

TAC Vista



TAC Menta

Technical Manual

TAC Vista

TAC Menta

Technical Manual

Copyright © 2008 TAC AB. All rights reserved.

This document, as well as the product it refers to, is only intended for licensed users. TAC AB owns the copyright of this document and reserves the right to make changes, additions or deletions. TAC AB assumes no responsibility for possible mistakes or errors that might appear in this document.

Do not use the product for other purposes than those indicated in this document.

Only licensed users of the product and the document are permitted to use the document or any information therein. Distribution, disclosure, copying, storing or use of the product, the information or the illustrations in the document on the part of non-licensed users, in electronic or mechanical form, as a recording or by other means, including photo copying or information storage and retrieval systems, without the express written permission of TAC AB, will be regarded as a violation of copyright laws and is strictly prohibited.

Trademarks and registered trademarks are the property of their respective owners.

Contents

INTRODUCTION

| | | |
|----------|-------------------------------|-----------|
| 1 | Introduction | 17 |
| 1.1 | Structure | 17 |
| 1.2 | Prerequisites | 18 |
| 1.3 | Terminology | 19 |
| 1.4 | New in this Edition..... | 21 |
| 1.5 | Typographic Conventions | 22 |

GETTING STARTED

| | | |
|----------|--|-----------|
| 2 | Planning the Project | 25 |
| 2.1 | ACME Inc. | 25 |
| 2.2 | The Example | 26 |
| 2.2.1 | The Project Folder Structure | 27 |
| 2.2.2 | The Functional Description..... | 27 |
| 2.3 | Application Programming For a TAC Xenta Device..... | 30 |
| 2.3.1 | Organizing the FBD | 31 |
| 3 | Setting Up an Application | 33 |
| 3.1 | Specifying a TAC Xenta Device..... | 33 |
| 3.2 | Specifying an I/O Module | 35 |
| 3.3 | Naming the Application | 37 |
| 3.4 | Adding an Application Comment | 38 |
| 3.5 | Saving the Application | 39 |
| 4 | Creating a Function Block Diagram | 41 |
| 4.1 | Adding a Drawing Title | 42 |
| 4.2 | Adding an Analog Input (AI) Function Block | 43 |
| 4.3 | Configuring an Analog Input (AI) Function Block..... | 44 |
| 4.4 | Adding a Binary Expression Block..... | 47 |
| 4.5 | Connecting a Signal Between Two Blocks | 48 |
| 4.6 | Creating a Module..... | 51 |
| 4.7 | Simulating the Design..... | 53 |
| 5 | Using Macro Blocks | 57 |
| 5.1 | Changing the Library Folder Path..... | 57 |
| 5.2 | Saving a Diagram as a Macro Block..... | 59 |
| 5.3 | Loading a Macro Block..... | 60 |
| 6 | Connecting Macro Blocks | 65 |
| 6.1 | Connecting Signals Between Macro Blocks | 65 |

| | | |
|----------|--|-----------|
| 7 | Creating Hierarchical Structures | 71 |
| 7.1 | Creating a Hierarchical Function Block | 72 |
| 7.2 | Expanding an Hierarchical Function Block | 73 |
| 7.3 | Creating Hierarchical Function Block in Levels | 73 |
| 7.4 | Navigating in the Hierarchical Structure | 78 |

REFERENCE

| | | |
|-----------|--|------------|
| 8 | TAC Menta Programming Fundamentals | 81 |
| 8.1 | Function Phase | 82 |
| 8.2 | Design Phase | 82 |
| 8.2.1 | Point Identification and Allocation | 82 |
| 8.2.2 | Naming of Points and Alarms | 82 |
| 8.2.3 | Structuring the Function Block Diagram | 83 |
| 8.2.4 | Using Modules in the Application | 84 |
| 8.2.5 | Menus in the OP | 84 |
| 8.3 | Test Phase | 86 |
| 9 | TAC Menta Overview | 87 |
| 9.1 | The Operation Modes | 87 |
| 9.2 | System of Units | 87 |
| 9.3 | Allowed Characters in Name Strings | 88 |
| 9.4 | Unique Application Program ID | 88 |
| 9.5 | Marking Standard Applications/Controllers | 89 |
| 9.6 | Program Licenses | 89 |
| 9.7 | Starting TAC Menta | 89 |
| 9.8 | Defining the TAC Menta Settings | 90 |
| 10 | Graphical Programming | 93 |
| 10.1 | Function Block Diagrams | 93 |
| 11 | Signals | 95 |
| 11.1 | Signal Types | 95 |
| 11.2 | Signal Names | 95 |
| 11.3 | Public Signals | 96 |
| 11.4 | Modules in Signal Names | 96 |
| 11.5 | Signal Names for TAC Network Variables | 97 |
| 11.5.1 | Signals without a Module Part in the Name | 97 |
| 11.5.2 | Signals with a Module Part in the Name | 98 |
| 11.6 | Accessing Signals | 98 |
| 11.7 | Input and Output Signals | 99 |
| 11.7.1 | The Physical Terminal Option | 100 |
| 11.7.2 | The Network Variable Option | 100 |
| 11.7.3 | Finding the Address for a TAC Network Variable in the Database | 101 |
| 11.7.4 | Finding the Address for a Network Variable | 103 |
| 11.7.5 | The Online Device Option | 105 |
| 11.7.6 | The SNVT Option | 106 |
| 12 | The Mouse and Function Keys | 111 |
| 12.1 | The Mouse | 111 |
| 12.2 | The Function Keys | 112 |

| | | |
|-----------|--|------------|
| 13 | The Edit Mode | 115 |
| 13.1 | The Device Specification | 116 |
| 13.1.1 | Specifying the type of TAC Xenta Device | 116 |
| 13.1.2 | Setting the Characteristics For the XIF File | 117 |
| 13.1.3 | Adding a TAC Xenta I/O Module | 119 |
| 13.1.4 | Editing a TAC Xenta I/O Module | 120 |
| 13.1.5 | Removing a TAC Xenta I/O Module | 121 |
| 13.1.6 | Adding an STR Wall Module | 121 |
| 13.1.7 | Editing an STR Wall Module | 124 |
| 13.1.8 | Removing an STR wall module | 125 |
| 13.2 | The Program Specification | 126 |
| 13.2.1 | Naming the Application | 128 |
| 13.2.2 | Using the Abbreviated Name for the Application | 128 |
| 13.2.3 | Defining the Cycle Time for the Application | 129 |
| 13.3 | Adding Function Blocks to the Diagram | 130 |
| 13.3.1 | Adding a Simple Block | 130 |
| 13.3.2 | Adding an Operator | 131 |
| 13.3.3 | Adding an Expression Block | 132 |
| 13.3.4 | Adding a Test Probe | 133 |
| 13.3.5 | Duplicating an Existing Function Block | 134 |
| 13.3.6 | Moving a Function Block | 135 |
| 13.3.7 | Copying a Function Block | 136 |
| 13.3.8 | Deleting a Function Block | 136 |
| 13.4 | Configuring a Function Block | 137 |
| 13.4.1 | Using Constants for Parameters | 140 |
| 13.4.2 | Configuring TAC Vista Alarm Texts | 141 |
| 13.5 | Connections | 142 |
| 13.5.1 | Drawing a Connection Line From a Block Output to Input | 143 |
| 13.5.2 | Creating a Branch on a Connection Line | 144 |
| 13.5.3 | Drawing a Connection Line From a Node | 145 |
| 13.5.4 | Drawing a Connection Line From the Work Area | 145 |
| 13.5.5 | Removing all Connections to a Function Block | 145 |
| 13.5.6 | Removing a Connection from an Input | 145 |
| 13.5.7 | Removing a Connection from an Output | 146 |
| 13.5.8 | Changing a Connection to Another Input | 146 |
| 13.5.9 | Changing a Connection to Another Output | 147 |
| 13.5.10 | Deleting a Connection | 147 |
| 13.5.11 | Breaking a Connection at an Arbitrary Point | 148 |
| 13.5.12 | Breaking a Connection at an Input | 148 |
| 13.5.13 | Breaking a Connection at an Output | 149 |
| 13.5.14 | Moving a Node | 149 |
| 13.5.15 | Highlighting a Connection | 150 |
| 13.5.16 | Orthogonal Connections | 151 |
| 13.6 | Operations on Groups | 152 |
| 13.6.1 | Selecting a Group | 152 |
| 13.6.2 | Deselecting a Group | 153 |
| 13.6.3 | Centering the Selection | 153 |
| 13.6.4 | Moving a Group | 153 |
| 13.6.5 | Deleting a Group | 154 |
| 13.6.6 | Copying and Pasting a Group | 154 |

| | | |
|-----------|--|------------|
| 13.6.7 | Copying a Selection to the Clipboard..... | 155 |
| 13.6.8 | Disconnecting a Group..... | 155 |
| 13.6.9 | Printing the Selection..... | 155 |
| 13.6.10 | Editing the Module Name..... | 156 |
| 13.7 | Finding and Replacing Text Strings..... | 156 |
| 13.7.1 | Finding a String..... | 156 |
| 13.7.2 | Replacing a String..... | 159 |
| 13.8 | Macro Blocks..... | 162 |
| 13.8.1 | Saving a Macro Block..... | 162 |
| 13.8.2 | Loading a Macro Block..... | 164 |
| 13.8.3 | Using Macro Commands in Comment Blocks..... | 165 |
| 13.9 | Hierarchical Function Blocks (HFB)..... | 167 |
| 13.9.1 | Creating an empty HFB..... | 168 |
| 13.9.2 | Adding an HFB I/O block..... | 169 |
| 13.9.3 | Creating an HFB of Existing Function Blocks..... | 170 |
| 13.9.4 | Expanding an HFB..... | 171 |
| 13.9.5 | Compressing an HFB..... | 171 |
| 13.9.6 | Navigating in a Hierarchical Structure of HFBs..... | 171 |
| 13.9.7 | Printing an HFB..... | 172 |
| 13.10 | Using Constants..... | 173 |
| 13.10.1 | Public Constants..... | 173 |
| 13.10.2 | Accessing Constants..... | 174 |
| 13.10.3 | Adding a Constant..... | 175 |
| 13.10.4 | Editing a Constant..... | 175 |
| 13.10.5 | Removing a Constant..... | 176 |
| 13.10.6 | Removing Unused Constants..... | 176 |
| 13.11 | Using the I/O Configuration Table..... | 176 |
| 13.11.1 | Opening the I/O Configuration Table..... | 177 |
| 13.11.2 | Changing I/O Bindings Using the I/O Configuration Table..... | 177 |
| 13.12 | Using the Time Schedule Table..... | 177 |
| 13.12.1 | Opening the Time Schedule Table..... | 178 |
| 13.12.2 | Changing a Time Schedule using the Time Schedule Table..... | 178 |
| 13.13 | Using the Alarm Text Table..... | 178 |
| 13.13.1 | Opening the Alarm Text Table..... | 179 |
| 13.13.2 | Changing an Alarm Using the Alarm Text Table..... | 179 |
| 13.14 | Using Local Trend Logging in TAC Xenta..... | 180 |
| 13.14.1 | Selecting a Trend Log..... | 181 |
| 13.14.2 | Defining The Number of Trend Logs In a TAC Xenta Device..... | 182 |
| 13.14.3 | Configuring a Trend Log..... | 182 |
| 13.15 | Setting Date and Time..... | 187 |
| 13.16 | Undo..... | 188 |
| 13.17 | Using an Associated Text File..... | 188 |
| 14 | The Simulation Mode | 189 |
| 14.1 | Changing to Simulation Mode..... | 190 |
| 14.2 | Executing the Simulation..... | 191 |
| 14.2.1 | Starting a Simulation..... | 192 |
| 14.2.2 | Simulating One Cycle Only..... | 192 |
| 14.2.3 | Simulating a Defined Number of Cycles..... | 192 |
| 14.2.4 | Stopping the Simulation at a Limit..... | 193 |

| | | |
|-----------|--|------------|
| 14.2.5 | Resetting the Cycle Counter..... | 193 |
| 14.2.6 | Stopping the Simulation..... | 193 |
| 14.3 | Highlighting a Connection..... | 194 |
| 14.4 | Simulating Physical Inputs Manually..... | 195 |
| 14.4.1 | Changing a Physical Analog Input Manually..... | 195 |
| 14.4.2 | Changing a Physical Binary Input Manually..... | 196 |
| 14.5 | Simulating Physical Inputs Automatically..... | 196 |
| 14.5.1 | Generating an Analog Input Signal Automatically..... | 196 |
| 14.5.2 | Generating a Binary Input Signal Automatically..... | 198 |
| 14.6 | Modifying Signals During Simulation..... | 200 |
| 14.6.1 | Modifying an Analog Signal..... | 200 |
| 14.6.2 | Modifying a Binary Signal..... | 201 |
| 14.7 | Modifying Block Parameters During Simulation..... | 202 |
| 14.7.1 | Viewing Parameters in a Function Block..... | 202 |
| 14.7.2 | Changing the Value of a Constant..... | 203 |
| 14.7.3 | Changing a Time Schedule..... | 203 |
| 14.7.4 | Changing the Binding of an I/O Point..... | 204 |
| 14.7.5 | Changing an Alarm Text..... | 204 |
| 14.8 | Simulation Using Test Probes..... | 205 |
| 14.9 | Simulating Executable Files..... | 205 |
| 14.10 | Generating Executable Code..... | 207 |
| 14.11 | Trend Logging..... | 208 |
| 14.11.1 | Adding an Analog Signal to the Recorder..... | 208 |
| 14.11.2 | Adding a Binary Signal to the Recorder..... | 209 |
| 14.11.3 | Removing a Signal From the Recorder..... | 210 |
| 14.11.4 | Clearing the Recorder..... | 210 |
| 14.11.5 | Editing the Range of a Recorded value..... | 210 |
| 14.11.6 | Stopping at a Limit..... | 211 |
| 14.11.7 | Restarting the Recorder..... | 211 |
| 14.11.8 | Resetting the Recorder..... | 212 |
| 14.11.9 | Viewing Sampled Values..... | 212 |
| 14.11.10 | Scanning Sampled Values..... | 213 |
| 14.11.11 | Defining the Sampling Period..... | 213 |
| 15 | The Logger Tool | 215 |
| 15.1 | Opening the Logger..... | 216 |
| 15.2 | Adding a Diagram Window..... | 217 |
| 15.3 | Configuring a Diagram Window..... | 218 |
| 15.4 | Adding a Signal to the Diagram Window..... | 219 |
| 15.5 | Viewing the Diagrams..... | 221 |
| 15.5.1 | Viewing the Diagrams Tiled Horizontally..... | 221 |
| 15.5.2 | Scrolling a Graph Horizontally..... | 221 |
| 15.5.3 | The Default Scaling of a Graph..... | 222 |
| 15.5.4 | Zooming in a part of a diagram..... | 222 |
| 15.6 | Saving a Log File..... | 223 |
| 15.7 | Viewing a Log File..... | 224 |
| 15.8 | Printing a Logger Diagram..... | 224 |
| 15.9 | Clearing the Logger Diagrams..... | 225 |
| 15.10 | Leaving the Logger Tool..... | 225 |
| 16 | On-line Mode Functions | 227 |

| | | |
|-----------|---|------------|
| 16.1 | Configuring the Serial Communication..... | 227 |
| 16.2 | Entering the Online Operation Mode | 229 |
| 16.3 | Optimizing the Refresh of Signals | 230 |
| 16.4 | The Connected Mode Options..... | 230 |
| 16.4.1 | Downloading to a Xenta Device..... | 231 |
| 16.4.2 | Uploading From the Xenta Device..... | 233 |
| 16.4.3 | Connecting To the Xenta Device | 234 |
| 16.5 | Execution Control in the Online Mode | 235 |
| 16.6 | Overriding a Physical I/O Signal..... | 236 |
| 16.7 | Modifying Parameter Value Blocks in the Online Mode | 238 |
| 16.8 | Modifying Public Constants in Online Mode..... | 239 |
| 16.8.1 | Turning Off the Updating of Public Constants..... | 239 |
| 16.9 | Modifying Binding Parameters in Online Mode | 239 |
| 17 | Memory Usage | 241 |
| 17.1 | Viewing the Calculated Memory Usage..... | 241 |
| 17.1.1 | Viewing the Advanced Memory Usage | 244 |
| 18 | Printing the Application Program Documentation | 247 |
| 18.1 | Setting up the TAC Menta Printout..... | 247 |
| 18.2 | Using Boundary Ties | 250 |
| 18.3 | Printing the TAC Menta documentation | 250 |
| 19 | The OP Configuration Tool | 255 |
| 19.1 | The Menu Structure Display | 257 |
| 19.2 | Creating the Menu Structure | 258 |
| 19.2.1 | Adding Menu Items | 258 |
| 19.2.2 | Sub Menu..... | 258 |
| 19.2.3 | Status | 259 |
| 19.2.4 | Alarm | 260 |
| 19.2.5 | Access Code | 260 |
| 19.2.6 | Date and Time/Daylight Saving | 261 |
| 19.2.7 | Week/Holiday Chart..... | 261 |
| 19.2.8 | TAC Service Menu..... | 261 |
| 19.3 | Editing an Existing Menu Structure Tree | 262 |
| 19.3.1 | Moving Menu Items | 262 |
| 19.3.2 | Changing the Operator Panel Display Layout..... | 262 |
| 19.3.3 | Copying and Pasting..... | 262 |
| 19.4 | Automatic Generation of an OP Menu Tree..... | 262 |
| 19.5 | OP Description Files..... | 264 |
| 19.5.1 | Data and Declarations Syntax in OP Description Files (.DOP) | 264 |
| 19.5.2 | OP Description File Example..... | 266 |
| 19.5.3 | Importing the OP Description File | 266 |
| 19.5.4 | Exporting the OP Description File | 267 |
| 19.6 | Defining a Character Set File | 267 |
| 19.7 | Menu Options | 268 |
| 19.7.1 | Menu Bar..... | 268 |
| 20 | The Download Wizard | 273 |
| 20.1 | How to Use the Download Wizard..... | 274 |
| 20.1.1 | The Dialog | 274 |
| 20.1.2 | The General Download Procedure | 275 |

| | | |
|-----------|---|------------|
| 20.2 | TAC Menta v3 Compatibility | 276 |
| 20.2.1 | .AUT Files..... | 276 |
| 20.2.2 | .COD Files..... | 276 |
| 20.2.3 | OP Menu Tree Files | 277 |
| 20.3 | Upgrading a TAC Xenta 300 Device to v3 | 277 |
| 20.3.1 | Updating the .AUT File with Application Data (Optional) | 277 |
| 20.3.2 | Upgrading System and Application Programs | 277 |
| 20.3.3 | Verifying Correct Operation (Optional But Recommended)..... | 278 |
| 21 | Simple Blocks | 279 |
| 21.1 | Simple Blocks, Overview..... | 279 |
| 21.1.1 | Connection Blocks | 280 |
| 21.1.2 | Physical I/O Blocks..... | 280 |
| 21.1.3 | Signal Sources | 281 |
| 21.1.4 | Logical Functions..... | 281 |
| 21.1.5 | Non-linear Functions..... | 282 |
| 21.1.6 | Delay Blocks | 282 |
| 21.1.7 | Controllers and Filters..... | 282 |
| 21.1.8 | Accumulators | 283 |
| 21.1.9 | System Variables..... | 283 |
| 21.1.10 | Time Schedules and Alarms..... | 283 |
| 21.1.11 | Transformation Functions | 284 |
| 21.2 | Simple Block Concepts | 284 |
| 21.3 | ACCUM – Accumulator | 285 |
| 21.4 | AHYST – Analog Hysteresis | 286 |
| 21.5 | AI – Analog Input | 287 |
| 21.6 | ALARM – Alarm | 293 |
| 21.7 | AND – Logical AND Gate..... | 295 |
| 21.8 | AO – Analog Output | 296 |
| 21.9 | BI – Binary Input | 299 |
| 21.10 | BO – Binary Output | 299 |
| 21.11 | CNT – Digital Input – Pulse Counter..... | 300 |
| 21.12 | CURVE – Curve Function | 301 |
| 21.13 | DATE – Day | 302 |
| 21.14 | DELAY – Delayed On/Off | 303 |
| 21.15 | DELB – Binary Value Delay | 304 |
| 21.16 | DELI – Integer Value Delay | 304 |
| 21.17 | DELR – Real Value Delay | 305 |
| 21.18 | DI – Digital Input | 305 |
| 21.19 | DO – Digital Output..... | 310 |
| 21.20 | DOPU – Digital Pulse Output..... | 312 |
| 21.21 | ENTH – Enthalpy..... | 313 |
| 21.22 | ERR – System Error..... | 316 |
| 21.22.1 | Error Codes for the ERR Function Block | 317 |
| 21.23 | ERROR – System Error in Xenta 700..... | 318 |
| 21.23.1 | Error Codes for the ERROR Function Block..... | 319 |
| 21.24 | FILT – First Order Filter | 320 |
| 21.25 | HOUR – Hour | 321 |
| 21.26 | HYST – Binary Hysteresis..... | 321 |
| 21.27 | INTEG – Integrator | 323 |

| | | |
|-----------|--|------------|
| 21.28 | II – Integer Input..... | 324 |
| 21.29 | IO – Integer Output | 325 |
| 21.30 | LIMIT – High/Low Signal Limit | 325 |
| 21.31 | MAX – Maximum Signal Selector..... | 326 |
| 21.32 | MIN – Minimum Signal Selector..... | 326 |
| 21.33 | MINUTE – Minute..... | 327 |
| 21.34 | MONTH – Month..... | 327 |
| 21.35 | NCYC – Program Cycle Counter | 328 |
| 21.36 | NOT – NOT Gate | 328 |
| 21.37 | OPT – Optimization | 329 |
| 21.38 | OR – OR Gate | 336 |
| 21.39 | OSC – Oscillator..... | 337 |
| 21.40 | PI – Pulse-Counter Binary Input | 338 |
| 21.41 | PO – Pulsed Binary Output | 339 |
| 21.42 | PIDA – PID Controller – Analog Output | 340 |
| 21.43 | PIDI – PID Controller – Incremental Output | 343 |
| 21.44 | PIDP – PID Controller – Analog Output..... | 346 |
| 21.45 | POLY – Polynomial Function | 349 |
| 21.46 | PRCNT – Percentage..... | 350 |
| 21.47 | PULSE – Pulse Generator | 351 |
| 21.48 | PVB – Binary Value Parameter..... | 352 |
| 21.49 | PVI – Integer Value Parameter..... | 352 |
| 21.50 | PVR – Real Value Parameter | 354 |
| 21.51 | RAMP – Ramp Filter..... | 354 |
| 21.52 | RI – Real Input | 355 |
| 21.53 | RO – Real Output | 355 |
| 21.54 | RST – Restart | 356 |
| 21.55 | RT – Run-Time Measurement..... | 357 |
| 21.56 | SECOND – Second | 358 |
| 21.57 | SEQ – Sequencer..... | 359 |
| 21.58 | SHB – Sample and Hold Binary Value | 361 |
| 21.59 | SHI – Sample and Hold Integer Value | 363 |
| 21.60 | SHR – Sample and Hold Real Value..... | 364 |
| 21.61 | SR – Set-Reset Flip-Flop..... | 365 |
| 21.62 | STRIN – STR Input..... | 367 |
| 21.63 | STROUT – STR Output | 368 |
| 21.64 | TCYC – Cycle Time..... | 369 |
| 21.65 | TRIG – Trigger..... | 370 |
| 21.66 | TSCH – Time Schedule..... | 370 |
| 21.67 | TSCHI – Time Schedule Block in Xenta 700 | 373 |
| 21.68 | VECTOR – Vectorial Curve Function | 374 |
| 21.69 | WDAY – Week Day..... | 375 |
| 21.70 | XOR – Exclusive OR Gate..... | 375 |
| 22 | Expression Blocks | 377 |
| 22.1 | Operands..... | 378 |
| 22.2 | Operators | 379 |
| 22.3 | Aritmethical Functions | 379 |
| 22.4 | Output..... | 380 |
| 23 | Operators | 381 |

| | | |
|-----------|--|------------|
| 23.1 | Constants | 382 |
| 23.2 | Logical Operators | 382 |
| 23.3 | Math Operators | 383 |
| 23.4 | Comparison Operators | 383 |
| 23.5 | Bit Operation Operators | 383 |
| 23.6 | Other Operators | 385 |
| 24 | Test Probe Blocks | 387 |
| 24.1 | Overview | 387 |
| 24.2 | TPAI – Test Probe for Analog Input | 388 |
| 24.3 | TPAO – Test Probe for Analog Output | 389 |
| 24.4 | TPDI – Test Probe for Digital Input | 389 |
| 24.5 | TPDO – Test Probe for Digital Output | 390 |
| 25 | The Menta Application File | 391 |
| 25.1 | Data Files | 391 |
| 25.2 | Renaming a TAC Menta application File | 393 |
| 26 | Error Messages | 395 |
| 26.1 | System Errors | 395 |
| 26.2 | FBD Compilation | 395 |
| 26.3 | Saving and Loading the Application Program in the Database | 397 |
| 26.4 | Simulation | 398 |
| 26.5 | Code Generation | 398 |
| 26.6 | Download | 399 |
| 26.6.1 | Error Codes | 399 |
| 26.7 | TAC Xenta Communication | 400 |
| 27 | Programming Hints | 401 |
| 27.1 | Program Cycle Time | 401 |
| 27.2 | Time Counter | 401 |
| 27.3 | Equality | 401 |
| 27.4 | Reset Counter | 402 |
| 27.5 | Bitwise Logical Operators | 402 |
| 27.6 | Multiplier Parameter in CNT Block | 402 |
| 27.7 | Sliding Average Value | 402 |
| 27.8 | TSCH Output | 403 |
| 27.9 | PIDI – DOPU | 403 |
| 27.10 | Day Shift | 403 |
| 27.11 | Expression Blocks | 404 |
| 27.12 | Start-Up Delay | 404 |
| 27.13 | Using the SNVT_reg_val in a TAC Xenta 700 Device | 405 |
| | Index | 409 |

INTRODUCTION

1 Introduction

1 Introduction

This manual describes a particular process. For information of specific products we refer to the manual of the product in question.

