMax = | 4000 / 20 000 | (field value span start / field value span end) |
| | PVMin / PVMax = | 0 / 800 m ³ | (display span start / display span end) |

| | Range: | Step width | Factor = steps per value of range |
|--|---------------|------------|-----------------------------------|
| | -20 ... 20 mA | ± 20 000 | 1 000 |
| | -10 ... 10 V | ± 10 000 | 1 000 |
| | -5 ... 5 mA | ± 5 000 | 1 000 |
| | -1 ... 1 mA | ± 10 000 | 10 000 |

Scaling of a potentiometer input signal

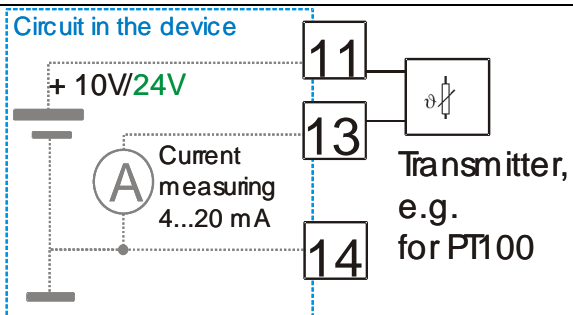
With potentiometer measurement, a display value within 0% for the span start and 100% for the span end is expected. Linear 2-point scaling is used for this purpose: The input span start (raw) value is set to the display span start and the span end value is set to the display span end.

In the engineering, the display value (PV) is shown as value of the input.

| | |
|---|--|
|  <p>+10V/24V</p> <p>Signal (noted): 2,45 ... 8.67 V (field value Fv) Range: 0 ... 100% (process value PV) FVMin / FVMax = 2.450 / 8.670 PVMIn / PVMax = 0 / 100 %</p> | <p>Example: Potentiometer connected to channel "In1":</p> <p>Select sensor supply: SSup_1 = 10V Select the input type: InpT_1 = -10 ... +10V Default setting (1:1 voltage conversion): FVMin_1 = PVMIn_1 = 0 FVMax_1 = PVMax_1 = 10,000</p> <p>Connect the potentiometer (see figure) Set the potentiometer to 0% or 100% and note the measured values FVMin_1 or FVMax_1 Set noted values for FVMin_1 / FVMax_1 Process values: Set PVMIn_1 = 0% and PVMax_1 = 100% (example: 0 ... 100%)</p> |
|---|--|

Two-wire transmitter

Linear 2-point scaling: The span start input (raw) value is set to the display span start and the span end value is set to the display span end.

| | |
|---|--|
|  <p>Circuit diagram</p> | <p>For example, passive transmitter in two-wire technology: PIN 11 is used for connecting the transmitter, and the absorbed current is measured via PIN 13. The common reference point for supply and measurement is internal (in the instrument, PIN 14). Settings: see below.</p> |
|---|--|

Transmitter

4 ... 20 mA

+10V/24V

Signal: 4 ... 20 mA (field value)
 Range: 0 ... 100% (process value)
 FVMin / FVMax = 4 000 / 20 000
 PVMIn / PVMMax = 0 / 100 %

Example: passive transmitter on channel "In 1":

Select sensor supply: $SSup_1 = 24\text{ V}$

Select the input type: $InpT_1 = -20 \dots +20\text{ mA}$

Connect transmitter (see figure)

Set field values: $FVMin_1 = 4.000$ and $FVMax_1 = 20.000$

Set process values: $PVMIn_1 = 0\%$ and $PVMax_1 = 100\%$
 (physical unit, example: 0 ... 100%)

Active transmitter

4 ... 20 mA

Signal: 4 ... 20 mA (field value)
 Range: 0 ... 100% (process value)
 FVMin / FVMax = 4 000 / 20 000
 PVMIn / PVMMax = 0 / 100 %

Example: passive transmitter on channel "In 1":

Select input type: $InpT_1 = -20 \dots +20\text{ mA}$

Connect transmitter (see figure)

Set field values: $FVMin_1 = 4.000$ und $FVMax_1 = 20.000$

Set process values: $PVMIn_1 = 0\%$ and $PVMax_1 = 100\%$
 (physical unit, example: 0 ... 100%)

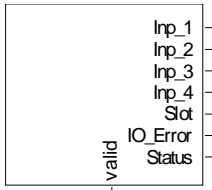
| In-/Outputs | | |
|-----------------|-------|---|
| Name | Type | Description |
| Inp_1 ... Inp_4 | Float | Conditioned measured value 1 |
| Slot | Float | Own block number |
| IO_Error | Float | Type of the communication error to the module in the I/O system |
| Status | Float | Status message of the module |
| valid | Bool | The communication to I/O system (hardware) is established. Measured values are valid. |

| Parameter | | | | | | | |
|------------------------|---------------------|------|--|--------|---------|------------------------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| FltV_1 ... FltV_4 | Error value i1 | Int | Substitute value with sensor fail | r/w | 0 | -30000 ... 30000 | |
| TFil_1 ... TFi1_4 | Filt.time i1[1/10s] | Int | Filter time constant of signal processing in s | r/w | 5 | 0 ... 9999 | |
| BFil_1 ... BFi1_4 | Filt.band i1 [1/10] | Int | Filter bandwidth | r/w | 5 | 0 ... 9999 | |
| FVMin_1 ... FVMin_4 | FVMin i1 | Int | Low field value | r/w | 0 | -30000 ... 30000 | |
| FVMax_1 ... FVMax_4 | FVMax i1 | Int | High field value | r/w | 20000 | -30000 ... 30000 | |
| PVMin_1 ... PVMin_4 | PVMin i1 | Int | Low process value at FVMin | r/w | 0 | -30000 ... 30000 | |
| PVMax_1 ... PVMax_4 | PVMax i1 | Int | High process value at FVMax | r/w | 20000 | -30000 ... 30000 | |
| SSup_1 ... SSUp_4 | Sensor supply i1 | Enum | Sensor supply | r/w | 0 | | |
| | | 10 V | 10 V | | 0 | | |
| | | 24 V | 24 V | | 2 | | |

| Configuration | | | | | | | |
|----------------------|----------------|-------------|---|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| ErrBeh | Error behavior | Enum | Behaviour in error condition | r/w | 0 | | |
| | | Last value | Error behaviour: Use of the last value is continued. | | 0 | | |
| | | Zero | Error behaviour: Value zero (value is set to "0"). | | 1 | | |
| | | Error value | Error behaviour: Error value (the pre-defined error value is used). | | 2 | | |
| Enab_1 ... Enab_4 | Activated | Enum | Activate input | r/w | 1 | | |
| | | No | Switched off, e.g. to prevent generation of error messages such as sensor break. The channel is not used. | | 0 | | |

| | | | | | | | |
|----------------------|------------|-------------|-----------------------|-----|---|--|--|
| | | Yes | Switched on | | 1 | | |
| InpT_1 ... InpT_4 | Input type | Enum | Sensor type | r/w | 3 | | |
| | | -10...+10 V | Voltage : -10...+10 V | | 0 | | |
| | | -5...+5 V | Voltage : -5...+5 V | | 1 | | |
| | | -1...+1 V | Voltage : -1...+1 V | | 2 | | |
| | | -20...20mA | Current -20...20mA | | 3 | | |

III-18.14 Railline 423-0 (Input module resistance-thermometer (No. 265))



RL_423-0

Abb. 652

The function block is used for configuration and parameter setting of a module with resistance thermometers. The inputs are for Pt100 or Ni 100 in two and three-wire connection, with galvanic isolation between inputs, logic circuitry and internal supply (the inputs are galvanically connected).

The conditioned measured values are available at its outputs.


Output **Slot** must be soft-wired with the input of a **RL_BK_CAN_(no.)** type function block.

Coding of output IO-Error

See table in the general part of the description of RL 400 system. Only if no error message is pending at output IO_Error, data is exchanged with the module in the rail line system.

Coding of output Status

| Bit | Description |
|-----|---|
| 0 | Alarm 0 = 1, if analog input detects an error (sensor break, short circuit at sensor, limits exceeded) and channel is active. |
| 1 | Not used. |
| 2 | Status 1 : EEPROM error |
| 3 | Not used. |

 **NOTE!**
For additional information: see operating note of the **rail line** I/O module RL_423-X.

Filter

A first order mathematic filter is built in. Time constant and band width are adjustable. The filter band width is the adjustable tolerance around the measured value within which the filter is active. Process value changes higher than the bandwidth are passed through directly (without filtering).

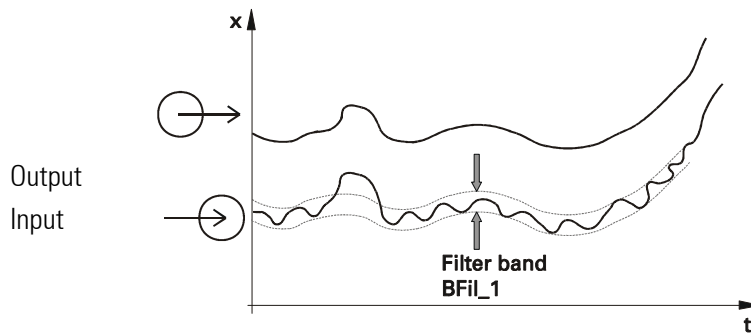


Fig. 9: Signal filtering using the filter time constant as long as it is within the bandwidth.

In-/Outputs

| Name | Type | Description |
|-----------------|-------|---|
| Inp_1 ... Inp_4 | Float | Conditioned measured value 1 |
| Slot | Float | Own block number |
| IO_Error | Float | Type of the communication error to the module in the I/O system |
| Status | Float | Status message of the module |
| valid | Bool | The communication to I/O system (hardware) is established. Measured values are valid. |

Parameter

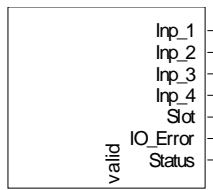
| ID | Name | Type | Description | Access | Default | Range | Off |
|----------------------|------------------------|------|--------------------------------------|--------|---------|-------------------|-----|
| Offs_1 ... Offs_4 | Offset i1 [1/10] | Int | Offset [1/10 Unit] | r/w | 0 | -4000 ... 4000 | |
| FiltV_1 ... FiltV_4 | Error val.i1 [1/10] | Int | Substitute value with sensor fail | r/w | 0 | -2000 ... 8500 | |
| TFil_1 ... TFil_4 | Filt.time i1[1/10s] | Int | Filter time [1/10 s] | r/w | 5 | 0 ... 9999 | |
| BFil_1 ... BFil_4 | Filt.band i1 [1/10] | Int | Filter bandwidth [1/10] | r/w | 5 | 0 ... 9999 | |

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|--------|-------------------|---------------|---|--------|---------|-------|-----|
| ErrBeh | Error behavior | Enum | Behaviour in error condition | r/w | 0 | | |
| | | Last value | Error behaviour: Use of the last value is continued. | | 0 | | |
| | | Zero | Error behaviour: Value zero (value is set to | | 1 | | |

| | | | | | | | |
|----------------------|---------------------|-------------|--|-----|---|--|--|
| | | | "0"). | | | | |
| | | Error value | Error behaviour: Error value (the pre-defined error value is used). | | 2 | | |
| Enab_1 ... Enab_4 | Enabled input 1 | Enum | Input activated | r/w | 1 | | |
| | | No | Switched off, e.g. to prevent error messages such as sensor break from being generated. The channel is not used. | | 0 | | |
| | | Yes | Switched on | | 1 | | |
| InpT_1 ... InpT_4 | Sensor type input 1 | Enum | Sensor type of the input | r/w | 0 | | |
| | | Pt100 | Pt100 (-200 ... 850°C, -140...1562°F) | | 0 | | |
| | | Ni100 | Ni100 (-60...180°C, -76...356°F) | | 1 | | |
| Unit_1 ... Unit_4 | Unit input 1 | Enum | Unit of the measurement signal (e.g. °C) | r/w | 0 | | |
| | | °C | Unit = °C, degrees Celsius | | 0 | | |
| | | °F | Unit = °F, degrees Fahrenheit | | 1 | | |

III-18.15 Railline 423-1 (Input module resistance-thermometer (No. 266))



RL_423-1

Abb. 653

The function block is used for configuration and parameter setting of a module with resistance thermometers. The inputs are for Pt100, Ni 100 or KTY 11-6, with galvanic isolation between inputs, logic circuitry and internal supply (the inputs are galvanically connected).

The conditioned measured values are available at its outputs.

Output **Slot** must be soft-wired with the input of a **RL_BK_CAN_(no.)** type function block.

Coding of output IO-Error

See table in the general part of the description of RL 400 system. Only if no error message is pending at output IO_Error, data is exchanged with the module in the rail line system.

Coding of output Status

| Bit | Description |
|-----|---|
| 0 | Alarm 0 = 1, if analog input detects an error (sensor break, short circuit at sensor, limits exceeded) and channel is active. |
| 1 | Not used. |
| 2 | Status 1 : EEPROM error |
| 3 | Not used. |



NOTE!

For additional information: see operating note of the **rail line** I/O module RL_423-X.

Filter

A first order mathematic filter is built in. Time constant and band width are adjustable. The filter band width is the adjustable tolerance around the measured value within which the filter is active. Process value changes higher than the bandwidth are passed through directly (without filtering).

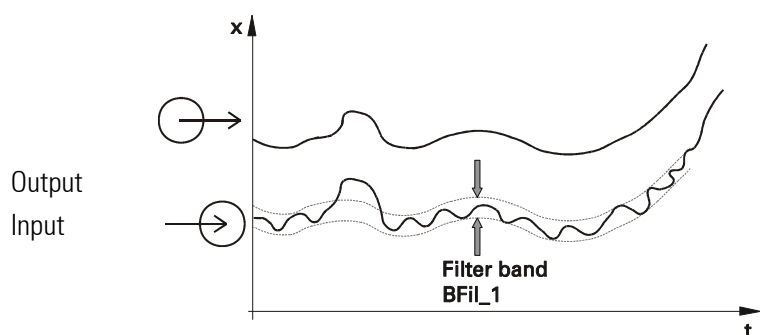


Fig. 9: Signal filtering using the filter time constant as long as it is within the bandwidth.

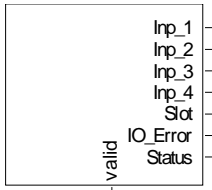
| In-/Outputs | | |
|-----------------|-------|---|
| Name | Type | Description |
| Inp_1 ... Inp_4 | Float | Conditioned measured value |
| Slot | Float | Own block number |
| IO_Error | Float | Type of the communication error to the module in the I/O system |
| Status | Float | Status message of the module |
| valid | Bool | The communication to I/O system (hardware) is established. Measured values are valid. |

| Parameter | | | | | | | |
|-------------------|---------------------|------|-----------------------------------|--------|---------|----------------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Offs_1 ... Offs_4 | Offset i1 [1/10] | Int | Offset [1/10 Unit] | r/w | 0 | -4000 ... 4000 | |
| FItV_1 ... FItV_4 | Error val.i1 [1/10] | Int | Substitute value with sensor fail | r/w | 0 | -2000 ... 8500 | |
| TFil_1 ... TFil_4 | Filt.time i1[1/10s] | Int | Filter time [1/10 s] | r/w | 5 | 0 ... 9999 | |
| BFil_1 ... BFil_4 | Filt.band i1 [1/10] | Int | Filter bandwidth [1/10] | r/w | 5 | 0 ... 9999 | |

| Configuration | | | | | | | |
|---------------|----------------|------------|--|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| ErrBeh | Error behavior | Enum | Behaviour in error condition | r/w | 0 | | |
| | | Last value | Error behaviour: Use of the last value is continued. | | 0 | | |
| | | Zero | Error behaviour: Value zero (value is set to "0"). | | 1 | | |

| | | | | | | | |
|----------------------|---------------------|-------------|--|-----|---|--|--|
| | | Error value | Error behaviour: Error value (the pre-defined error value is used). | | 2 | | |
| Enab_1 ... Enab_4 | Enabled input 1 | Enum | Input enabled | r/w | 1 | | |
| | | No | Switched off, e.g. to prevent error messages such as sensor break from being generated. The channel is not used. | | 0 | | |
| | | Yes | Switched on | | 1 | | |
| InpT_1 ... InpT_4 | Sensor type input 1 | Enum | Sensor type of the input | r/w | 0 | | |
| | | Pt1000 | Pt1000 (-200 ... 850°C, -140...1562°F) | | 0 | | |
| | | Ni1000 | Ni1000 (-60...180°C, -76...356°F) | | 1 | | |
| | | KTY | KTY | | 2 | | |
| Unit_1 ... Unit_4 | Unit input 1 | Enum | Unit of the measurement signal (e.g. °C) | r/w | 0 | | |
| | | °C | Unit = °C, degrees Celsius | | 0 | | |
| | | °F | Unit = °F, degrees Fahrenheit | | 1 | | |

III-18.16 Railline 423-2 (Input module resistance-thermometer (No. 267))



RL_423-2

Abb. 654

The function block is used for configuration and parameter setting of a module with resistance thermometers. The inputs are for Pt100 or Ni 100 in two and three-wire connection, Pt1000, Ni 1000 or KTY 11-6, with galvanic isolation between inputs, logic circuitry and internal supply (the inputs are galvanically connected). The conditioned measured values are available at its outputs.

Output **Slot** must be soft-wired with the input of a **RL_BK_CAN_(no.)** type function block.

Coding of output IO-Error

See table in the general part of the description of RL 400 system. Only if no error message is pending at output IO_Error, data is exchanged with the module in the rail line system.

Coding of output Status

| Bit | Description |
|-----|---|
| 0 | Alarm 0 = 1, if analog input detects an error (sensor break, short circuit at sensor, limits exceeded) and channel is active. |
| 1 | Not used. |
| 2 | Status 1 : EEPROM error |
| 3 | Not used. |



NOTE!

For additional information: see operating note of the **rail line** I/O module RL_423-X.

Filter

A first order mathematic filter is built in. Time constant and band width are adjustable. The filter band width is the adjustable tolerance around the measured value within which the filter is active. Process value changes higher than the bandwidth are passed through directly (without filtering).

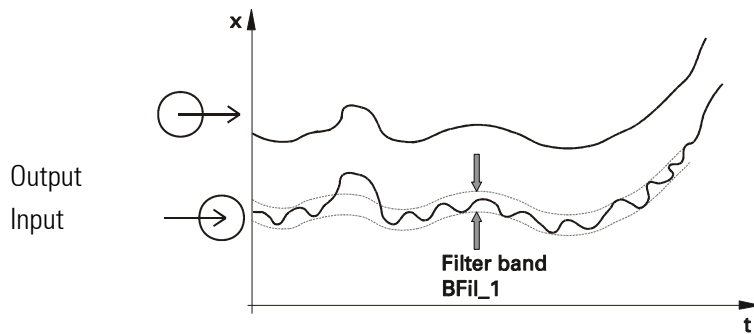


Fig. 9: Signal filtering using the filter time constant as long as it is within the bandwidth.

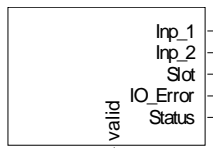
| In-/Outputs | | |
|-----------------|-------|---|
| Name | Type | Description |
| Inp_1 ... Inp_4 | Float | Conditioned measured value 1 |
| Slot | Float | Own block number |
| IO_Error | Float | Type of the communication error to the module in the I/O system |
| Status | Float | Status message of the module |
| valid | Bool | The communication to I/O system (hardware) is established. Measured values are valid. |

| Parameter | | | | | | | |
|----------------------|---------------------|------|--------------------------------------|--------|---------|-------------------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Offs_1 ... Offs_4 | Offset i1 [1/10] | Int | Offset [1/10 Unit] | r/w | 0 | -4000 ... 4000 | |
| FiltV_1 ... FiltV_4 | Error val.i1 [1/10] | Int | Substitute value with sensor fail | r/w | 0 | -2000 ... 8500 | |
| TFil_1 ... TFil_4 | Filt.time i1[1/10s] | Int | Filter time [1/10 s] | r/w | 5 | 0 ... 9999 | |
| BFil_1 ... BFil_4 | Filt.band i1 [1/10] | Int | Filter bandwidth [1/10] | r/w | 5 | 0 ... 9999 | |

| Configuration | | | | | | | |
|---------------|-------------------|------------|---|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| ErrBeh | Error behavior | Enum | Behaviour in error condition | r/w | 0 | | |
| | | Last value | Error behaviour: Use of the last value is continued. | | 0 | | |
| | | Zero | Error behaviour: Value zero (value is set to "0"). | | 1 | | |
| | | Error | Error behaviour: Error value (the pre- | | 2 | | |

| | | value | defined error value is used). | | | | |
|----------------------|------------------------|--------|--|-----|---|--|--|
| Enab_1 ... Enab_4 | Enabled input 1 | Enum | Input activated | r/w | 1 | | |
| | | No | Switched off, e.g. to prevent error messages such as sensor break from being generated. The channel is not used. | | 0 | | |
| | | Yes | Switched on | | 1 | | |
| InpT_1 ... InpT_4 | Sensor type input 1 | Enum | Sensor type of the input | r/w | 0 | | |
| | | Pt100 | Pt100 (-200 ... 850°C, -140...1562°F) | | 0 | | |
| | | Pt1000 | Pt1000 (-200 ... 850°C, -140...1562°F) | | 1 | | |
| | | Ni100 | Ni100 (-60...180°C, -76...356°F) | | 2 | | |
| | | Ni1000 | Ni1000 (-60...180°C, -76...356°F) | | 3 | | |
| | | KTY | KTY | | 4 | | |
| Unit_1 ... Unit_4 | Unit input 1 | Enum | Unit of the measurement signal (e.g. °C) | r/w | 0 | | |
| | | °C | Unit = °C, degrees Celsius | | 0 | | |
| | | °F | Unit = °F, degrees Fahrenheit | | 1 | | |

III-18.17 Railline 424-0 (Input module for thermocouple TC (No.270))



RL_424-0

Abb. 655

The function block is used for configuration and parameter setting of a module with thermocouple inputs TC. The inputs are galvanically isolated. Various thermocouples and a millivolt input are available as input types. The conditioned measured values are available at its outputs.

Output **Slot** must be soft-wired with the input of a **RL_BK_CAN_(no.)** type function block.

Coding of output IO-Error

See table in the general part of the description of RL 400 system. Only if no error message is pending at output IO_Error, data is exchanged with the module in the rail line system.

Coding of output Status

| Bit | Description |
|-----|--|
| 0 | Alarm 0 = 1, if analog input detects an error (sensor break, limits exceeded) and channel is active. Detection of sensor break only with input TC. |
| 1 | Not used. |
| 2 | Status 1 : EEPROM error |
| 3 | Not used. |



NOTE!

For additional information: see operating note of the **rail line** I/O module RL_424-X.

Filter

A first order mathematic filter is built in. Time constant and band width are adjustable. The filter band width is the adjustable tolerance around the measured value within which the filter is active. Process value changes higher than the bandwidth are passed through directly (without filtering).

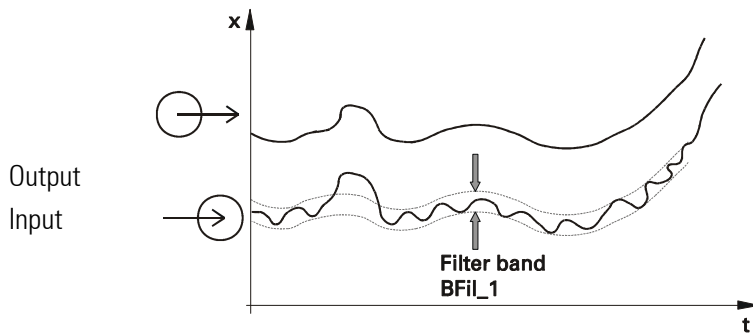


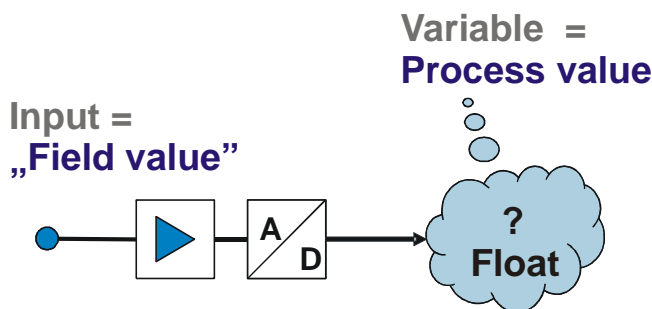
Fig. 9: Signal filtering using the filter time constant as long as it is within the bandwidth.

Scaling of a TC (thermocouple) input signal as a standard millivolt signal

If the input is configured as a millivolt input type, the displayed value should be adapted for the measured values. "Field values" are used as input values, i.e. the "raw measured values" as coming from the hardware input circuitry.

Normally, however, a value reflecting the meaning of the physical value in the system, for example the process temperature, instead of the actual input value in e.g. millivolt is used.

Linear 2-point scaling is used for this purpose: The span start input (raw) value is set to the display span start and the span end value is set to the display span end.



In the engineering, the display value (PV) appears as value of the input.

| | |
|--|--|
| <p>TC (mV linear)</p> <p>FVMin = -200 ≙ -20 mV</p> <p>FVMax = 600 ≙ 60 mV</p> | <p>Example: TC input (for mV linear):</p> <p>Signal: -20 ... 60 mV (field value FV)</p> <p>Range: -20 ... 60 mV (process value PV)</p> <p>FVMin / FVMax = -200 / 600</p> <p>PVMMin / PVMMax = -20 / 60 mV</p> |
|--|--|

Table: TC input (linear)

| | | |
|---------------|-------------------|--|
| Range: | Step width | Factor = number of steps per value of range |
| - 80...80mV | ± 8.000 | 100 |

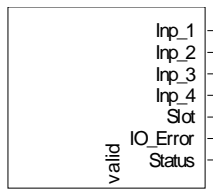
| In-/Outputs | | |
|-----------------|-------|---|
| Name | Type | Description |
| Inp_1 ... Inp_2 | Float | Produced measured value |
| Slot | Float | Own block number |
| IO_Error | Float | Type of the communication error to the module in the I/O system |
| Status | Float | Status message of the module |
| valid | Bool | The communication to I/O system (hardware) is established. Measured values are valid. |

| Parameter | | | | | | | |
|--------------------------|---------------------|------|-----------------------------------|--------|---------|---------------------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Offs_1 ... Offs_2 | Offset i1 [1/10] | Int | Offset [1/10 Unit] | r/w | 0 | -4000 ... 4000 | |
| FltV_1 ... FltV_2 | Error value i1 | Int | Substitute value with sensor fail | r/w | 0 | -30000 ... 30000 | |
| TFil_1 ... TFil_2 | Filt.time i1[1/10s] | Int | Filter time [1/10 s] | r/w | 5 | 0 ... 9999 | |
| BFil_1 ... BFil_2 | Filt.band i1 [1/10] | Int | Filter bandwidth [1/10] | r/w | 5 | 0 ... 9999 | |
| IFVMin_1 ... IFVMin_2 | IFVMin i1 | Int | Low field value | r/w | 0 | -30000 ... 30000 | |
| IFVMax_1 ... IFVMax_2 | IFVMax i1 | Int | High field value | r/w | 8000 | -30000 ... 30000 | |
| IPVMin_1 ... IPVMin_2 | IPVMin i1 | Int | Low Process value at FVMin | r/w | 0 | -30000 ... 30000 | |
| IPVMax_1 ... IPVMax_2 | IPVMax i1 | Int | High Process value at FVMax | r/w | 8000 | -30000 ... 30000 | |

| Configuration | | | | | | | |
|---------------|----------------|-------------|---|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| ErrBeh | Error behavior | Enum | Behaviour in error condition | r/w | 0 | | |
| | | Last value | Error behaviour: Use of the last value is continued. | | 0 | | |
| | | Zero | Error behaviour: Value zero (value is set to "0"). | | 1 | | |
| | | Error value | Error behaviour: Error value (the pre-defined error value is used). | | 2 | | |
| Func_1 ... | Function | Enum | Function of the input | r/w | 2 | | |

| | | | | | | | |
|-------------------------|------------------------|---------------------|---|-----|----|-------------------|--|
| Func_2 | input 1 | | | | | | |
| | | Channel disabled | Channel disabled | | 0 | | |
| | | No cold junct.comp. | No cold junct.comp. | | 1 | | |
| | | Cold junct.measured | Cold junct.measured | | 2 | | |
| | | Cold junct.constant | Cold junct.constant | | 3 | | |
| InpT_1 ... InpT_2 | Sensor type input 1 | Enum | Sensor type of the input | r/w | 0 | | |
| | | Typ L | Type L (-200...900°C, -328...1652°F), Fe-CuNi DIN | | 0 | | |
| | | Typ J | Type J (-210...1200°C, -346...2192°F), Fe-CuNi DIN | | 1 | | |
| | | Typ K | Type K (-270...1350°C, -454...2462°F), NiCr-Ni | | 2 | | |
| | | Typ N | Type N (-196...1300°C, -321...2372°F), Nicrosil-Nisil | | 3 | | |
| | | Typ S | Type S (-50...1760°C, -58...3200°F), PtRh-Pt10% | | 4 | | |
| | | Typ R | Type R (-50...1760°C, -58...3200°F), PtRh-Pt13% | | 5 | | |
| | | Typ T | Type T (-270...400°C, -454...752°F), Cu-CuNi | | 6 | | |
| | | Typ E | Type E (-270...1000°C, -454...1832°F), NiCr-CuNi | | 7 | | |
| | | Typ B | Type B (25...1820°C, 77 ... 3308°F), PtRh-Pt6% | | 8 | | |
| | | Typ W | Type W (0...2300°C, 32...4172°F) | | 9 | | |
| | | -80..80 mV | Voltage : -80...+80mV | | 10 | | |
| Unit_1 ... Unit_2 | Unit input 1 | Enum | Unit of the measurement value (°F) | r/w | 0 | | |
| | | °C | Unit = °C, degrees Celsius | | 0 | | |
| | | °F | Unit = °F, degrees Fahrenheit | | 1 | | |
| Temp_1 ... Temp_2 | Temp.cold junct. E1 | Int | Cold junction temperature | r/w | 0 | -2000 ... 8500 | |

III-18.18 Railline 424-1 (Input module with 2 thermocouple / O₂ inputs (No. 271))



RL_424-1

Abb. 656

The function block is used for configuration and parameter setting of a module with 2 thermocouple-(TC) and 2 O₂-inputs. With TC, various thermocouples and a millivolt input type are available. The inputs are galvanically isolated in groups of two.

The conditioned measured values are available at its outputs.

Output **Slot** must be soft-wired with the input of a **RL_BK_CAN_(no.)** type function block.

Coding of output IO-Error

See table in the general part of the description of RL 400 system. Only if no error message is pending at output IO_Error, data is exchanged with the module in the rail line system.