For information on how to install software, please refer to the instructions delivered with the software.

For information on specific products, please refer to the manual for the product in question.

If you discover errors and/or unclear descriptions in this manual, please contact your TAC representative.



Note

- We are continuously improving and correcting our documentation. This manual may have been updated.

Please check ExchangeOnline at <http://extranet.tac.com> for the latest version.

1.1 Structure

This handbook is divided into the following parts:

Introduction

The Introduction part contains information on how this handbook is structured and how it should be used to find information in the most efficient way.

Getting Started

The Getting Started section contains a step-by-step description of how to engineer or carry out different tasks. It also gives you guided instructions on how to complete a sample project. If you want more information, see the corresponding chapter in the Reference section of the manual.

Reference

The Reference part contains more easily comprehensible information about various parts of the Menta programming tool. It also gives you information on alternative solutions not covered in the Getting Started section.

1.2 Prerequisites

To be able to profit from the contents in this manual, you are recommended to read the following manuals:

- *TAC Xenta 401/301/280, Product Manual*
- *TAC Xenta OP, Operating Manual*

1.3 Terminology

| Term | Description |
|-------------------|---|
| TAC Xenta Devices | <p>All programmable TAC Xentas, 280/300/401, will be called Xenta devices throughout this manual.</p> <p>The Xenta 700 range will also be referred to as Xenta devices.</p> <p>The proper names are used when the Xenta 511/911 and the Xenta 901 are mentioned.</p> <p>The Xenta 422, 452 etc. will be referred to as I/O modules.</p> |
| LonWorks Devices | <p>All other devices will be called LonWorks devices, including the Xenta 100.</p> |
| Classic Network | <p>Classic Network refers to a TAC Vista system with a LonWorks network, TAC Xenta devices and/or LonWorks devices, using an LTA port connection/communication to the network.</p> <p>A Classic Network does NOT use an LNS database or SNVT bindings.</p> |
| LNS Network | <p>LNS Network refers to a TAC Vista system with a LonWorks network, TAC Xenta devices and/or LonWorks devices, using an LTA port with a VNI as the Network Interface (NI) application, and an LNS database.</p> <p>This type of LTA port is referred to as an LNS port in TAC Vista.</p> |
| OP | <p>Optional operator panel, used with Xenta 280/300/401 devices.</p> |
| LonWorks™ | <p>The standardized network, used for communication between the TAC Xenta devices.</p> |
| TAC Vista | <p>A PC based operator unit for the monitoring and control of air handling and heating systems.</p> |
| TAC Menta | <p>Application programming tool for TAC Xenta.</p> |
| OP configuration | <p>Programming tool for the TAC Xenta Operator panel optionally used with Xenta 280/300/401, included in TAC Menta.</p> |
| ASCII | <p>American Standard Code for Information Interchange.</p> |
| BMS | <p>Building Management System</p> |

| Term | Description |
|-----------------------|--|
| HVAC | Heating, Ventilation & Air Conditioning |
| ID | Plant specific names/descriptors of points etc. |
| I/O | Input/Output |
| B | Designation for a physical terminal in a TAC Xenta device, used for thermistor type inputs. |
| Cold Start | Restart after a power outage with a duration longer than the power failure protection time for the device, that is when the contents of the RAM in the TAC Xenta are no longer reliable. |
| FB | Function Block |
| FBD | Function Block Diagram. |
| OP Configuration Tool | Programming tool for the TAC Xenta Operator panel optionally used with the Xenta 280/300/401, included in TAC Menta. |
| OP | Operator Panel for the TAC Xenta controller. |
| SNVT | Standard Network Variable Type, which enables LonWorks™ network communication between nodes from different manufacturers. |
| TAC Menta | Application programming tool for TAC Xenta controllers. |
| TAC Xenta | A family of application specific standard controllers (TAC Xenta 3000) and freely programmable controllers (TAC Xenta 280/300/401/700) with a modular I/O configuration. |
| U | Designation for a physical terminal in a TAC Xenta device, used for universal type inputs. |
| Warm Start | Restart after a power outage with a duration of less than specified for the TAC Xenta device, that is when the RAM content of the device still is reliable. |
| X | Designation for a physical terminal in a TAC Xenta device, used for digital type inputs. |
| Y | Designation for a physical terminal in a TAC Xenta device, used for analog type outputs. |
| K | Designation for a physical terminal in a TAC Xenta device, used for relay type outputs. |

1.4 New in this Edition

The reference part of the manual is re-structured with the following main changes:

- The chapter "TAC Menta Programming Fundamentals" is moved to the reference part of the manual.
- The chapter "Introduction to TAC Menta" is removed. The revised content of the chapter has been distributed to the separate chapters.
- The chapter "General Concepts" is removed. The revised content of the chapter has been distributed to the separate chapters.
- The chapter "Function blocks, summary" is removed. The revised content of the chapter has been moved to a function blocks chapter.
- New chapters, including moved and revised contents, are made for the:
 - Function block diagrams
 - Signals
 - The Logger tool
 - On-line mode functions
 - Memory usage
 - Printing an application
 - The Menta Application file
- The chapter listing the supported SNVTs is removed from the manual.

In addition to this, several portions of texts throughout the reference part of the manual, are revised.

Only minor changes are made in the chapters covering the function blocks, Simple Blocks, Expression Blocks and Operators.

The chapters covering the OP Configuration tool and the Download Wizard are left unchanged.

1.5 Typographic Conventions

Throughout the manual the following specially marked texts may occur.



Warning

- Alerts you that failure to take, or avoid, a specific action might result in physical harm to you or to the hardware.



Caution

- Alerts you to possible data loss, breaches of security, or other more serious problems.



Important

- Alerts you to supplementary information that is essential to the completion of a task.



Note

- Alerts you to supplementary information.



Tip

- Alerts you to supplementary information that is not essential to the completion of the task at hand.

GETTING STARTED

- 2 Planning the Project
- 3 Setting Up an Application
- 4 Creating a Function Block Diagram
- 5 Using Macro Blocks
- 6 Connecting Macro Blocks
- 7 Creating Hierarchical Structures

2 Planning the Project

We are going to create an application for a Xenta 401 with five I/O modules for a fictional company, ACME Inc.



Important

The workflow described in this manual is valid for Xenta 401/300/280. For more information on the workflow for a Xenta 700 application, see *TAC Xenta Server - Controller, Technical Manual*.

2.1 ACME Inc.

The facility is a small, two-story office building, served by rooftop units. The first floor houses the Lobby, Accounts, Conference Room, and Marketing/Management. The second floor area houses Customer Support and Engineering. The system is managed using TAC Vista.

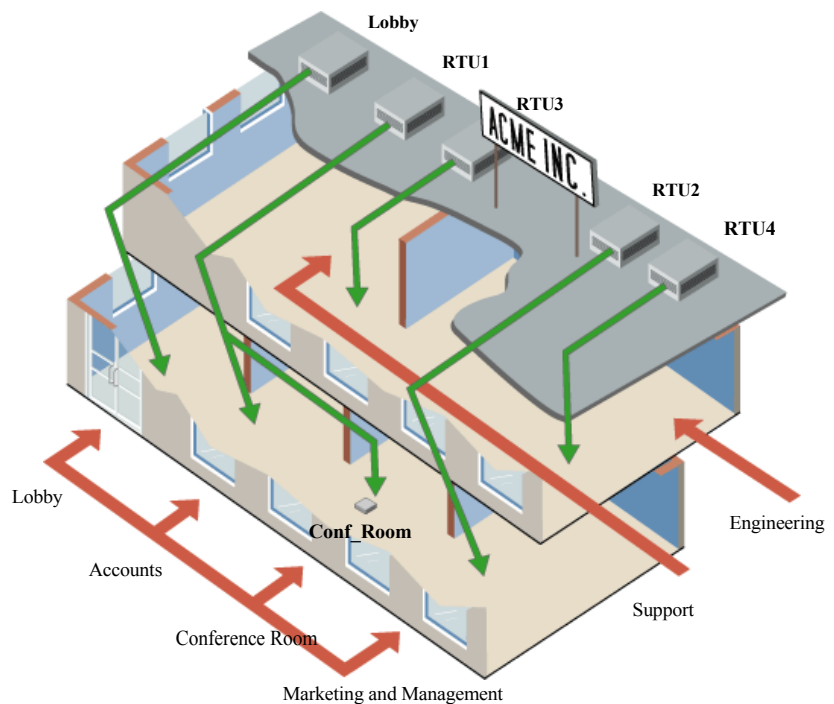


Fig. 2.1: The ACME Building.

2.2 The Example

In the example, we create an application for the Xenta 401 controlling the roof top unit RTU4 that serves six VAV (Variable Air Volume) units.

The Xenta 401 is called RTU4. Five I/O modules are required to control the roof top unit and the six VAV units. Two Xenta 422 and three Xenta 452. The I/O modules are called M1, M2, M3, M4, and M5.

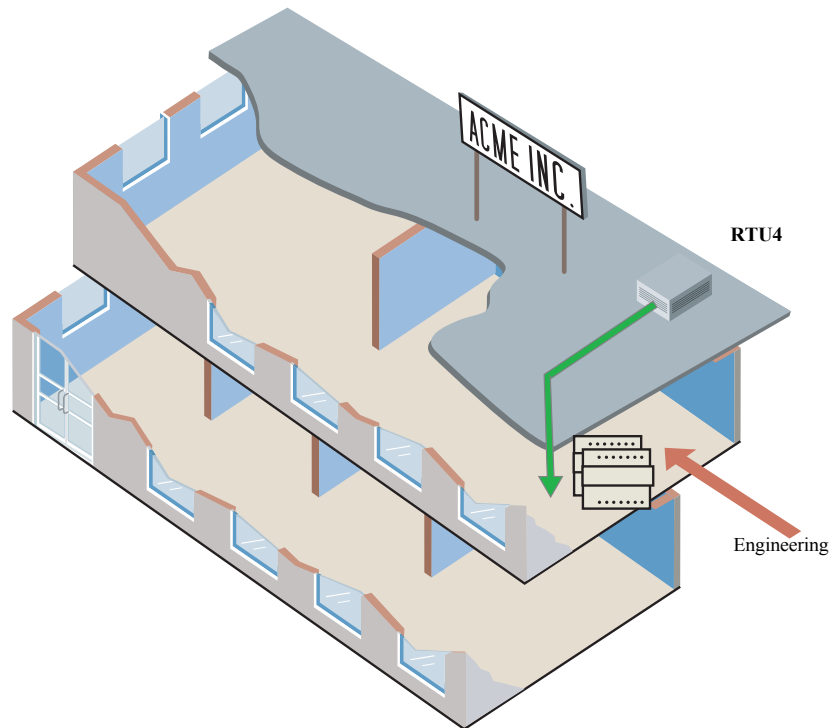


Fig. 2.2: RTU4



Important

The workflow described in this manual is valid for Xenta 401/300/280. For more information on the workflow for a Xenta 700 application, see *TAC Xenta Server - Controller, Technical Manual*.

2.2.1 The Project Folder Structure

A project for a complete system is best placed in a directory containing the folders and subfolders similar to the figure below.

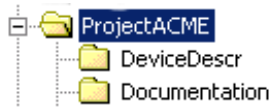


Fig. 2.3: The folder structure on the hard disk

A brief description follows of the intended use for the folders and their intended content:

- DeviceDescr – .mta files and .xif files for the LonWorks devices.
- Documentation – general information, for example, manuals, data sheets, functional descriptions, I/O lists and so on.

2.2.2 The Functional Description

To be able to create the application you need to know the required functionality of the Xenta controller. In the example, the required functionality of the Xenta 401 (RTU4) is as follows:

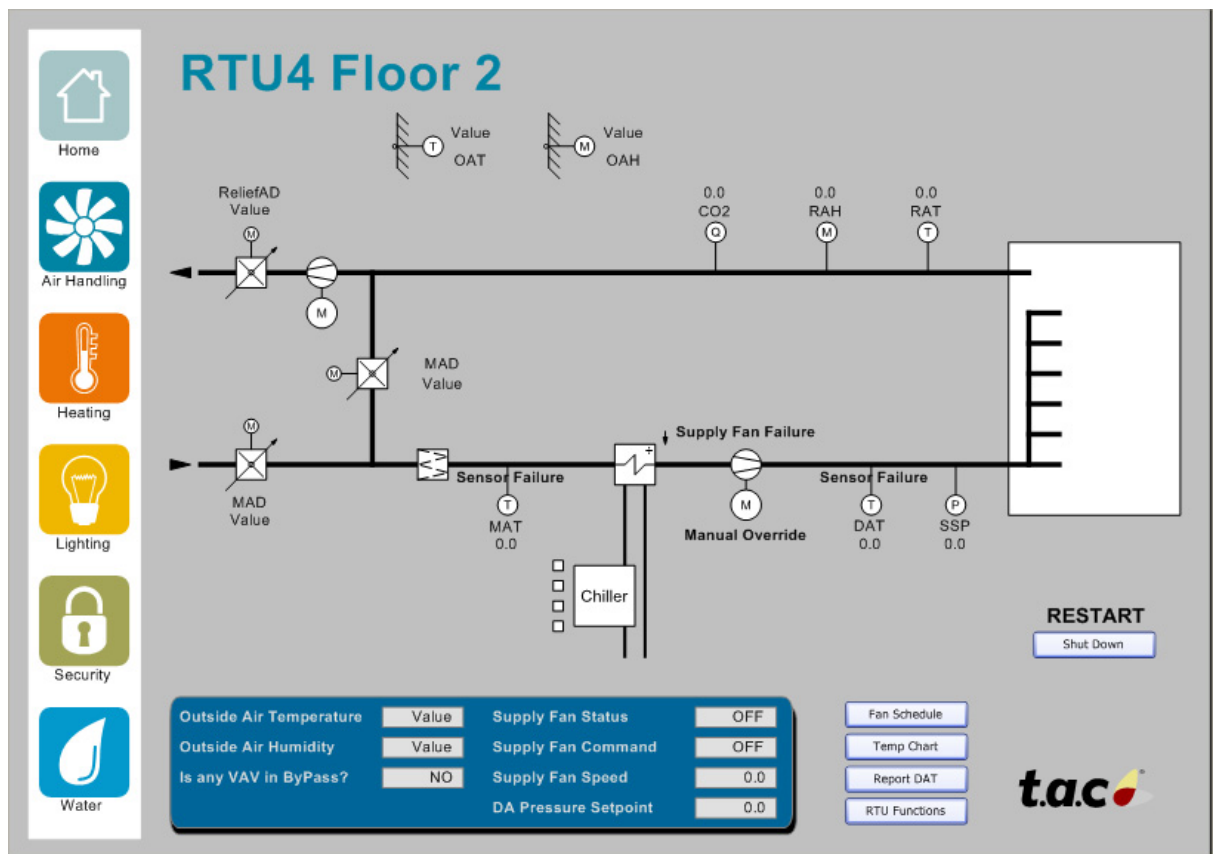


Fig. 2.4: Flow diagram

Abbreviations

| | |
|----------------------------------|----------------------------------|
| OAT = Outside air temperature | SF = Supply fan |
| OAH = Outside air humidity | RF = Relief fan |
| RAT = Return air temperature | RAD = Return air dampers |
| CO2 = Return air carbon dioxide | OAD = Outside air dampers |
| DAT = Discharge air temperature | C1–C4 = Compressor 1–4 |
| SSP = Supply air static pressure | VAV 4:1–6 = Zone controllers 1–6 |
| BSP = Building static pressure | MAT = Mixed air temperature |
| MAD = Mixed air damper | RAH = Return air humidity |

Fan Start/Stop

The air handling unit is to be started on the basis of the time schedule or whenever any zone is in bypass mode.

The unit can also be started in Timed Override mode by the operator via a software timer with an adjustable time. Additional commands before the timer has expired stops the Timed Override.

The operator can manually override the mode using the HMI or OP. The operator can select Auto, Off, or Manual running. An alarm will be issued when the unit is manually overridden.

A proofing function is to be provided with a time to state of 30 seconds. If the fan state input is in the wrong state for more than 30 seconds, the request for the fan is to be latched off and an alarm issued. The Alarm latch is to be reset whenever an off command is issued.

The start is to be delayed for an adjustable time following a power restore.

If the fan is running without the start relay being energized, there will be an alarm.

When the fan is started and the system is busy, as determined by the time schedule, an occupancy command is to be sent to all zone VAV terminals.

Fan Variable Speed Device Control

Supply air static pressure is to be controlled using a VSD (variable speed device). When the static pressure is above the set point, plus half the deadband, the signal to the VSD is to be reduced by the bump amount each sample. If the static pressure exceeds twice the setpoint, the signal to the VSD is to be reduced by 3 times the bump amount each sample. When the static pressure falls below the set point, minus half the deadband, the signal to the VSD is to be increased by 2% once each sample. When the static pressure is in the deadband, the signal to the VSD is not to be changed.

The control signal to the VSD has to be an adjustable startup value during a defined startup time period. The signal to the VSD is to be set to zero when the fan is off.

If communication with the pressure sensor's I/O Module is lost or if the pressure sensor fails, the VSD is to be run at startup speed.

Relief Fan and Damper Control

When the SFan is running, the Relief Damper modulates to maintain the static pressure of the building at setpoint. If the damper is 100% open and the static pressure exceeds 0.06 "wc for more than 10 minutes, the Relief Fan will start. When the Relief Damper is modulated to less than 40% and the static pressure of the building falls below -0.01 "wc, the Relief Fan will stop.

If there is a communication failure with I/O extension module 3, the control signal calculated for the VSD will be used as control value for the Relief Damper.

When the SFan is running, normal control is used. Otherwise the controller output is zero. If there is a static pressure sensor alarm, the controller output will be 100 percent.

Economizer Mode Calculations

The economizer mode can be applied to the outside air and the return air dampers. Calculations regarding enthalpy are to be made based on temperature and humidity measurements of the outside air and the return air. Offsets for calibrating the sensors are to be used. Alarms, representing sensor failures, are to be provided. Alarms for high and low Return air temperatures are also to be provided.

When the outside air enthalpy is -2 BTU/lb below the return air enthalpy and the fan is running, the economizer mode is to be enabled. The economizer mode is to be disabled if the unit supplying outside air enthalpy measurement is offline. Economizer mode is to be disabled when the outside air enthalpy rises above 0.5 BTU/lb below the return air enthalpy.

OA/RA Damper Control

When the fan is running and the economizer mode is disabled, a minimum position is to be imposed on the mixed air dampers. When the economizer mode is enabled, the OA/RA dampers are to modulate in order to maintain the mixed air temperature at setpoint. The opening position of the damper(s) must be limited using an operator's setting with a minimum value for the opening degree of the dampers. When the unit is off, the dampers must close.

DX Cooling Control

Four chillers are to be started/stopped in sequence and adjusted to the actual cooling demand, where the cooling demand is a function of the

supply fan (VSD) speed, adjusted with regard to the discharge air temperature and the maximum terminal load.

Each chiller stage is to have a defined minimum on and a minimum off time to be in service. There must also be a delay between the start of each chiller stage.

The chiller stages are to be started and stopped on the demand level using hysteresis.

The first chiller stage is to be started and stopped on the demand level, and the following three chiller stages may only start when Economizer mode is off.

Separate alarms are to be provided for each chiller stage.

Zone Values for Presentation and Alarms.

The difference between the Effective Setpoint and the actual Space Temperature for zones VAV_4:1, VAV_4:2, VAV_4:3, VAV_4:4, VAV_4:5, and VAV_4:6 is to be calculated for graphic display and alarms when levels exceed their limits.

I/O Alarms

There must be alarms when any expansion I/O module is offline and when an I/O is manually forced.

Chiller energy consumption

Data for the chiller compressor's energy consumption is to be totalled up and stored on both an hourly and a daily basis.

Trend Log

A Trend Log of the Discharge Air Temperature is to be defined.

OP menu tree

An automatically generated menu tree for the Operators panel is to be generated.

2.3 Application Programming For a TAC Xenta Device

The work with programming the application for a TAC Xenta device can be divided into a number of phases:

- Function phase.
- Design phase.
- Test phase.

For more information on application programming phases, see Section 8, "TAC Menta Programming Fundamentals"

2.3.1 Organizing the FBD

During the design phase we want to design blocks with the different functionalities, both together and in separate modules, using the Function Block Diagram (FBD).

If possible, we also want them to appear on one printable page each.