Coding of output Status

| Bit | Description |
|-----|---|
| 0 | Alarm 0 = 1, if analog input detects an error (sensor break, limits exceeded) and channel is active. Detection of sensor break only with input TC. |
| 1 | Not used. |
| 2 | Status 1 : EEPROM error |
| 3 | Not used. |



NOTE!

For additional information: see operating note of the **rail line** I/O module RL_424-X.

Filter

A first order mathematic filter is built in. Time constant and band width are adjustable. The filter band width is the adjustable tolerance around the measured value within which the filter is active. Process value changes higher than the bandwidth are passed through directly (without filtering).

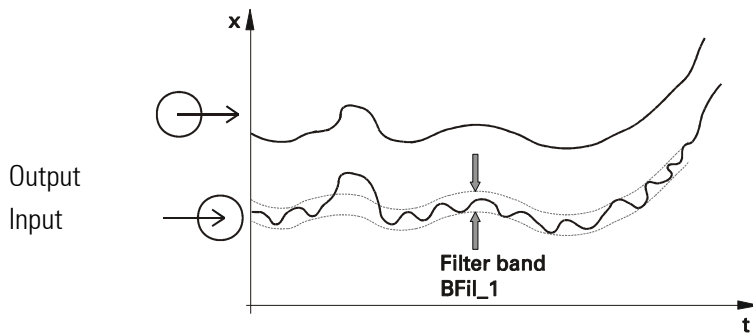


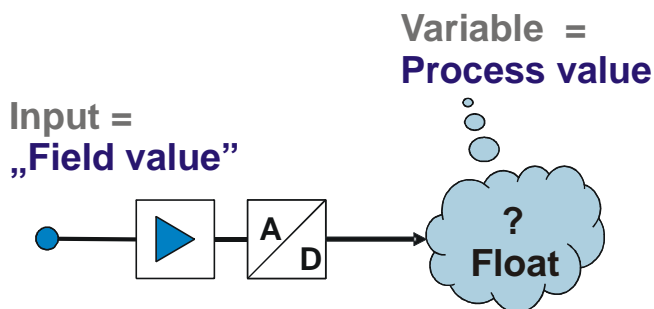
Fig. 9: Signal filtering using the filter time constant as long as it is within the bandwidth.

Scaling of a TC (thermocouple) input signal as a standard millivolt signal

If the input is configured as a millivolt input type, the displayed value should be adapted for the measured values. "Field values" are used as input values, i.e. the "raw measured values" as coming from the hardware input circuitry.

Normally, however, a value reflecting the meaning of the physical value in the system, for example the process temperature, instead of the actual input value in e.g. millivolt is used.

Linear 2-point scaling is used for this purpose: The span start input (raw) value is set to the display span start and the span end value is set to the display span end.



In the engineering, the display value (PV) appears as value of the input.

| | |
|--|--|
| <p>TC (mV linear)</p> <p>FVMin = -200 ≙ -20 mV</p> <p>FVMax = 600 ≙ 60 mV</p> | <p>Example: TC input (for mV linear):</p> <p>Signal: -20 ... 60 mV (field value FV)</p> <p>Range: -20 ... 60 mV (process value PV)</p> <p>FVMin / FVMax = -200 / 600</p> <p>PVMMin / PVMMax = -20 / 60 mV</p> |
|--|--|

Table: TC input (linear)

| Range: | Step width | Factor = number of steps per value of range |
|-------------|------------|---|
| - 80...80mV | ± 8.000 | 100 |

Scaling of O2 input signals

For explanation, see above: Scaling of a TC (thermocouple) input signal as a standard millivolt signal
 Linear 2-point scaling: The span start input (raw) value is set to the display span start and the span end input value is set to the display span end.

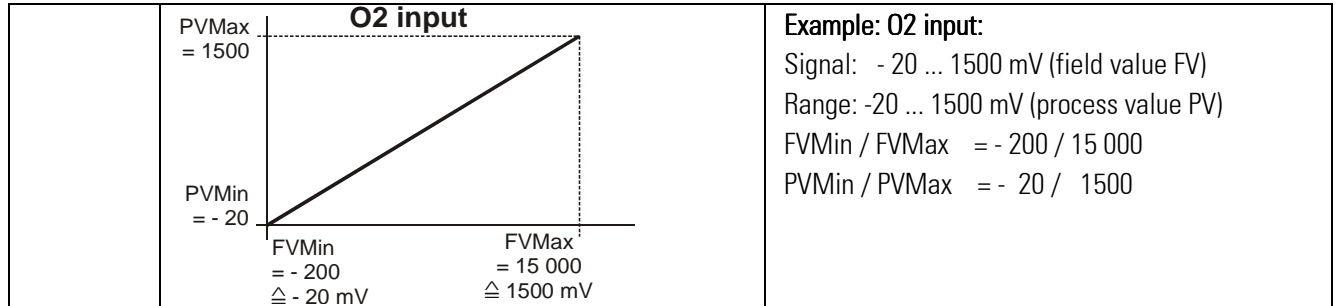


Table: mV input (see above: "Scaling of a TC (thermocouple) input signal as a standard millivolt signal")

| In-/Outputs | | |
|-----------------|-------|---|
| Name | Type | Description |
| Inp_1 ... Inp_4 | Float | Produced measured value |
| Slot | Float | Own block number |
| IO_Error | Float | Type of the communication error to the module in the I/O system |
| Status | Float | Status message of the module |
| valid | Bool | The communication to I/O system (hardware) is established. Measured values are valid. |

| Parameter | | | | | | | |
|--------------------------|---------------------|------|-----------------------------------|--------|---------|---------------------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Offs_1, Offs_3 | Offset i1 [1/10] | Int | Offset [1/10 Unit] | r/w | 0 | -4000 ... 4000 | |
| FltV_1 ... FltV_4 | Error value i1 | Int | Substitute value with sensor fail | r/w | 0 | -30000 ... 30000 | |
| TFil_1 ... TFil_4 | Filt.time i1[1/10s] | Int | Filter time [1/10 s] | r/w | 5 | 0 ... 9999 | |
| BFil_1 ... BFil_4 | Filt.band i1 [1/10] | Int | Filter bandwidth [1/10] | r/w | 5 | 0 ... 9999 | |
| IFVMin_1 ... IFVMin_4 | IFVMin i1 | Int | Low field value | r/w | 0 | -30000 ... 30000 | |
| IFVMax_1 ... IFVMax_4 | IFVMax i1 | Int | High field value | r/w | 8000 | -30000 ... 30000 | |
| IPVMin_1 ... IPVMin_4 | IPVMin i1 | Int | Low Process value at FVMin | r/w | 0 | -30000 ... 30000 | |
| IPVMax_1 ... | IPVMax i1 | Int | High Process value | r/w | 8000 | -30000 ... | |

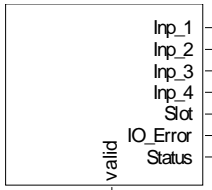
| | | | | | | | |
|----------------|-------------------|-----|---------------------|-----|---|---------------------|--|
| IPVMax_4 | | | at FVMax | | | 30000 | |
| Offs_2, Offs_4 | Offset i2 [1/100] | Int | Offset [1/100 Unit] | r/w | 0 | -10000 ... 10000 | |

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|----------------------|---------------------|---------------------|---|--------|---------|-------|-----|
| ErrBeh | Error behavior | Enum | Behaviour in error condition | r/w | 0 | | |
| | | Last value | Error behaviour: Use of the last value is continued. | | 0 | | |
| | | Zero | Error behaviour: Value zero (value is set to "0"). | | 1 | | |
| | | Error value | Error behaviour: Error value (the pre-defined error value is used). | | 2 | | |
| Func_1 ... Func_4 | Function input 1 | Enum | Function of the input | r/w | 2 | | |
| | | Channel disabled | Channel disabled | | 0 | | |
| | | No cold junct.comp. | No cold junct.comp. | | 1 | | |
| | | Cold junct.measured | Cold junct.measured | | 2 | | |
| | | Cold junct.constant | Cold junct.constant | | 3 | | |
| InpT_1 ... InpT_4 | Sensor type input 1 | Enum | Sensor type of the input | r/w | 0 | | |
| | | Typ L | Type L (-200...900°C, -328...1652°F), Fe-CuNi DIN | | 0 | | |
| | | Typ J | Type J (-210...1200°C, -346...2192°F), Fe-CuNi DIN | | 1 | | |
| | | Typ K | Type K (-270...1350°C, -454...2462°F), NiCr-Ni | | 2 | | |
| | | Typ N | Type N (-196...1300°C, -321...2372°F), Nicrosil-Nisil | | 3 | | |
| | | Typ S | Type S (-50...1760°C, -58...3200°F), PtRh-Pt10% | | 4 | | |
| | | Typ R | Type R (-50...1760°C, -58...3200°F), PtRh-Pt13% | | 5 | | |
| | | Typ T | Type T (-270...400°C, -454...752°F), Cu-CuNi | | 6 | | |
| | | Typ E | Type E (-270...1000°C, -454...1832°F), NiCr-CuNi | | 7 | | |
| | | Typ B | Type B (25...1820°C, 77 ... | | 8 | | |

| | | | | | | | |
|----------------------|------------------------|------------|---------------------------------------|-----|------|-------------------|--|
| | | | 3308°F), PtRh-Pt6% | | | | |
| | | Typ W | Type W (0...2300°C, 32...4172°F) | | 9 | | |
| | | -80..80 mV | Voltage : -80...+80mV | | 10 | | |
| Unit_1 ... Unit_4 | Unit input 1 | Enum | Unit of the measurement value (°F) | r/w | 0 | | |
| | | °C | Unit = °C, degrees Celsius | | 0 | | |
| | | °F | Unit = °F, degrees Fahrenheit | | 1 | | |
| Temp_1, Temp_3 | Temp.cold junct. E1 | Int | Cold junction temperature | r/w | 0 | -2000 ... 8500 | |
| Temp_2, Temp_4 | ProbeTemp E2[1/10] | Int | Probe temperature | r/w | 7300 | -2000 ... 8500 | |

III-18.19 Railline 424-2 (Input module for thermocouple TC (No. 272))



RL_424-2
Abb. 657

The function block is used for configuration and parameter setting of a module with thermocouple inputs TC. The inputs are galvanically isolated in groups of two. Various thermocouples and a millivolt input type are available as input types.

The conditioned measured values are available at its outputs.

Output **Slot** must be soft-wired with the input of a **RL_BK_CAN_(no.)** type function block.

Coding of output IO-Error

See table in the general part of the description of RL 400 system. Only if no error message is pending at output IO_Error, data is exchanged with the module in the rail line system.

Coding of output Status

| Bit | Description |
|-----|--|
| 0 | Alarm 0 = 1, if analog input detects an error (sensor break, limits exceeded) and channel is active. Detection of sensor break only with input TC. |
| 1 | Not used. |
| 2 | Status 1 : EEPROM error |
| 3 | Not used. |



NOTE!

For additional information: see operating note of the **rail line** I/O module RL_424-X.

Filter

A first order mathematic filter is built in. Time constant and band width are adjustable. The filter band width is the adjustable tolerance around the measured value within which the filter is active. Process value changes higher than the bandwidth are passed through directly (without filtering).

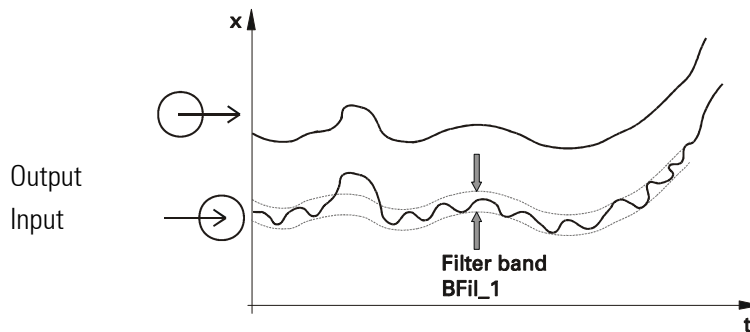


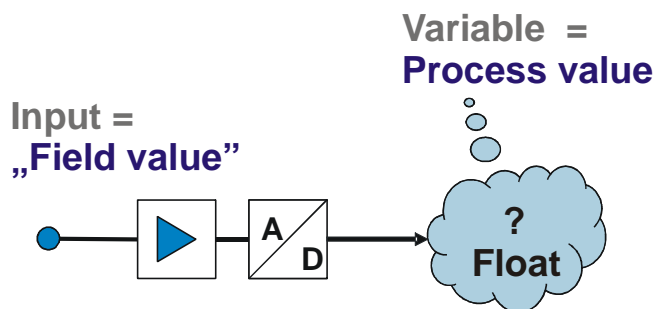
Fig. 9: Signal filtering using the filter time constant as long as it is within the bandwidth.

Scaling of a TC (thermocouple) input signal as a standard millivolt signal

If the input is configured as a millivolt input type, the displayed value should be adapted for the measured values. "Field values" are used as input values, i.e. the "raw measured values" as coming from the hardware input circuitry.

Normally, however, a value reflecting the meaning of the physical value in the system, for example the process temperature, instead of the actual input value in e.g. millivolt is used.

Linear 2-point scaling is used for this purpose: The span start input (raw) value is set to the display span start and the span end value is set to the display span end.



In the engineering, the display value (PV) appears as value of the input.

| | |
|------------------------------|---|
| <p>TC (mV linear)</p> | <p>Example: TC input (for mV linear):</p> <p>Signal: -20 ... 60 mV (field value FV)</p> <p>Range: -20 ... 60 mV (process value PV)</p> <p>FVMin / FVMax = -200 / 600</p> <p>PVMin / PVMMax = -20 / 60 mV</p> |
|------------------------------|---|

Table: TC input (linear)

| | | |
|---------------|-------------------|--|
| Range: | Step width | Factor = number of steps per value of range |
| - 80...80mV | ± 8.000 | 100 |

| In-/Outputs | | |
|-----------------|-------|---|
| Name | Type | Description |
| Inp_1 ... Inp_4 | Float | Produced measured value |
| Slot | Float | Own block number |
| IO_Error | Float | Type of the communication error to the module in the I/O system |
| Status | Float | Status message of the module |
| valid | Bool | The communication to I/O system (hardware) is established. Measured values are valid. |

| Parameter | | | | | | | |
|--------------------------|---------------------|------|-----------------------------------|--------|---------|---------------------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| Offs_1 ... Offs_4 | Offset i1 [1/10] | Int | Offset [1/10 Unit] | r/w | 0 | -4000 ... 4000 | |
| FltV_1 ... FltV_4 | Error value i1 | Int | Substitute value with sensor fail | r/w | 0 | -30000 ... 30000 | |
| TFil_1 ... TFil_4 | Filt.time i1[1/10s] | Int | Filter time [1/10 s] | r/w | 5 | 0 ... 9999 | |
| BFil_1 ... BFil_4 | Filt.band i1 [1/10] | Int | Filter bandwidth [1/10] | r/w | 5 | 0 ... 9999 | |
| IFVMin_1 ... IFVMin_4 | IFVMin i1 | Int | Low field value | r/w | 0 | -30000 ... 30000 | |
| IFVMax_1 ... IFVMax_4 | IFVMax i1 | Int | High field value | r/w | 8000 | -30000 ... 30000 | |
| IPVMin_1 ... IPVMin_4 | IPVMin i1 | Int | Low Process value at FVMin | r/w | 0 | -30000 ... 30000 | |
| IPVMax_1 ... IPVMax_4 | IPVMax i1 | Int | High Process value at FVMax | r/w | 8000 | -30000 ... 30000 | |

| Configuration | | | | | | | |
|---------------|----------------|-------------|---|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| ErrBeh | Error behavior | Enum | Behaviour in error condition | r/w | 0 | | |
| | | Last value | Error behaviour: Use of the last value is continued. | | 0 | | |
| | | Zero | Error behaviour: Value zero (value is set to "0"). | | 1 | | |
| | | Error value | Error behaviour: Error value (the pre-defined error value is used). | | 2 | | |
| Func_1 ... | Function | Enum | Function of the input | r/w | 2 | | |

| | | | | | | | |
|----------------------|------------------------|---------------------|---|-----|----|-------------------|--|
| Func_4 | input 1 | | | | | | |
| | | Channel disabled | Channel disabled | | 0 | | |
| | | No cold junct.comp. | No cold junct.comp. | | 1 | | |
| | | Cold junct.measured | Cold junct.measured | | 2 | | |
| | | Cold junct.constant | Cold junct.constant | | 3 | | |
| InpT_1 ... InpT_4 | Sensor type input 1 | Enum | Sensor type of the input | r/w | 0 | | |
| | | Typ L | Type L (-200...900°C, -328...1652°F), Fe-CuNi DIN | | 0 | | |
| | | Typ J | Type J (-210...1200°C, -346...2192°F), Fe-CuNi DIN | | 1 | | |
| | | Typ K | Type K (-270...1350°C, -454...2462°F), NiCr-Ni | | 2 | | |
| | | Typ N | Type N (-196...1300°C, -321...2372°F), Nicrosil-Nisil | | 3 | | |
| | | Typ S | Type S (-50...1760°C, -58...3200°F), PtRh-Pt10% | | 4 | | |
| | | Typ R | Type R (-50...1760°C, -58...3200°F), PtRh-Pt13% | | 5 | | |
| | | Typ T | Type T (-270...400°C, -454...752°F), Cu-CuNi | | 6 | | |
| | | Typ E | Type E (-270...1000°C, -454...1832°F), NiCr-CuNi | | 7 | | |
| | | Typ B | Type B (25...1820°C, 77 ...3308°F), PtRh-Pt6% | | 8 | | |
| | | Typ W | Type W (0...2300°C, 32...4172°F) | | 9 | | |
| | | -80..80 mV | Voltage : -80...+80mV | | 10 | | |
| Unit_1 ... Unit_4 | Unit input 1 | Enum | Unit of the measurement value (°F) | r/w | 0 | | |
| | | °C | Unit = °C, degrees Celsius | | 0 | | |
| | | °F | Unit = °F, degrees Fahrenheit | | 1 | | |
| Temp_1 ... Temp_4 | Temp.cold junct. E1 | Int | Cold junction temperature | r/w | 0 | -2000 ... 8500 | |

III-18.20 Railline 431 (Analog output module (No. 262))



RL_431

Abb. 658

The function block is used for configuration and parameter setting of a module with 4 analog outputs. The outputs may be current or voltage outputs.

The output values must be applied to its inputs.

Output **Slot** must be soft-wired with the input of a **RL_BK_CAN_(no.)** type function block.

Coding of output IO-Error

See table in the general part of the description of RL 400 system. Only if no error message is pending at output IO_Error, data is exchanged with the module in the rail line system.

Coding of output Status

| Bit | Description |
|-----|--|
| 0 | Not used. |
| 1 | Not used. |
| 2 | Status 1 : EEPROM error |
| 3 | Wrong_output_value; signal for a process value transmitted for a channel, that exceeds the admissible limits |



NOTE!

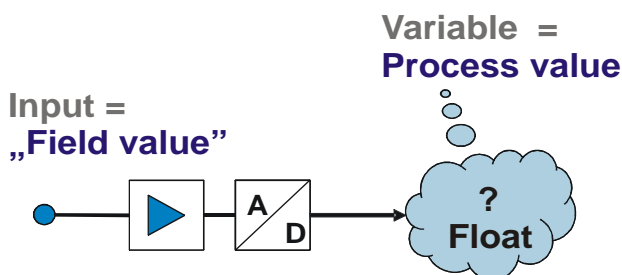
For additional information: see operating note of the **rail line** I/O module RL_431.

Standard output signal scaling

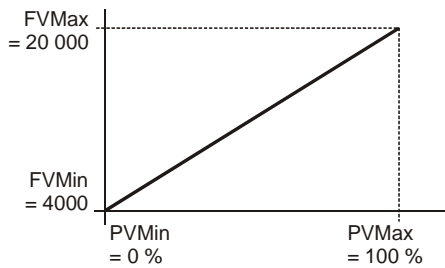
With the standard output signals, the field value should be adapted to the display value. The "field values" are output as electrical signals, i.e. the values the hardware needs for correct conversion into electrical signals.

Normally, however, a value reflecting the meaning of the physical value in the system, for example the per cent output value, instead of the actual output value in e.g. millivolt is used.

For this purpose, linear 2-point scaling is used: the output span start (e.g. in per cent) is set to the field value span start (e.g. in Volt) , and the output span end is set to the field value span end.



In the engineering, the "display" value is used. For a controller output, this is e.g. the per cent value . The function block calculates the corresponding "field value" and transmits it to the hardware.



Example: Output signal

| | | |
|------------------------|---------------|---|
| Signal: | 4 ... 20 mA | (field value FV) |
| Range: | 0 ... 100 % | (process value PV) |
| PVMin / PVMax = | 0 / 100 % | (display span start / display span end) |
| FVMin / FVMax = | 4000 / 20 000 | (field value span start / field value span end) |

| Range: | Step width | Factor = steps per value of range |
|---------------|------------|-----------------------------------|
| -20 ... 20 mA | ± 20 000 | 1000 |
| -10 ... 10 V | ± 10 000 | 1000 |
| -5 ... 5 mA | ± 5 000 | 1000 |
| -1 ... 1 mA | ± 10 000 | 10 000 |

In-/Outputs

| Name | Type | Description |
|------------------|-------|--------------|
| Outp_1 ... Outp4 | Float | Output value |

| Name | Type | Description |
|----------|-------|---|
| Slot | Float | Own block number |
| IO_Error | Float | Type of the communication error to the module in the I/O system |
| Status | Float | Status message of the module |
| valid | Bool | The communication to I/O system (hardware) is established. Measured values are valid. |

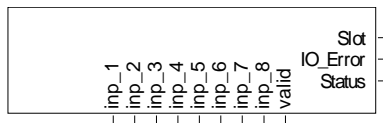
Parameter

| ID | Name | Type | Description | Access | Default | Range | Off |
|-------------------|------------------|------|--------------------------------------|--------|---------|---------------------|-----|
| FltV_1 ... FltV_4 | Error value 1 | Int | Substitute value with sensor fail | r/w | 0 | -30000 ... 30000 | |
| FVMin_1 ... | FVMin 1 | Int | Low field value | r/w | 0 | -30000 ... | |

| | | | | | | | |
|------------------------|---------|-----|--------------------------------|-----|-------|---------------------|--|
| FVMin_4 | | | | | | 30000 | |
| FVMax_1 ... FVMax_4 | FVMax 1 | Int | High field value | r/w | 20000 | -30000 ... 30000 | |
| PVMin_1 ... PVMin_4 | PVMin 1 | Int | Low process value at FVMin | r/w | 0 | -30000 ... 30000 | |
| PVMax_1 ... PVMax_4 | PVMax 1 | Int | High process value at FVMax | r/w | 20000 | -30000 ... 30000 | |

| Configuration | | | | | | | |
|------------------------|----------------|-------------|---|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| ErrBeh | Error behavior | Enum | Behaviour in error condition | r/w | 0 | | |
| | | Last value | Error behaviour: Use of the last value is continued. | | 0 | | |
| | | Zero | Error behaviour: Value zero (value is set to "0"). | | 1 | | |
| | | Error value | Error behaviour: Error value (the pre-defined error value is used). | | 2 | | |
| OutpT_1 ... OutpT_4 | Output type | Enum | Function of continuous output | r/w | 1 | | |
| | | -10...+10 V | Voltage : -10...+10 V | | 0 | | |
| | | -20...20mA | Standard -20...20mA signal | | 1 | | |

III-18.21 Railline 442 (Digital input module (No. 250))



RL_442

Abb. 659

The function block is used for configuration and parameter setting of a digital input module. The digital input signals are available at its outputs.

The various hardware version are not distinguished in the engineering, i.e. modules RL 442-0 digital input 2 x 4 24 VDC (pnp), RL 442-1 digital input 2 x 4 24 VDC (npn) and RL 442-2 digital input 2 x 4 contact (potential-free) are included into the engineering by the same RL442 function block.

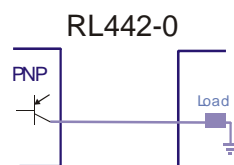


Abb. 660

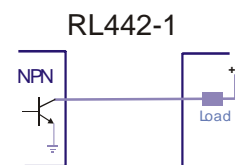


Abb. 661

Output **Slot** must be soft-wired with the input of a **RL_BK_CAN_(no.)** type function block according to module position.

Coding of output IO-Error

See table in the general part of the description of RL 400 system. Only if no error message is pending at output IO_Error, data is exchanged with the module in the rail line system.

Coding of output Status

| Bit | Description |
|-----|--------------------------|
| 0 | Not used. |
| 1 | Not used. |
| 2 | Status 1 : EEPROM error. |
| 3 | Not used. |



NOTE!

For additional information: see operating note of the **rail line** I/O module RL_442-X.

In-/Outputs

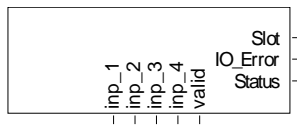
| Name | Type | Description |
|-----------------|------|--|
| inp_1 ... inp_8 | Bool | Signal at input 1 of the module |
| valid | Bool | The communication to I/O system (hardware) is established. Measured values |

| | | |
|----------|-------|---|
| | | are valid. |
| Slot | Float | Own block number |
| IO_Error | Float | Type of the communication error to the module in the I/O system |
| Status | Float | Status message of the module |

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|--------|---------------------|------------|--|--------|---------|-------------|-----|
| ErrBeh | Error behavior | Enum | Behaviour in error condition | r/w | 0 | | |
| | | Last value | Error behaviour: Use of the last value is continued. | | 0 | | |
| | | Zero | Error behaviour: Value zero (value is set to "0"). | | 1 | | |
| Plty | Polarity [Inp 1..8] | Int | Polarity of input 1 ... 8; direct = 0; invers = 1 | r/w | 0 | 0 ... 255 | |
| Time | Debounce [ms] | Int | Debounce time [ms] | r/w | 0 | 0 ... 30000 | |

III-18.22 Railline 443 (Digital input module AC (No. 251))



RL_443

Abb. 662

The function block is used for configuration and parameter setting of a digital input module. The logic is of the 115V/230V AC type.

The digital input signals are available at its outputs.

Output **Slot** must be soft-wired with the input of a **RL_BK_CAN_(no.)** type function block.

Coding of output IO-Error

See table in the general part of the description of RL 400 system. Only if no error message is pending at output IO_Error, data is exchanged with the module in the rail line system.

Coding of output Status

| Bit | Description |
|-----|--------------------------|
| 0 | Not used. |
| 1 | Not used. |
| 2 | Status 1 : EEPROM error. |
| 3 | Not used. |



NOTE!

For additional information: see operating note of the **rail line I/O module RL_443-0**.

In-/Outputs

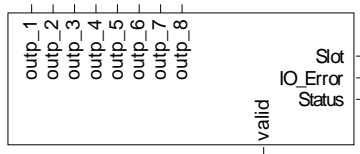
| Name | Type | Description |
|-----------------|-------|---|
| inp_1 ... inp_4 | Bool | Signal at input 1 of the module |
| valid | Bool | The communication to I/O system (hardware) is established. Measured values are valid. |
| Slot | Float | Own block number |
| IO_Error | Float | Type of the communication error to the module in the I/O system |
| Status | Float | Status message of the module |

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|----|------|------|-------------|--------|---------|-------|-----|
|----|------|------|-------------|--------|---------|-------|-----|

| ErrBeh | Error behavior | Enum | Behaviour in error condition | r/w | 0 | | |
|--------|---------------------|------------|--|-----|---|-------------|--|
| | | Last value | Error behaviour: Use of the last value is continued. | | 0 | | |
| | | Zero | Error behaviour: Value zero (value is set to "0"). | | 1 | | |
| Plty | Polarity [Inp 1..4] | Int | Polarity of input 1 ... 4; direct = 0; invers = 1 | r/w | 0 | 0 ... 15 | |
| Time | Debounce [ms] | Int | Debounce time [ms] | r/w | 0 | 0 ... 30000 | |

III-18.23 Railline 451 (Digital output module (No. 255))



RL_451

Abb. 663

The function block is used for configuration and parameter setting of a digital output module for 24V DC logic. The output values must be applied to its inputs.

The engineering does not distinguish between the two hardware versions **RL_451-0** (without protective diode) and **RL451-1** (with protective diode). The two modules are integrated into the engineering by the same function block.

Output **Slot** must be soft-wired with the input of a **RL_BK_CAN_(no.)** type function block.

Coding of output IO-Error

See table in the general part of the description of RL 400 system. Only if no error message is pending at output IO_Error, data is exchanged with the module in the rail line system.

Coding of output Status

| Bit | Description |
|-----|---|
| 0 | Alarm 0 = 1 no power supply for the output. |
| 1 | Alarm 1 = 1, if an error (open-circuit or short circuit) is detected on a channel that is activated and the error mask of which is enabled. |
| 2 | Status 1 : EEPROM error. |
| 3 | Wrong_output_value; signal for a process value transmitted for a channel, that exceeds the admissible limits |



NOTE!

For additional information: see operating note of the **rail line** I/O module RL_451-X.

In-/Outputs

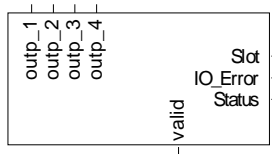
| Name | Type | Description |
|-------------------|-------|---|
| outp_1 ... outp_8 | Bool | Signal at output 1 of the module |
| Name | Type | Description |
| valid | Bool | The communication to I/O system (hardware) is established. Measured values are valid. |
| Slot | Float | Own block number |
| IO_Error | Float | Type of the communication error to the module in the I/O system |

| | | |
|--------|-------|------------------------------|
| Status | Float | Status message of the module |
|--------|-------|------------------------------|

Configuration

| ID | Name | Type | Description | Access | Default | Range | Off |
|--------|---------------------|-------------|---|--------|---------|-------------|-----|
| ErrBeh | Error behavior | Enum | Behaviour in error condition | r/w | 0 | | |
| | | Last value | Error behaviour: Use of the last value is continued. | | 0 | | |
| | | Zero | Error behaviour: Value zero (value is set to "0"). | | 1 | | |
| | | Error value | Error behaviour: Error value (the pre-defined error value is used). | | 2 | | |
| Plty | Polarity [Ou1..8] | Int | Polarity of output 1 ... 8; direct = 0; invers = 1 | r/w | 0 | 0 ... 255 | |
| Actv | Activity [Ou1..8] | Int | Activity of the output 1 ... 8; not active = 0; active = 1 | r/w | 255 | 0 ... 255 | |
| FltS | Err. state [Ou1..8] | Int | State of output 1 ... 8 in the fault condition | r/w | 0 | 0 ... 255 | |
| ErrM | Err. mask [Ou1..8] | Int | Mask for the fault condition for output 1 ... 8 | r/w | 0 | 0 ... 255 | |
| TktM | Pulse mask [Ou1..8] | Int | Mask for using the clock pulse for output 1 ... 8 | r/w | 0 | 0 ... 255 | |
| TktD | Start delay [ms] | Int | Clock pulse start delay [ms] | r/w | 1000 | 0 ... 30000 | |
| TktP | Period [ms] | Int | Clock pulse period [ms] | r/w | 100 | 5 ... 30000 | |
| TktE | Cyc. duration [ms] | Int | Clock pulse cycle duration [ms] | r/w | 50 | 0 ... 30000 | |

III-18.24 Railline 452 (Relay module (No. 256))



RL_452

Abb. 664

The function block is used for configuration and parameter setting of a module with relay outputs for 4 x AC (115/230V AC).

The output values must be applied to its inputs.

Output **Slot** must be soft-wired with the input of a **RL_BK_CAN_(no.)** type function block.

Coding of output IO-Error

See table in the general part of the description of RL 400 system. Only if no error message is pending at output IO_Error, data is exchanged with the module in the rail line system.

Coding of output Status

| Bit | Description |
|-----|--|
| 0 | Not used. |
| 1 | Not used. |
| 2 | Status 1 : EEPROM error |
| 3 | Wrong_output_value; signal for a process value transmitted for a channel, that exceeds the admissible limits |



NOTE!

For additional information: see operating note of the **rail line** I/O module RL_452-X.

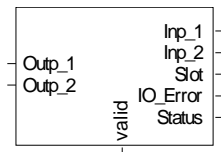
In-/Outputs

| Name | Type | Description |
|-----------------------|------|----------------------------------|
| output_1 ... output_4 | Bool | Signal at output 1 of the module |

| Name | Type | Description |
|----------|-------|---|
| valid | Bool | The communication to I/O system (hardware) is established. Measured values are valid. |
| Slot | Float | Own block number |
| IO_Error | Float | Type of the communication error to the module in the I/O system |
| Status | Float | Status message of the module |

| Configuration | | | | | | | |
|---------------|---------------------|-------------|---|--------|---------|----------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| ErrBeh | Error behavior | Enum | Behaviour in error condition | r/w | 0 | | |
| | | Last value | Error behaviour: Use of the last value is continued. | | 0 | | |
| | | Zero | Error behaviour: Value zero (value is set to "0"). | | 1 | | |
| | | Error value | Error behaviour: Error value (the pre-defined error value is used). | | 2 | | |
| Plty | Polarity [Ou1..4] | Int | Polarity of output 1 ... 4; direct = 0; invers = 1 | r/w | 0 | 0 ... 15 | |
| Actv | Activity [Ou1..4] | Int | Activity of output 1 ... 4; not active = 0; active = 1 | r/w | 15 | 0 ... 15 | |
| FltS | Err. state [Ou1..4] | Int | State of output 1 ... 4 in the fault condition | r/w | 0 | 0 ... 15 | |

III-18.25 Railline 461 (Analog combination module (No. 261))



RL_461

Abb. 665

The function block is used for configuration and parameter setting of a combi-module with 2 analog in- and 2 analog outputs.

Current or voltage signals can be measured or output. The inputs and outputs are galvanically isolated from the logic circuitry and from the internal supply voltage (inputs are galvanically connected, outputs are galvanically connected). The outputs can be selected individually for each channel. All outputs are protected against short circuit.

The conditioned measured values are available at its outputs. The output values must be applied to its inputs. Output **Slot** must be soft-wired with the input of a **RL_BK_CAN_(no.)** type function block.

Coding of output IO-Error

See table in the general part of the description of RL 400 system. Only if no error message is pending at output IO_Error, data is exchanged with the module in the rail line system.

Coding of output Status

| Bit | Description |
|-----|--|
| 0 | Alarm 0 = 1, if analog input detects an error (limits exceeded) and channel is active. |
| 1 | Not used. |
| 2 | Status 1 : EEPROM error |
| 3 | Wrong_output_value; signal for a process value transmitted for a channel, that exceeds the admissible limits |



NOTE!

For additional information: see operating note of the **rail line** I/O module RL_461.

Filter

A first order mathematic filter is built in. Time constant and band width are adjustable. The filter band width is the adjustable tolerance around the measured value within which the filter is active. Process value changes higher than the bandwidth are passed through directly (without filtering).

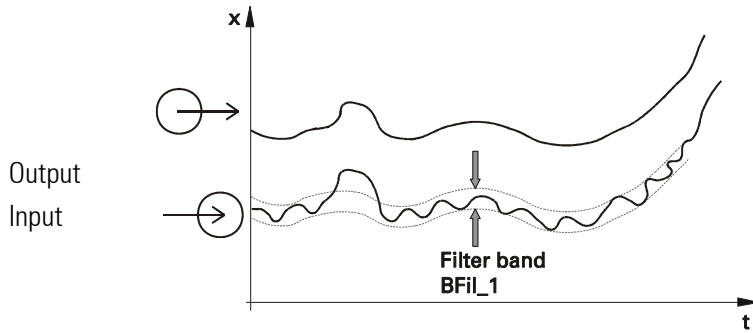
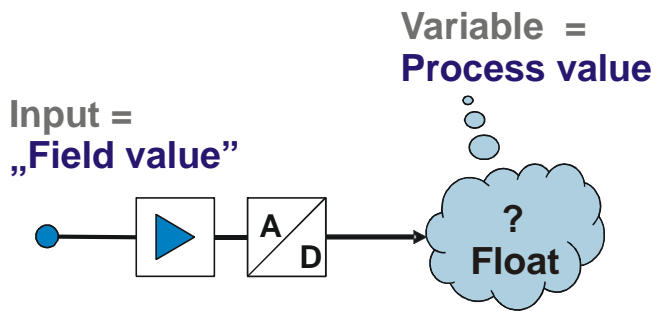


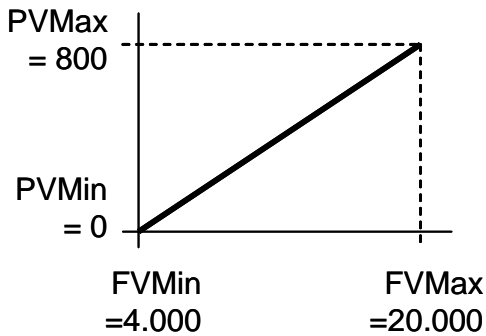
Fig. 9: Signal filtering using the filter time constant as long as it is within the bandwidth.

Standard input signal scaling

With the standard input signals, the displayed value should be adapted to the measured values. Originally, the measured values are "field values", i.e. "raw measured values" as coming from the input hardware. Normally, however, the "display value" reflecting the meaning of the physical value in the system, for example the filling level of a bowl instead of the actual input value (e.g. in millivolt) is used. For this purpose, linear 2-point scaling is possible: the input span start (= field value PV) is set to the display span start (=process value PV) and the span end value is set to the display span end.



In the engineering, the display value (PV) appears as value of the input.



Example: Standard input signal converted into level

| | | |
|------------------------|--------------------------|---|
| Signal: | 4 ... 20 mA | (field value FV) |
| Range: | 0 ... 800 m ³ | (process value PV) |
| FVMin / FVMax = | 4000 / 20 000 | (field value span start / field value span end) |
| PVMin / PVMax = | 0 / 800 m ³ | (display span start / display span end) |

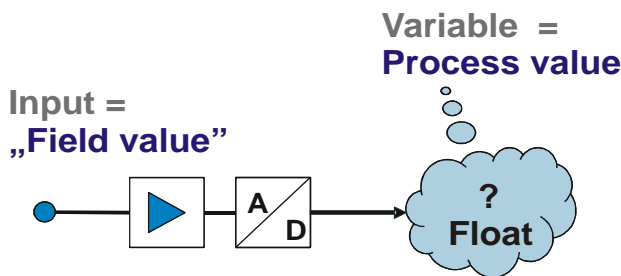
| | Range: | Step width | Factor = steps per value of range |
|--|---------------|------------|-----------------------------------|
| | -20 ... 20 mA | ± 20 000 | 1 000 |
| | -10 ... 10 V | ± 10 000 | 1 000 |
| | -5 ... 5 mA | ± 5 000 | 1 000 |
| | -1 ... 1 mA | ± 10 000 | 10 000 |

Standard output signal scaling

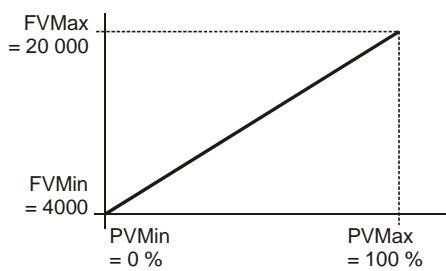
With the standard output signals, the field value should be adapted to the display value. The "field values" are output as electrical signals, i.e. the values the hardware needs for correct conversion into electrical signals.

Normally, however, a value reflecting the meaning of the physical value in the system, for example the per cent output value, instead of the actual output value in e.g. millivolt is used.

For this purpose, linear 2-point scaling is used: the output span start (e.g. in per cent) is set to the field value span start (e.g. in Volt), and the output span end is set to the field value span end.



In the engineering, the "display" value is used. For a controller output, this is e.g. the per cent value. The function block calculates the corresponding "field value" and transmits it to the hardware.



Example: Output signal

| | | |
|------------------------|---------------|---|
| Signal: | 4 ... 20 mA | (field value FV) |
| Range: | 0 ... 100 % | (process value PV) |
| PVMin / PVMax = | 0 / 100 % | (display span start / display span end) |
| FVMin / FVMax = | 4000 / 20 000 | (field value span start / field value span end) |

| | Range: | Step width | Factor = steps per value of range |
|--|---------------|------------|-----------------------------------|
| | -20 ... 20 mA | ± 20 000 | 1000 |

| | | | |
|--|--------------|----------|--------|
| | -10 ... 10 V | ± 10 000 | 1000 |
| | -5 ... 5 mA | ± 5 000 | 1000 |
| | -1 ... 1 mA | ± 10 000 | 10 000 |

In-/Outputs

| Name | Type | Description |
|----------------------|-------|--------------|
| Outp_1 ... Outp_2 | Float | Output value |

| Name | Type | Description |
|--------------------|-------|---|
| Inp_1 ... Inp_2 | Float | Conditioned measured value 1 |
| Slot | Float | Own block number |
| IO_Error | Float | Type of the communication error to the module in the I/O system |
| Status | Float | Status message of the module |
| valid | Bool | The communication to I/O system (hardware) is established. Measured values are valid. |

Parameter

| ID | Name | Type | Description | Access | Default | Range | Off |
|--------------------------|------------------------|------|--|--------|---------|---------------------|-----|
| IFItV_1 ... IFItV_2 | IError value i1 | Int | Substitute value with sensor fail | r/w | 0 | -30000 ... 30000 | |
| ITFil_1 ... ITFil_2 | IFil.time i1[1/10s] | Int | Filter time constant of signal processing in s | r/w | 5 | 0 ... 9999 | |
| IBFil_1 ... IBFil_2 | IFilt.band i1[1/10] | Int | Filter bandwidth | r/w | 5 | 0 ... 9999 | |
| IFVMin_1 ... IFVMin_2 | IFVMin i1 | Int | Low field value | r/w | 0 | -30000 ... 30000 | |
| IFVMax_1 ... IFVMax_2 | IFVMax i1 | Int | High field value | r/w | 20000 | -30000 ... 30000 | |
| IPVMin_1 ... IPVMin_2 | IPVMin i1 | Int | Low process value at FVMin | r/w | 0 | -30000 ... 30000 | |
| IPVMax_1 ... IPVMax_2 | IPVMax i1 | Int | High process value at FVMax | r/w | 20000 | -30000 ... 30000 | |
| OFItV_1 ... OFItV_2 | OError value o1 | Int | Substitute value with sensor fail | r/w | 0 | -30000 ... 30000 | |
| OFVMin_1 ... OFVMin_2 | OFVMin o1 | Int | Low field value | r/w | 0 | -30000 ... 30000 | |

| | | | | | | | |
|--------------------------|-----------|-----|--------------------------------|-----|-------|---------------------|--|
| OFVMax_1 ... OFVMax_2 | OFVMax o1 | Int | High field value | r/w | 20000 | -30000 ... 30000 | |
| OPVMin_1 ... OPVMin_2 | OPVMin o1 | Int | Low process value at FVMin | r/w | 0 | -30000 ... 30000 | |
| OPVMax_1 ... OPVMax_2 | OPVMax o1 | Int | High process value at FVMax | r/w | 20000 | -30000 ... 30000 | |

| Configuration | | | | | | | |
|------------------------|----------------|-------------|--|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| ErrBeh | Error behavior | Enum | Behaviour in error condition | r/w | 0 | | |
| | | Last value | Error behaviour: Use of the last value is continued. | | 0 | | |
| | | Zero | Error behaviour: Value zero (value is set to "0"). | | 1 | | |
| | | Error value | Error behaviour: Error value (the pre-defined error value is used). | | 2 | | |
| InpE_1 ... InpE_2 | Activated | Enum | Activate input | r/w | 1 | | |
| | | No | Switched off, e.g. to prevent error messages such as sensor break from being generated. The channel is not used. | | 0 | | |
| | | Yes | Switched on | | 1 | | |
| InpT_1 ... InpT_2 | Input type | Enum | Sensor type | r/w | 3 | | |
| | | -10...+10 V | Voltage : -10...+10 V | | 0 | | |
| | | -5...+5 V | Voltage : -5...+5 V | | 1 | | |
| | | -1...+1 V | Voltage : -1...+1 V | | 2 | | |
| | | -20...20mA | Current -20...20mA | | 3 | | |
| OutpT_1 ... OutpT_2 | Output type | Enum | Function of continuous output | r/w | 1 | | |
| | | -10...+10 V | Voltage : -10...+10 V | | 0 | | |
| | | -20...20mA | Standard -20...20mA signal | | 1 | | |

III-19 Universal programmer

III-19.1 General

The following section gives an overview of the functions provided by the programmer in *KS108 easy* and how to use them. Moreover, you will learn how to handle some typical tasks (for instance, defining a program).

The required environment for the function block PROGRAMMER includes the following components:

- The KS 108 instrument with the PMA and BlueDesign run-time software
- BlueDesign/PMA library with KS108 BlueSimulation
- I/O system
- BlueEdit for KS108

In the further course of this chapter, these components are described shortly with an explanation of their interactions. The chapter presents the steps required to create an operable program for KS108 that can be practised using an application example.

III-19.2 Prerequisites

To work with the PROGRAMMER, the following knowledge is required:

- Basic knowledge of the *Microsoft Windows*™ operating system

III-20 Installation and configuration

III-20.1 BlueDesign programming environment

III-20.1.1 Installing BlueDesign

The BlueDesign installation package can be found either on the CD delivered with *KS108 easy* or downloaded from the PMA homepage. Start the "*ibluedesign.exe*" installation software.

Follow the instructions displayed on the screen.



NOTE!

A detailed description for installation of BlueDesign is given in Chapter II "Development environment" of the KS108 easy manual.

III-20.1.2 Licencing BlueDesign

Without a valid licence, projects can be opened and handled in *BlueDesign* only during a short time. Enter the delivered licence under "?" in menu item „License..." of the main menu.



NOTE!

Information on BlueDesign licencing is given in Chapter II, section "Development environment" of the KS108 easy manual.

III-20.1.3 Configuring BlueDesign



NOTE!

For information on the configuration of BlueDesign, refer to Chapter II, section II "Operation of the development environment" in the KS108 easy manual.

III-20.1.4 Installing the Vario configurator

If you use the *Vario IO system*, you need the additional *VarioConfiguration* program. The installation package for *VarioConfiguration* can be found either on the CD delivered with *KS 108 easy* or downloaded from the PMA homepage. Start the "*varioconfiguration.zip*" installation package.

Follow the instructions displayed on the screen.



NOTE!

For a detailed description how to install the VarioConfiguration program, refer to Chapter II "Development environment" of the KS108 easy manual.

III-20.2 BlueEdit program editor

III-20.2.1 Installing, configuring and licencing BlueEdit

The installation package for BlueEdit can be found either on the CD delivered with *KS108 easy* or downloaded from the PMA homepage. Start the "*iblueedit.exe*" installation package.

Follow the instructions displayed on the screen.



NOTE!

For information on the installation of BlueEdit, refer to the BlueEdit operating manual.

BlueEdit requires a valid licence.



NOTE!

For information on the installation of BlueEdit, refer to the BlueEdit operating manual.



NOTE

For information on the configuration of BlueEdit, refer to the BlueEdit operating manual.

III-21 Components for program creation

The following section provides an overview of the components required to create a program for KS108 easy using the new PROGRAMMER function block.

Unless you have already worked using the PMA library or the development environment, we recommend reading the description "A practical example" in Chapter II Development environment of the *KS108 easy* manual first. In this section, the interaction of the *BlueDesign*, *BlueSimulation* and *KS108easy* components is explained at a practical example.

For program creation, a practical example for quick entrance is provided at the end of this chapter. See section Tutorial.

III-21.1 KS 108 easy with run-time environment

Basically, *KS 108 easy* is a computer with the LINUX operating system, i.e. in principle, the instrument might be used as a "normal" LINUX computer. Making *KS 108 easy* a powerful multi-function controller is achieved mainly by three software components:

- **BlueDesign run-time environment:** Permits the execution of *BlueDesign* applications on the instrument. User programs are created in the *BlueDesign* development environment and transferred to *KS 108 easy*, where they are executed in the run-time environment of the *BlueDesign* software.
- **PMA library:** Provides powerful function blocks for system operation which are used by the *BlueDesign* applications.
- **BlueEdit program editor:** For convenient recipe creation and management. Recipes are loaded into *KS108* dependent on requirement. The recipes make the PROGRAMMER function block a powerful but easy to operate programmer.

III-21.2 BlueDesign

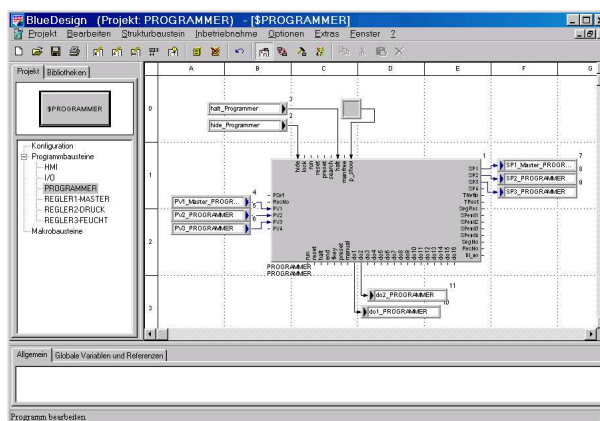


Abb. 666: BlueDesign

BlueDesign is a development environment for control applications. Applications are created in *BlueDesign* by selecting prefabricated components (e.g. programmers or controllers) using a graphic editor and connecting them. Programming knowledge is not required for this purpose.

Creating a project using *BlueDesign* includes the following steps: Developing the application, creating an operator interface, realizing the parameter setting and testing the application.



NOTE!

Detailed information how to create applications using *BlueDesign* and a practical example are given in Chapter II Development environment.

III-21.3 PMA library

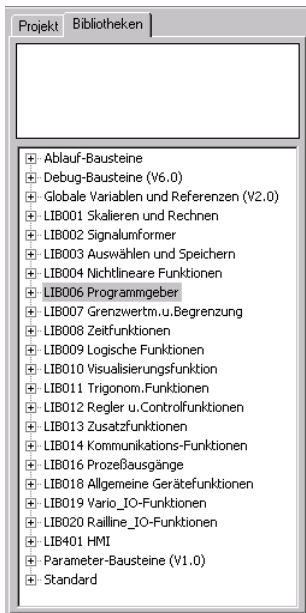



Abb. 667: PMA-Bibliothek

The engineering for *KS 108 easy* is developed using the *BlueDesign* tool based on the PMA library.

 **NOTE!**
For information on the PMA library, refer to Chapter II Development environment.

III-21.4 BlueSimulation

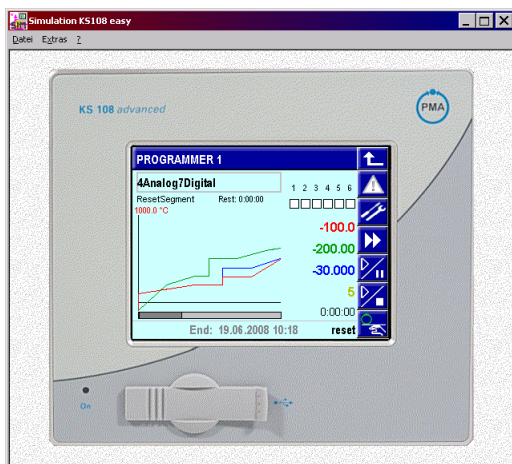


Abb. 668: BlueSimulation

The *BlueSimulation* tool simulates the *KS 108 easy* instrument. Behaviour and displays correspond to those of *KS108 easy*. Using the simulator, application development and program testing are possible also without the instrument.

III-21.5 BlueEdit

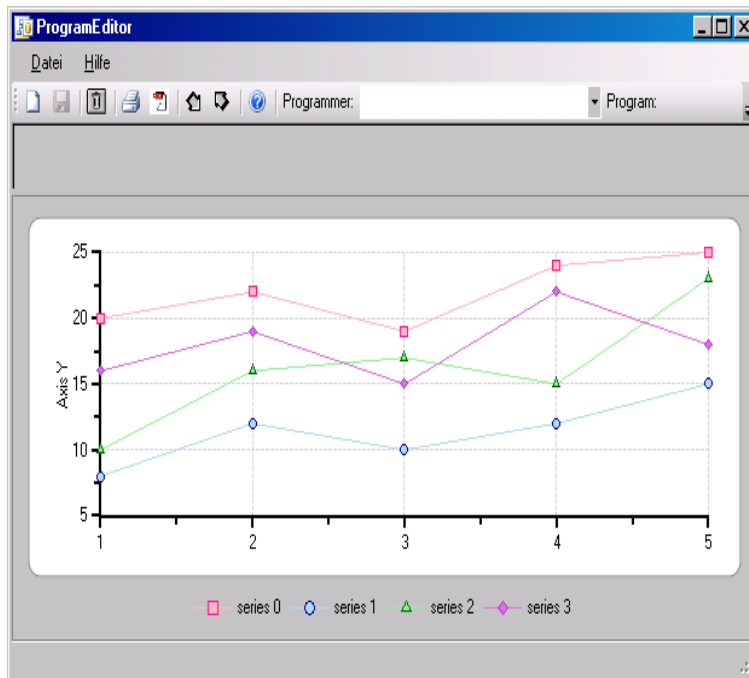


Abb. 669: BlueEdit with displayed profiles

The *BlueEdit* program editor is used for recipe and program creation, handling and management.

The basic parameters for the recipes are read out from an existing *BlueDesign* engineering which contains at least one PROGRAMMER function block. These include the number of available PROGRAMMER function blocks, whereby each of these function blocks comprises the recipe list, the number of analog and digital tracks. This is the framework for the recipes of this engineering for *KS108 easy*.

The recipes are created using *BlueEdit* and transferred into the *KS108 easy*. They can also be tested using the simulation. To work with the recipes, the engineering must run in *KS108 easy* with the correct parameters.



NOTE

Subsequent changes of the parameters (recipe list, number of tracks) in BlueDesign, KS108 easy or BlueEdit must be added manually, i.e. the parameters in the recipe and in the instrument must be identical.

III-22 From the idea to the programm sequence in the device

General

The programmer comprises a function block **PROGRAMMER** and at least one recipe file with a program. The program files are created conveniently using the *BlueEdit* program editor, however, they can be changed in the instrument. The programmer (**PROGRAMMER**) may have up to 4 analog and up to 16 digital tracks.

Each **PROGRAMMER** file includes a recipe with a program comprising an arbitrary number of segments. Additional recipes are added as one file for each recipe. The number of segments or recipes is limited only by the available memory size.

The directory in which the recipes are stored is determined in the engineering (use an own directory for each **PROGRAMMER!**). Recipe selection is possible on the operating page or via the analog input **RecNo**.

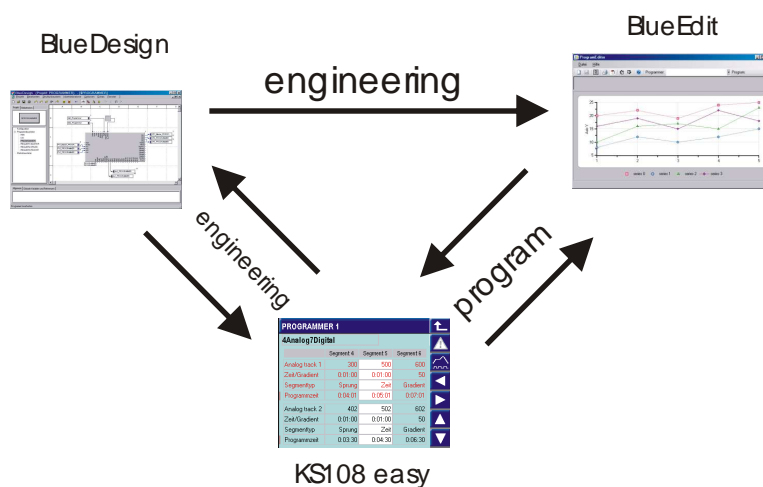


Abb. 670: Interaction of BlueDesign and KS108 easy

III-22.1 Preparation: Defining programs

III-22.1.1 Basic program structure

For using the universal programmer (FB PROGRAMMER), some basic parameters must be determined. These must be identical in the engineering and in the program and should be defined at the very beginning. Subsequent changes during the further course are possible. However, they require manual adaptations in the other components because these settings must be identical in the recipe and in the programmer configuration.

Programmer directory

A programmer may have a single recipe, however, a choice of several ones is normally offered. To enable the programmer in *KS108 easy* to find the programs and to change only its own ones, an own directory with a unique name must be specified for each programmer (i.e. for each PROGRAMMER function block). This directory must be entered into BlueDesign during configuration of the function block.

When loading the engineering into *KS108 easy*, this directory is created, unless it exists already.

The *BlueEdit* program editor reads out the directory for each programmer function block from the engineering and can then store the related recipes for each programmer in the right location in *KS108 easy*.

Number of analog tracks and master track

Each programmer can have up to 4 analog tracks. The first track is always the master track. The other tracks are coupled to this track either via the segment or via the time. For this reason, it is important to know the relationship of the tracks.

The names defined for the tracks should be as clear as possible.

Moreover, the operation of the programmer offers the possibility to assign a colour to each analog track so that the operator can realize the curves and values at a single glance.

Number of digital tracks: control tracks

Each programmer can have up to 16 digital tracks. The tracks are coupled to the master track via the segment and via the time.

The names defined for the tracks should be as clear and meaningful as possible.

During operation, the first six control tracks are displayed on the main page. For this reason, it is purposeful to assign the most important signals to control tracks DO1 to DO6.

Tip: To make an own segment for each switch point unnecessary, the user can set the status of the digital tracks independent of the relevant segment start and end by means of a switch-on delay and a duty cycle.

III-22.2 BlueDesign: Creating an engineering using the PROGRAMMER function block

The programmer in the engineering realizes the recipe-dependent program sequences represented by the controllers and by the remaining engineering with the application.

The most important settings and requirements for the programmer in the engineering are:

- Signal soft-wiring, i.e. connection of the PROGRAMMER to the overall engineering
- Configuration and parameter setting
- If necessary: user levels and passwords
- If necessary: HMI user visualization

Insert the PROGRAMMER function block into the engineering.

The programmer must be soft-wired dependent on requirements, whereby the following considerations must be taken into account:

- Are there any process values and where do they come from (e.g. search)?
- Which control inputs are required?
 - Which steps do you need to control via operation?
 - Does the operation have to be available at any time, or do you want to lock or suppress the display of operating pages when defined conditions arise?
 - Is manual operation required?
 - Do you need to make the program editing page available to permit modification of program data such as segment times or bandwidths?
- Which analog and digital tracks (SP1...SP4, do1 ... do16) are used, and where do you want to connect them?
- Which additional displays and signals do you want to use in the engineering?
- How many analog and how many digital tracks are provided?
- Which programmer behaviour after start-up and at the program end do you need to configure?
- Which are the couplings between the analog tracks?
- Which names do you want to use on the operating pages?
- What is the name of the recipe directory (unique name in *KS108 easy*)?
- Which users do you need to take into account? Is the engineering already provided with user (level) management via a PASSWORD function block, or do you have to create it now?
 - Which names do you want to assign to the users?
 - Which passwords (pass numbers) do you want to use?
 - Is the user management via the PASSWORD operating page accessible at any time?

The key data are the PROGRAMMER themselves and for every of these functionblocks the list of recipes and the amount of analog and digital tracks. If these are known and listed in the engineering, the creation of recipes can be started.

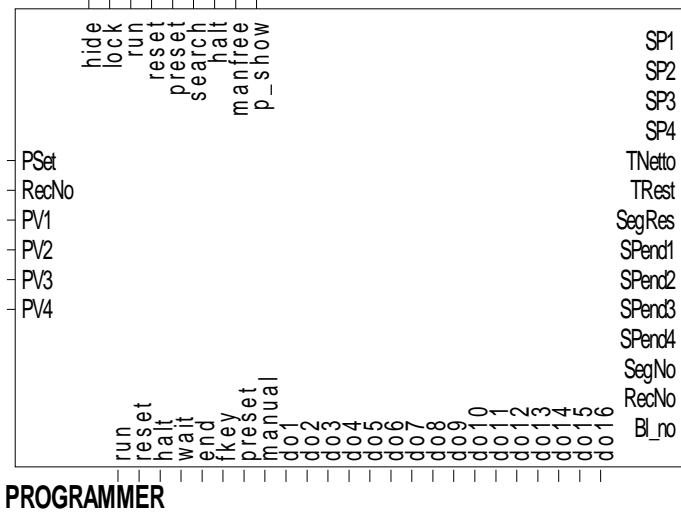


Abb. 671

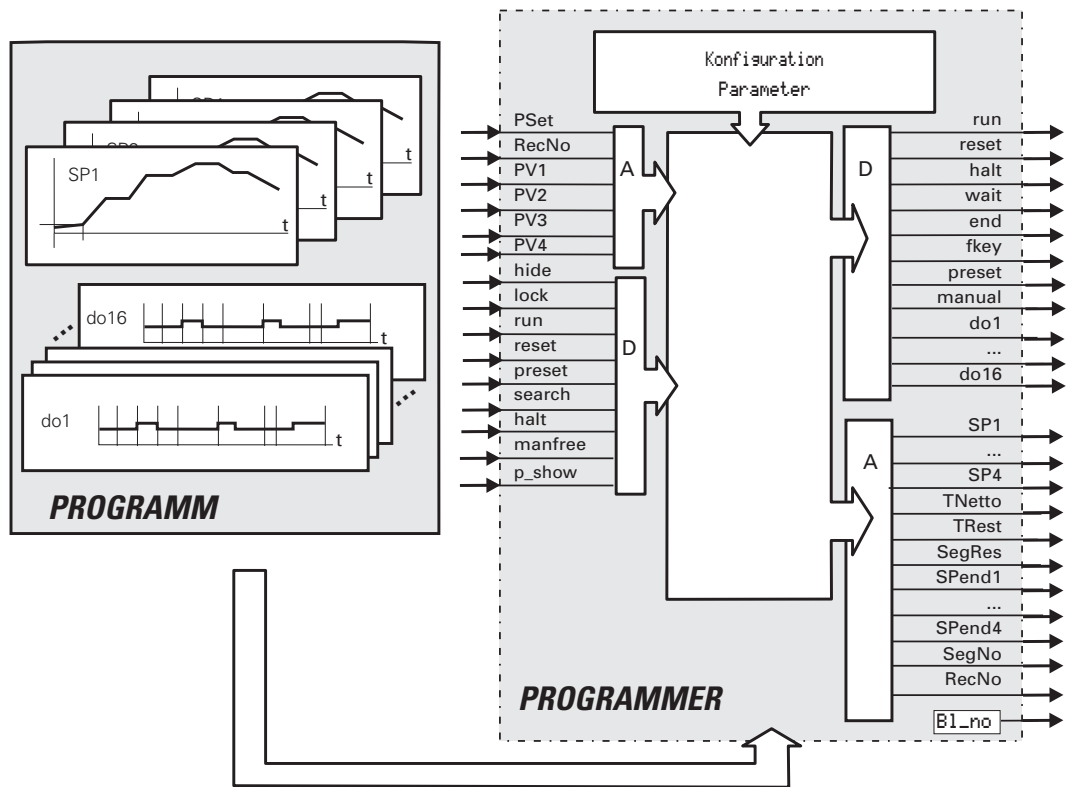


Abb. 672

If an error is detected when opening a recipe file, the reset value is output (status display on operating page: 'Error'). After an engineering download, the **Reset** status is active. If input **run** isn't soft-wired, **stop** is used.