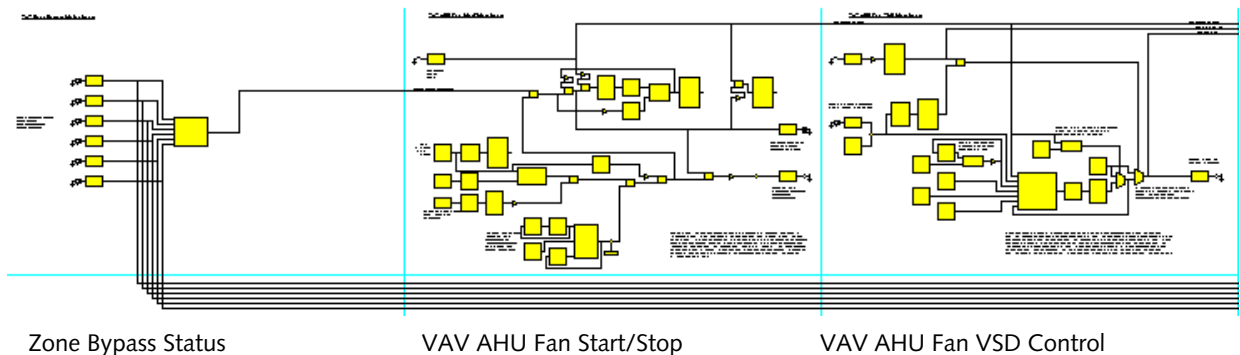
For more information on structuring the function block diagram, see Section 8.2.3, “Structuring the Function Block Diagram”

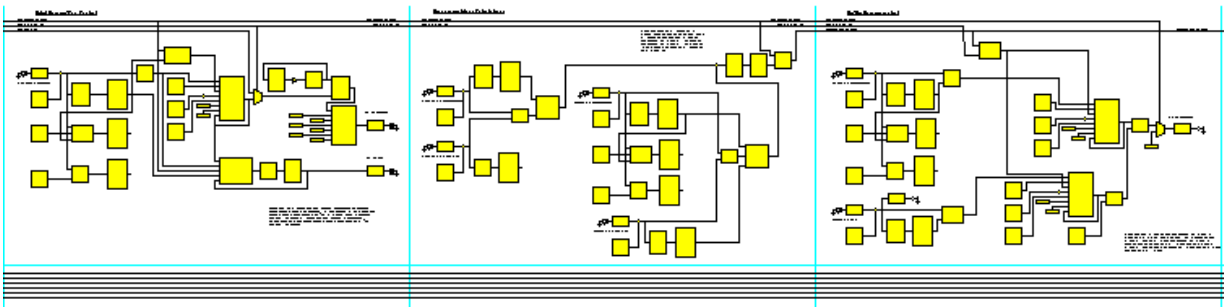
In the example, we can identify the major parts of the application:

- Zone Bypass state for Terminal Units
- VAV AHU Fan Start Stop.
- VAV AHU Fan VSD Control.
- Relief Damper/Fan control.
- Economizer Calculations.
- Outside/Return Air Damper Control.
- DX Cooling Control.
- Calculation of Zone Values for Display and Alarm.
- I/O Alarms

In the Function Block Diagram, we will create separate modules for these. In doing so, we will end up with an FBD of ten horizontal and two vertical printable pages. The lower five pages will only be used for showing "a bus like collection" of signals. We will try to place the inputs to the left and the outputs to the right on each page.

From left to right, the pages will display the following:

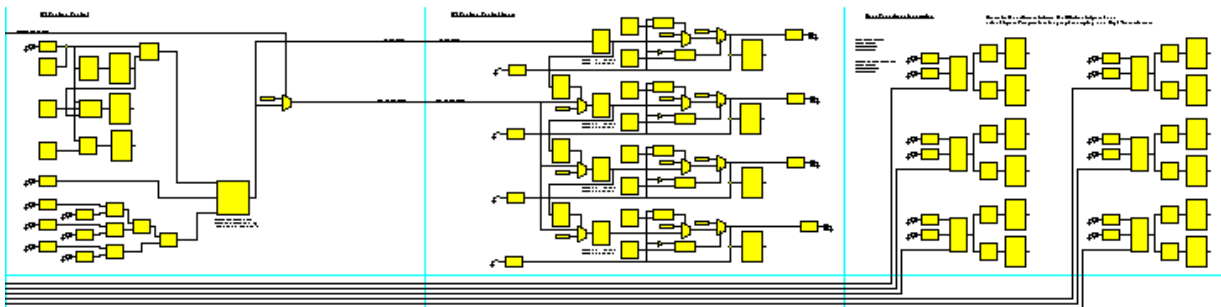




Relief Damper/Fan Control

Economizer Mode Calculations

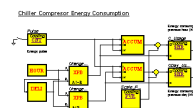
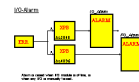
OA/RA Damper Control



DX Cooling Control

DX Cooling Control Logic

Zone Conditions for display



I/O Alarms and
Chiller Energy Consumption

3 Setting Up an Application

Before starting the detailed design of the application, the used type of device for the application is specified. When I/O modules and STR modules are used, these modules are also specified for the application.

In the example, the application is designed for a Xenta 401 with five I/O modules – two Xenta 422 and three Xenta 452.

3.1 Specifying a TAC Xenta Device

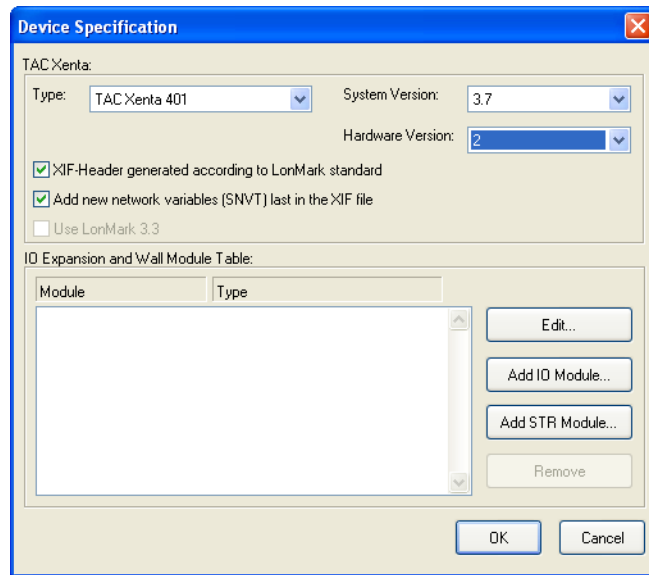
The Xenta device type, its system version and hardware version are specified in the application.

In the example, a Xenta 401 with the system version 3.7 and the hardware version 2 is specified.

To specify a Xenta device

- 1 In Menta, on the **Options** menu, click **Device Specification**.
- 2 In the **Device Specification** dialog box, in the **Type** list, select the device type. In the example, TAC Xenta 401.
- 3 In the **System Version** list, select the Xenta device system version. In the example, 3.7.
- 4 In the **Hardware Version** list, select the Xenta device hardware version. In the example, 2.
- 5 If a number of blocks with SNVTs will be used, ensure that the following two check boxes are selected.
 - **XIF-Header generated according to LonMark standard.**

- **Add new network variables (SNVT) last in the XIF file.**



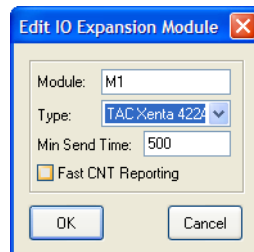
3.2 Specifying an I/O Module

All the required I/O modules are specified in the application.

In the example, two Xenta 422A and three Xenta 452A.

To specify an I/O module

- 1 On the **Options** menu, click **Device Specification**.
- 2 In the **Device Specification** dialog box, click **Add IO Module**.
- 3 In the **Edit IO Expansion Module** dialog box, in the **Module** box, type the name of the I/O module. In the example, M1.
- 4 In the **Type** list, select the type of the I/O module. In the example, TAC Xenta 422A.



- 5 Click **OK**.

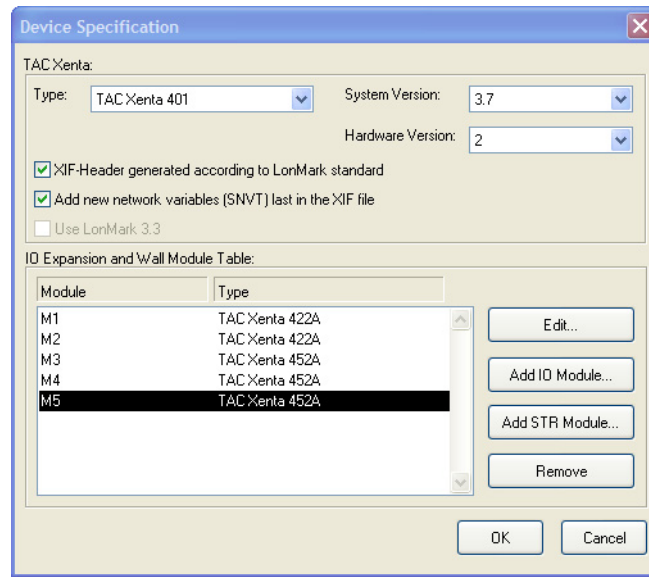
Repeat the procedure above to specify all the required I/O modules in the application.

In the example, add the following I/O modules:

Table 3.1: I/O modules in the example.

| Name | Type |
|------|----------------|
| M2 | TAC Xenta 422A |
| M3 | TAC Xenta 452A |
| M4 | TAC Xenta 452A |
| M5 | TAC Xenta 452A |

When finished, the **Device Specification** dialog box looks like this:



3.3 Naming the Application

Give the application a name and an abbreviation to be used in the menus of an Operator Panel (OP).

To name the application

- 1 On the **Options** menu, click **Program Specification**.
- 2 In the **Name** box, type the name of the application. In the example, RTU4.
- 3 In the **Type** box, type the type of application. In the example, Roof top unit serving six VAVs.
- 4 In the **Abbr** box, type an abbreviation for the application name. In the example, RTU4.



Note

- The abbreviation can contain a maximum of 4 characters.

- 5 In the **Cycle Time** box, type an appropriate value for the cycle time of the application. In the example, 1000.
- 6 Click **OK**.

Program Specification

Name: RTU4 Abbr: RTU4
Type: Cycle Time: 1000 ms
Author: Std. App. Date: 2006/3/23

Used Resources:

| Blocks | | I/O Signals | | | |
|--------|--|-------------|----|----|----|
| | | DI | AI | DO | AO |
| 0 | | 0 | 0 | 0 | 0 |

XIF file information:
Program ID: _____

Public Signal Table:

| Identifier | Type | Access | Units |
|------------|------|--------|-------|
|------------|------|--------|-------|

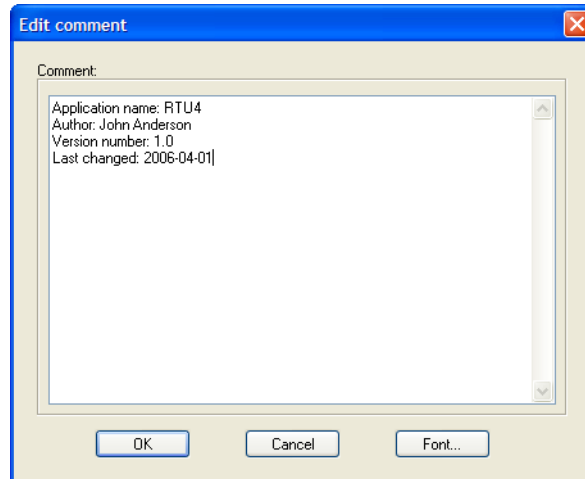
OK Cancel

3.4 Adding an Application Comment

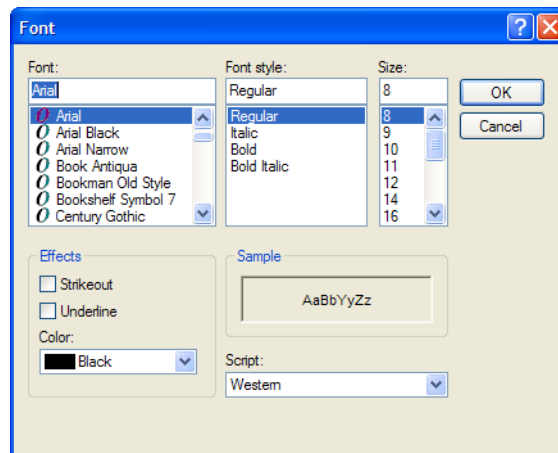
Each application can be assigned a comment where information such as the name of the application, the author, the version number and the most recent revision date can be added.

To add an application comment

- 1 Right-click the drawing area.
- 2 Click **Comment**.
- 3 In the **Comment** box, type the required information. In the example,



- 4 If required, click **Font**.
- 5 Format the text using the settings in the **Font** dialog box.



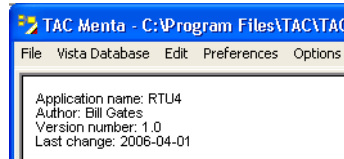
- 6 Click **OK**.



Note

- The font selection applies to the entire comment text.

- 7 Click **OK** to close the **Edit Comment** dialog box.
- 8 On the FBD sheet Click on the text block and hold the left mouse button down and drag the text block to a suitable position. The upper left corner is a suitable place for the comment since it appear in focus every time the application is opened. In the example, place the comment at coordinates (0,0).



- 9 Click outside the comment block to deselect the comment block.

3.5 Saving the Application

Save the application from time to time during design.

To save the application

- 1 On the **File** menu, click **Save**.
- 2 Browse to a suitable location for saving the application file. In the example, C:\ProjectACME\DeviceDescr.
- 3 Name the application file. In the example, RTU4.
- 4 Click **Save**.

4 Creating a Function Block Diagram

A function block diagram (FBD) in Menta is the graphic representation, of the application. It consists of a number of function blocks and connections between the function blocks.



Note

- For more information on how to work in Edit Mode, see Chapter 13, “The Edit Mode”, on page 115.

In the example application you will make a function block diagram to determine the status for six zones. One of the conditions for the AHU to run is that at least one of the 6 terminal unit is in occupancy mode. The solution is to use 6 function blocks of the Analog Input (AI) type, representing the terminal units. The combined result, calculated by an expression block (XPB), is used as one of the starting conditions for an AHU.

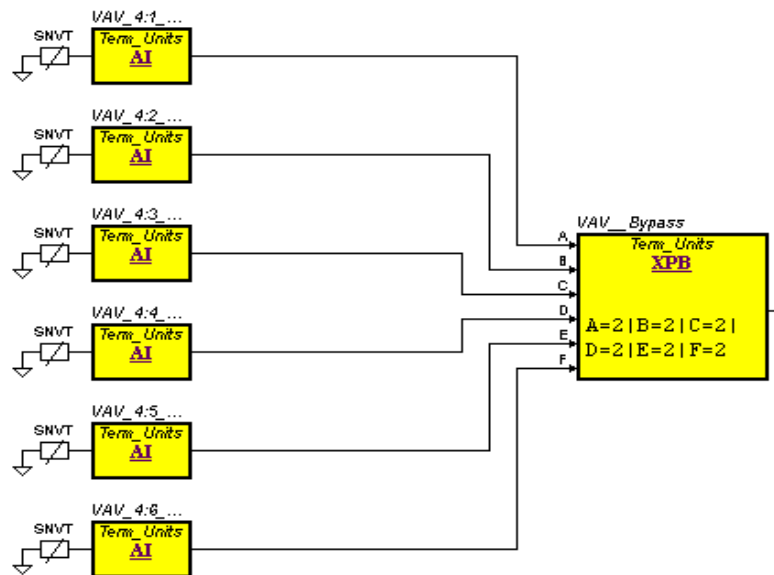


Fig. 4.1: When at least one of the 6 terminal unit is in occupancy mode (= 2) the AHU is set to run.

The occupancy sensors used here will have the value 2 when in occupancy mode.

4.1 Adding a Drawing Title

To simplify the identification of the individual pages at a printout of the function block diagram you are recommended to give each page on the function block diagram a title.

To add a drawing title

- 1 Right-click the FBD sheet.
- 2 On the shortcut menu, click **Comment**.
- 3 In the **Edit Comment** dialog box, type the title for the page. In the example, type “VAV Zone Bypass Status Logic”.
- 4 If required, click **Font**, select a suitable font and then click **OK** to close the **Font** dialog box. In the example, select 12pt Times New Roman, Bold, Underlined as font.
- 5 Click **OK** to close the **Edit Comment** dialog box.

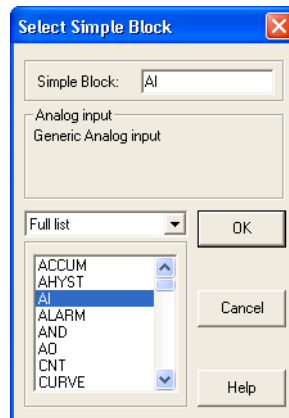
The newly created comment appears as a framed object on the FBD sheet with green borders.

- 6 Click on the text block and hold the left mouse button down and drag the text block to a suitable position. In the example, position it at (51,2).
- 7 Click outside the comment block to deselect the comment block.

4.2 Adding an Analog Input (AI) Function Block

To add an Analog Input (AI) function block

- 1 Right-click the FBD sheet.
- 2 On the shortcut menu, click **Simple Block**.
- 3 In the function list, click **AI**.



- 4 Click **OK**.

The inserted function block is selected by default and with green border.

- 5 Move the block to a suitable position on the FBD sheet. In the example, place the function block at the coordinates (50,39).
- 6 Click outside the frame (on the FBD sheet) to deselect the function block.

The function block has now been added but needs further configuration to function properly.

4.3 Configuring an Analog Input (AI) Function Block

An analog input (AI) function block needs to be configured to function properly.

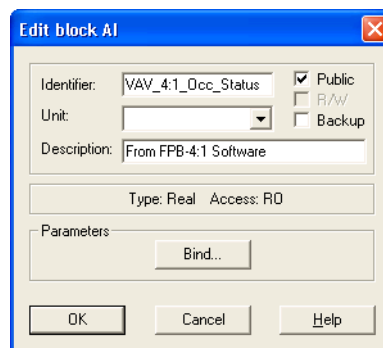


Note

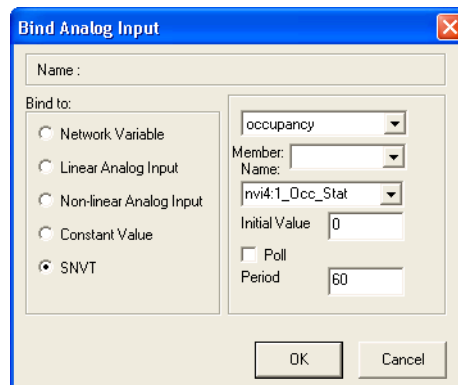
- For more information about the Analog Input (AI) function block, see the Help or Section 21.5, “AI – Analog Input”, on page 287.

To configure an analog input (AI) function block

- Double-click the analog input (AI) function block.
- In the **Edit block AI** dialog box, complete the input as required. In the example, see the screen capture below.



- Click **Bind**.
- In the **Bind Analog Input** dialog box, complete the input as required. In the example, see the screen capture below.



- Click **OK** to close the **Bind Analog Input** dialog box.
- Click **OK** to close the **Edit block AI** dialog box.

Adding function block by function block.

Repeat the steps above to add more analog input (AI) function blocks.

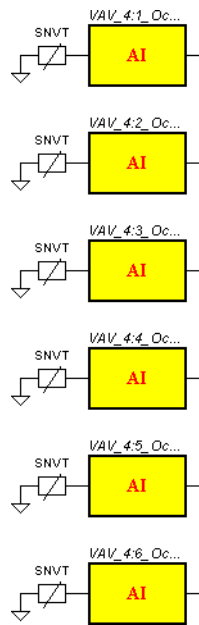
You can benefit from duplicating the first function block several times and then change some parameters while configuring each function block. For more information on how to duplicate a function block, see Chapter 13.3.7, “Copying a Function Block”, on page 136.

In the example, add function blocks for analog inputs using the data in the table below.

Table 4.1: Analog Input (AI) function block, 2–6.

| Function Block | Settings |
|---|---|
| VAV_4:2_Occ_Status Position: (50,51) | Identifier: VAV_4:2_Occ_Status Description: From FPB-4:2 Hardware Type: occupancy Name: nvi4:2_Occ_Stat |
| VAV_4:3_Occ_Status Position: (50,63) | Identifier: VAV_4:3_Occ_Status Description: From FPB-4:3 Rio Grande Type: occupancy Name: nvi4:3_Occ_Stat |
| VAV_4:4_Occ_Status Position: (50,75) | Identifier: VAV_4:4_Occ_Status Description: From FPB-4:4 R&D Services Type: occupancy Name: nvi4:4_Occ_Stat |
| VAV_4:5_Occ_Status Position: (50,87) | Identifier: VAV_4:5_Occ_Status Description: From FPB-4:5 Software Dev. Type: occupancy Name: nvi4:5_Occ_Stat |
| VAV_4:6_Occ_Status Position: (50,99) | Identifier: VAV_4:6_Occ_Status Description: From FPB-4:6 ASG Type: occupancy Name: nvi4:6_Occ_Stat |

When all the inputs are added, the FBD sheet should look like this:



4.4 Adding a Binary Expression Block

A binary expression block (XPB) is used to evaluate input signals using logic. The output is a binary signal – True/False or 1/0.



Note

- For more information about the binary expression block, see the Help or Chapter 22, “Expression Blocks”, on page 377.

In the example, a zone is in occupancy mode when the value of the input (SNVT) is 2. To detect if one (or more) units is running in occupancy mode, we use a (binary) expression block (XPB).

The phrase “if one (or more) units is running” equals the expression

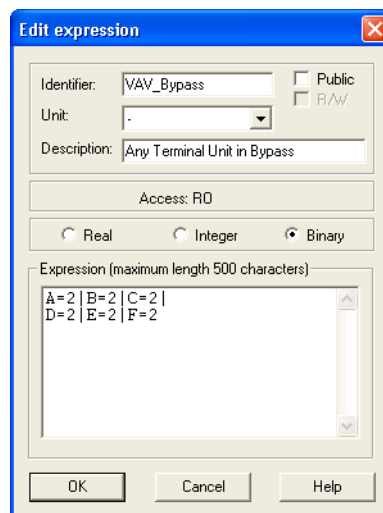
$$A=2|B=2|C=2|D=2|E=2|F=2$$

The character | is logical OR (bit wise).

If any of the 6 inputs (A–F) contain a value of 2, the block is set to True (the boolean value 1).

To add a binary Expression block

- Right-click the FBD sheet.
- On the shortcut menu, click **Expression**.
- In the **Edit expression** dialog box, complete the input as required. In the example, see the screen capture below.



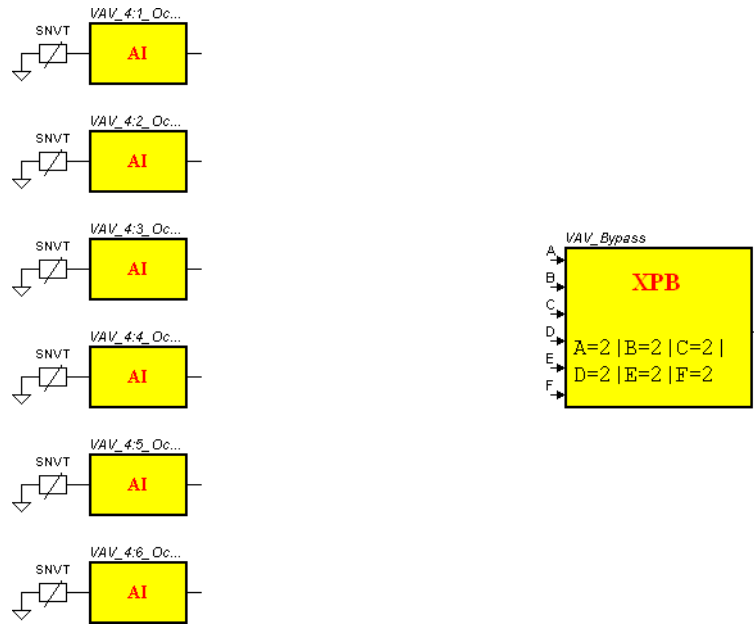
You may break the logic into several lines to reduce the horizontal space in the drawing. In the example we have split the logic into two lines.