Master and slave tracks

The first analog track is always the master track. All other tracks are slave tracks.

| In-/Outputs PROGRAMMER | | |
|-------------------------------|-------------|--|
| Name | Type | Description |
| PSet | Float | Preset value (time or segment, dependent on parameter PMode) |
| RecNo | Float | Required recipe number. The recipe number (setpoint) determines which recipe should be started next. Running recipes are not influenced. The selected recipe is activated only after the next reset. |
| PV1 | Float | Process value 1 for search run (master) |
| PV2 ... PV4 | Float | Process value 2 for search run |
| hide | Bool | Display suppression. hide = 1: the page is not displayed in the operation. |
| lock | Bool | Value adjustment blocking via operation. lock = 0: adjustment enabled, lock = 1: adjustment disabled. |
| run | Bool | Program stop/run. run = 0: stop, run = 1: run |
| reset | Bool | Continue/reset program. reset = 0: continue, reset = 1: reset |
| preset | Bool | Program preset, 1 = preset |
| search | Bool | Start programm search run, 1 = search run (operates on all analog tracks) |
| halt | Bool | Program interruption (e.g. due to an exceeded bandwidth detected outside the programmer). Output "run" remains active. halt = 0: program is not halted. halt = 1: program is halted. |
| manfree | Bool | Disable manual mode. manfree = 0: switch-over to manual mode is not permitted, manfree = 1: switch-over to manual mode is permitted. |
| p_show | Bool | Enable program editing. Display and adjustment of all segment parameters pertaining to a recipe. Calling up this page is done from the main operating page. |

| Name | Type | Description |
|----------------------|-------------|--|
| SP1 | Float | Programmer setpoint 1 (master) |
| SP2 ... SP4 | Float | Programmer setpoint 2 |
| TNetto | Float | Net program time. Elapsed program time without halt/stop times. (master) |
| TRest | Float | Remaining programmer time (master) |
| SegRes | Float | Remaining segment time (master) |
| SPend1 | Float | End value of current segment of analog track 1 (master) |
| SPend2 ... SPend4 | Float | End value of current segment of analog track 2 |
| SegNo | Float | Current segment number (master) |
| RecNo | Float | Current recipe number |
| Bl_no | Float | Own block number (e.g. for calling the operating page via CALLPG) |
| run | Bool | Program stop/run status; 0 = program stop, 1 = program run |
| reset | Bool | Program reset status; 1 = program reset |
| wait | Bool | Program wait status; 1 = Wait (one analog track has stopped the programmer at the end of a segment, because of a segment type with wait at the end). |

| | | |
|-----------------|------|---|
| halt | Bool | Program halt status (master); 1 = Program interruption (because of an external halt, an exceeding of the bandwidth or a wait on the other segment coupled slaves, which are still running) |
| end | Bool | Program end status (master); 1 = program end reached |
| fkey | Bool | F key status. Pressing the key causes a pulse. |
| preset | Bool | Single preset command: a pulse is output for the duration of one cycle (dependent on the programmer cycle time). Continuous preset command: the output is always active. preset = 0: no preset status, preset = 1: PROGRAMMER stands in preset status. |
| manual | Bool | Indication of manual mode, all tracks; manual = 0: none of the tracks of PROGRAMMER works in automatic mode, manual = 1: at least one track of PROGRAMMER works in manual mode. |
| do1 ... do16 | Bool | Status bit 1 |

| Parameter | | | | | | | |
|-----------|----------------|----------|---|--------|---------|-------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| PMode | Preset mode | Enum | Preset mode for input PSet or interface: Preset to segment or to time. | r/w | 1 | | |
| | | Segment | Preset to segment. Value at PSet input is used as target segment during a preset command (activated via preset input). | | 0 | | |
| | | Time | Preset to time. Value at PSet input is used as target program time during a preset command (activated via preset input). | | 1 | | |
| SMode | Search mode | Enum | Behaviour when activating search via search input or at program start | r/w | 0 | | |
| | | Segment | Search run in the segment. When starting the search run, setpoint SP is set to the value of input PV and approach to the segment end value is with the current gradient (TPrio = Grad.Prio) or in the current remaining segment time (TPrio). | | 0 | | |
| | | Programm | Search run in the program or program section. Search only in segments the gradients of which have the same sign. (Hold segment is neutral). The search cycle time may be longer than one processing cycle. | | 1 | | |
| | | Off | Search run switched off | | 2 | | |
| TPrio | Start priority | Enum | Start mode in search run determines the higher priority of gradient or | r/w | 0 | | |

| | | | | | | | |
|-----------------------|-----------------|----------|--|-----|--------|---------|--|
| | | | segment/time. | | | | |
| | | Gradient | Start mode in search run: Gradient has priority. When starting the search run, setpoint SP is set to the value of input PV and goes to the segment end value with the current gradient. | | 0 | | |
| | | Time | Start mode in search run: Time has priority. When starting the search run, setpoint SP is set to the value of input PV and approach to the segment end value is in the current remaining segment time. | | 1 | | |
| Dp1 ... Dp4 | Decimals 1 | Int | Number of digits behind the decimal point for display of the setpoint 1 | r/w | 3 | 0 ... 3 | |
| SPl01 ... SPl04 | Min. setpoint 1 | Float | Lower limit of set-point 1 | r/w | 0.0 | | |
| SPh1 ... SPh4 | Max.setpoint 1 | Float | Upper limit of set-point 1 | r/w | 1000.0 | | |

| Configuration | | | | | | | |
|---------------|---------------------|------------------|---|--------|---------|----------|-----|
| ID | Name | Type | Description | Access | Default | Range | Off |
| ATracks | Analog track number | Int | Number of the used analog tracks | r/w | 1 | 1 ... 4 | |
| DTracks | Contr.output number | Int | Number of the used control outputs | r/w | 6 | 0 ... 16 | |
| PwrUp | On mains recovery | Enum | Behaviour after mains recovery | r/w | 0 | | |
| | | Continue | Continue program at the point it was stopped by power failure. | | 0 | | |
| | | Search | Search run after mains recovery: forward and backward search from the point of failure, search function depends on parameter SMode. | | 1 | | |
| | | Continue at time | Continue at actual time. Search run after mains recovery: forward and backward search from the program time in which the program would be without | | 2 | | |

| | | | | | | | |
|-------------------------------|--------------------|------------------|--|-----|---|--|--|
| | | | power failure, search function depends on parameter SMode. Hint: If the program time from power failure to power recovery includes min. one segment with wait status at the end, the search run in the segment in which the program would be without power failure is omitted and the program stops at the first wait status without search run. | | | | |
| PEnd | On Program end | Enum | Behaviour at program end | r/w | 0 | | |
| | | Stop | After program end: stop (the setpoint of the last segment remains unchanged). | | 0 | | |
| | | Reset | After program end: reset (the start setpoint becomes active. The program restarts automatically, when the run condition has remained unchanged.) | | 1 | | |
| | | Reset + Stop | After program end: reset+stop (continuous reset, start setpoint with reset und stop). | | 2 | | |
| Coupling2 ... Coupling4 | Mast/slave coupl.2 | Enum | Coupling between slave track 2 and master track, slave 2 is coupled with time or segment of the master track. | r/w | 0 | | |
| | | Time coupling | Slave works on the same program time like the master (segments can be different). | | 0 | | |
| | | Segment coupling | Slave works every time in the same segment like the master (program time can be different). | | 1 | | |
| SpScale2 ... SpScale4 | Setpoint scaling 2 | Enum | Define scaling for the graph on the main page for track 2 independent from master or take over master scaling. | r/w | 0 | | |
| | | Own scaling | Own graphic scaling: SpLo ... SpHi | | 0 | | |
| | | Master scaling | Graphic scaling from master: SpLo1 ... SpHi1 | | 1 | | |

| | | | | | | | |
|----------------------------|---------------------------|--------|--|-----|---------------------|----------------------|--|
| Color1 ... Color4 | Setpoint color 1 | Enum | Color of graph on main page | r/w | 0 | | |
| | | Red | Red | | 0 | | |
| | | Blue | Blue | | 1 | | |
| | | Green | Green | | 2 | | |
| | | Yellow | Yellow | | 3 | | |
| TChart | Visible time [min] | Float | Visible time for graph on main page | r/w | 60.0 | 1.0 ... 1800000.0 | |
| A1Title ... A4Title | Name analog track 1 | Text | Name for analog track 1 | r/w | Analog track 1 | | |
| Unit1 ... Unit4 | Unit 1 | Text | Unit of setpoint 1 | r/w | Unit | | |
| D1Title ... D16Title | Name contr. output1 | Text | Name for control output 1 | r/w | Control output 1 | | |
| RecDir | Recipe directory | Text | Name of the directory, in which the recipes of the function block are stored | r/w | PROGRAMME R | | |

Segment types

| Name | Type | Description | Access | Default | Range |
|-----------------|----------------|---|--------|---------|-------|
| Segment type | Enum | Setpoint behaviour in the segment. The setpoint can be held or changed with a ramp or a step change. Continuation is automatic or manual ("Wait for operation"; configurable). | r/w | 8 | |
| | Time | Time segment: The setpoint changes linearly in the segment duration from the start value to the target setpoint of the relevant segment. The gradient is a function of this change (start value = end value of the previous segment) | | 0 | |
| | Rate | Rate to setpoint: The setpoint changes linearly with the adjusted gradient from the start value to the target value of the relevant segment. The segment duration is a function of this change (start value = end value of the previous segment). | | 1 | |
| | Dwell | Dwell: The end setpoint of the previous segment is kept constant for the time. | | 2 | |
| | Step | Step to setpoint: The setpoint goes to the adjusted target setpoint immediately. | | 3 | |
| | Time + wait | Time to setpoint and wait. The setpoint changes linearly from the start value to the target setpoint in the duration of the segment. The programmer goes to stop condition at the end of segment. | | 4 | |

| | | | |
|--------------|--|---|--|
| Rate + wait | Rate to setpoint and wait: The setpoint changes linearly from the start value to the target setpoint with the adjusted gradient. The segment duration is a function of this change (start value = end value of the previous segment). At the segment end, the programmer goes to the stop condition. | 5 | |
| Dwell + wait | Dwell and wait: Dwell segment: The end setpoint of the previous segment is kept constant during the segment time. At the end of the segment, the programmer goes to the stop condition. | 6 | |
| Step + wait | Step and wait: Step segment: The programmer goes to the adjusted target setpoint immediately and waits at the segment end. | 7 | |
| End | The last segment in a program is the end segment. When reaching the end segment, the setpoint output last is held. | 8 | |

Additional parameters that are determining for a program step:

| Name | Type | Description | Access | Default | Range |
|-----------------------------------|-------|--|--------|---------|-----------------|
| (Segment) time / gradient TpGr | Float | Time or gradient for the segment. The duration of a segment can be determined directly, or as a gradient and a setpoint difference SP - segment start setpoint. Whether segment time or gradient are concerned is determined in parameter Segment type (Type). | r/w | 0.0 | 0.0 ... 1800000 |

| Name | Type | Description | Access | Default | Range |
|----------------|-------|--|--------|---------|-------|
| Setpoint SP | Float | End value for the segment. Target value at the end of the first segment. Approach to this value is from the last valid setpoint (from the process value at the beginning of the 1 st segment). After elapse of the program, the controller continues controlling using the target setpoint adjusted last. | r/w | 0.0 | |

Synchronization

The analog slave tracks can be selected either as coupling by segment or by time to the master. A **time coupled** slave follows the master time, also if the master stops or changes its time because of search or preset. So it runs independently from the other slaves to its program end, where it is waiting until all tracks reach their end. After that the programmer executes its end function, which depends on the PEnd configuration.

If **segment coupling** is selected all tracks wait ('halt' state) for each other at the end of each segment, until all the tracks have ended this segment. This can happen, if the tracks have different segment times, if a search or a bandwidth violation has occurred or if a track is in a wait state at segment end.

All digital tracks are coupled by segment to the master. The segment times of these tracks are taken from the master, they do not have separate segment times.

Analog tracks



HINT!

The first analog track is always the master track. All other tracks are slave tracks.

Start setpoint

The start setpoint is part of the recipe: setpoint of the 1st segment (segment with segment number 0). Start setpoint is active, if PROGRAMMER is in status **reset**. If no recipe is selected or the selected recipe is not available, e.g. the recipe does not exist or is not correct, or it does not match the configuration, the effective setpoint is set 0.

Operation preparation and end position

Each program starts at an initial position. This position is used and maintained with reset or first programmer set-up.

With program start from rest position, the first programmer segment runs. The program starts from the instantaneous process value at the time of start command, if the corresponding process value was soft-wired at PV of the PROGRAMMER and **search run** was configured.

With step change mode, the setpoint of the first segment is activated immediately.

At program end, depending on configuration (PEND), either

- 0=Stop: the setpoint of the last segment is maintained

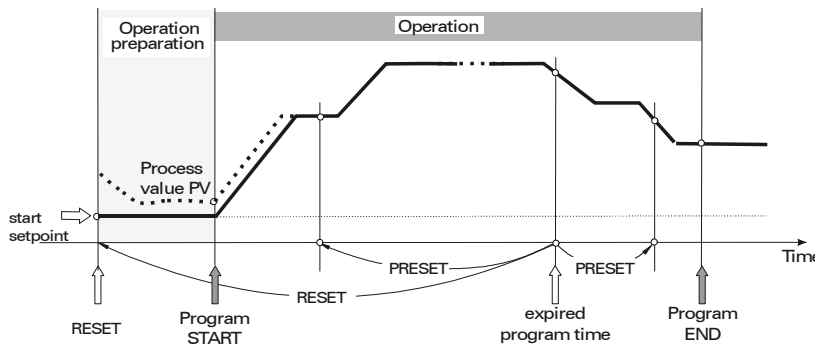


Abb. 673 Profile, with the end position kept unchanged

- 1 = Reset: the programmer goes to rest position **SP** of the first segment (segment 0). The program restarts automatically, if run is still valid.

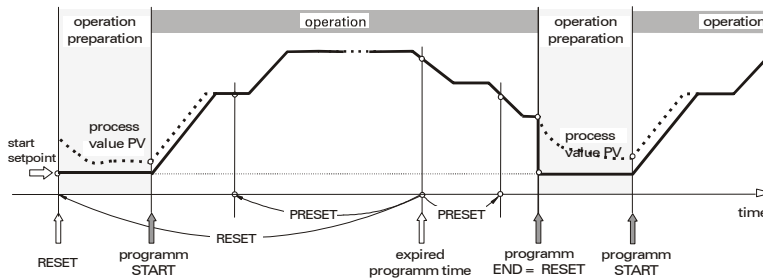


Abb. 674

- 2 = Reset + Stop: Programmer goes to rest condition **SP** of the first segment (segment 0) with reset and stops there.

At program end, the number of the last segment increased by 1 is output as active segment number (**SegNo** output). This is required to bring the slave block to the end status safely with a segment preset

Search Run

The term *search run* describes the behaviour of the programmer to reach the end value starting from the current process value. A search run can be performed in the current program segment or over several program segments.

A search run is used in the following cases:

- After program start: via the operator interface or via the **run** input.
- After a program restart using the **reset** command on the programmer operating page, or via the **reset** input.
- Via the **search** input.

Configuring the Programmer

In the *BlueDesign* parameter dialogue, you can determine, if a programmer can perform search runs and which type of search can be performed.

For this purpose, the *SMode* parameter offers the following options:

- Segment: search runs are made in the program segment.
- Program: search runs are made in the program section. A program section is defined as a sequence of program sections with gradient of the same direction (only upwards or only downwards). Change in direction limits a program section.
- Off: No search runs are made.

| Search mode | |
|-------------|----------|
| 0: | Segment |
| 1: | Programm |
| 2: | Off |

Abb. 675: Parameter "SMode"

Search Run in the Program Segment

With a search run, the process value (*PV*) is set as programmer set-point first. Then the process value is controlled to the segment end value either using the current gradient or using the current remaining segment time (depending on the programmer configuration).

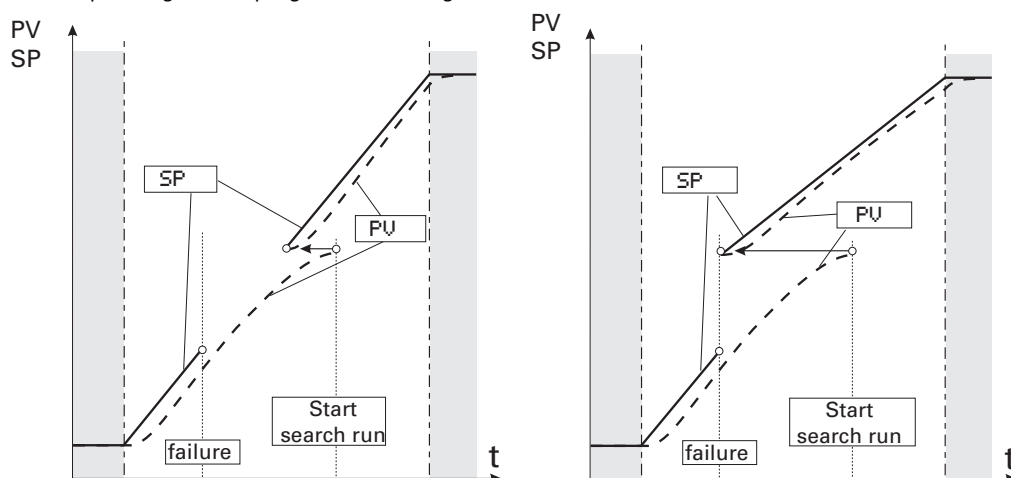


Abb. 676: Search run with the current gradient/the current remaining segment time

Configuring the programmer is done using the *TPrio* parameter in the *BlueDesign* parameter dialogue. When making a search run, the following information must be taken into account:

- If setting "TPrio = Gradient" is used for the search run and the search value is out of the current segment, the program is continued at the segment point next to the search value.
- If the start value is equal to the end value (a segment without gradient) the program is continued at the segment start.
- With a jump segment, start is always at the beginning of the segment. The target set-point is assigned to the process value *PV*.

Search Run in the Program Section

A search run in the program section is limited to a section of several segments that have the same sign as the gradient, whereby hold segments are not considered as changes of sign. If the current program section contains hold segments, a search run is performed only, if this program section contains at least one more program segment which isn't a hold segment. If this segment is directly preceded or followed by another hold segment, the search run is realized only in the current segment.

Depending on the number of segments included in the search run, the run-time may be very long. For this reason, searching is divided into several partial runs. Each partial run checks only one segment. This is an automatic procedure which needs no interaction from operator.

i *NOTE!*
 If the value of parameter "TPrio" is "1: time", the search run is always limited to the current program segment.
 Program segments with a final waiting state (e.g. "time + wait") do not limit the search range (exception: if a search run after a power failure is concerned).
 A search run can lead to termination of the program.

Particularities of a search run with the PROGRAMMER

Dependent on master/slave coupling, a search run can have various effects:

| Master/slave coupling | Search run function | Search run of a single track | Simultaneous search run of all tracks |
|-----------------------|-----------------------|---|--|
| Time coupling | Search run in segment | Search run is permitted only for the master. As all slave tracks follow the master track time, no search run occurs with the slave tracks. | |
| | Search run in program | | |
| Segment coupling | Search run in segment | A search run is performed for the track. At the segment end, all tracks with segment coupling wait for each other before starting the next segment. (*) | |
| | | | A search run is performed for all tracks with segment coupling. At the segment end, they wait for each other before starting the next segment. (*) |

| | |
|-----------------------|---|
| Search run in program | <p>Permitted only for the master track.</p> <p>The master track performs a search run. All tracks with segment coupling follow the master into the new segment and perform a search run in this segment. If this segment search run isn't successful, the track continues at the segment start or end value, dependent on the setpoint (start or end) which is closer to its process value. If the process value (input PV) of a slave track with coupled segments is not soft-wired, the slave track continues at the value of the elapsed time of the master track in the segment.</p> <p>At the segment end, all tracks with segment coupling wait for each other before starting the next segment (*).</p> |
|-----------------------|---|

(*) With segment coupling of at least one analog track, the master track operates according to the same principle.

Dependent on parameter **SMode** (no search run / search run in the segment / search run in the program), the **search** input acts on all analog tracks simultaneously. Via the operating pages or the interface, a search run in the segment or in the program can be started independently of **SMode**. The search run for a single track can be started in the same way.

During a step or a dwell segment, the search run is without effect on this track.

If a slave track with segment coupling following the master with a search run in the program arrives at a dwell or step segment, the slave track continues at the value of the elapsed time of the master track in the segment and it waits there for the other tracks, before the next segment is started in common.

Preset

A preset is used to change to a defined time in the program.

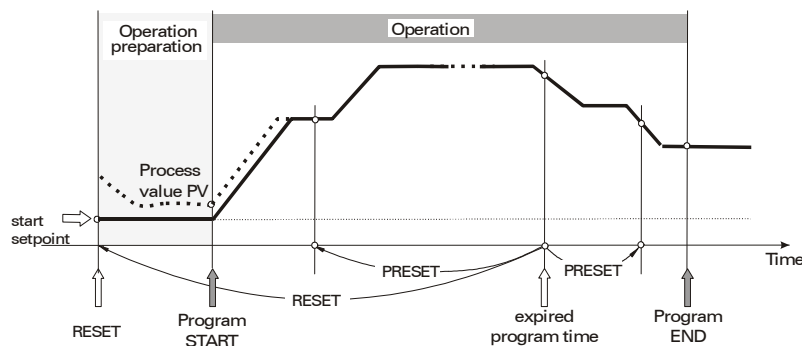


Abb. 677

For a detailed description how to call up the preset via operation, see operating page "Review profiles".

- A slave track with time coupling follows the master track.
- A slave track with segment coupling changes into the new segment of the master track and follows the time of the master track within this segment.

Bandwidth monitoring

To prevent the separation between setpoint and process value from becoming too large, bandwidth monitoring is used. This function checks, if the control deviation is within the permissible limits and stops the programmer ("Halt status"), as soon as the separation becomes too big. As soon as the control deviation has decreased sufficiently, the bandwidth monitoring terminates the **halt** status and the program is continued.

If the separation between setpoint **SP** and process value **PV** exceeds the bandwidth (**BW**) parameter, **halt** is activated:

$$HALT \leftarrow (|SP - PV| < bandwidth)$$

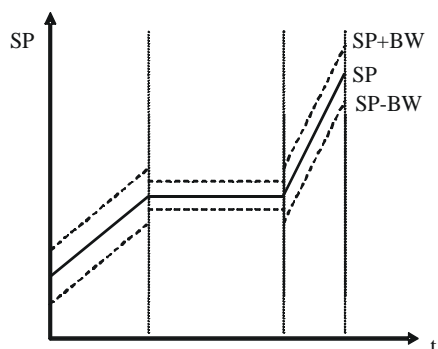


Abb. 678 Bandwidth around the setpoint SP

- Each analog track has its own process value (PV) input and its own bandwidth.
- There is one bandwidth for each segment. Value **BW** determines the separation between process value **PV** and setpoint **SP** without triggering the **halt** status.
- If the bandwidth is exceeded, a track with time coupling stops, until **PV** is within the permissible bandwidth again. During this time, all tracks with time coupling including the master track are stopped. A track with segment coupling shows the same bandwidth behaviour, but the other tracks continue up to their segment end. At the segment end, these tracks including the master track wait, until all tracks with coupled segments have reached the segment end. While the bandwidth is exceeded, **halt** is signalled at the **halt** output and on the corresponding operating pages (main page and detailed page).
- Bandwidth monitoring can be switched off by setting the function to "OFF".

Control tracks (digital tracks)

Start values

The start values of the control tracks are included in the recipe: the values of the first segment (segment no. 0). The start values are active, when the **PROGRAMMER** is in the **reset** status. If no recipe is selected, or if the selected recipe is not available, e.g. because the recipe doesn't exist or isn't o.k., or because it doesn't suit the configuration, all control tracks are set to 0.

Segment time

The control tracks are coupled to the master track via the segment numbers. To reduce the division into segments, the two parameters **t+** and **t-** can be used:

- Delay **t-**: delay after which the control track reaches its adjusted value. Up to this time, the control track remains on the inverted value.
- Switch-on time **t+**: The control track remains on its adjusted value, until switch-on time **t+** has elapsed. If **t+** is set to "0", the switch-on time ends at the segment end.

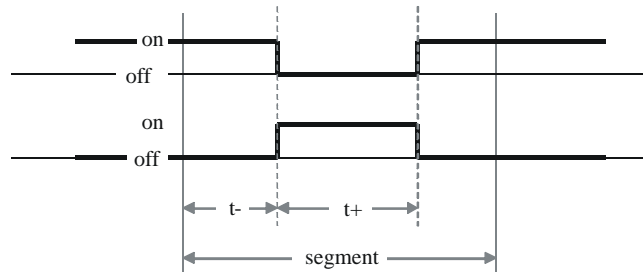


Abb. 679 Switch-on and off times within a segment

Recipes

The recipe management, i.e. creating, deleting and editing, is performed using the BlueEdit program editor. A program can be changed also using the PROGRAMMER editing page.

Fill in the basic data for the recipes in BlueDesign: Recipe directory (each PROGRAMMER must have an own directory), number of required analog tracks, number of required digital tracks.



CAUTION!

When using several PROGRAMMER function blocks, the recipes can be changed alternately.

As the PROGRAMMER function blocks start with the same recipe directory when edited in BlueDesign (default setting), they can access the same recipes.

For this reason:

- A unique recipe directory name must be used for each PROGRAMMER function block.

For creating the recipes, the engineering must be read in using BlueEdit. Subsequently, suitable recipes can be created and edited. These are uploaded into the instrument as required. Recipes in the instrument can be downloaded, edited, stored and reloaded.

Each PROGRAMMER uses its own directory specified in the **RecDir** configuration. The recipe file names have a defined structure of recipe number (001 ... 999) and recipe name: "RecNo_RecName.prf". During the start-up phase or after downloading the engineering, the PROGRAMMER generates its recipe directory, if it still doesn't exist.



NOTE!

A recipe directory that isn't required any more is not deleted automatically. It must be deleted using an external program (e.g. FTP tool), for instance the program editor.

Temporary recipe change

While PROGRAMMER is running, the user can change the active recipe temporarily, i.e. until the next **reset**.



CAUTION!

Changes of the active program made on the editing page are effective IMMEDIATELY.

This may cause unpredictable and dangerous situations in the process.

For this reason:

- Consider the effects before making changes in the current program and take measures to prevent potentially hazardous situations.

The temporary change is made on the PROGRAMMER editing page and changes will become active only during the future execution of the recipe (see editing page).

**NOTE!**

When quitting the editing page, a memory dialogue for permanent storage of recipe changes opens. If only a temporary change is required, "No" (= don't save changes permanently) must be selected.

Recipe change – program selection

The required recipe can be selected externally via the analog **RecNo** input, or internally using the recipe number adjusted via operation/interface.

**NOTE!**

During an active program sequence, switch-over to a different recipe on the programmer operating page is not possible. Recipe changing is possible only during the reset status.

Halt status

1. For example, used for external bandwidth monitoring

The **halt** status can be switched on and off using control input **halt**. Unlike the **stop** status, the **run** status remains unchanged during the **halt** status (the **run** output remains active). Status display is **halt**.

2. Internal bandwidth monitoring

The **halt** status can be triggered by internal bandwidth monitoring (see section Bandwidth monitoring). Status **halt** is displayed.

3. Synchronization of tracks with coupled segments

The **halt** status can be triggered when a track must wait for another track, because the tracks are coupled via segment coupling to the master (see section Synchronization). Status display is **halt**.

III-22.3 BlueEdit: recipes



NOTE

For information on recipe creation and handling, refer to the BlueEdit manual.

III-22.4 KS108 easy : Starting the programmer operation

III-22.4.1 Loading the engineering

In principle, you can address various target devices with BlueDesign. However, it is only possible to work with one device at a time. To address the device, BlueDesign needs information where to find the device .



NOTE

Detailed information on how to build up the communication is given in Chapter II Development environment "Logging on to the target system" in the KS108 easy manual.

Open the required engineering created in BlueDesign. To load it into *KS108 easy* log on to the target system from the Run menu and establish the connection.

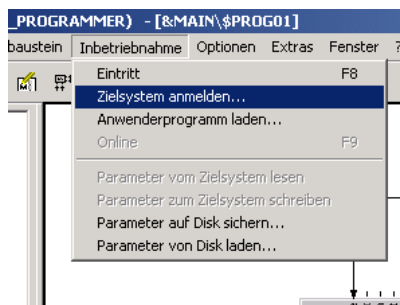


Abb. 680: Connecting to the target system

Status indication

The status of connection to the device is displayed by symbols (the simulator display is shown as an example below):

| Symbol | Meaning |
|-----------------------|--|
| Simulation KS108 easy | The connection is active.. |
| Simulation KS108 easy | The connection was configured, but it is not used. konfiguriert. |
| Simulation KS108 easy | The connection isn't active. |

Loading the user program



DANGER!