- Click **OK** to close the **Edit expression** dialog box.

The inserted function block is selected by default and with green border.

- 5 Move the block to a suitable position on the FBD sheet. In the example, place the function block at the coordinates (111,64).
- 6 Click outside the frame (on the FBD sheet) to deselect the function block.

With the expression block added, the FBD sheet in the example should look like this:



4.5 Connecting a Signal Between Two Blocks

Connections in Menta transport signals (data) between the function blocks in the function block diagram.

For more information on connections, see Chapter 13.5, “Connections”, on page 142.

In the example you will connect the input blocks to the expression block.

To connect a signal between two blocks

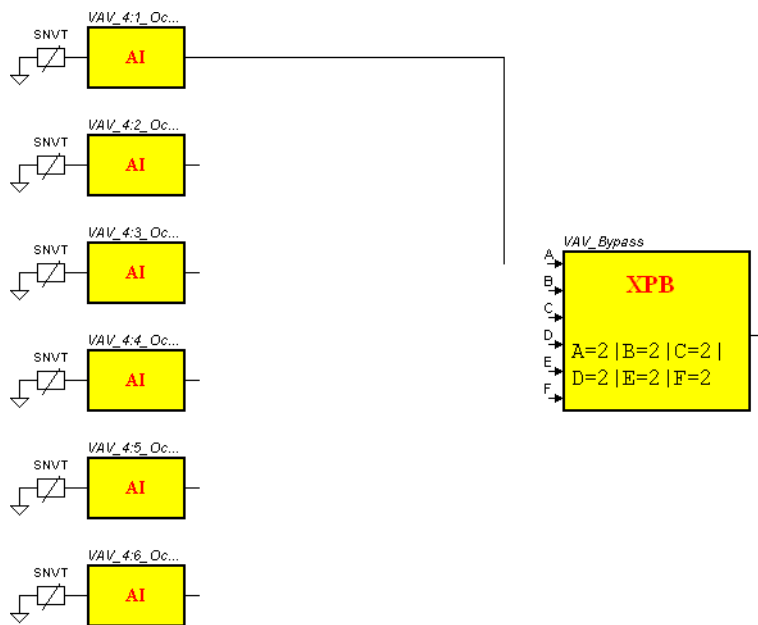
- 1 Click the output of the block. The cursor changes into a X. In the example, click the output of the VAV_4:1_Occ_Status block.
- 2 With the left button pressed, draw a connection line to the input on the target block. In the example, the input A on the XPR expression block.



Note

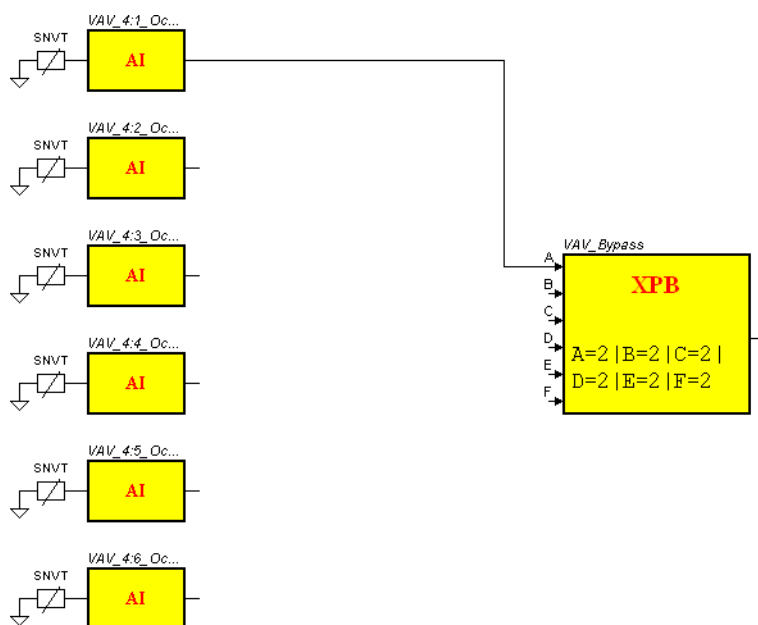
- You can change direction of the connection line (insert “corners”), by clicking the mouse and take a new direction.

So far, the drawing should look like this:



- 3 Point to the required input (the cursor changes into a square). In the example, point to the input A on the VAV_Bypass expression block.
- 4 Click to connect the output with the input. The green line turns black.

The drawing should look like this:



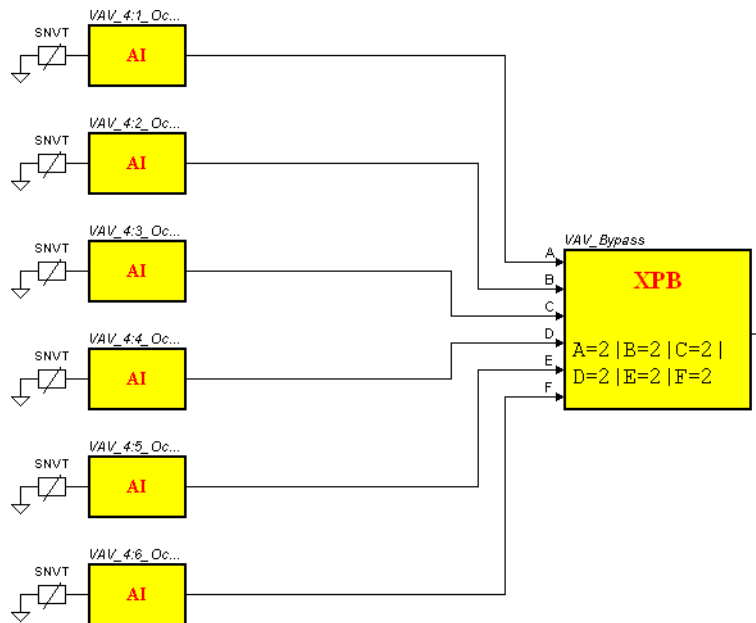
If required, repeat the procedure above to connect all other signals. Connecting signals one by one.

In the example, connect the remaining AI blocks to the corresponding inputs in the expression block.:

Table 4.2: Connection pairs.

| Output on AI Block | Input on XPB Block |
|--------------------|--------------------|
| VAV_4:2_Occ_Status | B |
| VAV_4:3_Occ_Status | C |
| VAV_4:4_Occ_Status | D |
| VAV_4:5_Occ_Status | E |
| VAV_4:6_Occ_Status | F |

When finished, the drawing should look like this:



4.6 Creating a Module

Using a module part in the signal name will let you to divide the application program in parts so that the parts can be handled separately in TAC Vista.

For more information on the use of modules in Menta see, Chapter 8.2.4, “Using Modules in the Application”, on page 84

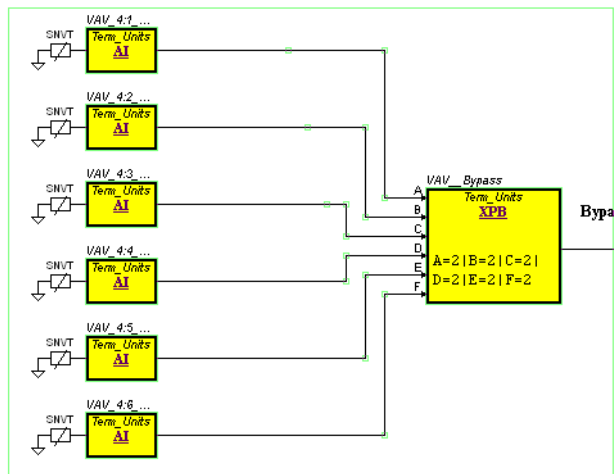
For more information on how to select groups of blocks on the FBD sheet, see Chapter 13.6, “Operations on Groups”, on page 152.

For more information on how to edit module name, see Chapter 13.6.10, “Editing the Module Name”, on page 156

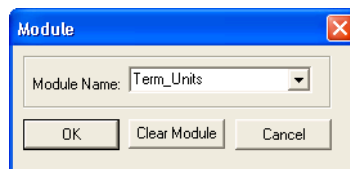
In the example we will take the 6 AI function blocks and the XPB expression block and group them into a module, “Term_Units”.

To create a module

- 1 In the diagram window, select the blocks to be included in the module. In the example, select all blocks.



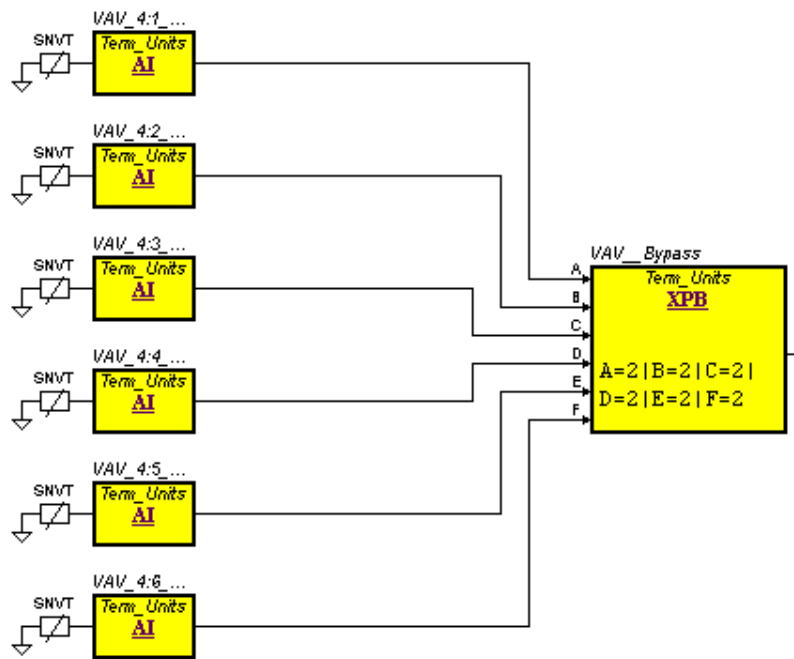
- 2 Right-click the FBD sheet.
- 3 On the shortcut menu, click **Module**.
- 4 In the **Module Name** box, type the name of the module. In the example, “Term_Units”.



- 5 Click **OK**.

The module name is shown in each block. The block type, for a block in a module, will be underlined and in purple (see the AI and XPB labels in the blocks in the figure below).

When finished, the FBD sheet should look like this:

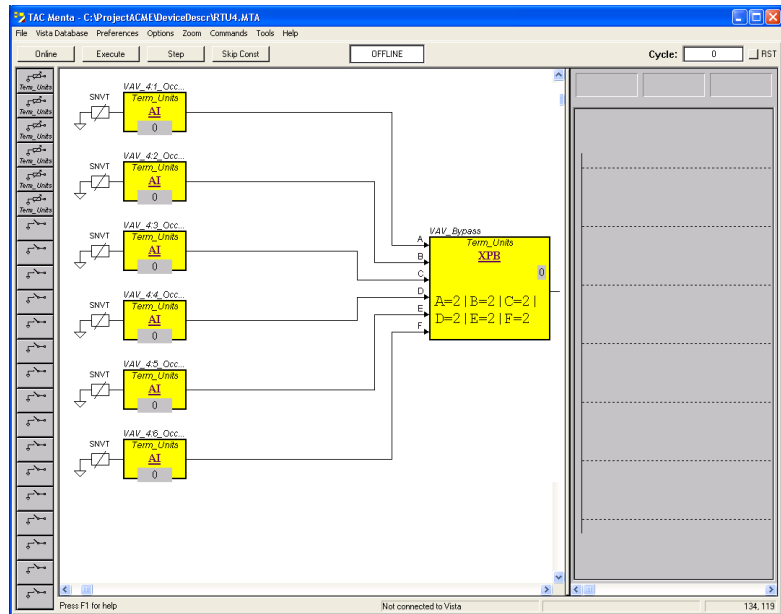


4.7 Simulating the Design

A design test, using the simulation mode in Menta will mostly be a test of expression and calculation blocks. In a running installation, there will be other factors such as errors made when naming the SNVT variables, etc. that cannot be captured with the simulation. The application must be tested in an operational network to guarantee correct design.

To simulate the design

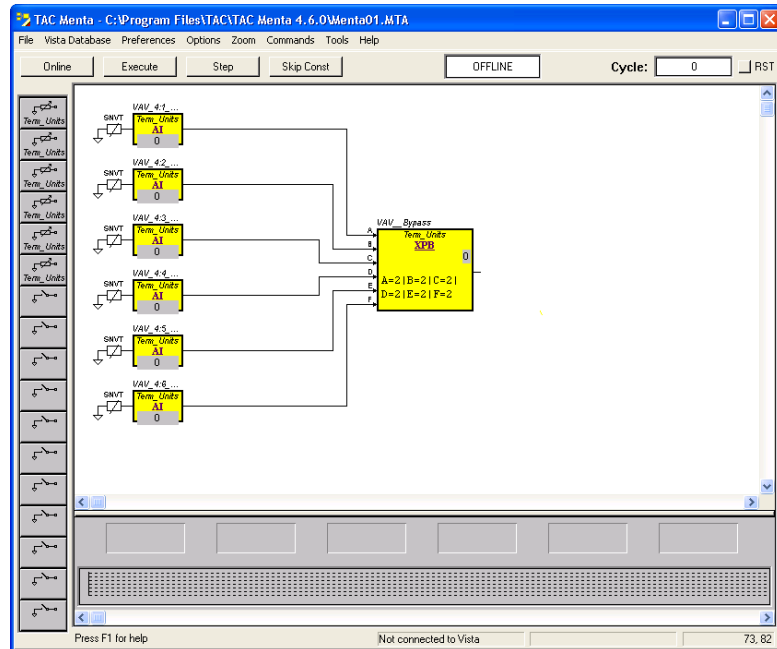
- 1 On the **Options** menu, click **Simulate**.



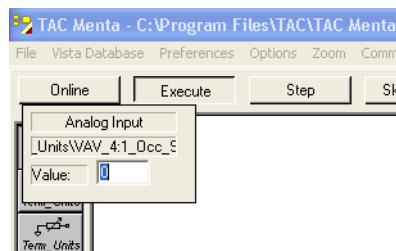
This will change Menta from Edit mode to Simulate mode and display the FBD sheet in the simulation window. In the left margin you will find the 6 analog inputs represented by 6 buttons. The text shown on each button is brief. To see the full identifier, hover a button and look at the status bar in the lower left corner of the window.

Since there is, in this case, no meaning in tracing the values in the logger area (the gray area), click on the separator between the drawing area and the logger area and move the separator to the right

so there will be only one area – the drawing area. Alternatively, you can double-click the separator to make it horizontal.

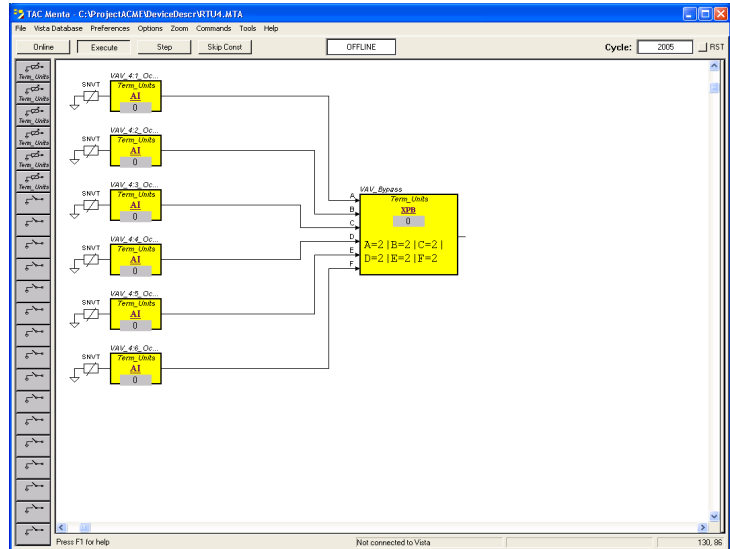


- 2 On the toolbar, click **Execute**. The cycle counter to the right will start.
- 3 Click any of the buttons in the left margin to open a dialog to enter a test value for the Analog Input (AI) function block and see how that affects output of the XPB function block.
 - a In the example, click the first Analog Input (AI) function block.

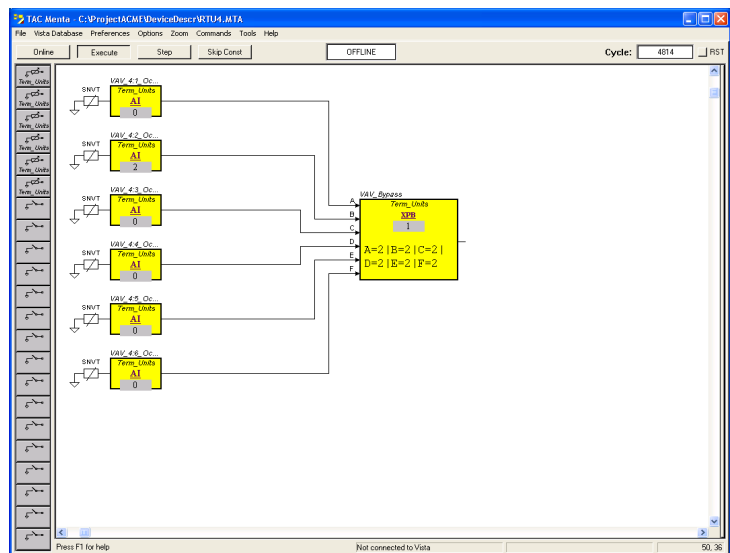


- b Type a test value. In the example, 0.
- c Press **Return**.

Observe the state (the gray field just below the function block type) of the VAV_Bypass block (XPB). It should remain 0 (off) as all inputs are 0 (zero).



- d Click any of the first 6 buttons in the left margin and enter a value of 1 followed by Return. No change of the XPB block state is expected. Not until the input is set to 2 will there be a change.
- e Repeat the procedure above but enter a value of 2. The XPB block should now show a state of 1 (On).



- f Enter a value of 3 and confirm that the state of the XPB returns to 0 (zero). Enter values for the other inputs, the XPB should remain at 1 as long as at least one input has a value of 2.
- 4 To stop simulation mode, click **Execute** on the toolbar.
 - 5 To resume Edit mode, on the **Options** menu, click **Edit**.

The function block diagram for this part of the application is now ready. The remaining parts of the total application will be added to the function block diagram by using macro blocks for each part.

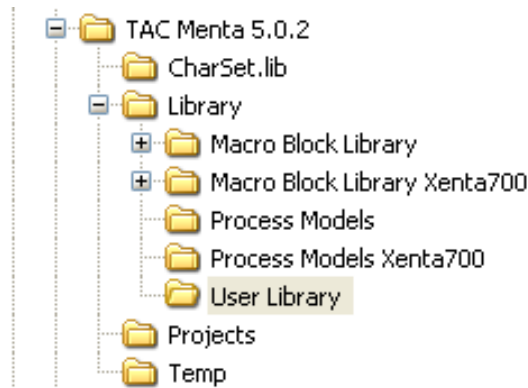
5 Using Macro Blocks

Necessary time for designing an application can be reduced by reusing existing designs and load them as parts to the function block diagram as so called macro blocks. Macros blocks for Menta are by default localized in the Library folder.

5.1 Changing the Library Folder Path

When a macro block is loaded, Menta by default looks in the Library folder from the installation folder for Menta. The folder path can be changed.

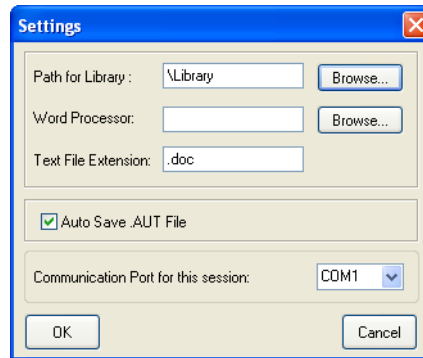
In the example you change the path to use a folder created by the installation of Menta and reserved for the user's own macro block files/applications.



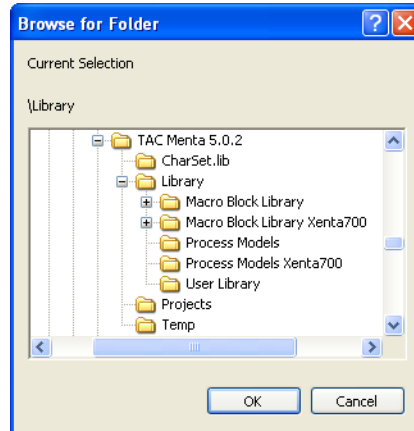
To change the Library folder path

- 1 On the **Preferences** menu, click **Settings**.

- 2 In the **Path for Library** box, click **Browse** and select the folder for the library. In the example, C:\Program Files\TAC\TAC Menta 5.x.x\Library\User Library.



- 3 In the **Browse for Folder** dialog, select the required folder for the library. In the example, C:\Program Files\TAC\TAC Menta 5.x.x\Library\User Library



- 4 Click **OK**.
- 5 Click **OK**.

5.2 Saving a Diagram as a Macro Block

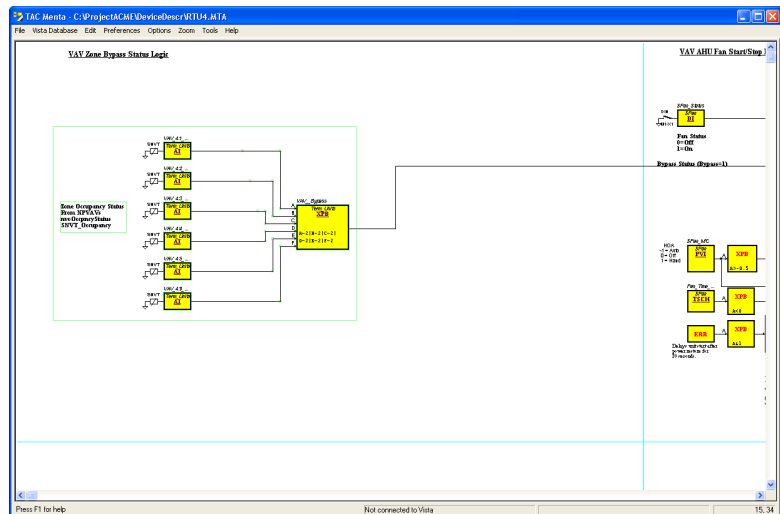
A group of blocks and connections can be saved as a macro block for future re-use.

For more information on saving a macro block see, Chapter 13.8.1, “Saving a Macro Block”, on page 162.

In the example, you save the function block diagram in the project folder.

To save a drawing as a macro block

- 1 Select all function blocks and connections you want to save as a macro block. In the example, select the VAV Zone Bypass Status Logic drawing.



- 2 Right-click the selection, and in the **GROUP** menu click **Save**.
- 3 In the **Save As** dialog box, in the **Save in** box, browse to the location where you want the macro to be stored. In the example browse to C:\Program Files\TAC\TAC Menta 5.x.x\Library\User Library.
- 4 In the **File name** box, type the file name for the macro block with the extension .aut. In the example, My_VAV_Zone_Bypass_Status_Logic.aut.
- 5 Click **Save**.
- 6 Click outside the selection.

5.3 Loading a Macro Block

Previously saved macro blocks can be added to the function block diagram by loading them to the diagram window.

For more information on loading a macro block see, Chapter 13.8.2, “Loading a Macro Block”, on page 164.

In the example, you will the enlarge the application by adding a macro block for handling the start and stop of the AHU to the function block diagram.

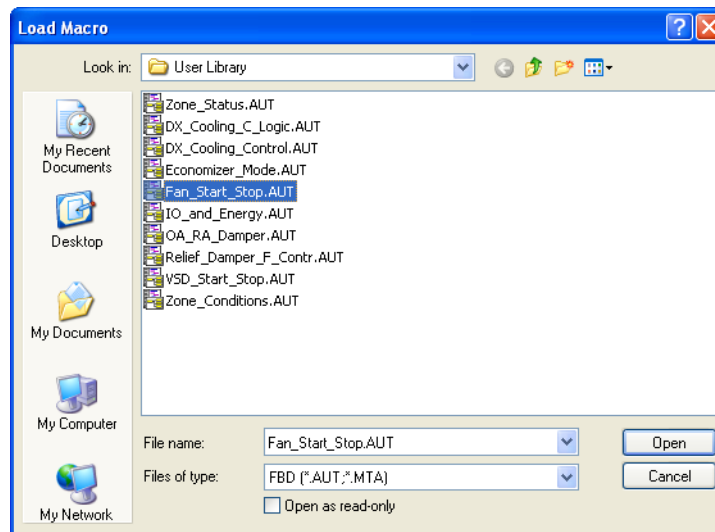


Tip

- Use the **Zoom out** command to get an satisfactory overview of the blocks in the diagram window.

To load a macro block

- 1 Click in the FBD sheet to clear any possible selection.
- 2 Move to a blank area in the FBD sheet, preferably a new “page”.
- 3 Right-click the FBD sheet and click **Load Macro**.
- 4 In the **Load Macro** dialog, select the macro block to load. In the example, Fan_Start_Stop.aut.



- 5 Click **Open**.

All items in the inserted macro block are selected by default and have green borders.

- 6 Move the block to a suitable position on the FBD sheet. In the example, place the macro block to the right of the existing blocks for the zones.
- 7 Click outside the macro block to de-select the items.

When finished the FBD should look something like the figure below

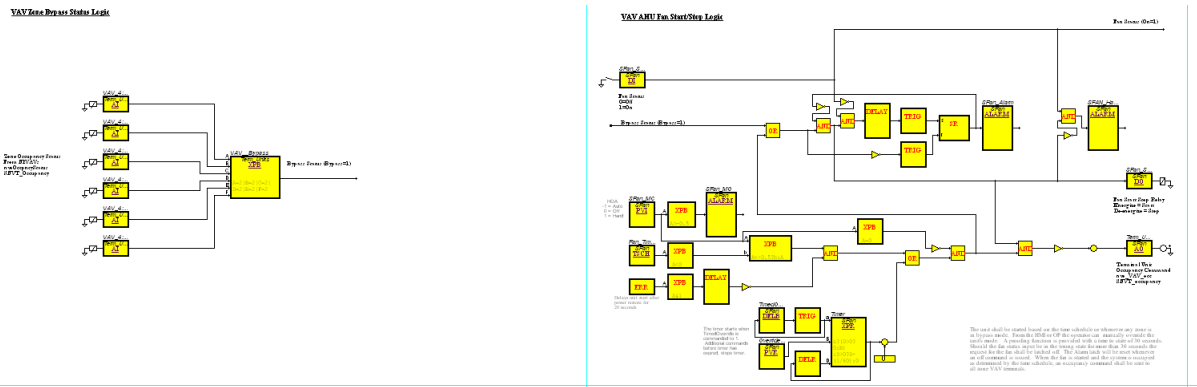


Fig. 5.1: The FBD with the Fan_Start_Stop macro added to the right.

Loading macro blocks one by one

Continue with loading all macro blocks available and required for the design.

In the example, the remaining parts of the complete application will be added to the function block diagram by using macro blocks.

Repeat the procedure above and add the macro blocks according to the table:

Table 5.1: Macro blocks.

| Macro Block | File Name |
|---|---------------------------|
| VAV AHU Fan VSD Stop Logic | VSD_Start_Stop.aut |
| Relief Damper/Fan Control | Relief_Damper_F_Contr.aut |
| Economizer Mode Calculations | Economizer_Mode.aut |
| OA/RA Damper control | OA_RA_Damper.aut |
| DX Cooling Control | DX_Cooling_Control.aut |
| DX Cooling Control Logic | DX_Cooling_C_Logic.aut |
| Zone Conditions for display | Zone_Conditions.aut |
| I/O Alarms and Energy Consumption Calc. | IO_and_Energy.aut |

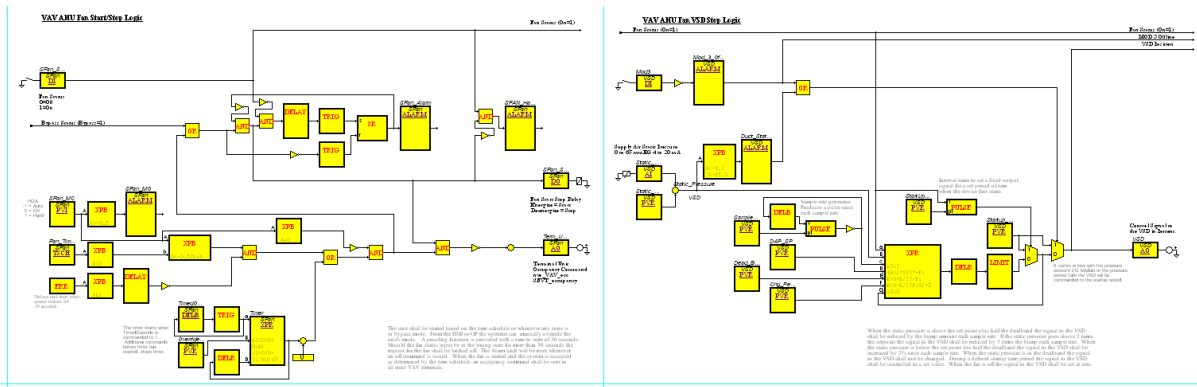


Fig. 5.2: The FBD with the VSD_Start_Stop macro added to the right.

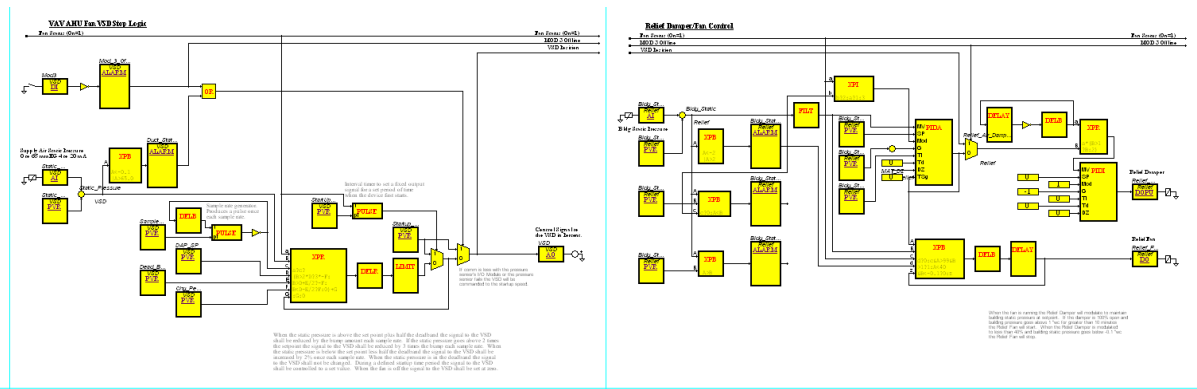


Fig. 5.3: The FBD with the Relief_Damper_F_Contr macro added to the right.

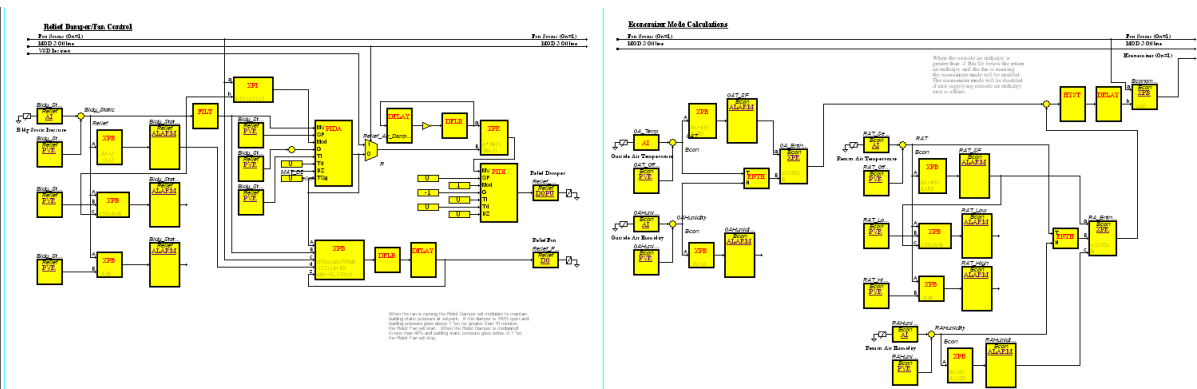


Fig. 5.4: The FBD with the Economizer_Mode macro added to the right.

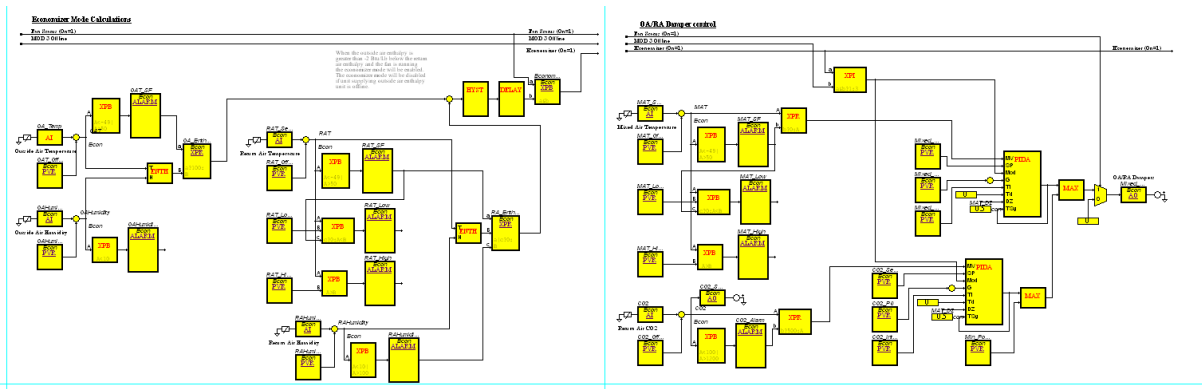


Fig. 5.5: The FBD with the OA_RA_Damper macro added to the right.

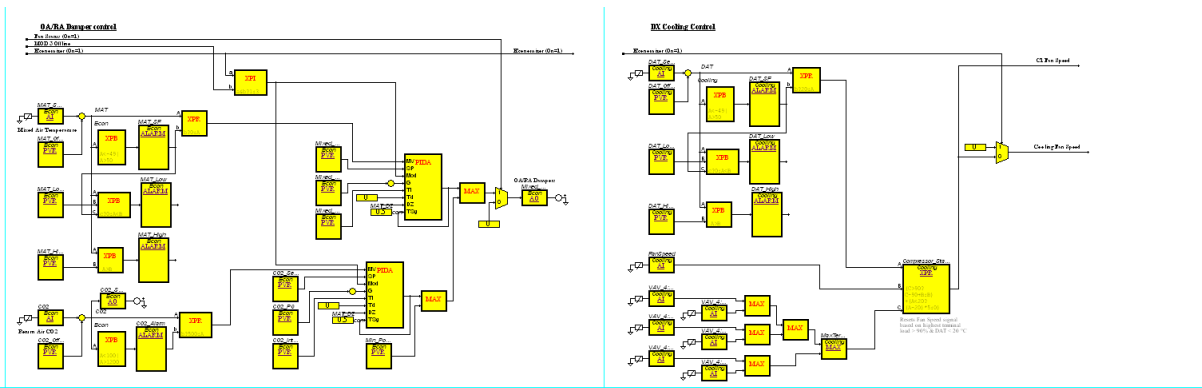


Fig. 5.6: The FBD with the DX_Cooling_Control macro added to the right.

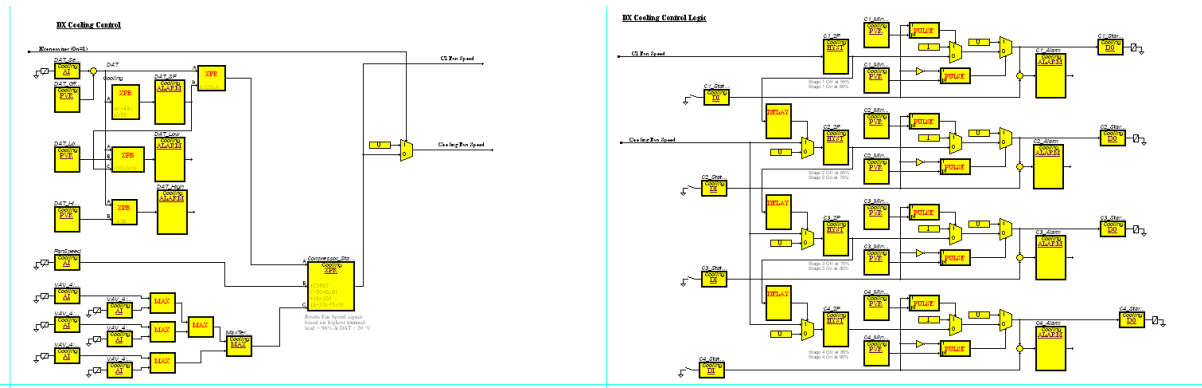


Fig. 5.7: The FBD with the DX_Cooling_C_Logic macro added to the right.

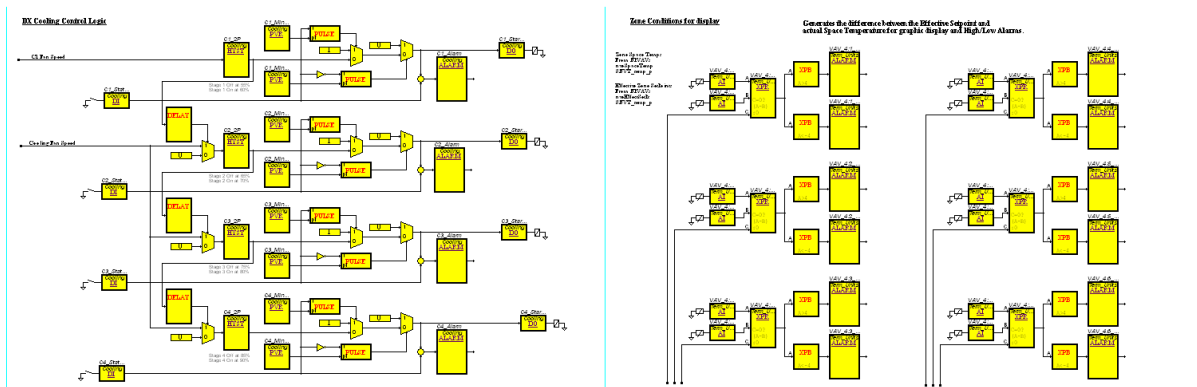


Fig. 5.8: The FBD with the Zone_Conditions macro added to the right.

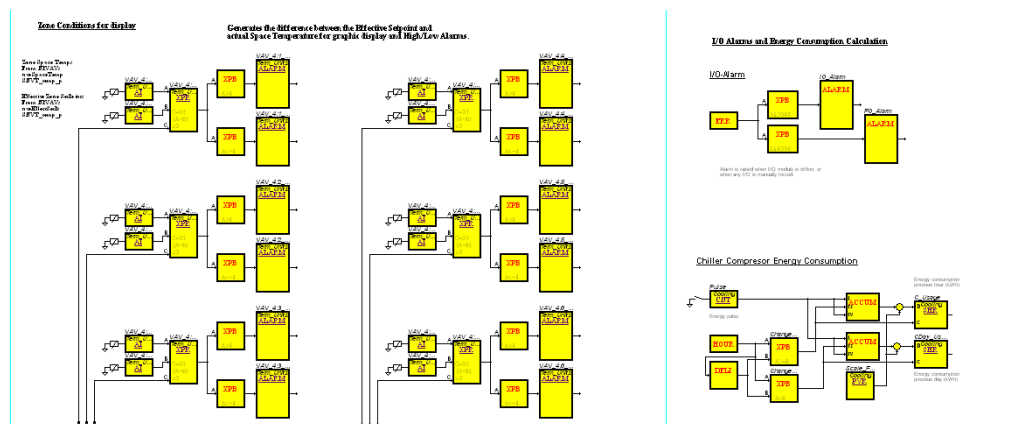


Fig. 5.9: The FBD with the IO_and_Energy macro added to the right.

In the example you have now loaded all macro blocks to the diagram window but signals are not connected between the macro blocks. You will connect them in the next chapter.

6 Connecting Macro Blocks

When the macro blocks are placed on the diagram window, signals between them need to be connected. Good comments and naming the connection nodes in the macro blocks simplify this work.

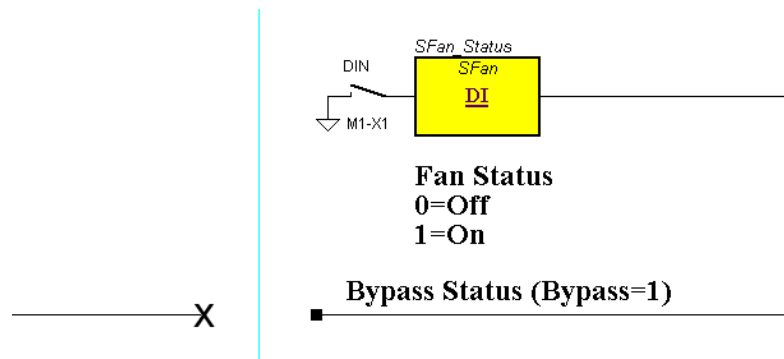
6.1 Connecting Signals Between Macro Blocks

The signals are connected the same way as directly between function blocks. For more information on how to connect signals, see Chapter 13.5.1, “Drawing a Connection Line From a Block Output to Input”, on page 143.

In the example you connect the signal indicating the status of the zones.

To connect signals between macro blocks

- 1 On a macro block, click the required output signal. In the example, click the output signal of the VAV Zone Bypass Status Logic macro block.

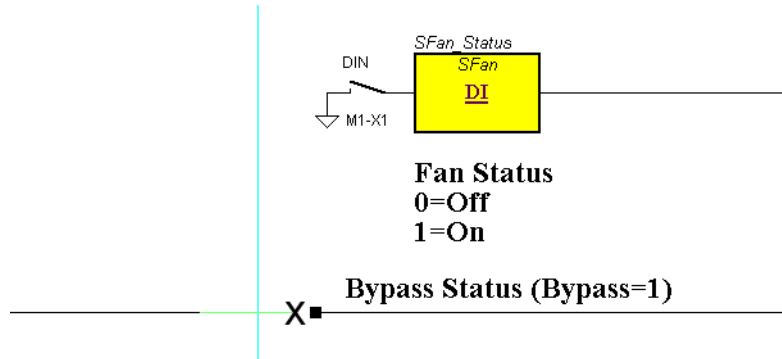


- 2 Draw a connection line towards the input signal on the target macro block. In the example, the Bypass Status signal node in the VAV AHU Fan Start/Stop Logic.



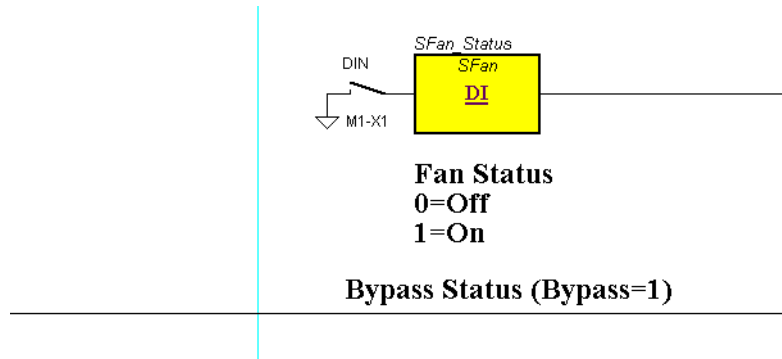
Tip

- If required, change direction of the connection insert “corners” by clicking and take a new direction.



- 3 Point to the required input signal connection node and then click the node. In the example, the Bypass Status signal node in the VAV AHU Fan Start/Stop Logic.

The drawing should look like:



Connecting signals one by one

Repeat the procedure above to connect all necessary signals between the macro blocks.

In the example, connect the signals between the macro blocks according to the table:

Table 6.1: Macro block connections.

| Macro Block | Macro Block Connections |
|--|-------------------------|
| <p>Source (output): VAV AHU Fan Start/Stop Logic</p> <p>The signal: Fan Status</p> <p>Target (input): VAV AHU Fan VSD Stop Logic</p> | |

Table 6.1: Macro block connections. (Contd.)

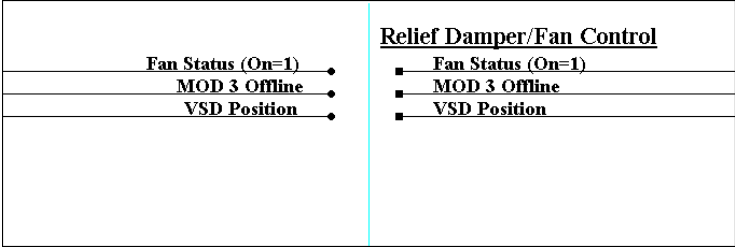
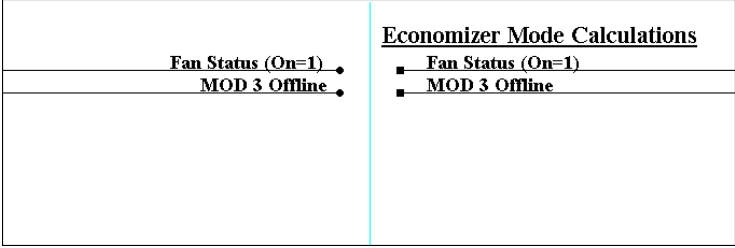
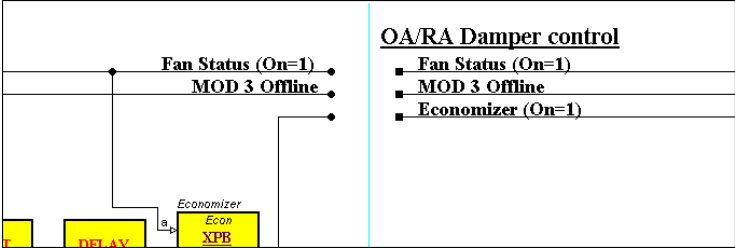
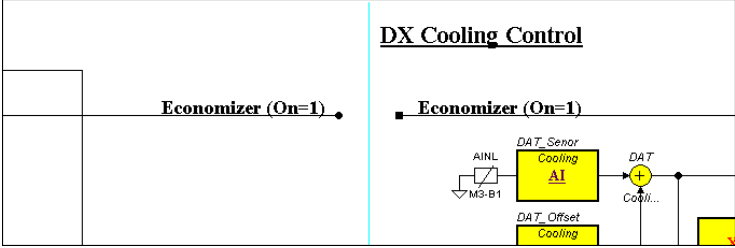
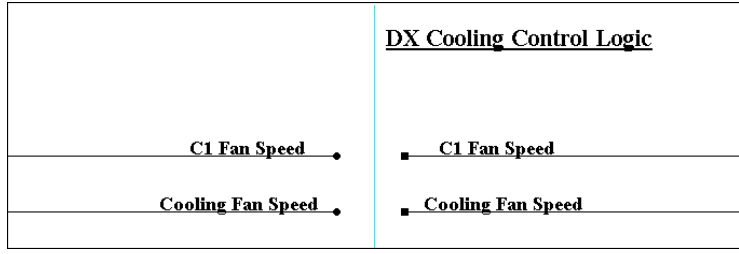
| Macro Block | Macro Block Connections |
|---|--|
| <p>Source (output): VAV AHU Fan Start/Stop Logic</p> <p>The signals: Fan Status MOD 3 Offline VSD Position</p> <p>Target (input): Relief Damper/Fan Control</p> |  |
| <p>Source (output): Relief Damper/Fan Control</p> <p>The signals: Fan Status MOD 3 Offline</p> <p>Target (input): Economizer Mode Calculations</p> |  |
| <p>Source (output): Economizer Mode Calculations</p> <p>The signals: Fan Status MOD 3 Offline Economizer</p> <p>Target (input): OA/RA Damper control</p> |  |
| <p>Source (output): OA/RA Damper control</p> <p>The signal: Economizer</p> <p>Target (input): DX Cooling Control</p> |  |

Table 6.1: Macro block connections. (Contd.)

| Macro Block | Macro Block Connections |
|--|--|
| Source (output): DX Cooling Control The signals: C1 Fan Speed Cooling fan Speed Target (input): DX Cooling Control Logic |  |

In the example it remains to connect the 6 outputs of VAV Zone Bypass Status Logic to 6 inputs of Zone Conditions for display. The the output and input signals are located far apart in the diagram window. To make it easier to follow the signals in the diagram, this work requires some more complicated drawing of the connections.