This symbol is a warning against injury or against a hazard of material damage by unpredictable functions and movements in the process.

When using *KS 108 easy* in connection with other instruments/facilities, the program transmission may cause a risk of reactions of these instruments/actuators, etc.

For this reason:

- Before connecting, consider the effects of a program update and take suitable protective measures.
- Prior to establishing the connection, make sure that the right program is loaded and
- ensure that the program is free of faults.

1. Open menu "Download ...":

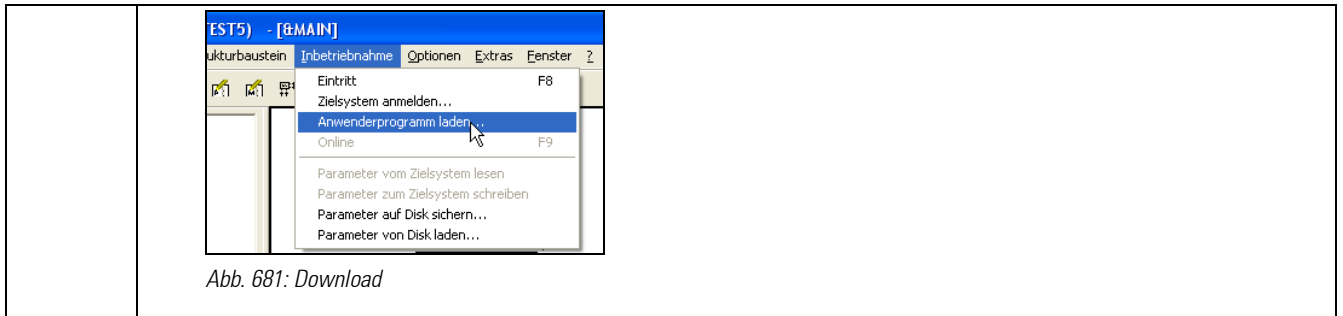


Abb. 681: Download

III-22.4.2 Loading a recipe

In principle, you can address various devices with the *BlueEdit* program editor. To address the *KS108 easy*, you must provide *BlueEdit* with the information where to find the device.



NOTE!

Detailed information on how to establish the connection is given in the *BlueEdit* operating manual.

Open the required recipe created in *BlueEdit*. To load it into *KS108 easy* establish the connection.



DANGER!

This symbol is a warning against injury or against a hazard of material damage due to unpredictable functions and movements in the process.

When using *KS 108 easy* in connection with other instruments/facilities, the program transfer may cause a risk of reactions of these instruments/actuators, etc.

For this reason:

- Before connecting, consider the effects of a program update and take suitable protective measures.
- Prior to establishing the connection, make definitely sure that the right program is loaded and
- ensure that the program is free of faults.

1. Open menu "Download ..."

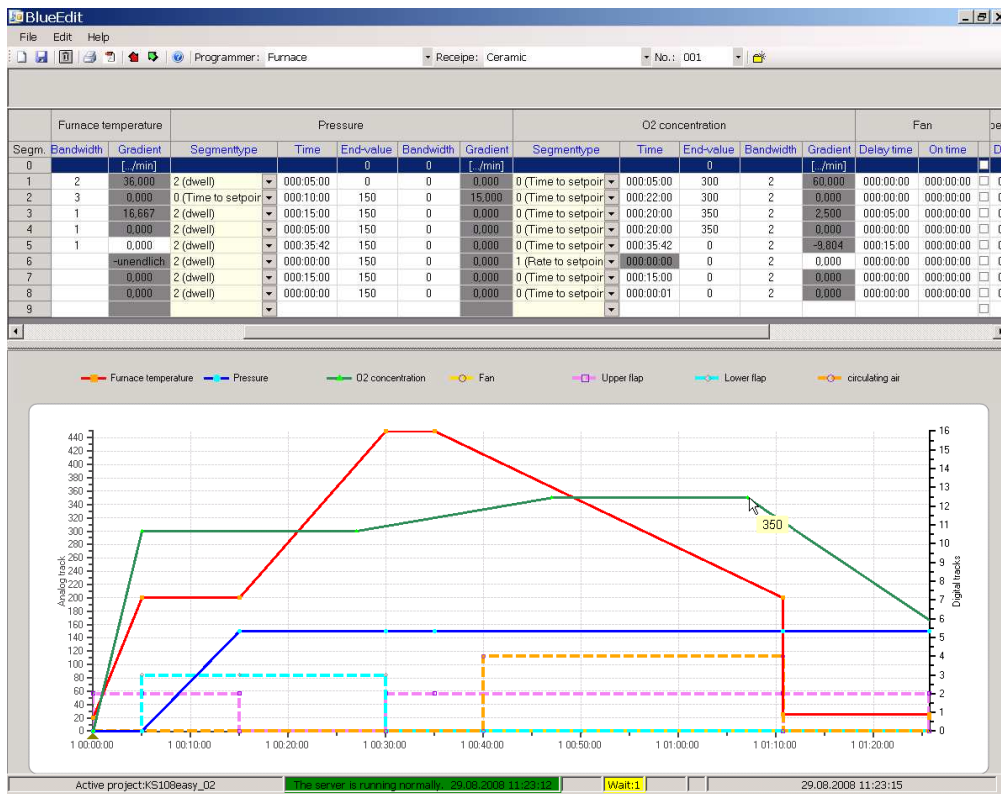


Fig. 682: Loading a recipe

III-22.4.3 Starting the programmer

When the engineering and the related recipes were downloaded into KS108 easy or into the BlueSimulation, the programmer can be started.

If the engineering and the recipes are a match and if they use the same recipe directory for the various programmers, the first recipe is downloaded and can be started.

As the operation can be influenced extensively by the engineering, the operating principles are described in the following section.

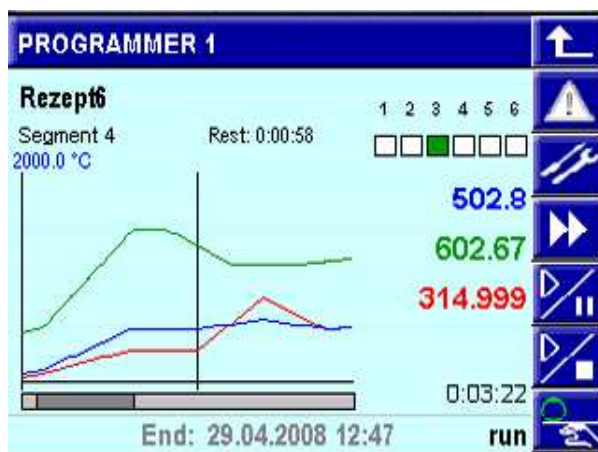





Abb. 683: Loading a recipe

Programmer control via keys and

The programmer can be controlled via the following elements:

- the digital function block inputs,
- the interface, or
- using control keys  and .

The  key offers a choice of the following functions:

- **reset**: Selectable from any program position
- **preset**: From any program position. **preset** is used to select the review page (**preset**) to change to a different time in the program
- **segment search** : Start a search run in the segment.
- **program search** : Start a search run in the program.

The  key controls the programmer (fkey output generates a pulse when a key is activated).

For both keys, the rule saying that the states at soft-wired control inputs must be given priority over the operation. The following diagram describes the sequence of states dependent of the actions:

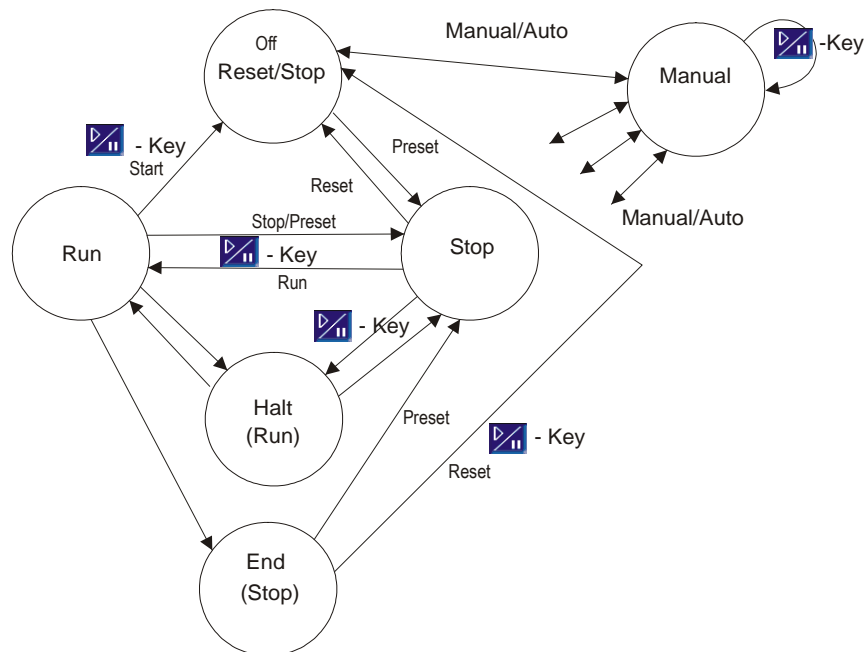


Fig. 684: Interaction between statuses of program and possible switch overs

Programmer control: General



- Recipes may be selected during the reset status.
- The process value is only shown on the detailed pages, if the process value input is soft-wired.
- The setpoint can be adjusted during manual operation (status "man").
- The 3 status displays are (depending on operating status):
 - Status left: man
 - Status middle: Estimated program end time
 - Status right: stop / run / reset / search / error / halt



NOTE!

If the inputs (function block inputs) shown in the following table are used by the engineering, this signal cannot be changed on the operating page (front-panel operation).

This concerns inputs **run**, **reset**, **preset** and **search** (as shown in the following table):

| Input fields | | Operation | Display | FB input |
|--|-----------------------|--|--|-----------------------|
| Recipe name on main page | | Selecting the required recipes is not possible, if the RecNo input is soft-wired. | Indicates the current recipe name. | RecNo |
|  (Selection key) | reset | Switching the programmer to segment 0. | The programmer is switched to segment 0 | reset |
| | preset | Select Preset via the review page | Preset page is activated | preset, Preset |
| | segment search | Starting a search run in the segment | Programmer makes a search run in the segment | search |
| | program search | Starting a search run in the program | Programmer makes a search run in the program | search |
|  key | | Program control | Changes of the status display (bottom right) | run / reset |

Programmer operation

Operating structure

The **PROGRAMMER** has several operating pages selectable on a main page in the menu of operating pages. If the **hide** input is soft-wired, no **PROGRAMMER** operating page is displayed.

As shown in the following diagram, there is a main page for the programmer on which a branch to the detailed pages of the tracks can be made by tapping the relevant key. Changing to the parameter page is possible from the detailed pages of the tracks. The editing page is selectable only, when digital input **p_show** is soft-wired and set.

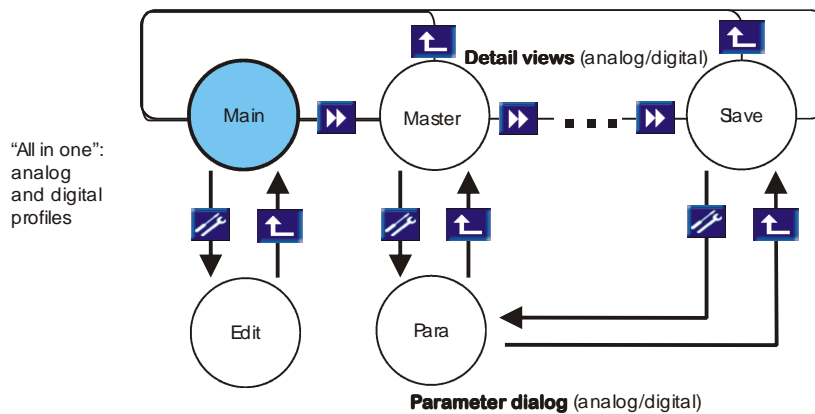
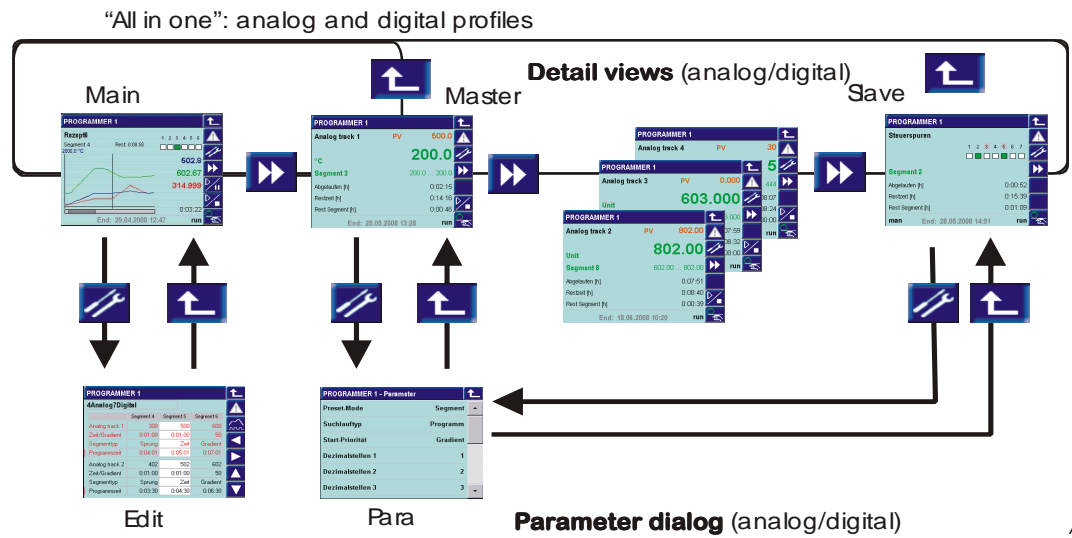


Abb. 685 Operating structure



686 Operating structure

Abb.

Main operating page

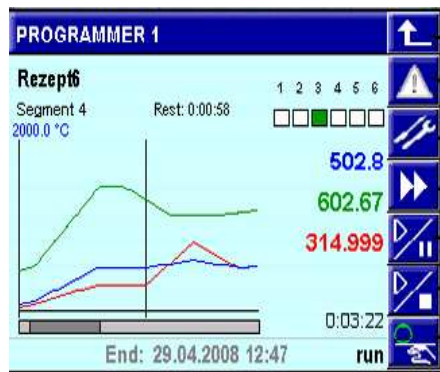


Abb. 687

Keys on the right section of the operating page:

- Key (line) 1: return
- Key 2: Access to the alarm page
- Key 3: Access to the parameter page
- Key 4: Change to the detail page
- Key 5: Programmer control
- Key 6: Programmer control selection
- Key 7: Auto / manual switch-over

Display on left section of main page:

0: Programmer name

1: Active recipe

2: Active segment

3: Remaining time of active segment

4: Current status of control tracks 1 to 6 (status)**5 - 8: Current setpoint of analog track (up to 4)**

9: Elapsed program time (since start)

10: Maximum setpoint (SPhi1) and unit (Unit1) of master

11: Time stamp in the program curve

12: Setpoint curve in the program

13: Bargraph of program time (bar length to overall length as displayed time section to overall program time)

14: Display of manual operation "man"

15: Estimated program end time

16: Program status: run / stop / reset / search / halt / end / error

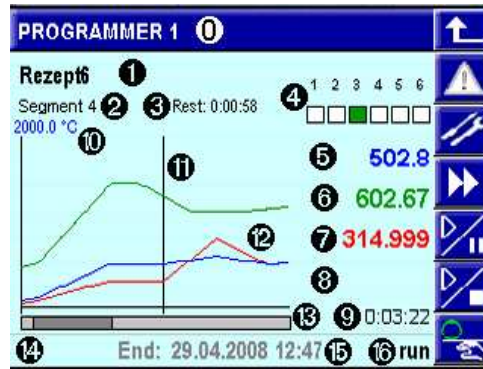


Abb. 688

The items marked in bold letters may include controls with variable values

Display on the main page

- Up to 4 setpoints with their curves are displayed; a vertical line marks the current time in the program. The visible time range is defined in configuration **TChart**. It is always updated, and the current position with the vertical time stamp remains always in the medium range.
- The Y-axis is determined by scaling of the master track with its maximum and minimum setpoint (**SPhi1**, **SPhi1**). Depending on configuration, the slave tracks are scaled like the master track, or provided with their own scaling. The scaling of individual slave tracks is not shown on the display.
- The setpoints belonging to the current time marked on the trend curve are displayed in the same colour as the setpoint curves on the right. The current program time is shown as a value below the current setpoints.
- The bar of the bargraph indicates the visible time range in relation to the overall program time.
- Items shown with a frame can be changed by tipping with your finger:
 - Recipe (only in status `reset`)
 - Setpoints (only during manual mode `man`)
 - Control tracks (only in manual mode `man`)
- An overview of the status of the first 6 control tracks is given by the "LEDs" (on / off, "LED" = Square at beginning of the line). Details and further control tracks are found on a list which can be called up by tipping on the LED field with your finger. All control tracks are listed with status and name.

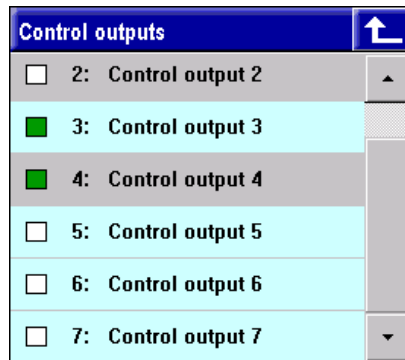



Abb. 689

Control track operation:

1. Symbol: white = off, green = on
2. Number of control track (1 ... max. 16)
3. Name of control track. The individual control tracks can be switched over: gray = manual mode, blue = automatic mode, if at least 1 control track is in manual mode.

- Tipping on a setpoint opens a window with display of track name, setpoint and unit.
- Switch-over to manual mode from the main page, marked by display "man" on the status line, puts all analog and digital tracks into manual mode. The variable items are marked in gray colour: setpoints are shown with a gray frame, control tracks are shown with a gray background.
 - Tipping on a setpoint opens a numeric pad for changing the setpoint.
 - Tipping on the LED field opens the list of control tracks. Change the status (off/on) of an individual control track by tipping on the corresponding icon. Tip on the name of the control track to switch it over between automatic and manual operation (gray background).

In status line "man" is displayed as long as at least 1 track is in manual operation. Tipping on  with "man" displayed all tracks are switched to automatic operation.

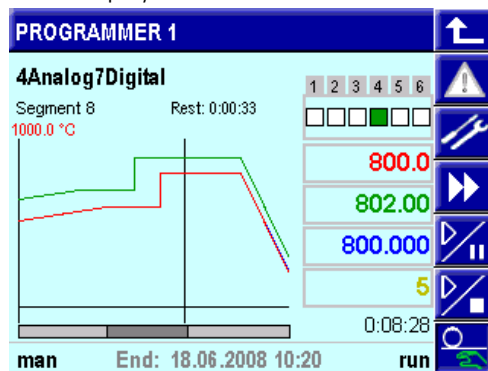






Abb. 690

Tapping  on the main page switches all tracks to manual or to automatic operation in common.

- The program can be set to a preset value. Preset to a segment or to a time in the program is possible. Preset can be activated via the profile overview page ( displays the list: "reset / preset / search", **preset** changes to the overview page). Tap the navigation keys to select the required time or segment. When leaving the page, the preset is or isn't activated ("OK" / "No"). If the preset is activated, the elapsed time is adapted and switch-over to the main operating page occurs (see operating page "Review profiles".)
- Program setting on the operating page: Direct access to the recipe parameter setting (= editing page) is enabled, when control input **p_show** = „1" is set at the programmer function block. Access is using the parameter key .
- The **run / stop / reset / preset** states are always valid for the overall PROGRAMMER, i.e. for all tracks. They can be switched over centrally on the main operating page, unless they are determined via control inputs.

Detailed pages of the analog tracks

When manual mode is activated on the operating page of a track (key ) , only this track is switched to the manual mode. All other tracks remain in the automatic mode.

**NOTE!**

On the status line of the main operating page, "man" is displayed, if at least 1 analog or digital track is in the manual mode.

The setpoint of a track can be adjusted only, if the track is in manual mode.

A search run can be started in the current segment. Additionally, a program search run can be started on the operating page of the master track (track 1).

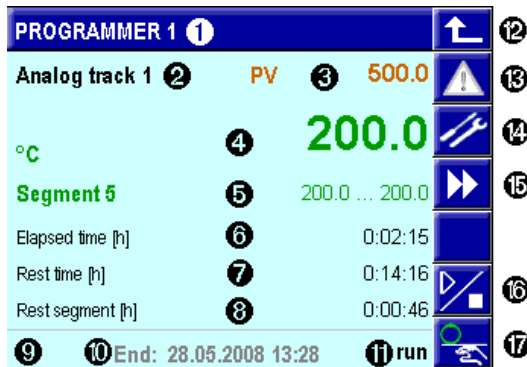


Abb. 691

- 1: PROGRAMMER name
- 2: Track name
- 3: Process value (if any) as a PV with value
- 4: Setpoint unit and **current setpoint**
- 5: Segment name/segment start and end value
- 6: Elapsed program time (master track)
- 7: Remaining program time (master track)
- 8: Remaining segment time (master track)
- 9: Display of manual mode: "man"
- 10: Estimated time for program end (master track)
- 11: Program status: stop, run, reset, search, error, halt (master track)
- 12: Return key
- 13: Key for opening the alarm page
- 14: Key for opening the parameter page
- 15: Track change key
- 16: Search run key (master track: selection dialogue)
- 17: automatic/manual switch-over key

Detailed pages of control tracks (digital tracks)



The screenshot shows a control interface with the following elements:

- 1:** PROGRAMMER name (PROGRAMMER 1)
- 2:** Control outputs (1-7) with status indicators (green for on, white for off)
- 3:** Segment name (Segment 5)
- 4:** Elapsed time [h] (0:00:52)
- 5:** Rest time [h] (0:15:39)
- 6:** Rest segment [h] (0:01:09)
- 7:** Status line (man, End: 28.05.2008 14:51, run)

Navigation keys are shown on the right side of the screen:

- 11:** Return key
- 12:** Key for opening the alarm page
- 13:** Key for opening the parameter page
- 14:** Key for track changing
- 15:** auto/manual switch-over key

Fig. 692

- The control tracks are displayed as LEDs.
- The name of the segment in which the master track works instead of the current segment number is displayed.
- The time estimated for program end is displayed on the status line.
- When tipping on the LED field, the control track list is opened. Each defined control track is displayed with number, name and a symbol for the status (off/on).
- Tipping on key , the **automatic/manual** key, also opens the list of control tracks. Tip on the name of the control track to switch it over between automatic and manual operation. On the list, the line is shown with a gray background during manual mode.
- When activating the manual mode on the operating page of control tracks () , only the control track selected on the list is switched to manual operation. All other tracks remain in the automatic mode.
- The status of a track can be changed only when it is in manual mode.
- Manual mode of the control tracks is shown by a gray field around the number of the control track. During manual operation, change the off/on status of the individual control track by tipping on the corresponding icon. Additionally, "man" is displayed on the status line, when at least 1 control track is in the manual mode.

Programmer parameters

The parameter page can be opened from the detailed pages of the tracks. On these pages, basic functions are listed, for example, the **search run type**, and can be changed. Some parameters are valid for the overall PROGRAMMER (e.g. **Preset-Mode**), others are valid for single tracks (e.g. setpoint ranges).



| PROGRAMMER 1 - Parameter | | ↑ |
|--------------------------|----------|---|
| Preset mode | Time | ▲ |
| Search mode | Segment | |
| Start priority | Gradient | |
| Decimals 1 | 1 | |
| Decimals 2 | 2 | |
| Decimals 3 | 3 | ▼ |

See table: Parameters

Abb. 693

Editing page

All recipe parameters (analog and digital tracks) are edited on a **single** scrollable page. When opening the page, the active recipe is displayed. Tap on the recipe name to switch over to other recipes.

The upper part always shows the master track in the colour configured with **Color1**. In the lower part, another track is displayed. For track paging, tap keys  and .

The following items can be changed:

- Recipe name (i.e. file selection)
- Setpoint/control track
- Time(s) or gradient
- Bandwidth



WARNING!

Changes on the editing page in the active recipe are effective immediately.

Changes in the program sequence may cause potentially hazardous conditions in the process.

For this reason:

- Consider the effects of the changes and take appropriate measures.

When leaving the editing page, a (memory) dialogue is opened. In this dialogue, the operator decides, if the changes are or aren't stored permanently. Changes in the current program sequence are taken into account in the further program sequence and deleted subsequently, unless they are saved.

Changes in another programm are not taken into account and are deleted, unless they are saved.

If the program is reset via the digital **reset** input or via the interface during editing on the editing page while the memory dialogue is still active, all changes are cancelled and the dialog is closed.

| | Segment 4 | Segment 5 | Segment 6 |
|----------------|-----------|-----------|-----------|
| Analog track 1 | 300 | 500 | 600 |
| Time/Gradient | 0:01:00 | 0:01:00 | 50 |
| Segment type | Step | Time | Rate |
| Program time | 0:04:01 | 0:05:01 | 0:07:01 |
| Analog track 2 | 402 | 502 | 602 |
| Time/Gradient | 0:01:00 | 0:01:00 | 50 |
| Segment type | Step | Time | Rate |
| Program time | 0:03:30 | 0:04:30 | 0:06:30 |

Abb. 694

For description of **segment types**:
see the Segment table.

Bandwidth:

If the separation between setpoint **SP** and process value **PV** exceeds the bandwidth parameter (**BW**), **halt** is activated (for description, the section Bandwidth monitoring).

- 1: PROGRAMMER name
- 2: Recipe (file name)
- 3 to 6: **Master track** (M=Master, always visible):
 - 3: Track name
 - 4: Depending on segment type: time/gradient
 - 5: Segment type (cannot be changed via operation)
 - 6: Change field, marked by a line. Switch-over by tipping:
 - a) Elapsed program time or
 - b) Bandwidth: Adjust the symmetric bandwidth per segment on the line.
- 7 to 10: one of the slave tracks (S=slave, 4 lines, paging with up/down keys), lines as 3 to 6:
 - 7: Track name
 - 8: depending on segment type: time/gradient
 - 9: Segment type (cannot be changed via operation)
 - 10: Program time/band width change field
- 11: Current segment (white background)
- 12: View and operation of 3 segments, scrolling by the up/down key
- 13: Return key
- 14: Key for opening the alarm page
- 15: Key for opening the overview page
- 16: Key for "Segment paging" right/left
- 17: Key for paging up/down slave tracks

Digital tracks on the editing page:



| | Segment 4 | Segment 5 | Segment 6 |
|------------------|-----------|-----------|-----------|
| Analog track 1 | 300 | 500 | 600 |
| Time/Gradient | 0:01:00 | 0:01:00 | 50 |
| Segment type | Step | Time | Rate |
| Bandwidth | OFF | 2 | 3 |
| Control output 1 | off | on | off |
| Delay t- | 0:00:00 | 0:00:00 | 0:00:00 |
| On time t+ | 0:00:00 | 0:00:00 | 0:00:00 |
| Program time | 0:04:01 | 0:05:01 | 0:07:01 |

Abb. 695

Same picture as above, but

- 1) **Bandwidth** on the master display
 - 2) **Control track** on the slave display.
- Switch-on delay t-** : delay after which the control track goes to its adjusted value.
- Switch-on time t+** : The control track remains on its adjusted value during switch-on time t+ .
- For a description of parameters, see section Segment time under Control tracks.
- on** = control track is on, **off** = control track is off.

Programmer page Review profiles

To provide a program overview, the "Review profiles" programmer page is opened. Tap  to select it on the editing page (tap  to open the editing page in the main menu). On this page, you can check the

program, for example, after creation or after changes. The page shows the curves of the analog tracks with setpoints in a graph. Tap the 4 arrow keys to scroll within the segments (◀▶) or per segment (◀▶) throughout the program time. Apart from the program time as a reference, the segments with segment names, the analog tracks with the setpoints and the first 6 digital tracks with states are displayed.

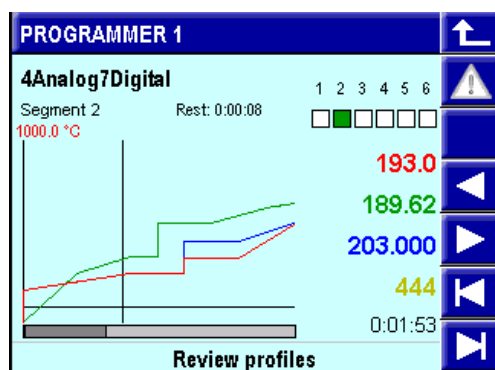





Abb. 696

The overview page is also used to make a preset for jumping to a defined program point. When tapping key  to select **preset** on the main page of the PROGRAMMER, the overview page is opened automatically. Tap on the arrow keys to navigate to the required position in the program and on the return key to leave the page. With the following prompt if a jump to the position is required, the preset can be activated or cancelled.

Note: paging with   is done by 1 pixel at a time (the time for each pixel is a function of the 80-pixel width for the configured visible time range).

III-23 Program management: BlueEdit

Any number of recipe files can be created using *BlueEdit*. Only those required and actually needed are downloaded into *KS108 easy*.

The programs must be available for use in the *KS108 easy*. They can be changed during operation and are also saved in this form.

**DANGER!****Electronic devices can fail!**

As with any electronic system, a failure may cause damage to the memory.

For this reason:

- It is indispensable to maintain an external archive with back-up copies of data such as recipes and engineering.

To permit replacement of the instrument in case of failure, the recipe files just like the engineering files must be kept in an external archive outside *KS108easy*. For this reason, the recipes must be read out by the programmer and saved in an archive for the eventuality of important program changes in the instrument.

TIPS:

Clear and meaningful naming of programmer, recipes and tracks facilitates the programmer operation considerably, i.e. it makes sense to spend a reasonable amount of time for this work.

**NOTE!**

For additional information on recipe management, please, refer to the BlueEdit operating manual.

III-24 Tutorial: A practical example of a "programmer project"

KS108 easy is a powerful and flexible multifunction unit. In combination with the *BlueDesign* programming environment and the *BlueEdit* program editor, applications with the most exacting programmers can be realized.

Adapting the instrument to the required application is done using the *BlueDesign* development environment. *BlueDesign* is an easy-to-operate, powerful development environment for control applications. To create an application in *BlueDesign*, select prefabricated components (e.g. controllers or filters) using a graphic editor and link them to each other. You can select the prefabricated components from a "library". Programming knowledge is not required to develop an application with *BlueDesign*.

Now, you can create recipes for the programmer using *BlueEdit*. The *BlueEdit* program editor permits creation, editing and management of recipes for existing engineering. The recipes are adapted to the selected engineering automatically. Access to the recipes in *KS108 easy* is possible at all times. Normally, the track sequences are edited graphically, however, it is also possible to use values for this purpose.