For more information on drawing connections, see Chapter 13.5, “Connections”, on page 142.

- Create a node at the output from the VAV_4:1_Occ_Status analog input block.

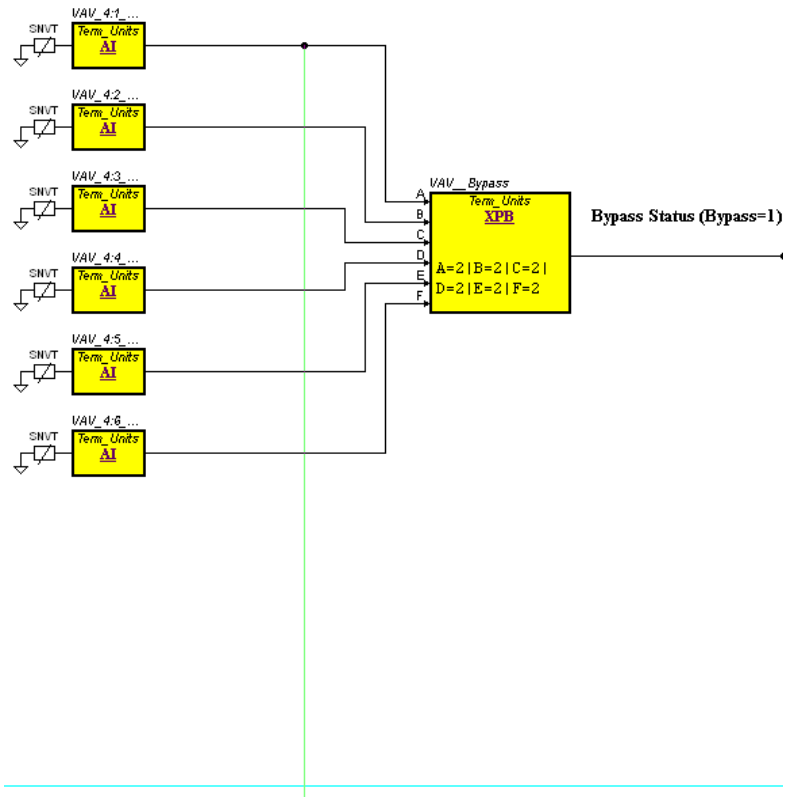


Fig. 6.1: VAV4_1 node created.

- Draw a connection line to the bottom of the diagram according to the figure:

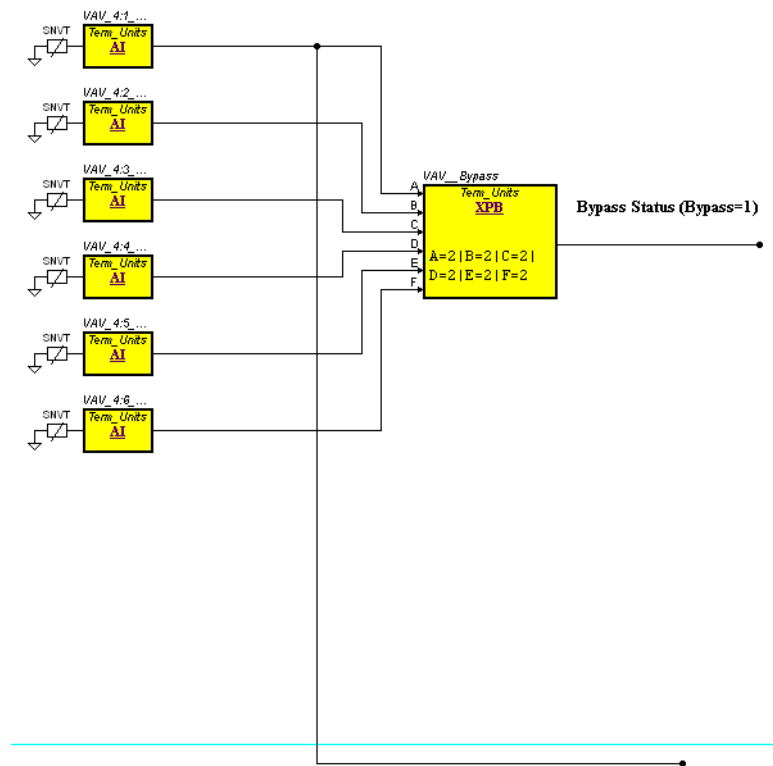


Fig. 6.2: VAV4_1 node and connection line created.

- Click and draw the connection line horizontally to the right in the diagram window and connect to the input node (C input) of the VAV4:1_Temp_Error expression block.

If we disregard the blocks in between, the finished connection looks like the figure below:

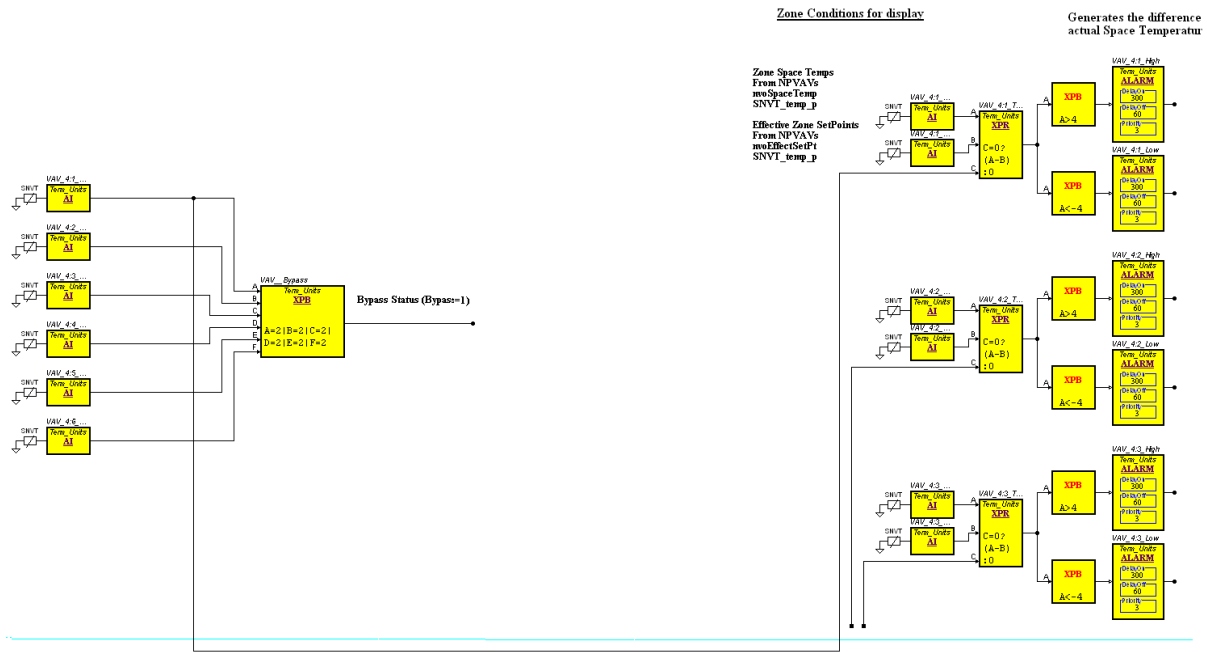


Fig. 6.3: VAV4_1 node connected.

Connect in the same way also the signals:

- VAV_4:2_Occ_Status to the C input of the VAV4:2_Temp_Error block.
- VAV_4:3_Occ_Status to the C input of the VAV4:3_Temp_Error block.
- VAV_4:4_Occ_Status to the C input of the VAV4:4_Temp_Error block.
- VAV_4:5_Occ_Status to the C input of the VAV4:5_Temp_Error block.
- VAV_4:6_Occ_Status to the C input of the VAV4:6_Temp_Error block.

7 Creating Hierarchical Structures

To get a better overview of a function block diagram in Menta you can group blocks and connections in the diagram into hierarchical function blocks (HFBS).



Important

- When an hierarchical function block diagram has been created, it can not be reverted to its original (flat) structure.

A diagram with HFBS is a graphical presentation of the application for printing or viewing on the screen. The application is downloaded to the TAC Xenta device as an ordinary function block diagram, not as hierarchical function blocks.

For more information on hierarchical function blocks, see Chapter 13.9, “Hierarchical Function Blocks (HFB)”, on page 167.

7.1 Creating a Hierarchical Function Block

A hierarchical function block contains a group of function blocks and connections.

For more information on how to handle blocks and connection in groups, see Chapter 13.6, “Operations on Groups”, on page 152.

In the example you create a hierarchical function block for the control of the DX cooling, used in RTU4 application.

To create a hierarchical function block

- 1 Select the required part of the application. In the example, all blocks and texts for the DX cooling control and logic. See figure below:

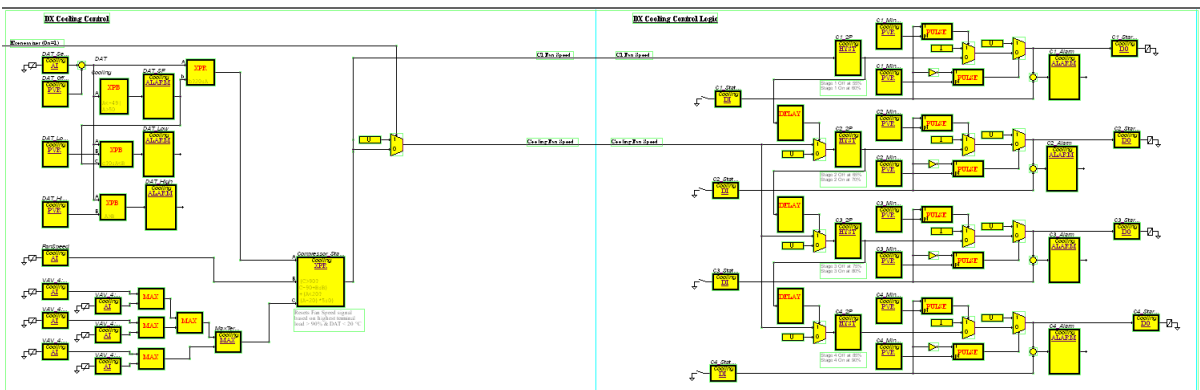


Fig. 7.1: The selected blocks in DX_Cooling.

- 2 Right-click, and then click **Create HFB**.
- 3 In the **Edit Hierarchy** dialog, in the **Identifier** box, type a suitable designation. In the example, type “DX_Cooling”.
- 4 In the **Description** box, type a suitable description. In the example, type “Cooling control and logic for RTU4”.
- 5 Click **OK**.

The hierarchical function block for the DX cooling appears in the diagram window like the figure below:

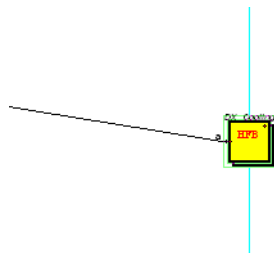


Fig. 7.2: The hierarchical function block for DX_Cooling

7.2 Expanding an Hierarchical Function Block

You can expand a hierarchical function block when you want to view the details.

In the example you expand the DX_Cooling HFB.

To expand a hierarchical function block

- 1 Right-click the required HFB. In the example, DX_Cooling.
- 2 Click **Expand HFB**.

All blocks in the hierarchical function block will fill the diagram window.

- 3 Right-click and then click **Compress HFB** to view the total function block diagram.



Tip

- To expand an HFB you can also double-click the block.

7.3 Creating Hierarchical Function Block in Levels

An HFB can contain other HFBs and connections. You can create HFBs using several levels.

In the example you create an HFB for each stage of compressor logic as HFBs within the existing HFB.

- 1 Expand the HFB in which you want to create the HFB. In the example, DX_Cooling.
- 2 Select the part of the application you want to create as HFB. In the example, all blocks and texts for the stage 1 compressor. See the figure below:

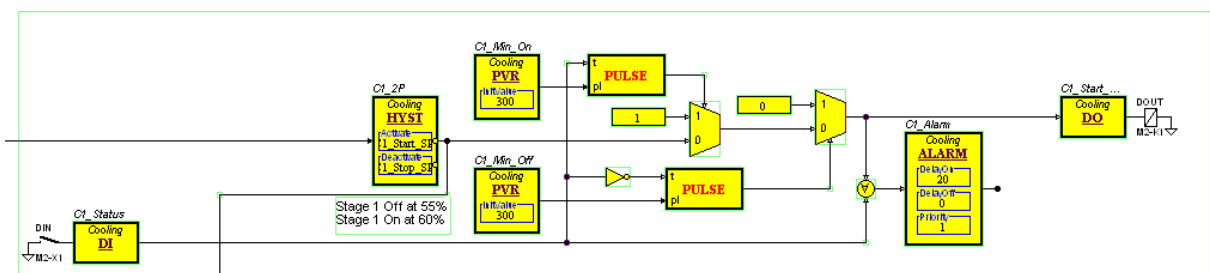


Fig. 7.3: The selected blocks for stage 1 compressor.

- 3 Right-click, and then click **Create HFB**.
- 4 In the **Edit Hierarchy** dialog, in the **Identifier** box, type a suitable designation. In the example, type “Compressor_1”.
- 5 In the **Description** box, type a suitable description. In the example, type “Logic for compressor 1”.

6 Click OK.

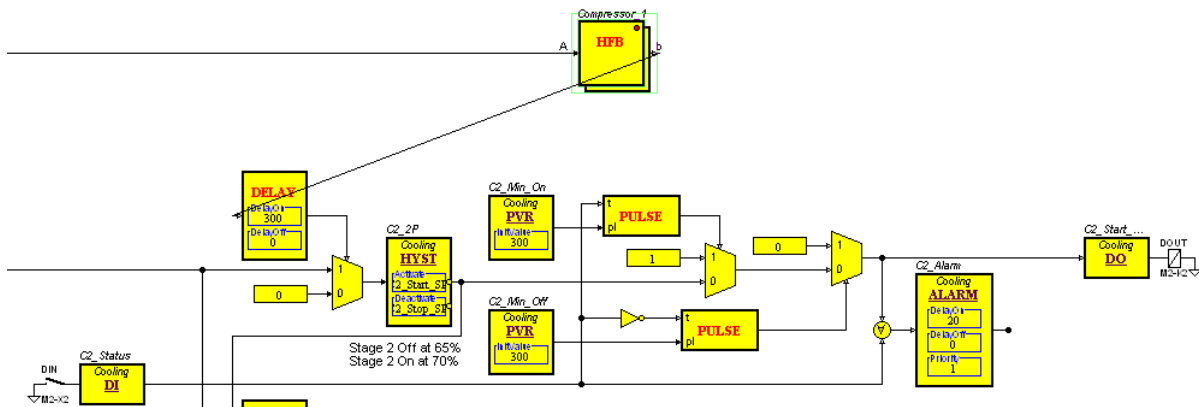


Fig. 7.4: The HFB for the stage 1 compressor.

Creating HFBs one by one

Repeat the procedures above to create all required hierarchical function blocks.

In the example, create an HFB for each of the remaining three compressor stages according to the figure and table below:

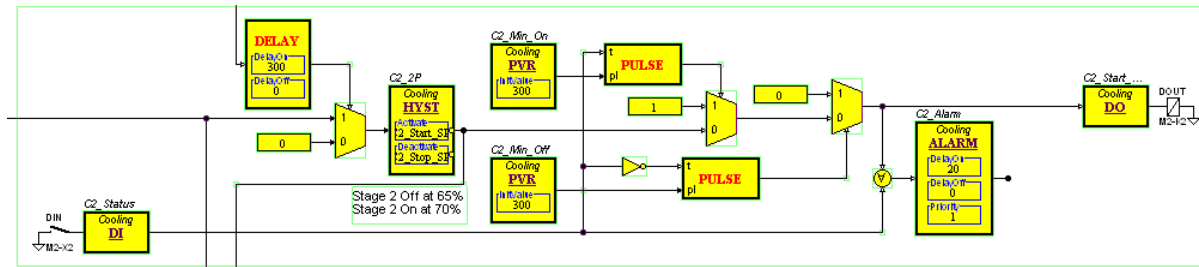


Fig. 7.5: Typical function blocks for the compressor stages 2, 3 and 4.

Table 7.1:

| Identifier | Description |
|--------------|------------------------|
| Compressor_2 | Logic for compressor 2 |
| Compressor_3 | Logic for compressor 3 |
| Compressor_4 | Logic for compressor 4 |



Tip

- You can select blocks and rearrange the block diagram dragging them to a suitable location.

When all HFBs for the compressor are created they will look like the figure below:

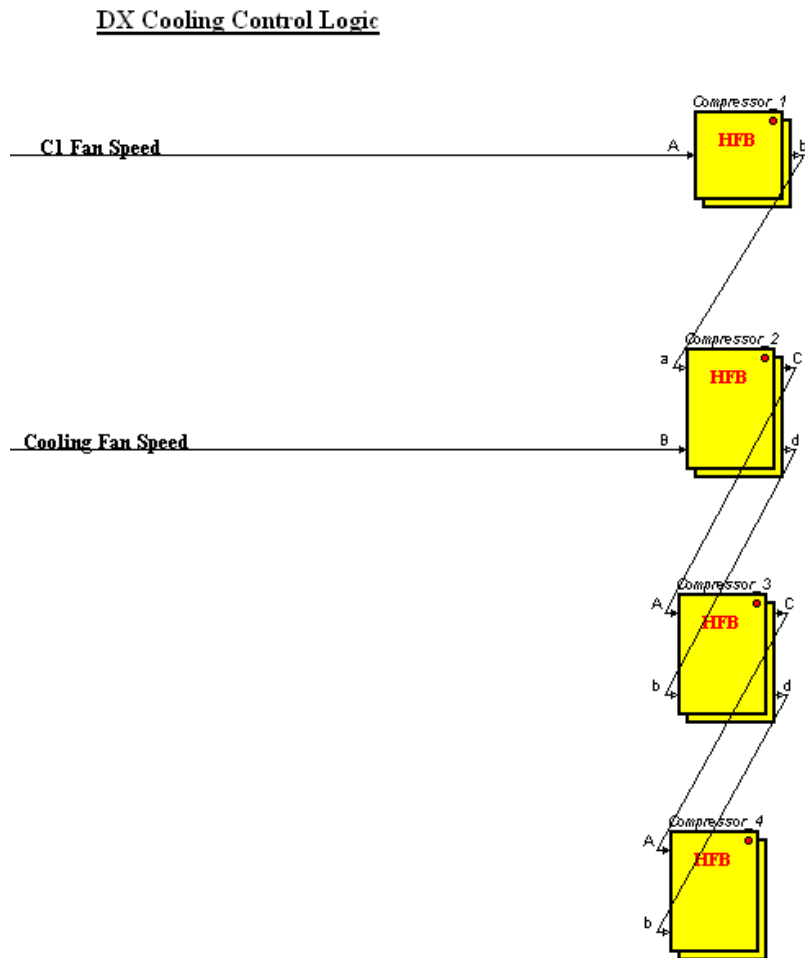


Fig. 7.6: The four HFBs for compressor stages.

In the example, proceed with creating one HFB for all AHU function blocks.

- The complete VAV Zone Bypass Status Logic
- The complete VAV AHU Fan Start/Stop Logic
- The complete VAV AHU Fan VSD Stop Logic
- The complete Relief Damper/Fan Control
- The complete Economizer Mode Calculations
- The complete Economizer Mode Calculations
- The complete OA/RA Damper control

Table 7.2:

| Identifier | Description |
|------------|-------------------------------------|
| RTU4_AHU | The air handling functions for RTU4 |

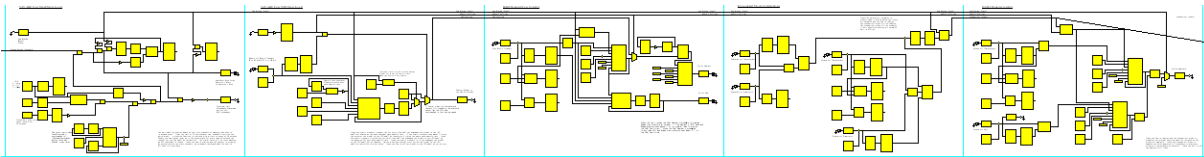


Fig. 7.7: Function blocks for the RTU4_AHU HFB.

After some repositioning of the function blocks, the diagram window with the RTU4_AHU HFB may look like the figure below:

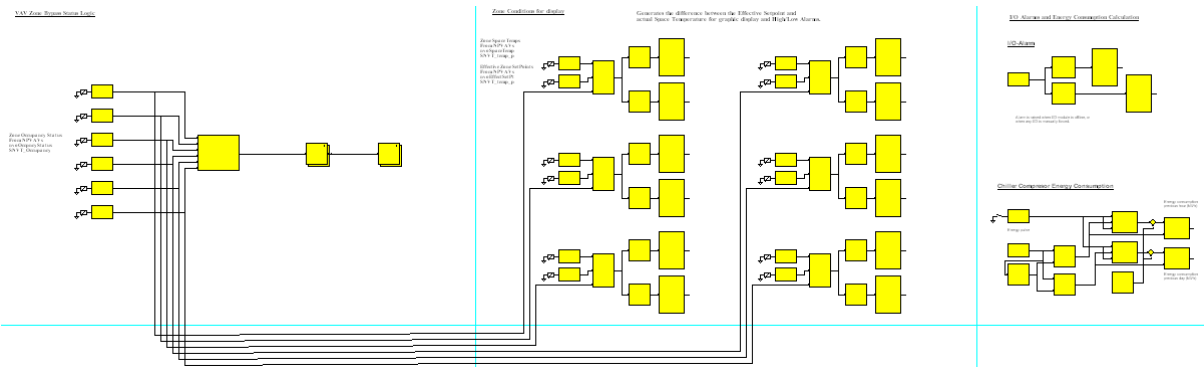
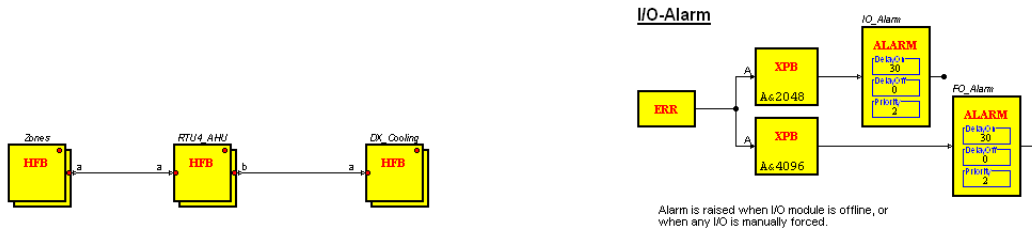


Fig. 7.8: The diagram window with RTU4_AHU HFB.