Content

In this chapter, a simple example will teach you how to create an application with a programmer for *KS 108 easy* using the PMA library, the *BlueDesign* development environment and the *BlueEdit* program editor.



NOTE

Information on the fundamentals of working with *BlueDesign* is given in sections "The components of the development environment" and "Working with the development environment" of the *KS108 easy* manual.

This section contains information on the following subjects:

- Using the PMA library in *BlueDesign*
- Application parameter setting by means of *BlueDesign*
- Transfer of the application to *KS 108*
- Recipe creation using *BlueEdit*
- Transfer of the recipe to *KS 108*
- Starting the program

The time expenditure for studying the example is approx. three to four hours.

Prerequisites

You should have the following knowledge:

- Basic knowledge of the *Microsoft Windows™* operating system
- Knowledge of the standard programming languages to DIN EN 61131-3 (especially Function Block Diagram and Sequential Function Chart)

Technical prerequisites

The technical prerequisites are:

- *BlueDesign* version 1.6.1.0 (or higher)
- The *BlueSimulation 108* software must be installed (see Chapter *Installing the software on PC*)
- *BlueEdit* version 1.0 (or higher)

The components

A short overview of the *BlueDesign* development environment, the PMA library and *BlueEdit*: is given below.

BlueDesign

The *BlueDesign* development environment supports the overall development cycle of a project:

- **Developing an application:** Applications are composed of application blocks. These application blocks are selected from a library. An application can be structured according to various criteria. Moreover, templates can be created, so that frequently used parts of the application can be simply copied for using them again. These reusable blocks are termed macro blocks.

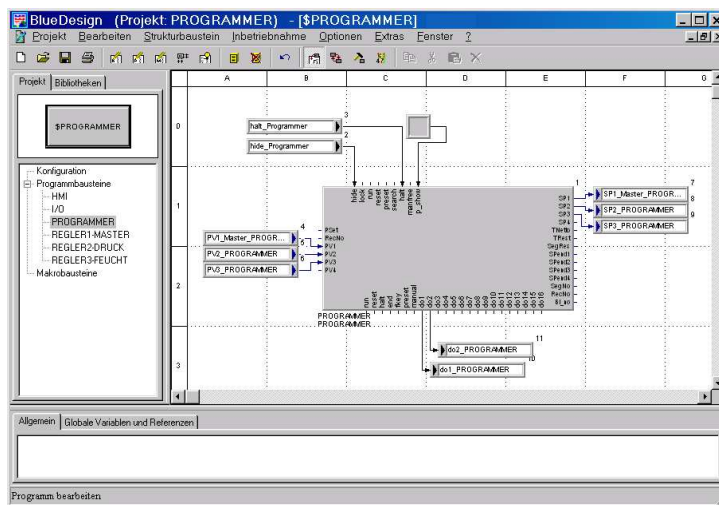


Fig. 697: Programmer project: Example for developing an application

- **Creating an operator interface:** A graphic editor is used to create the operator interface. This editor provides functions to place controls, displays and symbols on the worksheet and to configure them. The appearance of these items is largely identical to the one of the operator interface used on *KS 108* later.

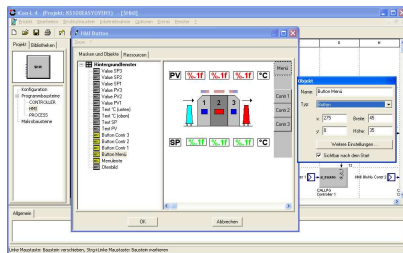


Fig. 698: Programmer project: Example for creation of an operator interface

In our example, we use mainly standard PMA operating pages. These are created automatically by using function blocks with operating pages (programmings, controllers, visualization library).



NOTE!

A description of the operator interface is not provided in this example. For this purpose, refer to the tutorial "A practical example" from Chapter II Development environment in the *KS108* easy manual.

- **Parameter setting:** The out-of-the-box functions of the PMA library are ready for use. To adapt them to your individual requirements, however, they must be configured. This is done using parameters which can be simply entered or selected in *BlueDesign*. For numerous parameters, value lists offering a choice

of possible options are displayed. Moreover, the entries are checked. If a value is inadmissible, it is corrected into the next permissible value.

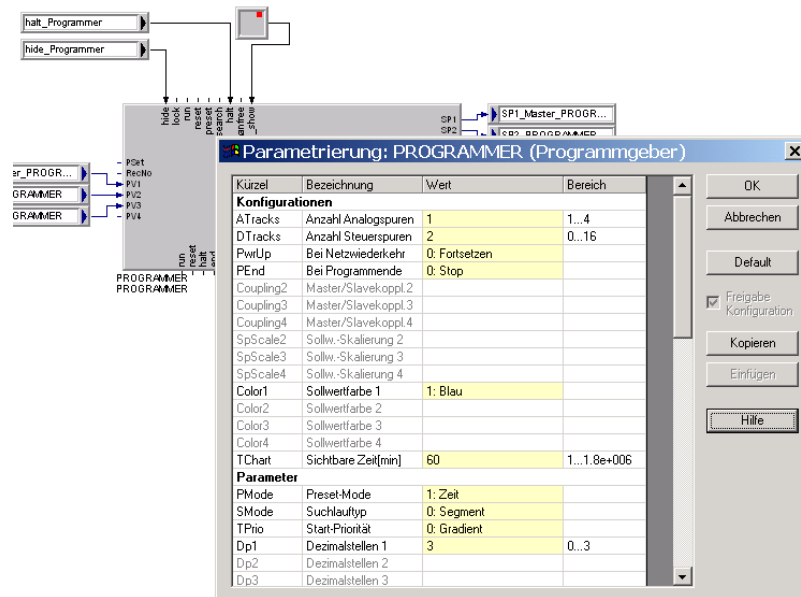


Fig. 699: Programmer project: Parameter dialogue

In the parameter dialogue, the on-line help is offered. It provides detailed information on each function block.

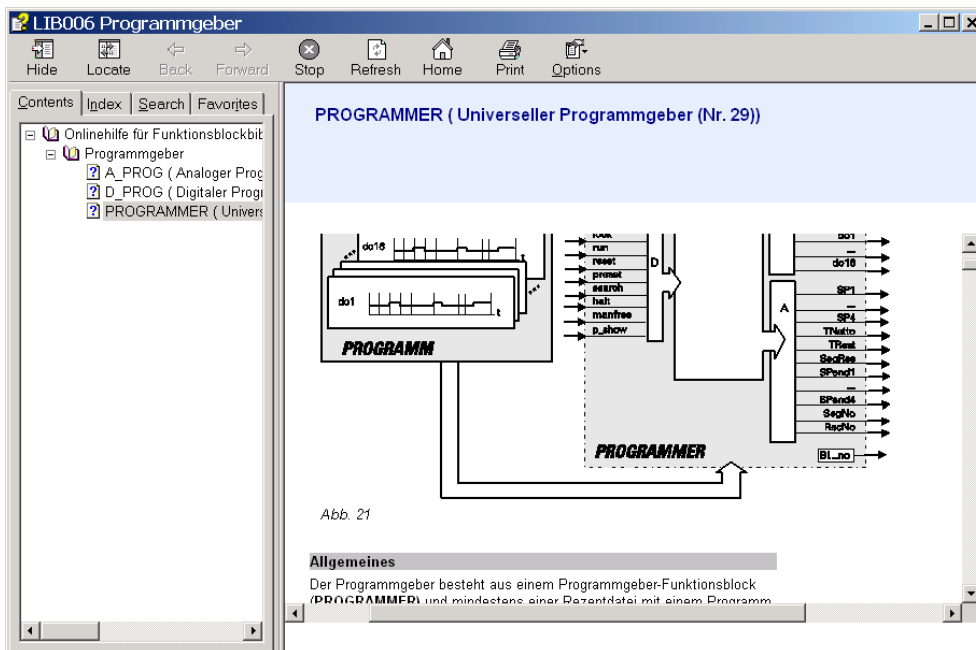


Fig. 700: Programmer project: Parameter dialogue

- Testing an application:** There are two possibilities to localize application errors: When you position the mouse pointer on a connecting line, the relevant values are displayed (no. 2 in Fig. 701). Alternatively, special blocks from the PMA library can be used. As trouble shooting in applications is also called "Debugging", these are termed "Debug blocks". You can use these blocks to display information on the program during runtime (1 and 3 in Fig. 701). They are displayed exclusively in the development environment (i.e. not on the instrument).

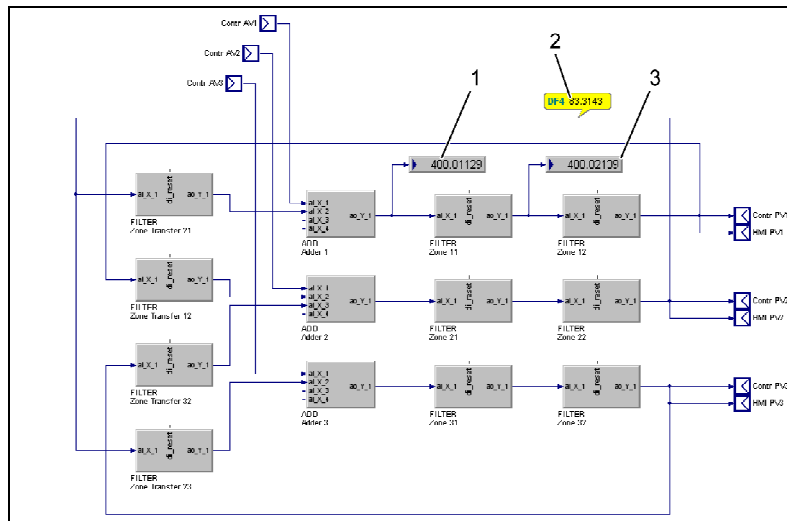


Fig. 701 Programmer project: Debugging example

PMA library

The PMA library comprises a large number of function blocks covering all functions normally used for operation of a process. These include among others:

- Mathematic functions
- Logic functions
- Alarm and limit value functions
- Controllers
- Programmers

Moreover, visualization for numerous library functions is made available to you automatically, e.g. for the controller termed "Control" that will be used in our example:

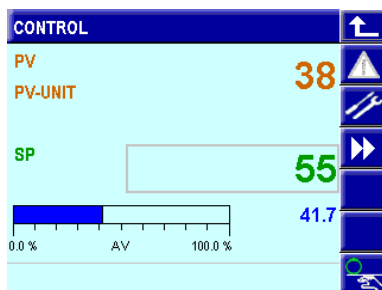


Fig. 702: Programmer project - Control, with examples

or for the universal programmer called PROGRAMMER

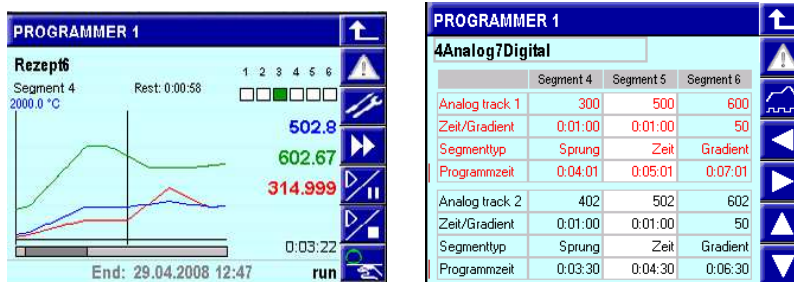


Fig. 703: Programmer project - programmer, with examples

The PMA library facilitates the program development in two respects:

- **Speed:** It speeds up the application design, because applications can be built up almost completely using functions of the library and the user interface is created to a large extent automatically.
- **Quality:** The PMA library offers optimized functions which can be considered as a "black box". Thus potential programming errors are avoided.

BlueEdit

The *BlueEdit* programmer offers all functions required for recipe creation, operation and management:

- Creation of recipes for the engineering
- Reading recipes from KS108 easy, loading recipes into KS108 easy
- Recipe management

The recipes are displayed graphically for the largest part. For precise information, the track values are displayed simultaneously:

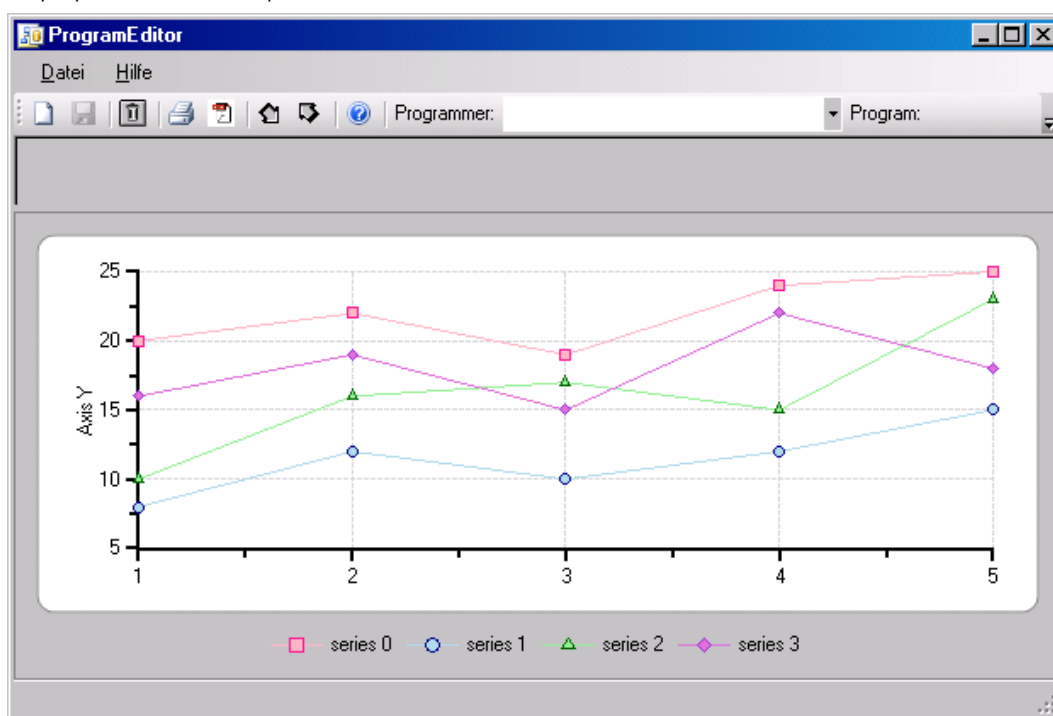


Fig. 704: Programmer project: Program editor

BlueEdit offers the following advantages for program design:

- **Speed:** Recipe creation is speeded up, because the graphic display provides at-a-glance information on the curves at all times.
- **Reliability:** Due to the clarity of recipe management in *BlueEdit*, recipes can be found easily. You can set up your own recipe "library".

The example

The application example to be created must meet the following requirements: It must permit control and simulation of an oven. For checking the heat resistance, a thermal profile is required. The current process temperature must be displayed on the controller operating page. The programmer operating pages must permit the entry of small program changes, e.g. of setpoint and runtime of two program steps. The access to the operation can be limited using codes to be entered at various levels. The main operating page of the application is the programmer:

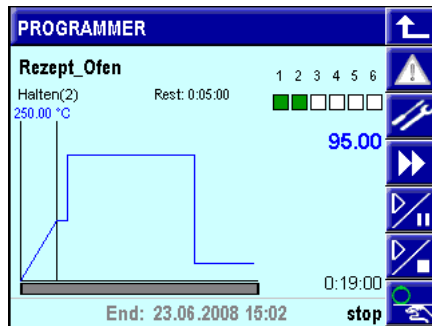


Fig. 705: Programmer project: Oven, main programmer operating page

Description

The following characteristics for control of the oven are required:

- Process values, setpoint and correcting variable are displayed on the controller operating page.
- The setpoint and duration of the program for checking can be adapted.
- Access to the operation is enabled by 3 access authority levels, otherwise, the access is disabled completely.
- The setpoint may be within 0 and 400.

Simulation

The behaviour of the oven must be simulated. Simulation is based on a the following assumption:

- Setpoint changes are realized by the oven with a delay (2nd order).

III-24.1 Step 1: Creating a new project

At first, you must create a new project:

1. **Creating a project:** To create a project, proceed as follows:
 - Click menu item "Project/New".
 - Give the project a name (in this example: "PROGRAMMER EXAMPLE").

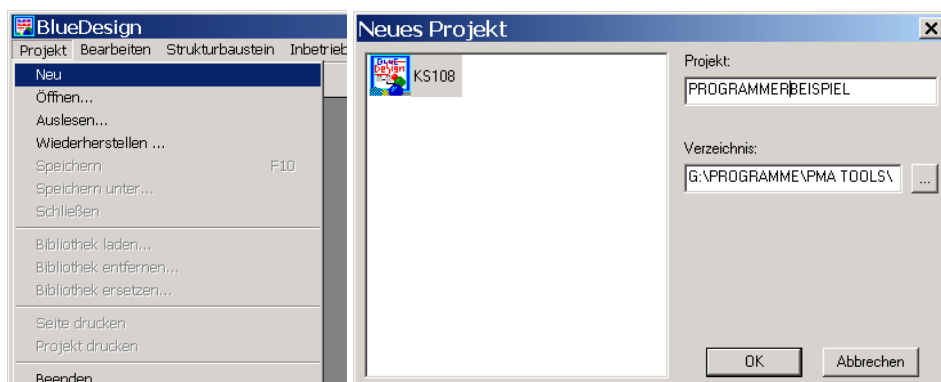


Fig. 706: Programmer project: BlueDesign dialogue window "New Project"

2. **De-activating the quick start wizard:** Now *BlueDesign* may start a quick start wizard. In this wizard, you can choose which action you want to carry out next. For the example, we don't need the wizard.
 - To de-activate this option, tick checkbox "Show quickstart next time".
 - Then click button "Program" in the quick start window.



Fig. 707: Programmer project: Quick start wizard

- 3. Creating a new program block:** To create a (new) program block, enter the name "PROCESS" into the input field and click button "OK".
In the structure tree on the left, there is now a program block with the name "PROCESS" .

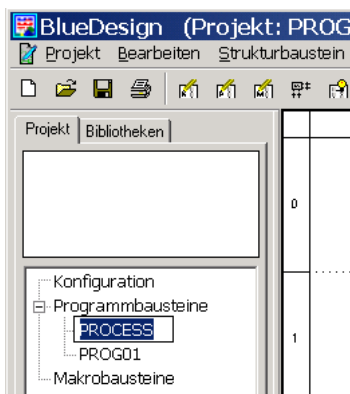


Fig. 708: Programmer project: BlueDesign dialogue window "Program block", project overview

- 4. Deleting the PROG01 program block:** The automatically generated program block "TASK01" is not required any more. Mark the block using the mouse and select command "Delete" in the context menu.



Fig. 709: Programmer project: BlueDesign dialogue "Delete program block"

Digression: How to build up projects in *BlueDesign*

As shown in the figure below, tab "Project" provides three main categories for subdivision of your project.

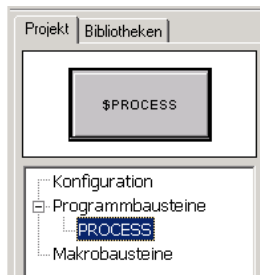


Fig. 710: Programmer project – Project structure in BlueDesign

- Program blocks:** Program blocks are the basic units making up your project. Program blocks consist of function blocks (for example controllers, filters, totalizers, etc.), links between function blocks as well as inputs and outputs. An example of such a function block is given in Fig. 701. Your application may consist of max. 15 program blocks.
- Configuration:** In the "Configuration", you can make the settings for interaction of your program blocks. Each program block is executed as a separate "task" and is given its own computing time by the run-time environment. For this, the run-time environment invokes the program blocks cyclically (as shown below). However, determination that (some) program blocks are invoked at longer intervals is also possible via the "cycle time", i.e. they are sometimes skipped.

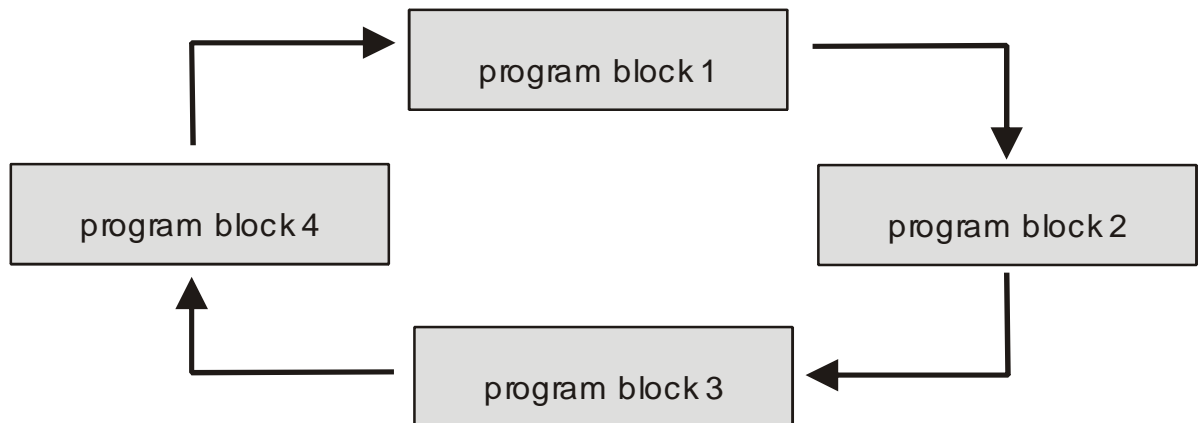


Fig. 711: Program tasks are invoked cyclically

Moreover, the order according to which the program tasks are invoked ("priority") can be determined. These settings can be made during "Configuration".

- Macro blocks:** When identical combinations of tasks are used in your program blocks, macro blocks can facilitate working. By means of macro blocks, you can create a template that may be used like any other blocks in your projects.

This explanation of the project structuring process may seem abstract and implausible. However, the whole purpose of the project structure will become transparent in the further course of the example.

III-24.2 Step 2: Creating a controller



NOTE

In this chapter, the properties of the components (controllers, filters, etc.) are explained only to the extent required for the example. More detailed information on these components is given in Chapter "Function library".

The oven must be controlled using a controller. For this purpose, a controller from the PMA library is used. Proceed as follows:

- Creating a program block:** Create a new program block and give it the name "CONTROLLER". Proceed as described in "Step 1". The result should look as shown below:

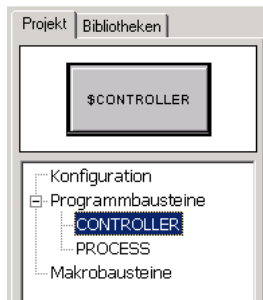


Fig. 712: Programmer project: New program block "CONTROLLER"

2. **Selecting tab "Libraries"**: Now, open tab "Libraries" and select the required block in the next step.

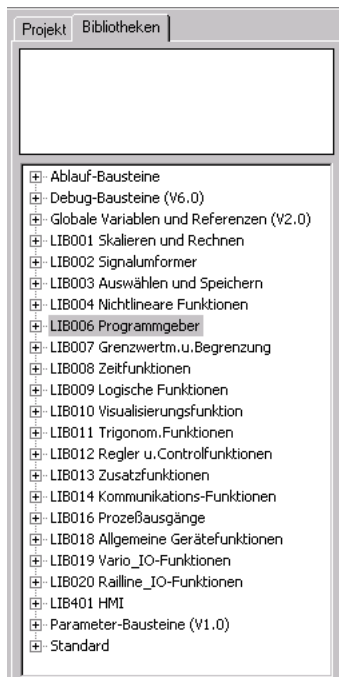



Fig. 713: Programmer project: The Libraries tab

A list of the available libraries is displayed. Libraries with characters "LIB" at the beginning are part of the PMA library. The remaining libraries provide general blocks such as input and output fields, or debug and comment fields.

3. **Selecting a controller**: The controller we need is in library "LIB012 Controllers". Click symbol  to open the library.

Now, select controller "CONTROL".

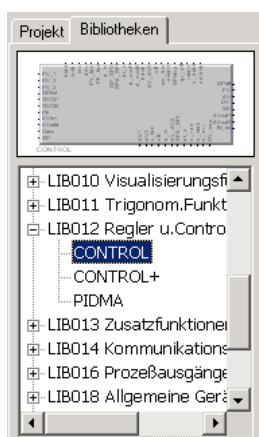


Fig. 714: Programmer project: Selecting controller "CONTROL"

The selected controller is displayed in the upper part of the tab window.

4. **Using the controller:** Click the controller symbol displayed in the upper part of the tab window. Keep the left mouse key pressed down and draw the block right into the worksheet. Then release the left mouse key.

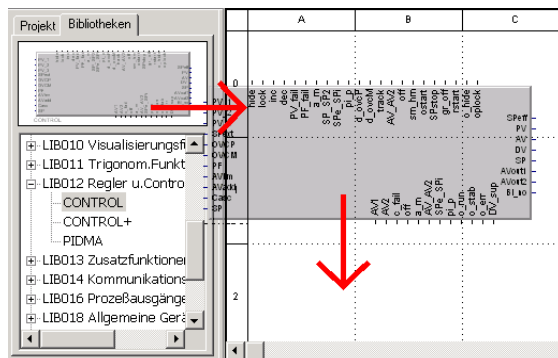


Fig. 715: Programmer project: Using controller "CONTROL"

The result should look as shown below:

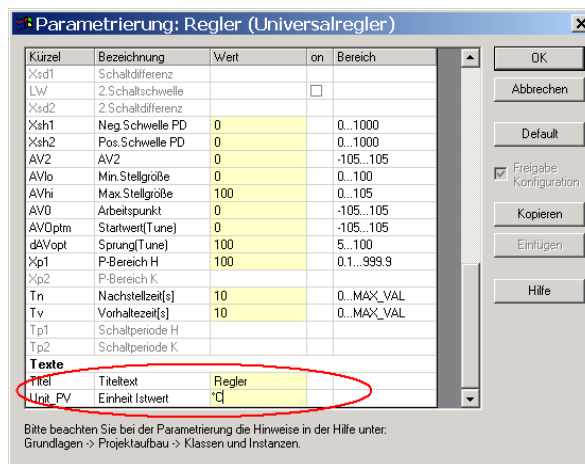


Fig. 716: Programmer project: Using controller "CONTROL"

5. **Defining a name:** Select the controller with the right mouse key. In the context menu, click command "Parameter Dialog ...". Scroll down in this dialogue window, until you find entry "Titel". Enter the name "Control" into column "Value" (shown with a yellow background). For "Unit_PV", enter the unit "°C" into the column "Value" (shown with a yellow background). Click button "OK" to save your entry.

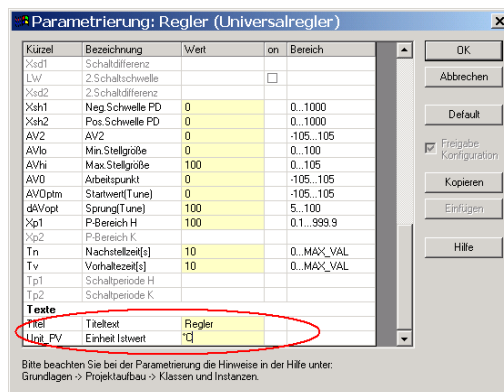


Fig. 717: Programmer project: Parameter setting window

6. **Determining inputs and outputs:** Now, connect the controller input and the controller output. As the actual application logic of our project must be located in program block "PROCESS", a possibility to

transmit an input value or to save the output value is required. For this purpose, we use the blocks "Input" resp. "Output" from library "Standard".

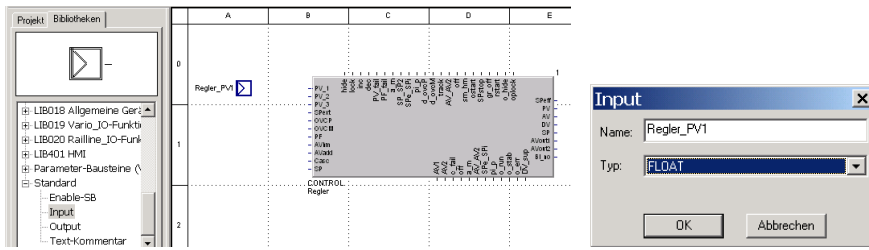


Fig. 718: Programmer project: Using inputs

- Select block "Input" and draw the symbol into the worksheet. Window "Input" opens.
- Assign the name "Controller_PV1" to the block.
- Select data type "FLOAT". Click button "OK" to save your entry.
- Create an input with the name "Control_SPext".
- Then create an output. For this purpose, use the Output block. Give the output the name "Controller_AVout1".

The result should look as shown below:

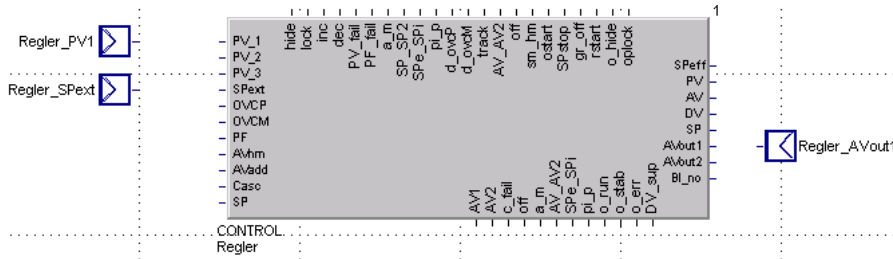


Fig. 719: Programmer project: Creating inputs and outputs



NOTE!

The data type "Bit/Bytes" is offered. For the inputs and outputs, however, we need the data type "Float". Always make sure to select the appropriate data type. Otherwise, it is not possible to make the connection between e.g. the input and the block.

7. Connecting inputs and outputs: Now, connect the input and output with the controller.

- For this, position the mouse pointer on the connecting element of the input. The mouse pointer changes into a pen (as shown below).
- Click with the left mouse key. The mouse pointer changes into a cross.
- Now, draw the mouse pointer from the input block to the "ai_PV_1" input of the controller. Then click on the target point of the connection to create the line.

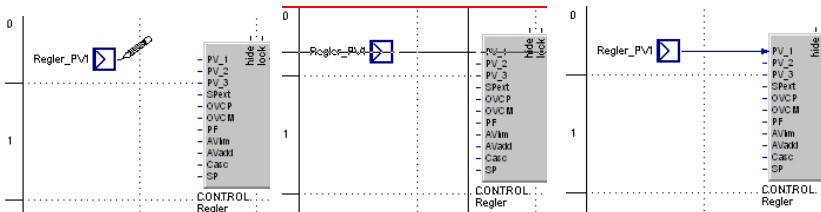


Fig. 720: Programmer project: Drawing a connecting line.