After more repositioning of the function blocks, and also creating the HFB for the zones, the diagram may look like the figure below:

I/O Alarms and Energy Consumption Calculation



Chiller Compressor Energy Consumption

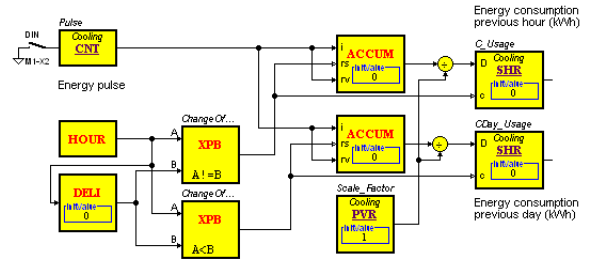


Fig. 7.9: The root level of the diagram window.

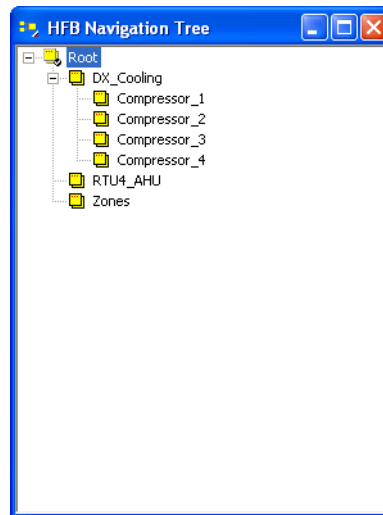
7.4 Navigating in the Hierarchical Structure

Finding a required part of a block diagram can be made easier using a navigation tree in Menta.

In the example, you use the navigation tree to find the logic design for the compressor 1

To navigate in the hierarchical structure

- 1 In the **Options** menu, click **Show HFB Navigation Tree**.



- 2 In the **HFB Navigation Tree** dialog, click the required HFB. In the example, **Compressor_1**.

The HFB for compressor_1 logic is expanded and shown in the diagram window.



Notes

- The **HFB Navigation Tree** dialog remains open to allow fast navigation in the diagram.
- You close the dialog by clicking **Close**.

REFERENCE

- 8 TAC Menta Programming Fundamentals
- 9 TAC Menta Overview
- 10 Graphical Programming
- 11 Signals
- 12 The Mouse and Function Keys
- 13 The Edit Mode
- 14 The Simulation Mode
- 15 The Logger Tool
- 16 On-line Mode Functions
- 17 Memory Usage
- 18 Printing the Application Program Documentation
- 19 The OP Configuration Tool
- 20 The Download Wizard
- 21 Simple Blocks
- 22 Expression Blocks
- 23 Operators
- 24 Test Probe Blocks
- 25 The Menta Application File
- 26 Error Messages
- 27 Programming Hints

8 TAC Menta Programming Fundamentals

Programming an application for a TAC Xenta device can be divided into three phases:

- Function phase, when you read the functional specification is read, and add your own analysis. During this stage, you will determine to what extent you can reuse applications that are already designed.
- Design phase, when you create the application software, menus for an optional Operator panel, and the user documentation partly in parallel.
- Test phase, which can include the following activities:
 - Off-line functional tests of the program modules as they are integrated during the design phase. This is often done using Menta in the simulation mode.
 - A final system test where the complete application is downloaded to a TAC Xenta device and tested.

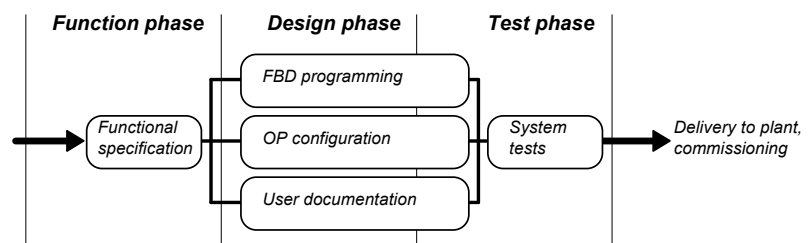


Fig. 8.1: Application programming phases.



Important

- Before you program a Xenta device in a project, you need to consider Project Analysis and System Configuration:
 - Project Analysis: These planning activities greatly affect the number of engineering hours that will be needed later on in the project. The more complete this is made, the less need there will be to supplement the programming.
 - System Configuration: Consider the layout of the installed system early in the process.

8.1 Function Phase

During the Function phase, the functional specification is read and interpreted. Your own analysis is added and an overall solution is formed. During this stage, you will sometimes need to acquire more data regarding parts in the installation, that affect the programming. An important part of this phase is to find the available applications in your library that are ready to use or may only require small changes.

8.2 Design Phase

During the Design phase, points are identified, named, and allocated. A structure of the function block diagram is also planned.

8.2.1 Point Identification and Allocation

The importance of a good point allocation is often underestimated. There is a very close relationship between a well-prepared point allocation and the efficiency that can be achieved later on in the project.

With good point allocation, copying techniques can substantially reduce the engineering hours needed for the design and manufacturing of electrical panels. Installation and commissioning work will also benefit from an efficient point allocation.

8.2.2 Naming of Points and Alarms

The naming of points and signals is an important part of design work. I/O and signals are best given names that are descriptive and easy to recognize. The customer or end user has often specific demands on the use of acronyms or designations. The same applies, even in a higher degree, for alarm texts.

8.2.3 Structuring the Function Block Diagram

In the function block diagram (FBD), the function blocks process input signal(s) and generate a single output signal. During program execution, the output signal is forwarded to other blocks along connections representing the data flow, from left to right.

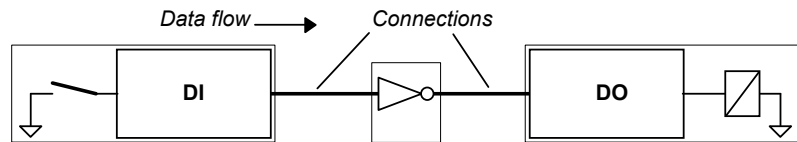


Fig. 8.2:

The first step in the design phase is to make an outline structure:

- Identify the main functions
- Determine where to graphically locate these function groups in relation to each other. The grouping can be done in several ways. The most important thing is to make the FBD easy to understand. Function blocks that perform a function together should preferably be located together.

The second step is to graphically locate the groups. Create a left-to-right logical sequence in the FBD. If there is a connection between two groups, place a group that delivers an output to the left of a group that receives an input. To standardize this, we recommend a logical order from left-to-right: START CONDITION - STOP CONDITION - GENERAL CONTROL - PID CONTROL SEQUENCES - ALARM HANDLING.

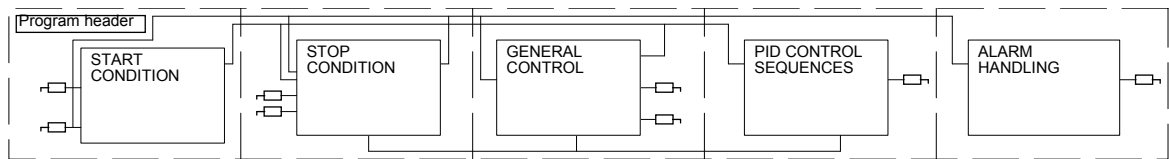


Fig. 8.3:

Use the following rules of thumb when you locate blocks and groups:

- Enter a framed comment containing the last edition date of the FBD in the upper left corner. History notes, describing the revision, can also be included. Update this information whenever you make a program revision.
- Try to structure every page so that the physical input blocks are arranged in a column to the left, and the physical output blocks are in a column to the right.
- Adjust the function block diagram to the page break lines to make a printout easy to read.
- Make space between the groups so that it will be possible to add connections and additional functions.

- Add new pages to the right when necessary, but try to use only one page in a vertical direction.
- Locate connections like a bus with common left-to-right connections at the top of the diagram. Avoid mixing connections running in the opposite direction. Put connections in a separate bus at the bottom of the page when running in the opposite direction. Mark the direction of these signals.

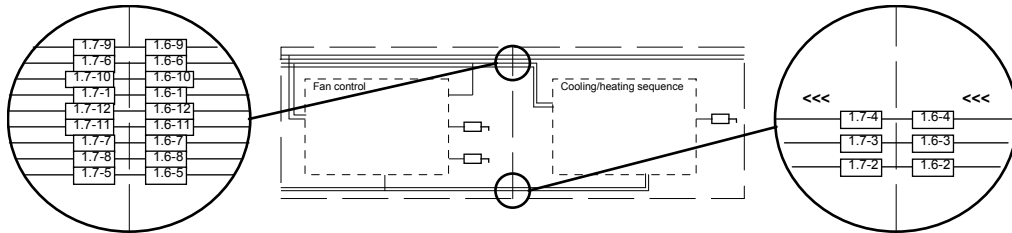


Fig. 8.4:

- Re-use tested macro blocks from the Macro Block Library or your own library as much as possible.

8.2.4 Using Modules in the Application

Using a module part in the full signal name will enable you to divide the application program into different parts so that the parts can be handled separately in Vista. Public signals within a Module will be located in the same logical unit in the Vista database. Signals that have a module part in the name will also form submenus when the Operator Menu (OP) is generated.

Structure the application program so that all public signals that you want to display together will belong to the same module (and have the same module name).

8.2.5 Menus in the OP

The contents of the menus in an OP for Xenta 280etc are determined by two factors:

- The two upper levels are determined when the network is created.
- The application levels are determined when the application is created using Menta.



Important

- Xenta 700 devices have no operator panel menus.

Upper Level Menus

The structure of the upper level menus in an OP is created when you create Xenta groups and give the Xenta devices their network names. The OP displays a view of the network:

- Xenta groups on the top level
- Xenta device within each group on the next level.

These menus contains network information and are available in all Xenta devices in the network. Whenever a new Xenta device is added (or renamed) in a network, new information must be downloaded to all devices in the group with the new (or renamed) device.

The preparatory work of determining the need for all Xenta devices, including grouping and naming them, is of major importance in order to avoid unnecessary work.

TAC Xenta OP Display

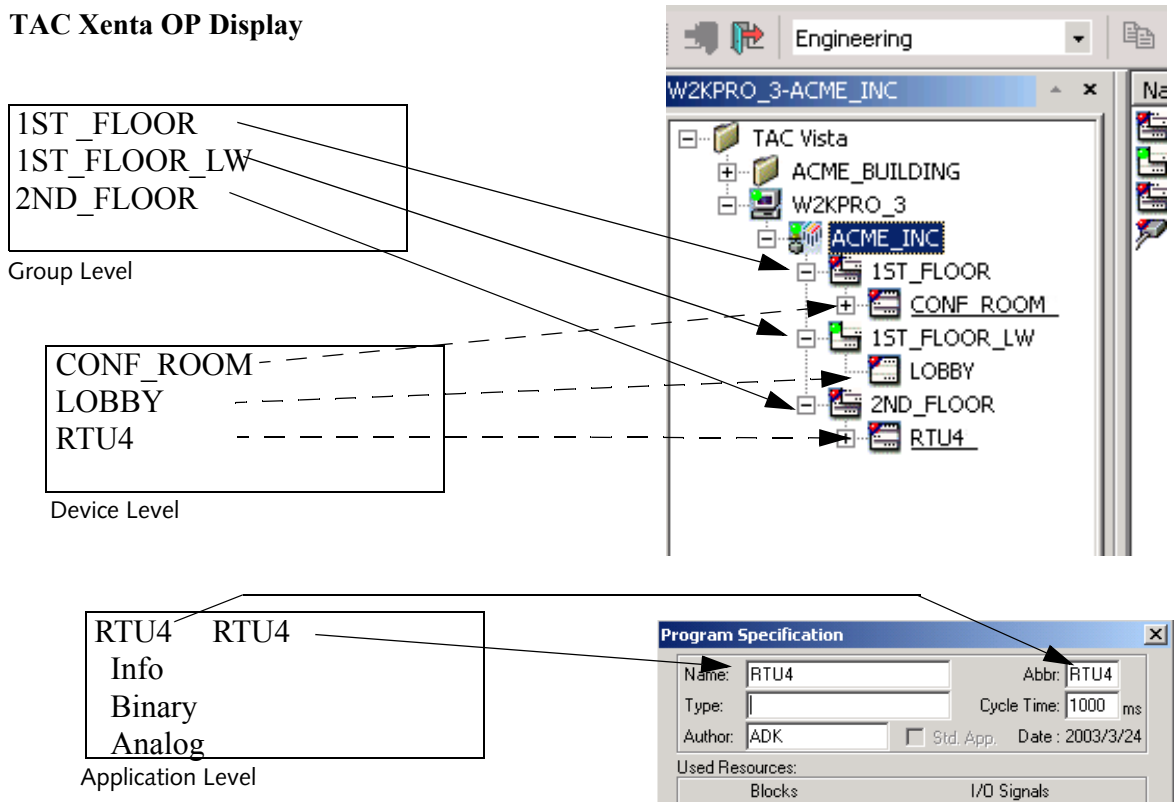


Fig. 8.5:

Application Level Menus

To open the application menu, select the Xenta group and then the Xenta device from the menu tree.

The Program Specification dialog box displays the application name and an abbreviation of the Xenta device identification.

- the application name (**Name**).
- the Xenta device identification (**Abbr**) as an abbreviation, shortened to a maximum of four characters.

The contents of the remaining lines in these device specific menu windows are created when you use the OP configuration tool.

These menus only contain device specific information and will only be available in the specific Xenta devices. The revised menu tree is only downloaded to the actual Xenta device.

8.3 Test Phase

Testing an application is done using the Simulation mode in Menta. Testing an application is usually done offline, when the Xenta device is not connected.

If you need to start/stop the program and to alter values/states, test the application in online mode (with the device connected).

The possibility to start/stop program execution and to alter values/states can be helpful when troubleshooting an application.

9 TAC Menta Overview

TAC Menta is used for graphically designing control software for TAC Xenta 280/300/401 and Xenta Server 700 devices. You design the application using combinations of different function blocks and the produced signals are transferred to other function blocks using connections.

9.1 The Operation Modes

When you design an application in TAC Menta you can work in two modes:

- the Edit mode: Use this mode to design a program.
- the Simulation mode: Use this mode to run the software and show the state of the signals.

For more information on the edit mode, see Chapter 13, “The Edit Mode”

You can simulate the application two ways:

- Offline: The TAC Xenta device is not connected.
- Online: The TAC Xenta device is connected to the PC by a serial port.

Simulating the application using the online mode via a serial port is not applicable for Xenta Server 700 devices.

For more information on the simulation mode, see Chapter 14, “The Simulation Mode”

Both of the simulation modes contain the **OP Configuration** command. This command is used to design an operator panel for Xenta 280/300/401 devices.

The **OP Configuration** command is not available when Menta is started from XBuilder.

9.2 System of Units

In TAC Menta, a number of blocks use either the SI (metric) unit system or the Inch-Pound unit system.

The national settings of the Windows© operating system, used in the PC in which the application is created, determines the unit system for

some of the blocks. The definition of the signal signal determines the unit system for other blocks.

9.3 Allowed Characters in Name Strings

Alphanumeric strings are used in TAC Menta to name signals (blocks) and constants. You can use national character sets in name strings.

There are restrictions in the use of characters in name strings.

The space character is not allowed, but there are three substitutes:

- / the slash character
- _ the underscore character
- : the colon character

The following characters are reserved for use as separators or system object identifiers in monitoring and supervising systems:

- , the comma character
- ; the semicolon character
- - the minus character
- . the full stop character
- \$ the dollar character
- ' the apostrophe character

Double-quotes are not allowed. If you type a name string with double-quotes ("Signal_name", for example), the double-quotes are deleted when the dialog box is closed."

9.4 Unique Application Program ID

The program ID is an automatically generated string that contains information about the application and its signals. Certain positions have a specified meaning. For example, 80 in the leftmost position indicates an application according to the Lonmark standard.

The last position of the string is a checksum. The checksum is dependent of the structural and functional parts of the program, and is not dependent on parameter values or I/O configuration data. You can change parameters and configuration data without affecting the program identity.

Sometimes it is of value to verify whether the application program in a device is identical to a specific source code, for instance from a TAC standard application library. For this, and other reasons, an application has a unique ID.

You can view the program ID in the **program ID** box, in the **Program Specification** dialog.

9.5 Marking Standard Applications/Controllers

A TAC Xenta controller is marked as either a Standard or a Custom (user programmable) controller. The marking is unique for each controller, and is generated from the neuron ID via a special algorithm.

Only standard applications can be downloaded to a controller when it is marked as Standard controller.

Only custom applications can be downloaded to a controller when it is marked as Custom controller.

Source code files for standard applications have unique marks, generated from the checksum of the function block diagram. This checksum is also used as the unique program ID.

9.6 Program Licenses

To use the TAC Menta programming tool you need an installed TAC software licence.

For more information on how to acquire and install the licence for TAC Menta, see the document "TAC Licenses, Installation Manual".

Applications saved in TAC Menta 4.0 (or higher) can not be used by versions of TAC Vista older than 3.2. To support older TAC Vista versions, you can save TAC Menta 3.1 applications in TAC Menta 4.0.

9.7 Starting TAC Menta

You can start TAC Menta manually or automatically from certain other applications, for example TAC XBuilder.

You can start TAC Menta by clicking the TAC Menta icon in the TAC Tools program group. TAC Menta can also be started from within other programs like TAC Vista Explorer, and from the TAC Vista Device Plug-In.

Depending on how TAC Menta is started, the available commands and function blocks differs.

You can run multiple instances of the Menta simultaneously. More than one instance can also be connected on line, if the required serial ports are available.

To start TAC Menta

- On the Windows taskbar, click **Start**, point to **All Programs**, point to **TAC**, point to **TAC Tools**, and the click **Menta**.

To start TAC Menta from another program, right-click a device and then click **Edit**.

9.8 Defining the TAC Menta Settings

Before starting your work in TAC Menta, you can define a few system settings.

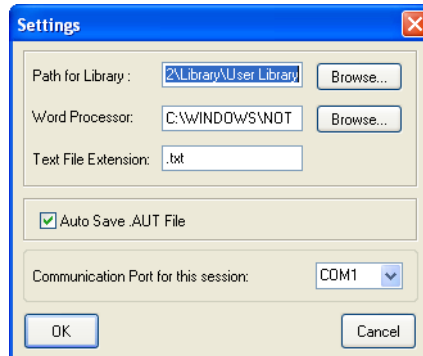
These settings establish the location of a library of applications, the type of editor and file extension, and the serial port for communication with devices.

Table 9.1: The Settings options.

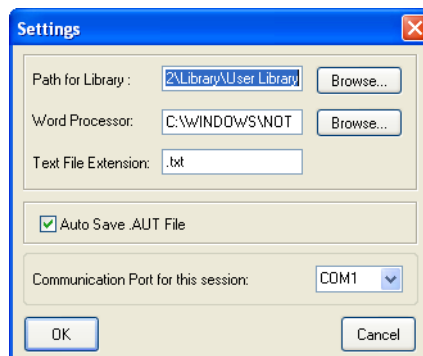
| Option | Description |
|--------------------------------------|--|
| Path for Library | <p>Use this box to specify the path to an application library, used by TAC Menta.</p> <p>Browse to the required folder using the Browse button.</p> |
| Word processor | <p>Use this box to specify the path to a text editor, used by TAC Menta.</p> <p>Browse to the required application using the Browse button.</p> |
| Text File extension | <p>Use this box to enter the required file extension, for example .txt, used for an optional associated text file.</p> <p>For more information about the associated text file, see Chapter 13.17, “Using an Associated Text File”.</p> |
| Auto Save .AUT File | <p>Select this check box to automatically save the current function block diagram to the AUTOSAVE.AUT file before changing from edit mode to simulation mode.</p> |
| Communication port for this session. | <p>Use this list to specify the COM port used for a serially connected TAC Xenta device type 280, 300, 401.</p> |

To define the TAC Menta settings

- 1 On the menu bar, in the **Preferences** menu, click **Settings**.



- 2 In the **Settings** dialog, click the **Browse** button and browse to the required library folder for the TAC Menta Library.
- 3 Click the **Browse** button and browse to the required Word processor (editor), used by TAC Menta when editing text files.
- 4 In the **Text File Extension** box, type the required file extension.
For more information about the associated text file, see Chapter 13.17, “Using an Associated Text File”
- 5 Select the **Auto Save .AUT File** check box, if required.
- 6 In the **Communication Port for this session** list box, select the required port
- 7 Click **OK**.



10 Graphical Programming

When you program an application using TAC Menta, you define the application graphically in the function block diagram (FBD). The FBD has two fundamental elements:

- Function blocks (FBs), which processes the data
- Connections, which transport the data (signals)

10.1 Function Block Diagrams

The function blocks in the diagram process input signals to the blocks and generate a single output signal.

Each function block can have one or more parameters that are processing the input signals. The parameters can be defined as numeric values or as identifiers, (constants).

The output signal is forwarded to other blocks. The signal follows the route defined by the connections, which represents the data flow during program execution.

The data normally flows from left to right in the diagram, except when a connection is used to close a feedback loop.

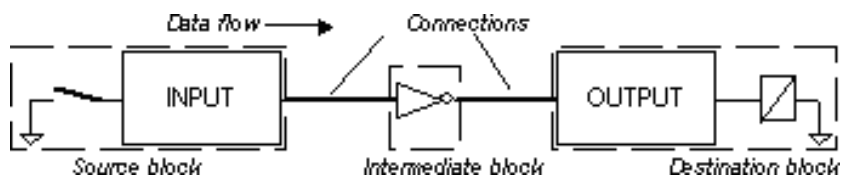


Fig. 10.1: A simple function block diagram (FBD).

In general, blocks without inputs, (source blocks) are located on the left in the diagram.

The blocks where outputs not are connected to other blocks (destination blocks), are often placed to the right in the diagram.

Intermediate blocks, which make calculations and logical decisions, are placed between the source and destination blocks. These intermediate blocks are oriented in the direction of the data flow.

The Menta application is a cyclic application that is executed at a constant time interval, the program cycle.

During each program cycle, changes originating in the source blocks are propagated to the destination blocks via the intermediate blocks.

11 Signals

Signals, their handling and the response to them, are the most important details in the TAC Menta application.

The signals in an application are variables. These variables are generated as outputs of function blocks and are transferred to inputs in other blocks using connections.

All function blocks in TAC Menta have one output only. Each function block generates a unique signal.

11.1 Signal Types

Four types of signals are used in TAC Menta:

- Integer: Signed 16 bit number (range: -32768 to 32767).
- Real: Signed real 32 bit number in the IEEE format with a precision of 6 decimals (range: 3.4×10^{-38} to 3.4×10^{38}).
- Binary: 1 bit to represent binary values (0/1 = FALSE/TRUE). The binary type is also called Boolean or digital.
- Analog signal: either a real or an integer signal

Normally you can only connect outputs to inputs of the same type in the function block diagram.

11.2 Signal Names

Public signals in TAC Menta always have names that are alphanumeric strings. For information on allowed characters in signal names, see Chapter 9.3, “Allowed Characters in Name Strings”

The last part of a signal name can have a maximum length of 20 characters. This is equal to the length of a line in the OP display and the maximum length of an ID string in TAC Vista. The part must not consist of digits only.

The signal name can be composed of two parts: the Module and Signal part. These parts are divided by the character “\” (backslash). For example, ECON\MAT, where the string ECON is an optional module part of the signal name.