- Connect the controller output "ao_AVout1" to a block of the "Output" type. The result should look as shown below:

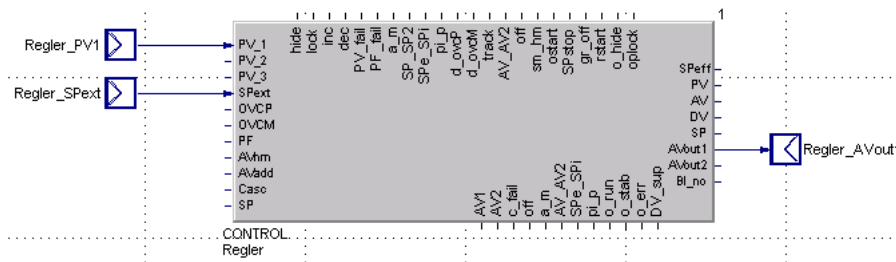


Fig. 721: Programmer project: Connecting inputs and outputs

8. **Saving the changes:** Save your changes. For this purpose, click button "Save project".

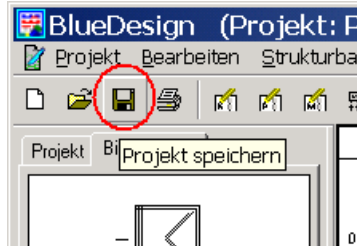


Fig. 722: Programmer project: Save project

III-24.3 Step 3: Creating a programmer



NOTE!

In this chapter, the properties of the components used (programmer, etc.) are described only to the extent required for the purpose of the example. Detailed information on these components is given in Chapter "Function library".

We want to make the oven go through a temperature profile. For the purpose of the temperature profile, we use a programmer from the PMA library.

Proceed as follows:

1. **Creating a program block:** Create a new program block with the name "PROGRAMMER". For this, proceed as described in "Step 1". The result should be:



Fig. 723: Programmer project: New program block "PROGRAMMER"

2. **Selecting tab "Libraries":** Go to tab "Libraries" to select the required blocks in the next step. Now you see the available libraries in the lower part of the window.
3. **Selecting the programmer:** The required programmer is in library "LIB006 Programmers". Click symbol to open the library. Select the programmer "PROGRAMMER".



Fig. 724: Programmer project: Selecting programmer "PROGRAMMER"

The selected block is displayed in the upper part of the tab window.

4. Using the programmer: Draw the block right onto the worksheet.

The result should look as shown below:

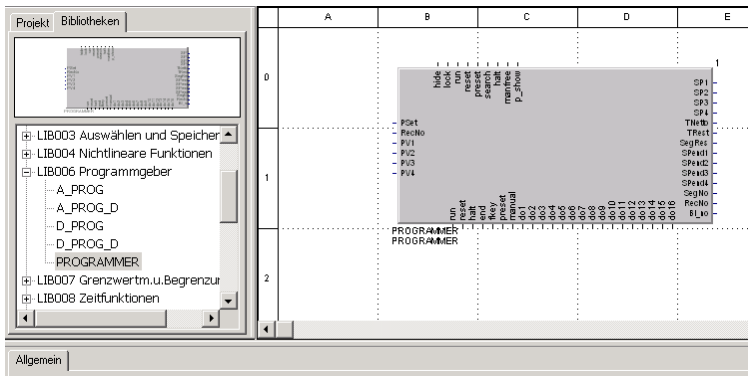


Fig. 725: Programmer project: Working with "PROGRAMMER" blocks

5. Setting inputs: For our example, you want to be able to change the program. For this purpose, you need the standard operating page (editing page) of the programmer. To activate the editing page, input "p_show" must get a positive signal. For this, use function block "Parameter (Bit)" from library "Parameter blocks".

- Select block "Parameter (Bit)" and draw the symbol onto the worksheet. Window "initial setting" is opened.

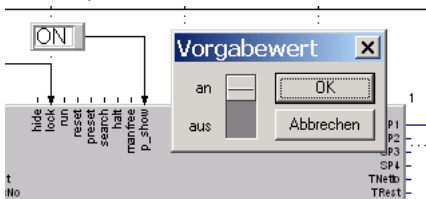


Fig. 726: Programmer project: Using block "Parameter (Bit)"

- Set the switch to position "on". Click "OK" to save your entry.
- Connect the "Parameter (Bit)" to input "p_show".

6. Determining the program inputs and outputs: Now, connect the programmer inputs and output. This means that you have to make the connections to our process and to the controller. As with the controller, use blocks "Input" resp. "Output" from the "Standard" library for this purpose.

- Select block "Input" and draw the symbol onto the worksheet. Window "Input" is displayed. Give the block the name "Programmer_PV1".
- Select data type "FLOAT". Then click button "OK" to save your entry.
- Create a FLOAT output using block Output. Give the output the name "Programmer_SP1".

- Create two BIT outputs: one for control of a ventilator with the name "do1_ventilator", and another one for control of the door lock "do2_door lock".
- Create another BIT output for the password operating page with the name "Programmer END". We will use it to change to the operating page of the access privilege at the end of the program.

**NOTE**

The offered data type is "Bit/Bytes". However, we need data type "Float" for the PV and SP inputs and outputs. Make sure to select the appropriate data type. Otherwise, it is not possible to connect e.g. the input and the block.

7. Connecting inputs and outputs: Now, connect the input and output to the programmer as follows:

- Input "Programmer_PV1" to input "PV1".
- Output "Programmer_SP1" to output "SP1".
- Output "do1 ventilator" to output "do1".
- Output "do2 door lock" to output "do2".
- Output "Programmer END" to output "end".

The result should look as shown below:

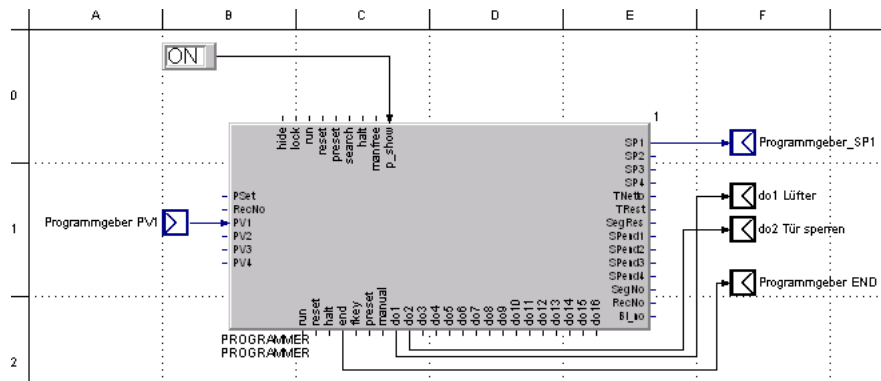


Fig. 727: Programmer project: Creating inputs and outputs

8. Saving changes: Click button "Save project" to save your changes.

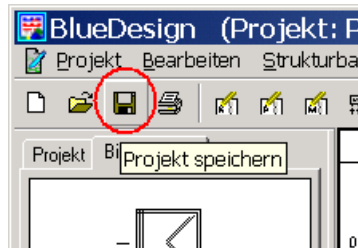


Fig. 728: Programmer project: Save project


III-24.4 Step 4: Creating the password management

**NOTE**

In this chapter, the properties of the components used (function block PASSWORD, etc.) are described only to the extent required for the purpose of the example. Detailed information on these components is given in Chapter "Function library".

We want the operation of the oven to permit a number of changes and operator interventions for various users via an access management function. For access management, function block PASSWORD from the PMA library must be used.

Proceed as follows:

1. **Creating a program block:** Create a new program block and give it the name "PASS". Proceed as described in "Step 1".
2. **Selecting the "Libraries" tab:** Go to tab "Libraries" to select the required block in the next step. Now you can see the list of the available libraries.
3. **Selecting the access management:** The required PASSWORD function block is found in library "LIB013 Supplementary functions". Click symbol  to open the library. Select programmer "PASSWORD" and draw it onto the worksheet.

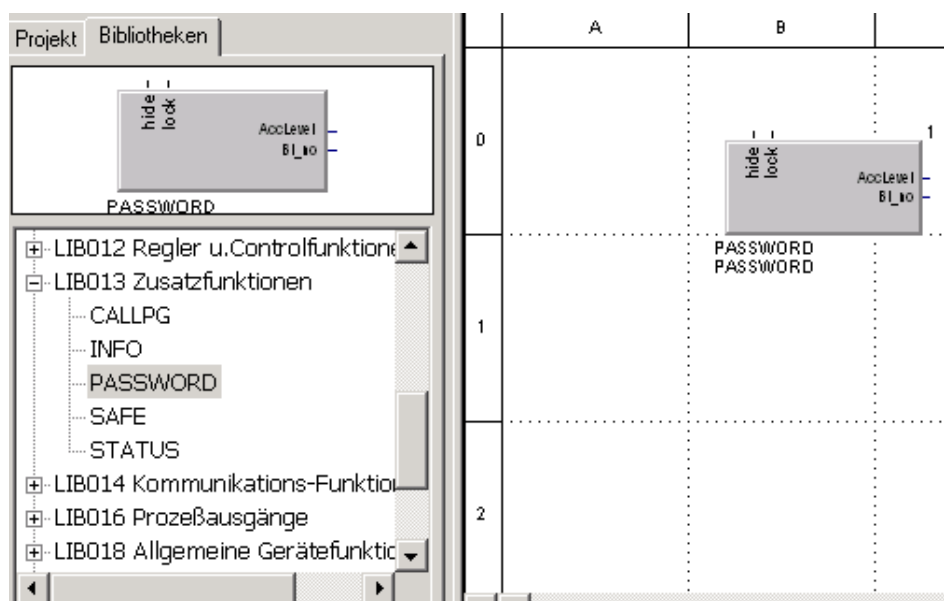


Fig. 729: Programmer project: Selecting "PASSWORD"

4. Setting inputs:

For the event that the programmer arrives at the program end, we want to make the password available. Only with extended access privilege, it is possible to select a recipe and to change the program via the editing page and parameters and configurations become available.

The PASSWORD function block signals the access level via output "AccLevel". When the programmer is in the end position this output is used to display the PASSWORD operating page. For this purpose, function block "CALLPG" is provided.

- Select block "CALLPG" from library "LIB013 Supplementary functions" and draw the symbol onto the worksheet. Connect output "BL_no" of the PASSWORD with input "BlockNo" of block CALLPG.
- For switching over, select block "Alarm" from library "LIB007 Limits and limit values" and draw the symbol onto the worksheet. Connect output "AccLevel1" of the PASSWORD to input "X_1" of block ALARM.
- Select block "AND" from library "LIB009 Logical functions" and draw the symbol onto the worksheet. Connect output "alarm" of the ALARM block to input "d_1" of the AND block. Connect the output "z_1" of the AND block with input "d_1" of the CALLPG block.

Now the program should look roughly as shown below:

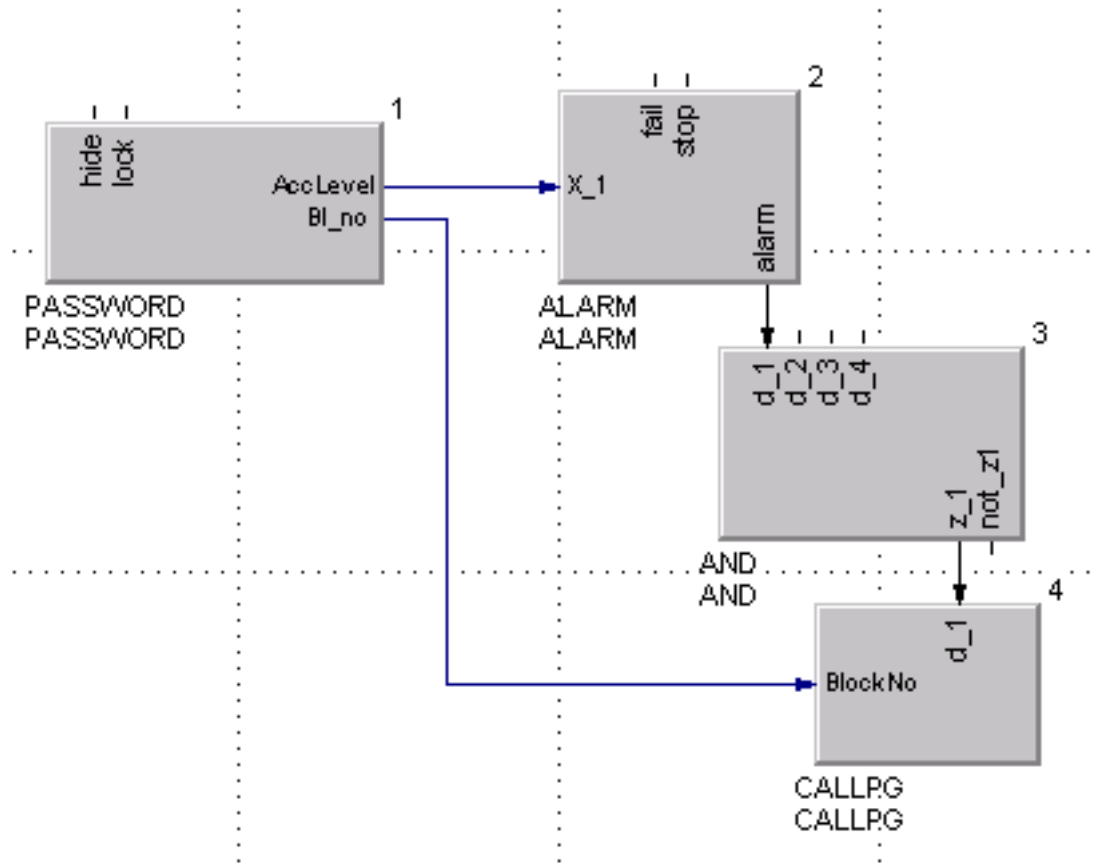


Fig. 730: Programmer project: Purposeful display of an operating page

5. Determining the program inputs and outputs: Now an input must be connected. As with the controller, we use the "Input" block from library "Standard" for this purpose.

- Set block "Input" and assign the name "Programmer END" to the block.
- Select data type "BIT". Then click button "OK" to save your entry.

The result should look as shown below:

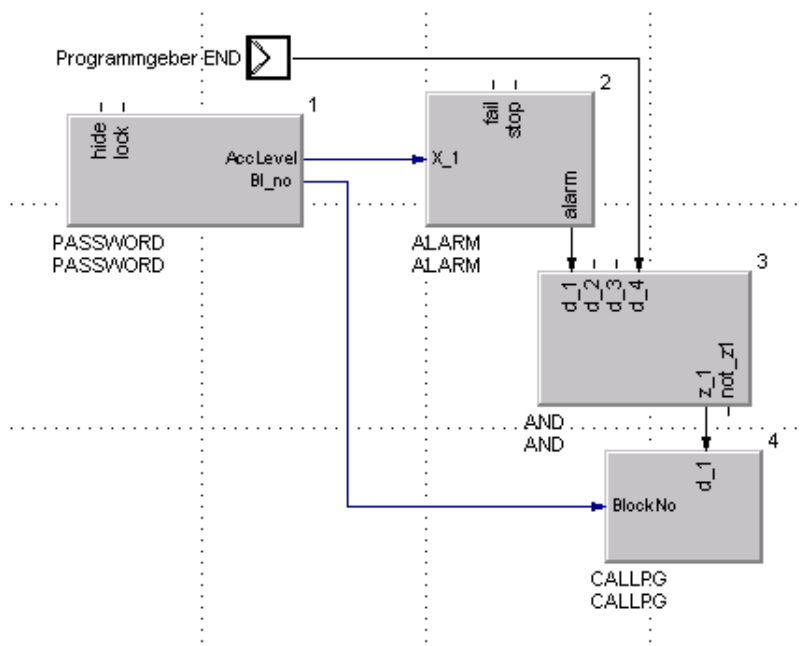


Fig. 731: Programmer project: Creating an input

6. Saving changes: Save your changes. For this, click button "Save project".

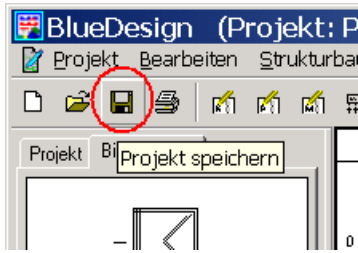



Fig. 732: Programmer project: Save changes

III-24.5 Step 5: Creating the simulation

As mentioned above, the program example is intended to simulate the behaviour of the oven. The behaviour of the oven is simulated using a filter from the PMA library. Task of the filter is to provide the input variable with a delay (to be defined) as an output variable. The delay will be determined by a parameter (for further information, refer to section "III-24.7 Step 7: Determining parameters").

 **NOTE**
 For additional information on these components, refer to Chapter "Function library".

We want to use 2 control tracks: A ventilator must run dependent on the program and the door must be blocked while the program is running.

For changing the program, operators must have the corresponding access privilege.

1. **Selecting the PROCESS program block:** Open tab "Project" and *double-click* to select program block "PROCESS".

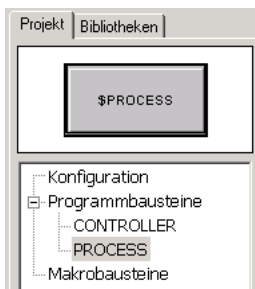


Fig. 733: Programmer project: Selecting program block "PROCESS"

2. **Adding a filter:** We want to simulate the behaviour of an oven chamber. To simulate an oven chamber, we use two filters connected in series. Task of the filters is to smoothen the (often quite important) changes of the output variable so that the filter output value is similar to the behaviour of an oven chamber. The following graph is intended as an illustration (the upper curve shows the output value, the curve in the middle shows the value after passing the first filter and the bottom curve shows the final value after passing the second filter):

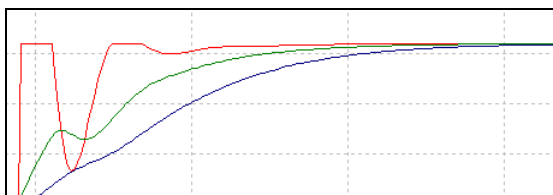


Fig. 734: Programmer project: Filter curves

To add the filters, proceed as follows:

- Go to tab "Libraries" and open library "LIB008 Timing functions".
- Click block "Filter" and draw two filter blocks onto your worksheet (as described above).

- Assign the name "Zone 1" to the first filter and the name "Zone 2" to the second filter. For this, click the filter and select command "Parameter Dialog ..." in the context menu. Enter the name into Column "Titel" of this dialogue.
- Connect the two filters as shown below.

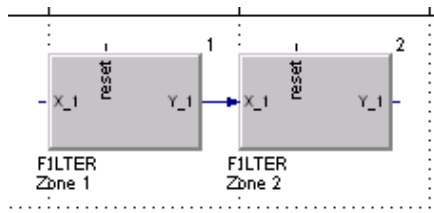


Fig. 735: Programmer project: Two series-connected filters

3. **Adding a scaling block:** a scaling block must be added to the simulation for the following reasons: Conversion of the current correcting variable provided by the controller into a temperature value is required. The correcting variable is specified in per cent (values between 0 and 100 are possible), the oven temperature is specified in degrees (values between 0 and 400 are possible). For these facts, the correcting variable is multiplied by factor 4 in the scaling block. You will probably wonder where this information comes from, i.e. where the scaling block knows, for example, that the input variable must be multiplied by factor 4. These values are defined as parameters later (also refer to section "Defining parameters").
Add a scaling block to the program, connect it as shown in the following figure and assign a name to it. Note: the scaling block can be found in library "LIB001 Scaling and computing"; the block name is "SCAL".

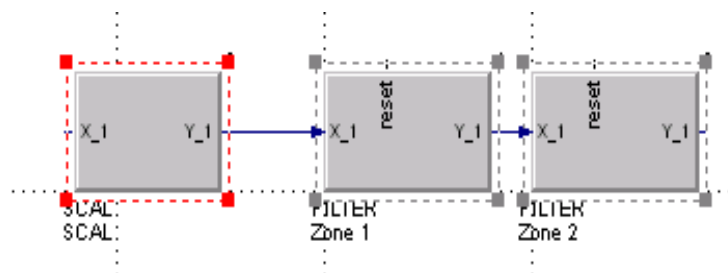


Fig. 736: Programmer project: Scaling block and filters

4. **Preparing the connection to the controller:** The controller correcting variable must be used as an input variable for simulation. To make the data available, a block of the "INPUT" type is used again. Select a block of the "INPUT" type from library "STANDARD". Assign the name "Controller AVout1" to it and select data type "FLOAT".
Connect the block to input "X_1" of the scaling block.
To provide the data to the controller and to the programmer, an "OUTPUT" type block from the "STANDARD" library must be used. Assign the name Controller PV1 to the block and select the "FLOAT" data type. Connect the block to the "Y_1" output of the last filter "Zone 2".
The result should be look roughly as shown below:

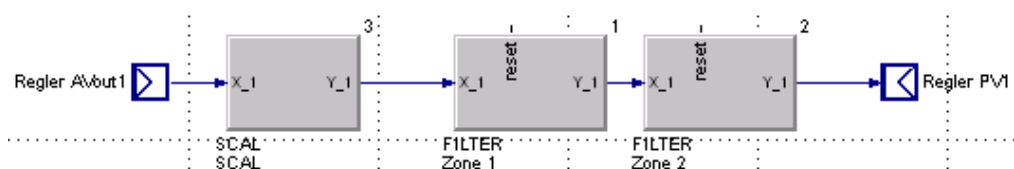


Fig. 737: Programmer project: Program with input



NOTE

If an error message "Error! Invalid connection" is displayed when making an attempt to connect two blocks, you have probably selected a wrong data type for an output. In this case, delete the relevant output and create a new one. To delete a block, position the

mouse cursor on the block and select command "Delete" in the context menu.

- Additional connection :** In a real project, inputs and outputs are connected via an I/O system. In our example, we simulate the ventilator and the door lock using an LED. Insert 2 LEDs from library "Debug blocks". Insert a BIT input "do1 ventilator" and connect it to the first LED. Insert a BIT input "do2 lock door" and connect it to the second LED.

The result should look roughly as shown below:

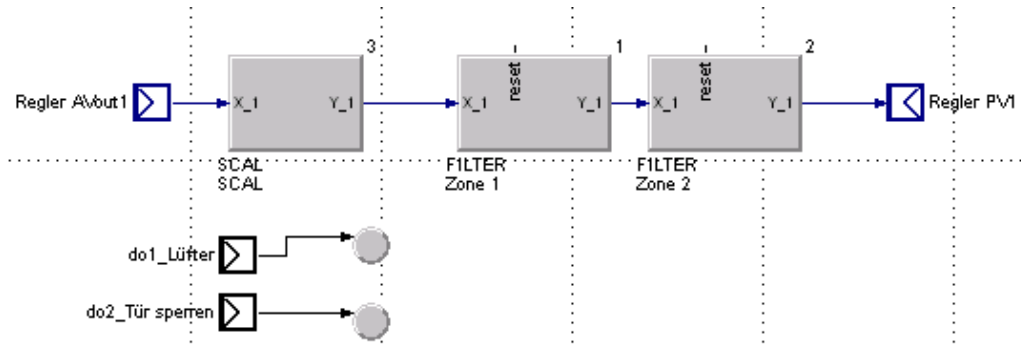


Fig. 738: Programmer project: Program with input



NOTE

If an error message "Error! Invalid connection" is displayed when making an attempt to connect two blocks, you have probably selected a wrong data type for an output. In this case, delete the relevant output and create a new one. To delete a block, position the mouse cursor on the block and select command "Delete" in the context menu.

- Saving changes:** Save your changes. For this purpose, click button "Save project".

III-24.6 Step 6: Determining interfaces, connecting program blocks

So far, we have created the program blocks "CONTROLLER", "PROCESS", "PROGRAMMER" and "PASS". Each of these program blocks has an input and an output interface. However, these inputs and outputs still cannot be accessed from "outside". Moreover, the program blocks in our project are never called up. For this reason, the following steps are required at this point:

- Creating the main program:** The "Configuration" is the uppermost level of a project. All program blocks must be provided at this level.
 - Double-click node "Configuration" on tab "Project" to display the "Configuration" worksheet.
 - Click node "CONTROLLER".
 - Draw the "CONTROLLER" block onto the configuration worksheet.
 - Proceed identically with program blocks "PROCESS", "PROGRAMMER" and "PASS".

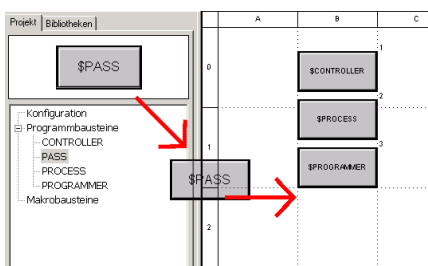


Fig. 739: Programmer project: Creating the "Configuration"

**NOTE!**

Program blocks are executed only if they are found in the "Configuration". Configuration without program blocks doesn't make sense, because the program would lack functionality (a program without "tasks").

The two program blocks don't have inputs and outputs (interfaces). The inputs and outputs must still be assigned to make them visible at the higher level. For this, proceed as follows:

2. **Selecting the "PROCESS" program block:** Double-click program block "PROCESS" in the tree structure to select it.
3. **Executing the Design context menu command:** Click command "Design" in the context menu.

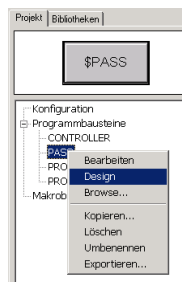


Fig. 740: Programmer project: Selecting the "Design" command

Now a worksheet for input and output assignment is displayed in the right section of the window.

4. **Assigning inputs:** First, the inputs must be assigned. Position the mouse pointer above input "Controller AVout1" in column "Inputs". A hand symbol appears as a mouse pointer (see below). Click with the left mouse key. Keep the mouse key pressed and draw the input to the left border of the program block symbol. Then release the left mouse key. You may also change the design into the normal PMA view showing the name of the function block at the bottom left and the inputs/outputs with designations inside. Position the digital inputs "do1_Ventilator" and "do2_lock door" at the top.

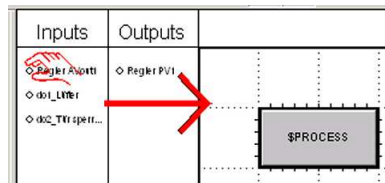


Fig. 741: Programmer project: Assigning inputs

Then position the outputs on the right side of the dialogue window. The result should look as shown below:

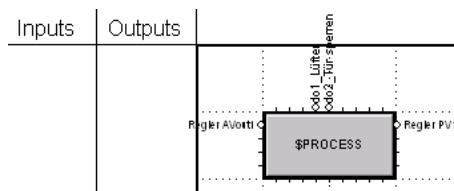


Fig. 742: Programmer project: Inputs assigned

5. **Adapting the labels:** As standard, the input and output names are displayed outside the program block. However, names can be displayed also inside the blocks to increase the clarity. With the name at the bottom left, the clarity is improved. For this, click button "Connector labels inside or outside". For the name, click "Name bottom left, outside".

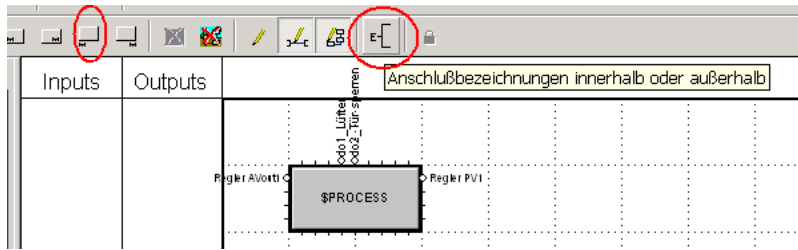


Fig. 743: Programmer project: Adapting the layout

6. **Adapting the block size:** The appearance of the program block symbol is unacceptable. For this reason, position the mouse pointer in the bottom right corner of the block. The mouse pointer changes into a double arrow. Keep the left mouse key pressed down to enlarge the symbol (as shown below).

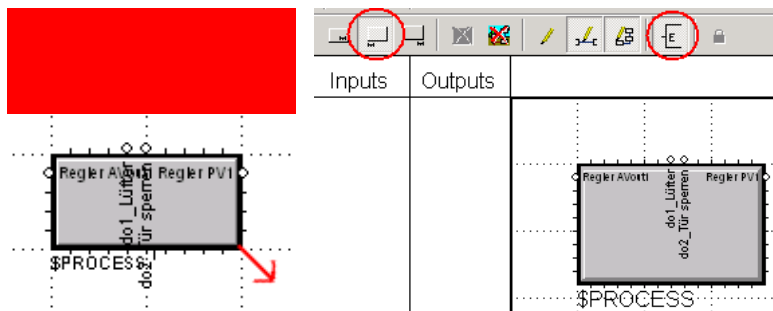


Fig. 744: Programmer project: Adapting the block size

7. **Configuring the "CONTROL" block:** Proceed analogously with the "CONTROL" program block. The appearance of the result should be:

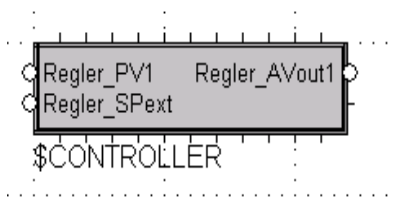


Fig. 745: Programmer project: Configuring program block "CONTROL"

8. **Configuring the "PROGRAMMER" program block:** Proceed analogously with the "PROGRAMMER" program block. The appearance of the result should be as shown below:

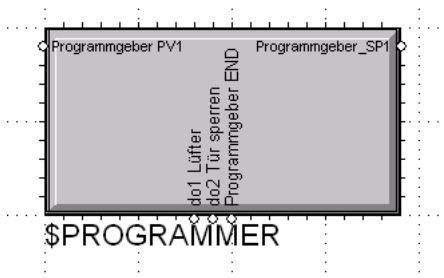


Fig. 746: Programmer project: Configuring program block "PROGRAMMER"

9. **Configuring the "PASS" program block:** Proceed analogously with the "PASS" program block. The appearance of the result should be as shown below:

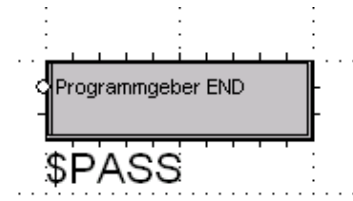


Fig. 747: Programmer project: Configuring program block "PASS"

10. **Connecting program blocks:** So far, the program blocks stand side by side without connections. The controller doesn't know the simulation, and vice versa. The programmer and the access privilege (PASS) must be included as well. For this reason, we have to connect the program blocks in the next step. At first, connect output "Control_AVout1" of the controller with the input "Control_AVout1" of the simulation.

The "Control_PV1" simulation output must be connected to the controller input "Control_PV1" and to the "Controller_PV1" programmer input.

This is done as follows:

- On tab "Project", double-click on item "Configuration" in the tree structure.
- Connect the two program blocks as shown below.

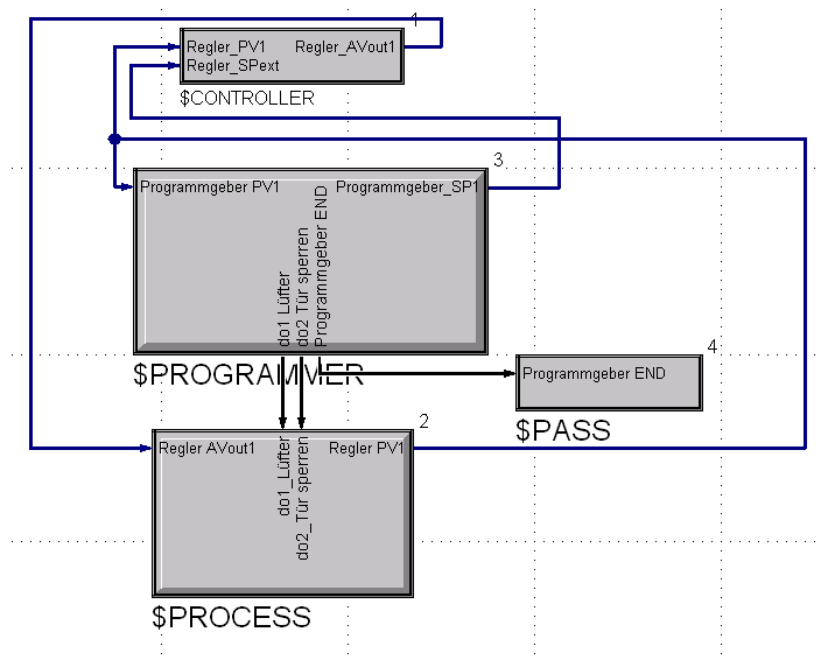


Fig. 748: Programmer project: Connecting program blocks

11. **Saving changes:** Save your changes. For this, click button "Save project".

III-24.7 Step 7: Determining parameters

Now the PMA library components used for the example must be configured. For configuration, parameters are used to determine the behaviour and properties of the blocks.

Digression: Parameter types

BlueDesign offers two basic methods to work with parameters. One possible method was already explained: The names of blocks or the data type of inputs and outputs were defined by parameters.

This was done in the *BlueDesign* editing mode. In this mode, program blocks, macro blocks and connections are handled and parameters can be entered. However, there is a problem: Multiple program blocks of the same type can be used in a program. Example: For control of two ovens each with three chambers in a project, the result might look as shown below:

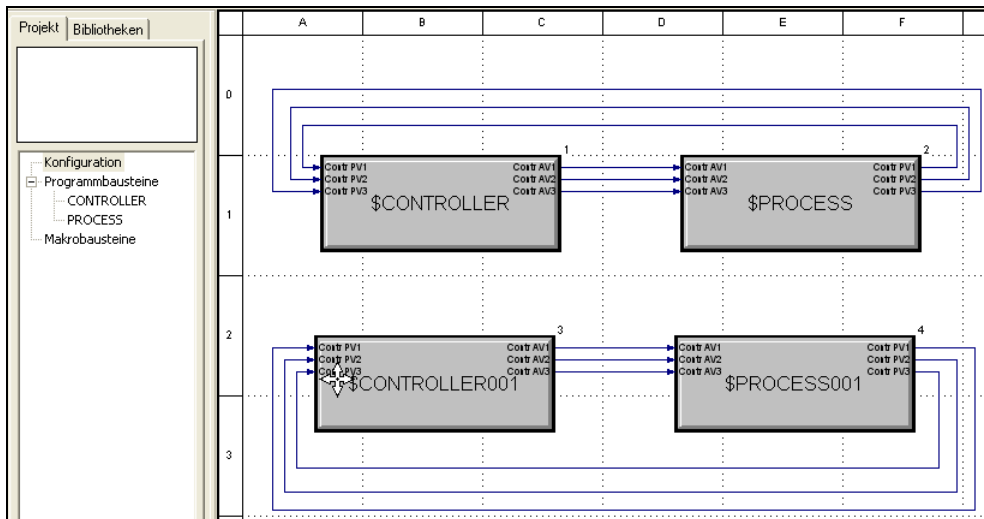


Fig. 749: Programmer project: Multiple use of program blocks

In the example shown above, two program blocks of types "CONTROL" and "PROCESS" are used. Now the maximum oven temperature and various simulation properties, etc. are determined. If we would determine the parameters in the editing mode, the parameters of each copy of the same program block type would be identical. This is not reasonable. After all our two ovens may have different properties (e.g. maximum temperature) or a different behaviour.

BlueDesign offers a simple solution to this problem: Apart from parameter editing in the editing mode, the parameters can be edited also in the "Run mode". In this mode, parameters for each copy of a block or program block can be defined individually.

Note on terminology: a template is also termed a "class" and a copy an "Instance".

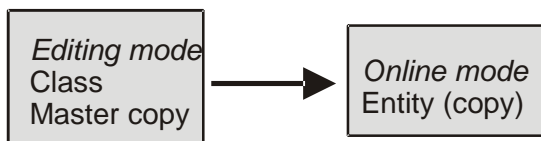



Fig. 750: Editing mode and online mode

Parameters entered during the online mode relate exclusively to the instance, *never* to the class ("template"). This can be compared to the following situation: If you copy a page from a book, the remarks you make on the copy aren't found in the original.

Summary:

- **Editing mode:** Parameters entered in the editing mode are valid for all instances ("copies") of a block. These parameters can be overwritten in the Run mode.
Purpose: These parameters should be used only to determine general properties such as data types.
- **Run mode:** Parameters entered during the Run mode are valid only for the relevant instance ("copy") of a block. They overwrite the parameters entered into the corresponding instance ("Copy") during the editing mode.
Purpose: Determination of the (virtual) parameters for a block.

 **NOTE!**
As a basic rule: Enter the parameters during the Run mode. Use the editing mode for parameter input only, if you are sure that a parameter must be applicable to all instances of a block.

Entering parameters

To determine parameters, proceed as follows:

1. **Starting the Run mode:** Start the Run mode using menu command "Run/Enter".

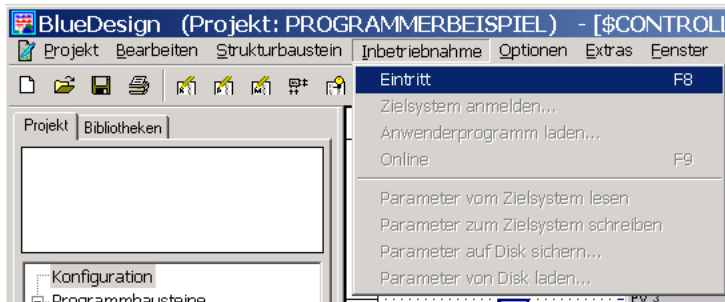


Fig. 751: Programmer project: Starting the Run mode

In the tree structure on the left side, an overview of the project instances is shown. Click item "\$Controller".

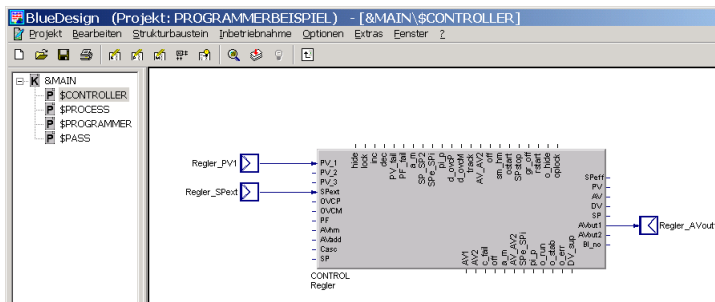


Fig. 752: Programmer project: Run mode

2. **Configuring the "Controller":** Position the mouse pointer on controller "Controller". Select command "Parameter Dialog ..." of the context menu. We want to determine the setpoint using a program, i.e. as an external setpoint. Therefore, assign value "1: Setpoint/cascade" to parameter "SPfunc". The maximum temperature of the oven chamber must be 400 °C, so assign value "400" to parameter "SPhi". As further parameters, we have changed the **title** "Controller" and entered the unit "°C" under **Unit_PV**. Click button "OK" to save the entry.



NOTE

Additional information on the parameter meaning is given in Chapter "Function block reference".

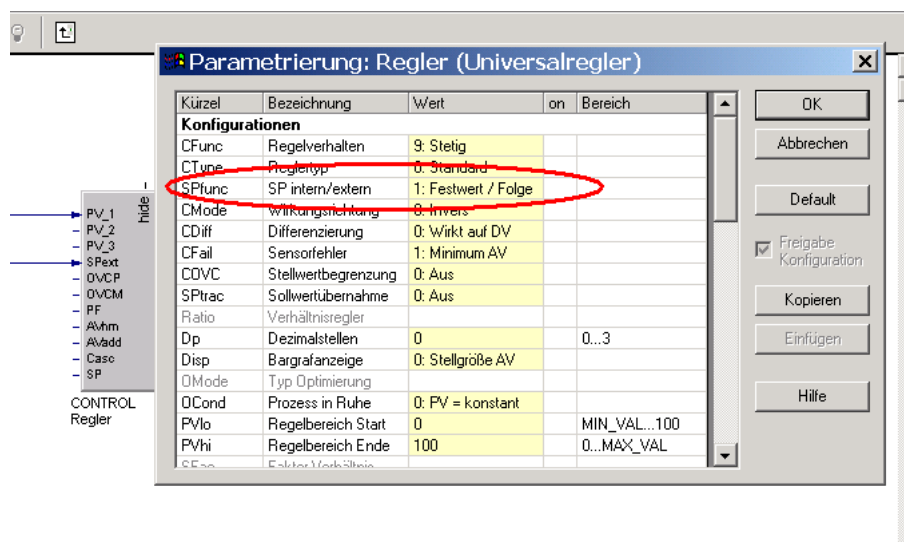


Fig. 753: Programmer project: Configuring the setpoint for the controller

3. **Changing over to program block "PROCESS":** Now, click the program block name in the tree structure to change over to program block "\$PROCESS".
4. **Configuring filters:** Two filters are used to simulate the behaviour of the oven. To simulate a more or less realistic behaviour of the oven, the filter input variable must be applied to the output with a delay of 10 seconds.

For this purpose, assign the value of 10 to the time constant of "T" of both filters.

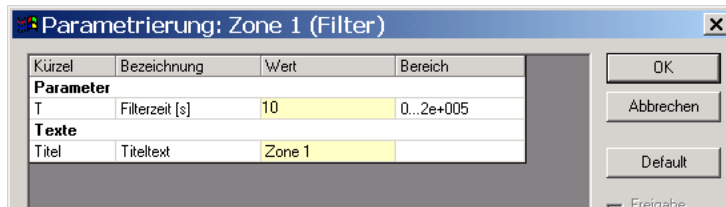


Fig. 754: Programmer project: Configuring the time constant for a filter

5. **"Scaling block:** The scaling block is required to convert the current controller correcting variable into a temperature value. The correcting variable is specified in per cent, the oven temperature is specified in degrees Celsius (maximum 400). To show these conditions, the correcting variable is multiplied by factor 4 in the scaling block.

Enter factor "4" for input "X1" of parameter "A1".

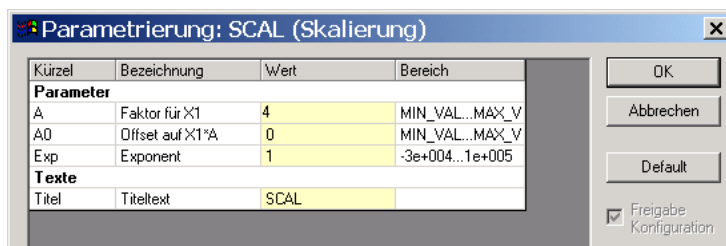


Fig. 755: Programmer project: Configuring a scaling block

6. **Changing over to program block "PROGRAMMER":** Click on the program block name in the tree structure to change over to program block "\$PROGRAMMER".
7. **Configuring a PROGRAMMER:** To determine a temperature profile, the programmer PROGRAMMER is used. Some parameters for the correct function and some additional parameters to increase the operating convenience must be determined:

Specify the number of the analog tracks as "1" and the number of the control tracks as "2".

The recipe directory as indicated is left unchanged: "PROGRAMMER"

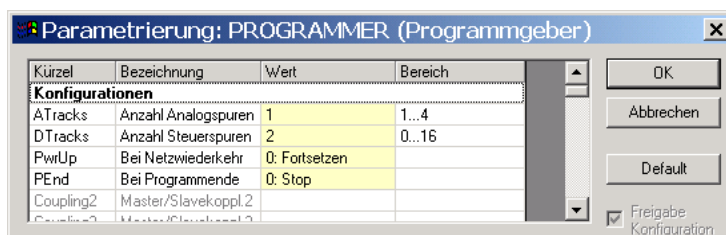


Fig. 756: Programmer project: Configuring a time constant for a filter

Define a name "Temperature" for the analogue tracks and "Ventilator" and "Door lock" for the control tracks.

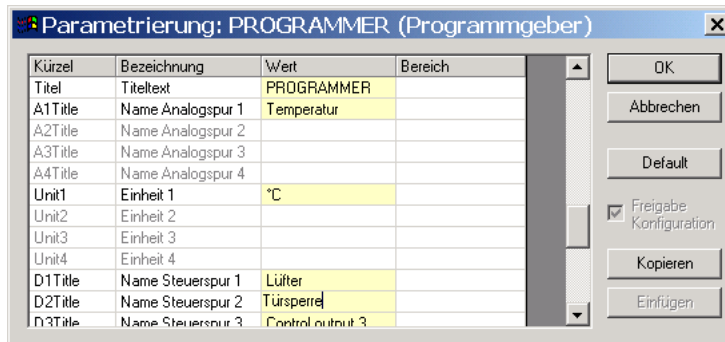


Fig. 757: Programmer project: Configuring a time constant for a filter

8. **Changing over to program block "PASS":** Click the program block name in the tree structure to change to program block "\$PASS".
9. **Configuring the access privilege:** The access to the application must be limited using different privileges. For this, use function block PASSWORD. The password operating page must be opened when exiting the program.

We use 3 of the 4 possible levels: e.g. operator, supervisor and engineer. Enter these names for levels 1 to 3 and set the passwords, for example, "11", "22" and "33":

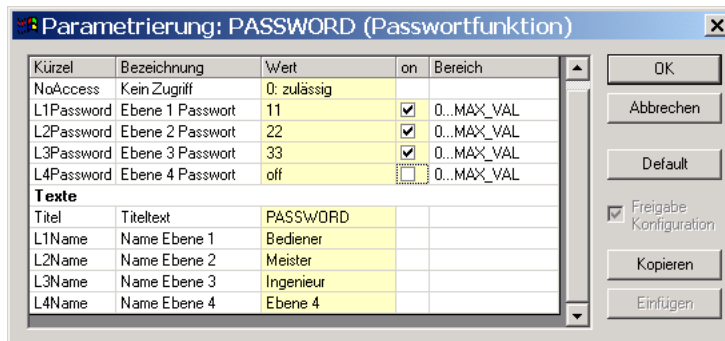


Fig. 758: Programmer project: Configuring the access privilege using function block PASSWORD

The operating page is opened via function blocks CALLPG, ALARM and AND. For this purpose, set the minimum alarm limit to "LimL" = 2. The PASSWORD operating page is invoked only with access level 2 (Supervisor).

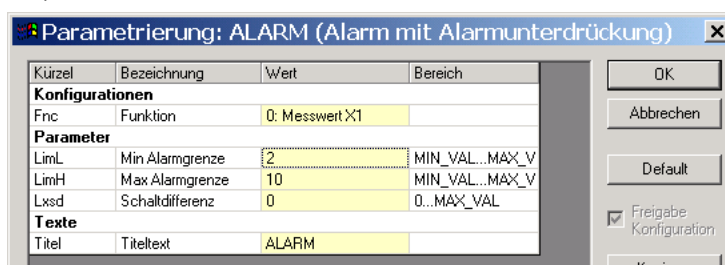


Fig. 759: Programmer project: Configuring a time constant for a filter

10. **Saving changes:** Click button "Save project" to save your changes.

III-24.8 Step 8: Generate a symbol file

In the previous steps, we have created an application using BlueDesign. For the program, we need a file providing the required data for recipe creation. This file is realized using a simple export function: Select item "Symbol file" in menu "Tools":

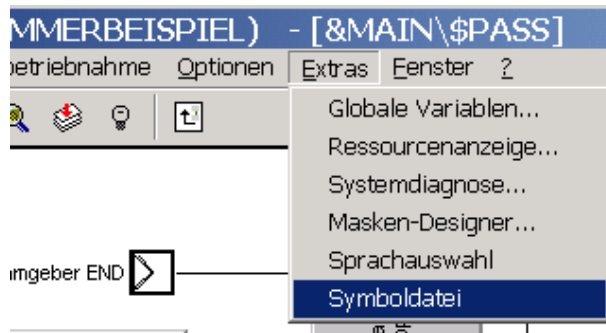


Fig. 760: Programmer project: Configuring a time constant for a filter

A window to specify the storage path for the file is opened. Click save to confirm:

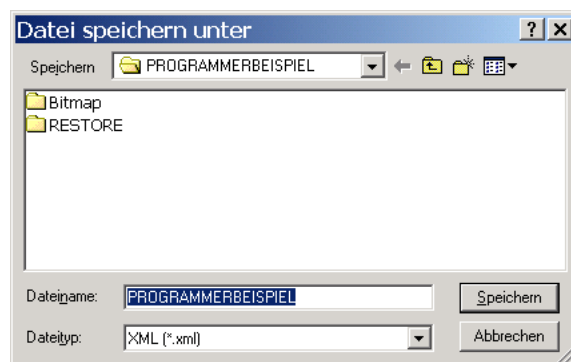


Fig. 761: Programmer project: Configuring a time constant for a filter

Now all preparations in BlueDesign for the application with a programmer were made.

III-24.9 Step 9: Creating a recipe

In the previous steps, we have created an application for control and simulation of the oven. We have also created the programmer and set up an access privilege. Only the recipe is still lacking.



NOTE!

Detailed information you find in the manual of BlueEdit 9499-040-91211.

1. **Step** Open the configurator of BlueEdit BlueEdit-Config from PMA Tools.
2. **Step** Create a new project "Tutorial programmer".
3. **Step** Blend in the toolbox <View><Toolbox> and drag an interface (Modbus on TCP/IP) from the toolbox and drop it onto the spreadsheet.
4. **Step** Click on the symbol of the interface (symbol is colored black) and drag a device (KS108 easy) from toolbox and drop it to the spreadsheet.
5. **Step** Clicking on the device brings in a table on the right to fill in the interface settings (IP address).
6. **Step** Drag a programmer from the toolbox and drop it onto the spreadsheet.
7. **Step** Search for the name of the symbol file (*.xml). Displayed is a list of the available program blocks of the engineering with the comprised function blocks.
8. **Step** Select the programmers (PROGRAMMER; double click on block name) from the XML file and confirm with "Execute". A dialog window opens for controlling the imported parameters of the programmer. Close the window with "Execute".
9. **Step** Save the project and close the BlueEdit configurator.
10. **Step** Open the program editor of BlueEdit from PMA Tools. The just created project is opened automatically.

- 11. Step** Create a new recipe (<File><New>) with recipe name and number.
- 12. Step** Fill in the settings of the tracks in the displayed table (segment name, segment type, segment time / gradient, segment end value and bandwidth) and store the recipe. For the example see figure below:

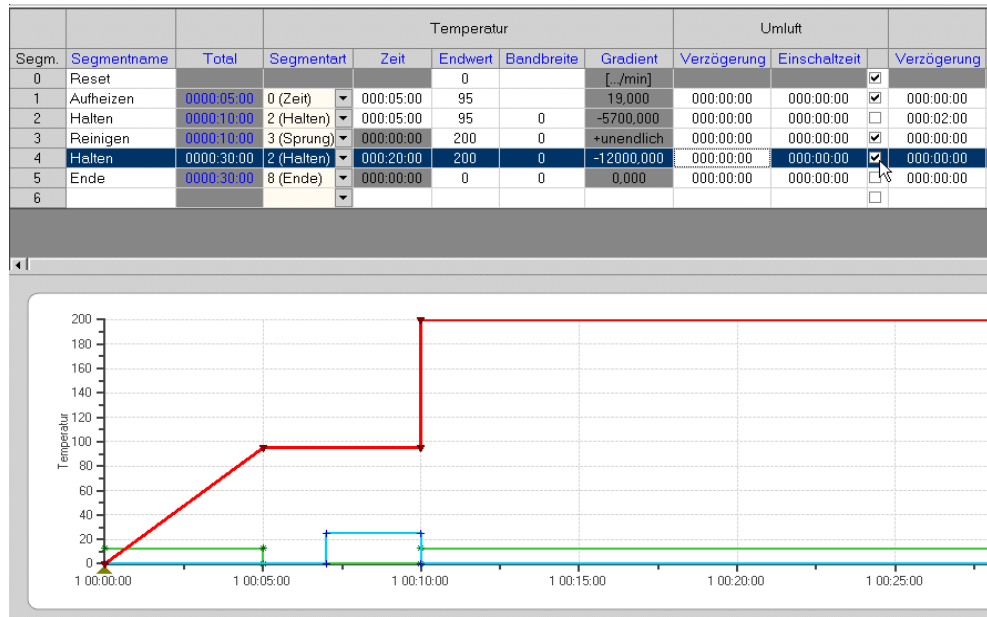


Abb. 762: recipe of example

- 13. Step** Send the recipe to KS 108 easy or the simulation. (For the example you send the recipe after loading the engineering into the simulation, see next step.

III-24.10 Step 10: An application test

Now it's time for testing the application. For this purpose, the *KS 108* simulation software will be used.

- Starting BlueSimulation:** Start the BlueSimulation application.
- Starting the Run mode:** Put your BlueDesign project in the Run mode. For this purpose, use the menu command "Run/Enter".

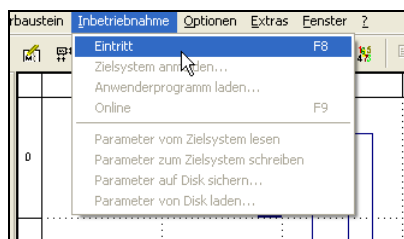


Fig. 763: Programmer project: Starting the Run mode

- Starting dialogue "Logon to Target System":** Now you have to transfer the application to the simulating software. For this, use menu command "Run/Download ...". Dialogue window "Logon to Target System" is displayed.



Fig. 764: Programmer project: Dialogue "Logon to Target system"

4. **Connecting to the target system:** If several target systems are displayed in the left part of the dialogue window, select entry "Simulation *KS108 easy*". Then click button "Connect".

Note: Additional settings are not required for working with the simulator.

After establishing the connection to the target system, information relating to the application is displayed in the dialogue window (name and size of the application, state, etc.).



Fig. 765: Programmer project: Dialogue "Target System" (2)

5. **Closing the dialogue:** Click button "OK" to finish the dialogue. The application is transferred. The transmission status messages are displayed on tab "General".

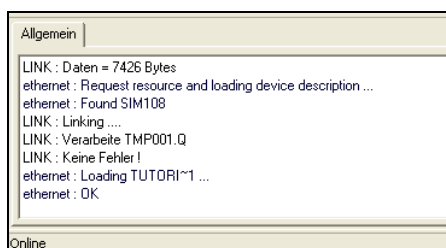


Fig. 766: Programmer project: Status messages

6. **Testing the application:** Now you can test the application using the "BlueSimulation" software.

IV Index

A

A_PROG .. 89, 96, 353, 395, 410, 411, 415, 417, 418, 421,
423, 424, 433, 434, 536, 537

Absolute value..... 227

Adder 192, 193, 200, 201

Alarm history 56, 57

Alarm page 44, 55, 56, 382

AND gate..... 225, 255

Application development 23, 54

B

Bargraph 44, 60, 69, 71, 99, 161, 164, 176, 213, 340, 359,
453, 463, 483, 503, 704

Batteries 23

Battery 107

Block diagram..... 31

C

CAN interface 27, 31, 33, 604, 605, 614, 617, 620, 625

CAN system..... 616, 619, 623, 627

Cascade 76, 318, 365, 520, 521, 536

Cascade control..... 76, 520, 536

Cleaning..... 109

CoDeSys..... 114

Commissioning 39

Communication protocol 26

Condensation of water..... 17

Constant selection..... 308

Control function..... 500

Control parameter 72, 85, 464, 484, 504, 537

Control track. 99, 100, 102, 104, 448, 453, 454, 456, 458,
694, 704, 705, 707, 709

Controller 71, 75, 76, 77, 78, 83, 199, 214, 216, 217, 460,
463, 483, 503, 520, 526, 527, 537, 538, 541, 608, 722,
729, 731, 733, 735

Controlling the setpoint 524

Copyright..... 16

Customer service..... 14

D

Data logger 61, 62, 376, 380

Data type conversion..... 271

Device failure..... 22

Differentiator..... 287

Dimensions 28

Display 12, 28, 45, 58, 59, 60, 65, 71, 72, 89, 97, 99, 101,
102, 138, 157, 158, 176, 177, 183, 342, 350, 359, 360,
364, 369, 399, 422, 423, 425, 426, 434, 435, 451, 453,
455, 456, 463, 464, 483, 484, 503, 504, 521, 702, 704,
706, 707

Disposal 17

E

EEPROM..... 313, 356, 361, 384, 628, 634, 637, 640, 643,
647, 652, 656, 659, 661, 663, 665, 667

e-function..... 235, 236, 293, 296

EMC 29

Environmental protection..... 24

Ethernet 26, 27, 29, 31, 32, 39, 43, 52, 53, 406, 408, 556,
557, 558, 559, 560

Exclusive OR gate..... 260

Explanation of symbols 15

Explosion protection..... 22

External setpoint 521, 523

F

Filter 191, 293, 296, 628, 634, 637, 640, 643, 647, 652,
667, 728

Function 12, 54, 89, 95, 107, 185, 191, 200, 225, 243,
271, 281, 283, 316, 339, 342, 344, 346, 350, 360, 366,
395, 396, 399, 402, 460, 462, 481, 482, 502, 552, 554,
712, 719, 723, 725, 728, 735

G

Global variables..... 145

H

HMI 160, 161, 164, 169, 170, 171, 201, 202, 203, 205,
212, 213, 214, 216, 217, 218, 335, 337, 339, 342, 344,
346, 349, 350, 682

I

installation.... 12, 14, 20, 35, 39, 115, 116, 117, 577, 579,
586, 588, 593, 596, 675, 676

Installation..... 12, 13, 35, 115, 116, 117, 675

Instructed person..... 21

Instruction..... 21

Integrator..... 290

Intended use..... 24

- Interfaces 26, 31, 39
- Inverter 256
- IP address..... 50, 148, 149, 172, 175, 387, 404, 556, 557, 558, 559, 560, 738
- L**
- Limitation of liability..... 16
- Limiting control 533
- Linearization function 240
- LINUX 30, 120, 677
- Logical functions..... 121, 182, 726
- M**
- Macro block 123, 140, 141, 142, 143, 144, 145, 158, 185, 220, 719
- Main menu..... 46, 395
- Main operating page 98, 452, 703
- maintenance 12, 14, 20, 22, 24, 40, 106, 107, 110, 111
- Manual.... 1, 71, 89, 95, 97, 102, 420, 423, 456, 526, 536, 537, 707
- Manual mode..... 71, 89, 95, 97, 102, 456, 707
- Mask Designer.... 160, 170, 202, 203, 206, 207, 213, 335, 337, 339, 342, 344, 346, 350
- Monoflop..... 264
- Mounting..... 28, 35, 36, 37, 556
- Mounting cutout 35, 36
- Multiplication / division 230
- N**
- Non-linear functions..... 12, 240
- O**
- Operating elements 121, 181
- operating instructions. 562, 564, 566, 567, 568, 569, 570, 571, 572, 574, 583, 607, 609, 611, 613
- Operating personnel 21
- Ores 72, 74, 82, 88, 464, 484, 504
- Owner..... 20
- P**
- Parameter selection..... 384
- PMA library 121, 122
- Power supply..... 32, 38, 112
- Practical example ... 12, 13, 198, 199, 200, 203, 204, 205, 207, 208, 218, 219, 220, 221
- Program block 123, 129, 131, 132, 138, 158, 185, 196, 220, 335, 718, 719, 731
- Program control element..... 182
- Program memory..... 28
- Program selection..... 433
- Programming..... 120, 122, 677, 712
- Q**
- Qualified electrician 21
- R**
- Ratio controller 528
- Recipe 89, 90, 91, 92, 95, 96, 99, 103, 104, 418, 422, 423, 425, 433, 437, 448, 449, 451, 453, 457, 458, 680, 695, 696, 702, 704, 708, 709, 712, 716
- Recipe name ... 89, 95, 103, 418, 422, 425, 433, 451, 457, 702, 708
- Recipes..... 92, 417, 422, 448, 449, 450, 677, 695, 701
- RS 232 27, 31, 34
- RS 485..... 27, 30, 31, 34
- S**
- Safety 12, 19, 22, 29, 521, 536
- Scaling .. 12, 192, 227, 233, 577, 579, 585, 588, 593, 596, 630, 644, 648, 649, 653, 729, 736
- SD card. 26, 27, 28, 30, 42, 56, 57, 61, 374, 376, 377, 380
- Segment.. 89, 95, 101, 104, 411, 417, 426, 427, 430, 442, 444, 445, 446, 448, 455, 458, 688, 689, 691, 692, 694, 706, 709
- Segment types 442, 688
- Self-optimization 71, 72, 73, 74, 78, 79, 81, 82, 83, 84, 86, 464, 484, 504
- Setpoint functions 521
- Specialized personnel 39
- Standard 26, 136, 163, 188, 208, 211, 212, 528, 529, 629, 656, 668, 669, 722, 724, 727
- Status function 397
- step controller..... 77, 83, 84
- Step function 266
- Storage..... 17
- T**
- Target system 740
- Technical data 28
- Time functions 287
- Touchscreen..... 28
- Tracking..... 524, 525, 538
- Transport..... 17
- Trend 44, 64, 65, 66, 67, 177, 369, 373
- Trigonometric functions 12, 245
- Troubleshooting..... 111
- Two-point controller..... 541
- Type plate 30

U

USB . 26, 27, 28, 29, 33, 39, 40, 41, 55, 56, 57, 61, 62, 63,
374, 376, 377, 380

User display 44, 59, 364

V

V_ALARM 44, 55, 381, 382

V_BAR 44, 60, 353, 355, 359, 395

V_DISPLAY 44, 58, 353, 360, 364, 365, 366, 367, 395

V_LOGGING 44, 61, 62, 368, 370, 374, 376, 377, 378,
379, 395

V_TREND 44, 64, 368, 369, 370, 372, 395

Value list 121, 139, 181

Vario 13, 115, 116, 554, 555, 556, 557, 561, 586, 589,
594, 597, 675