For more information on the module part of a signal name, see Section 11.4, “Modules in Signal Names”.

11.3 Public Signals

Many signals in an application represent intermediate values. In most applications, only a reduced number of the signals in the application are useful for other nodes in the network. Declaring the signals of interest as public signals allows them to be read, and modified by other nodes.

Public signal names can not differ only in using upper and lower case characters. If so, they are considered identical and an error message is shown at compilation.

All public signals in the application are compiled in the Public Signal Table, an automatically compiled part of the program specification.

There are two types of signals in the table:

- DIG: Binary signal.
- ANA: Analog signal (real or integer).

Signals are also classified as signal as read-only or read/write. For more information on reading and writing a signal, see Section 11.6, “Accessing Signals”.

11.4 Modules in Signal Names

You can add a module part to the name of a signal to make signals with same name unique. Using modules in the signal name enables you to divide an application into separate systems, such as multiple heating groups. This provides two major advantages:

- You can use the same signal name in multiple systems, and make them unique using the module part in the name. For example, AHU1\T1_SP and AHU2\T1_SP.
- All public signals in the same Module are treated as a group and located together in the database when they are imported to a supervisory system like TAC Vista.

The Module name is a part of the signal name for a block in a module, using the syntax:

- Module name\Signal.

The optional Module part of a signal name can have a maximum length of 12 characters. The Module name can be entered or edited for a single block or a group of blocks. For more information on editing a module name, see Chapter 13.6.10, “Editing the Module Name”.

A block in a module is indicated in the function block diagram by showing the block type underlined and with purple letters.

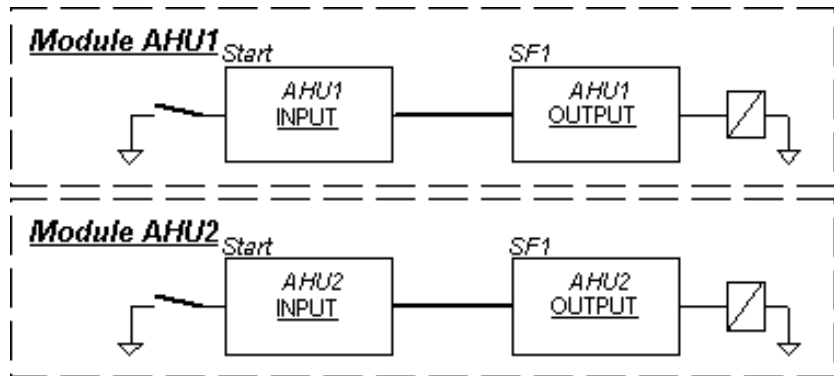


Fig. 11.1:

11.5 Signal Names for TAC Network Variables

Signals in other TAC devices in the network can be used in the application as inputs, configured as TAC Network Variables (TACNVs).

You define the network variable as the network address for the public signal in the other TAC device. The network address is a character string of maximum length 20\12\20 characters. You can not change the network address when the application is running.

11.5.1 Signals without a Module Part in the Name

The syntax for addressing a TAC Network Variable without a module part in the signal name is the following:

- \Device\Application\Signal,

where “Device” is the name of the TAC Xenta device in the network.

“Application” is the name of the application, as entered in the **Name** box in the **Program specification** dialog box.

11.5.2 Signals with a Module Part in the Name

When the public signal name includes a module part, you use the Module name in the Network address for the TAC Network variable (TACNV). The following syntax is used for the network address:

- `\Device\Module\Signal`

where “Device” is the name of the TAC Xenta device in the network, for example `\RPU1\AHU1\OutdoorTemp`.



Note

- You can also address signals without Module name using the syntax:
`\Device\Signal`
Use double backward slashes and , omit the application name.

11.6 Accessing Signals

The permissions for a user to view, enter, or modify a block output signal is defined by the access class for the signal. The following classes are used:

- RO: Read-only. The signal can be read from the OP and/or from other nodes in the network, but not modified.
- RW: Read/write. The signal can be read and modified from the OP and/or from other nodes in the network.

The signals in blocks where the computed output value depends only on the current value(s) of the input signal(s) are read-only (RO). The AI, DI, AO, and DO blocks are examples of such function blocks.

If the output of such a block is modified via the network, the modification will be restored by the application in the next program cycle. Subsequent blocks that use the signal, may detect and reflect the transitory change.

For other block types, such as integrators and accumulators, you can allow external write operations on the block outputs. An example of this is when you modify the value of a block output and the new value is computed by adding or subtracting the increment of the previous output value.

11.7 Input and Output Signals

Signals from connected equipment are introduced to the function block diagram for TAC Xenta 280/300/401 devices, using the function blocks for physical I/Os.



Important

- A Menta application designed for a Xenta 700 has no I/O blocks for physical signals. The Menta application uses so called connection blocks for inputs and outputs. These connection blocks are connected to signals in physical I/O. For more information on connection blocks, see Chapter 21.1.1, “Connection Blocks”

You can configure the function blocks for either physically connected signals or signals received via the connected network. Depending on the block type and binding, you use different sets of configuration parameters in the I/O function blocks:

- Physical terminals for inputs and outputs
- Network Variable
- Online device
- SNVT

Two special options are available for certain I/O blocks:

- Constant Value
- Not connected

You can view a list of the bindings of all I/O blocks in the I/O Configuration Table.

| Name | Type | Bound to | Override |
|----------------------------|------|--------------------------|----------|
| Term_UnitsWAV_4:2_S... | AI | temp_p_nvi4:2_SetPt | |
| Term_UnitsWAV_4:3_S... | AI | temp_p_nvi4:3_SetPt | |
| Term_UnitsWAV_4:4_S... | AI | temp_p_nvi4:4_SetPt | |
| Term_UnitsWAV_4:5_S... | AI | temp_p_nvi4:5_SetPt | |
| Term_UnitsWAV_4:6_S... | AI | temp_p_nvi4:6_SetPt | |
| Econ\RAT_Sensor | AI | Non linear input (M3-B2) | |
| Econ\CO2 | AI | Linear input (M3-U3) | |
| Econ\RAHumiditySen | AI | Linear input (M3-U2) | |
| Relief\Rldg_Static | AI | Linear input (M3-U4) | |
| Relief\Relief_Air_Dmprs... | DOPU | Output pulse (M1-K3) | |
| Relief\Relief_Fan | DO | Digital output (M1-K2) | |
| VSD\Mod3 | DI | Online device | |
| OA_Temp | AI | Non linear input (M3-B4) | |
| Econ\DAHumiditySen | AI | Linear input (M5-U1) | |
| Econ\CO2_SNVT | AO | ppm_nvoCO2 | |
| Cooling\Pulse | CNT | Pulse counter (M1-X2) | |
| DDI | DI | Digital input (M0-X1) | |
| ao1 | AO | Analog output (M0-Y1) | |

The transfer of data between TAC Xenta application programs are programmed with the generic input blocks AI and DI in accordance with the principle of reading data from other nodes, not writing to them.

For more information on the I/O function blocks, see Chapter 21.1.2, “Physical I/O Blocks”.

11.7.1 The Physical Terminal Option

You use the physical terminal option when signals are connected to physical terminals in TAC Xenta devices or I/O modules.

When the I/O block is bound to a physical terminal, the module number and terminal reference are displayed next to the I/O block symbol in the function block diagram.

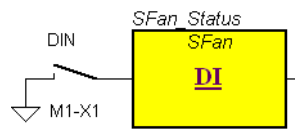


Table 11.1: Options in the Bind dialog box

| | |
|---------------|---|
| Mod Number | The Mod Number list shows the available terminals. A TAC Xenta device is displayed by name: for example, Xenta 301. Xenta I/O modules are displayed by module number and model: for example, M1 (422). |
| Terminal Ref. | The Terminal Ref list shows the available terminals in the device or I/O module. The terminals are shown with the type and number: for example B1-B4, U1-U4, X1-X4, Y1-Y4 and K1-K4. Already allocated terminals are indicated by the Number sign (#) after the point designation. |

11.7.2 The Network Variable Option

You can use a public signal or constant from another device in the network using an I/O block and specifying the network address for the required variable. You can allow several controllers to share a sensor signal of common interest, such as outdoor temperature, which is physically connected to one of them.

For more information on naming the network variables, see Section 11.5, “Signal Names for TAC Network Variables”.

When the I/O block is bound to a remote network signal, “NET” is displayed next to the I/O block symbol in the function block diagram.

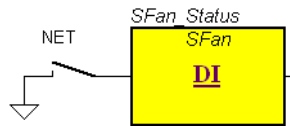


Table 11.2: Options in the Bind dialog box

| | |
|-----------------|---|
| Network Address | <p>The Network Address is a reference to a public signal in another TAC Xenta device in the network.</p> <p>The network address in a digital input (DI) block must always point to a binary signal in the remote device.</p> <p>The Network address in an analog input (AI) block can point to either a real or an integer signal. The output of the AI block always is real.</p> |
| Delta | <p>A Delta is the smallest change of the signal value that causes an update via the network. The default value is 0.5.</p> |
| Period | <p>The Period is an integer representing the maximum time interval in seconds between two updates of the value.</p> <p>The value is updated when the time interval has passed after the last update, whether the signal has changed or not.</p> <p>Default value is 60.</p> |



Note

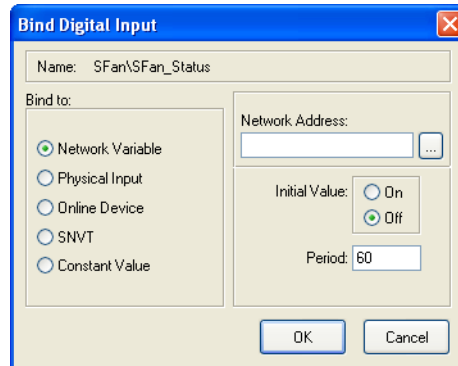
- The values for **Delta** and **Period** should always be selected in so that a high network load is avoided. Try updating signals via the network on a 60 or a 30 second basis, rather than every second.

11.7.3 Finding the Address for a TAC Network Variable in the Database

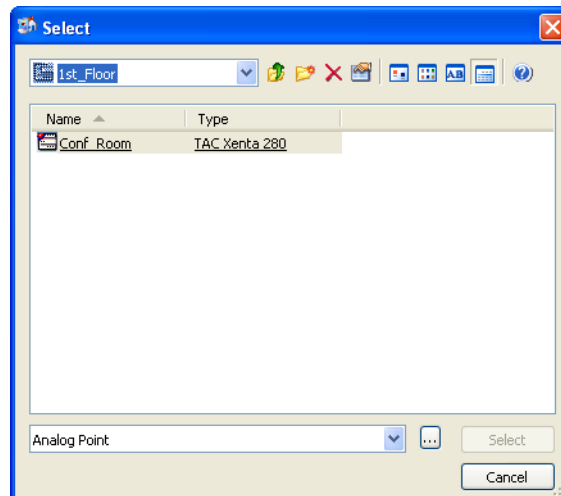
When you edit an I/O block where you want to use a TAC network variable you can browse the TAC Vista database to find a network variable.

To find the address for a TAC network variable in the database

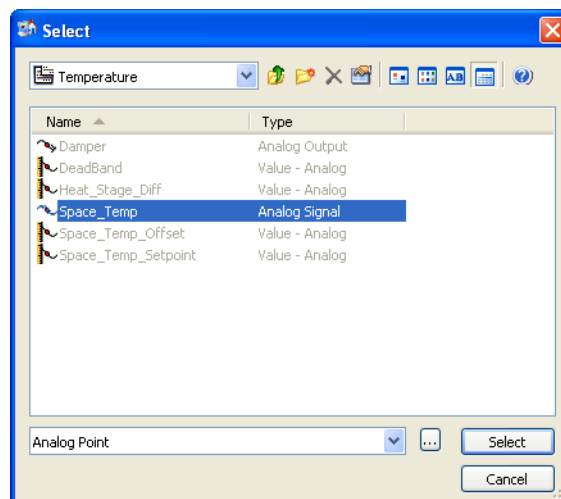
- 1 In the **Bind Input** dialog box, in the **Network Address** box, click the browse button.



- 2 In the **Select** dialog box, browse to the signal in the TAC Vista database.



- 3 Select the variable in the network and click **Select**.



11.7.4 Finding the Address for a Network Variable

When you edit an I/O block where you want to use a TAC network variable, you can also search the *.MTA file to find a network variable.

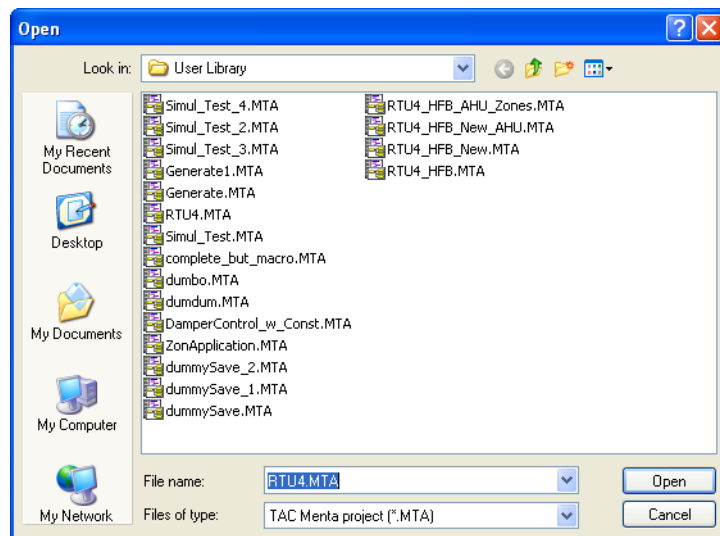


Important

- Ensure that the network address string is retained when the Xenta device is connected in the network.
The name of the Xenta device in the network can differ from the name specified in the application.

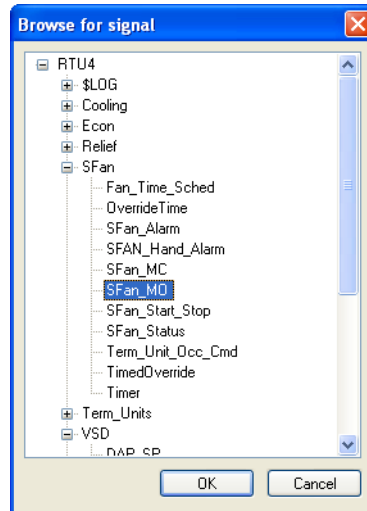
To find the address for a network variable

- 1 In the **Bind Input** dialog box, in the **Network Address** box, click the browse button.
- 2 In the **Open** dialog box, browse to the application file (.MTA file) that contains the variable.

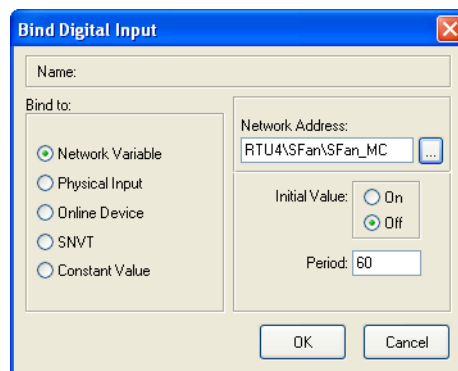


- 3 Click **Open**.

- 4 In the **Browse for signal** dialog box, select the variable and click **OK**.



- 5 You can view the network address in the **Network Address** box.



- 6 Click **OK**.

11.7.5 The Online Device Option

In applications where alternative actions must be taken if the communication has ceased, you might need information on the status of other devices in the network.

You can use the online device option of the digital input function block (DI) to monitor the communication status of an I/O module or another device in the network.



Important

- The **Online Device** option can only monitor the I/O modules defined for the TAC Xenta device where the application is running.
- The block output is true (1) if the I/O module specified in the Network Address is online. Otherwise, the block output is false (0).

You can also use the online device option to monitor the communication status of a monitoring and supervising system, such as TAC Vista, by specifying the name of the LonWorks network as the network address.

When the I/O block is configured to monitor the network status, “DEV” is displayed next to the I/O block symbol in the function block diagram.

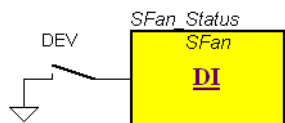


Table 11.3: Online device option in the Bind dialog box

| | |
|----------------|--|
| Device Address | <p>The device address is the name of the monitored device, I/O module, or network.</p> <p>The name is a character string.</p> <p>The I/O module is specified as the number of the I/O module as it is numbered in the Device Configuration; for example 1, 2 or 3</p> <p>The output of the digital input block is true (1) if the monitored device, I/O module, or network is online. Otherwise the output is false.(0).</p> |
|----------------|--|

For more information on the online device option, see Chapter 21.18, “DI – Digital Input”.

11.7.6 The SNVT Option

Signals from other devices in the network can also be used in the application by configuring the physical I/O block using the SNVT option. This option is primarily intended for communicating with equipment from other manufacturers, such as lighting or intelligent actuators, via the network.



Important

- An I/O block, configured with the SNVT option, must also be bound to a variable of the same SNVT type using a LonWorks binding tool.
- There will be no network communication between two bound SNVT variables, unless either the output variable is set to Send a new value periodically (or on Delta change), or the input variable is set to Poll the output periodically.

When the I/O block is bound to a SNVT variable, “SNVT” is displayed next to the I/O block symbol in the function block diagram.

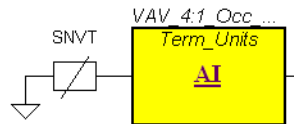


Table 11.4: SNVT option in the Bind dialog box

| | |
|---------------|---|
| Type | The variable type is selected from the pre-defined types list. A structured type of SNVT consists of several variables which are called the members. The members are automatically available in the Members list box. |
| SNVT Name | The signal name is stored in the .XIF file and is displayed in external tools. The maximum length of the name string is 16 characters. |
| Initial Value | The Initial Values parameter specifies the initial block output value. |

Table 11.4: SNVT option in the Bind dialog box

| | |
|--------|---|
| Period | <p>The period value defines the maximum time interval, in seconds, between two updates of the value.</p> <p>Setting the Period value to 0 (zero) for an output block (AO or DO) updates the external variable only on changes according to the Delta parameter.</p> <p>The default value is 60.</p> |
| Delta | <p>The Delta parameter is the smallest change of the signal that causes an update via the network.</p> <p>The Delta parameter is only used for output SNVTs.</p> |
| Send | <p>The Send option specifies whether the external input variable is automatically updated or not.</p> <p>The Send option is only used for output SNVTs.</p> |
| Poll | <p>The Poll option defines whether the external signal is polled or not.</p> <p>The Poll option is only used for input SNVTs.</p> |



Notes

- A structured SNVT consists of several variables, which are called the members.
- Each member of the structured SNVT you want to use in the application must be represented by a block in the function block diagram.
- Normally, you use function blocks for all members of the structured SNVT.
- The TAC Macro block library includes Macro blocks for all supported SNVTs of structured type.

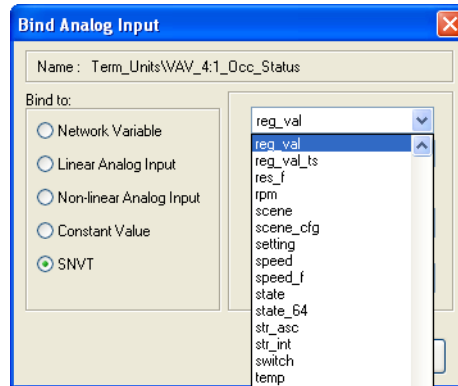
For more information on the SNVT option, see Chapter 21.1.2, “Physical I/O Blocks”.

When you configure the I/O block as SNVT you name the SNVT: for example, "nvi4:1_SpaceTemp". The name may differ from the block name. The type values which are, optional Members of a structured SNVT, initial value, period, and delta are also configured for the SNVT.

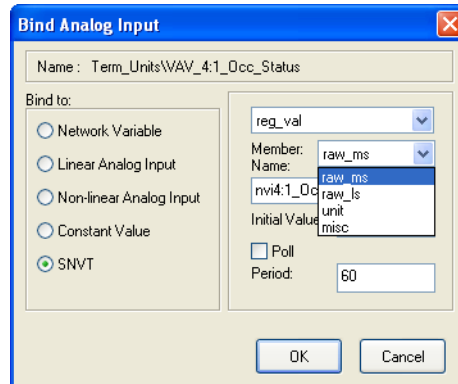
You use a LonWorks binding tool to bind the defined SNVT to a signal of the same SNVT type.

Binding to a SNVT

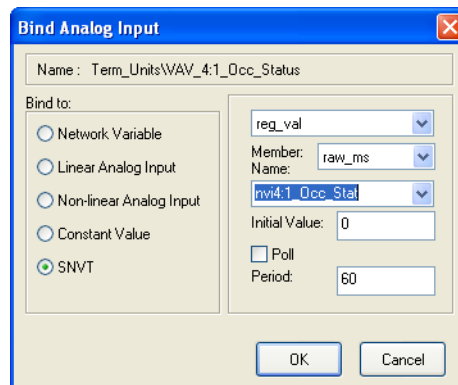
- 1 In the **Edit block** type dialog, click **Bind**.
- 2 In the **Bind** dialog for the I/O, click the **SNVT** option.
- 3 In the SNVT type list, select the required SNVT.



- 4 If the SNVT is a structured type, select the required Member in the **Member** list.



- 5 In the **Name** box, type a name for the SNVT.



- 6 You can select the **Poll** option for an input I/O block.
- 7 You can select the **Send** option for an output I/O block.

- 8** In the **Delta** box, enter a value for an analog output I/O block.
- 9** In the **Period** box, enter a value for the maximum time interval between updates.
- 10** In the **Initial Value** box, enter a value for the initial block output value.
- 11** If the I/O signal is a digital signal, click the required initial value option either **On** or **Off**.

12 The Mouse and Function Keys

12.1 The Mouse

The options for using the mouse vary, depending on whether you are working in the Edit or the Simulation mode.

Each action has a different effect depending on the area where the cursor is when the action is taken. The possible areas of action for the mouse in the application program window are:

Table 12.1:

| | |
|------------|---|
| Block | Rectangle composed of the yellow area in the function blocks. |
| Output | All function blocks have an exit which is represented by a small segment protruding from the right hand side of the symbol. |
| Input | Some function blocks have inputs which are shown as arrowheads. |
| Connection | This is any intermediate point on a direct line connection segment. |
| Node | The end points of a connection are denominated nodes. |
| Background | Any area of the application program window which is not one of the above areas belongs to the background. |

Finally, the definitive option depends on the editing context. Various possibilities exist for the context, distinguished by the cursor shape:

- Normal, the cursor is the classical arrow.
- Block insert from Tool bar, the cursor is an arrow with a block.
- Zoom in, the cursor is a magnifying glass.
- Group selected, the cursor is an open hand.
- Draw a connection, the cursor is an X. The cursor symbol changes to an X within a square when it is possible to connect to an input.
- Open a selection rectangle, the cursor is a cross.

- Draw a group, the cursor is crossing arrows.

12.2 The Function Keys

Table 12.2:

| Key | Description |
|--------|---|
| Delete | Deletes a selected block or group of blocks. |
| F1 | Help. |
| F2 | Starts or stops (Toggle) the continuous execution of the application program. Same as clicking on the Execute button in Simulation mode. |
| F3 | Stepwise execution of the application program. Same as clicking on the Step button in Simulation mode. |
| F5 | Toggles between FBD and tabular simulation mode. |
| F7 | Searches for a block or a comment that contains a given text string in the block name, the block type, or in a parameter. Same as the Edit – Find command. The key works in both Edit and Simulation mode. |
| F8 | Searches for a given text string in comments, block names, and block parameters and replaces the string with the entered text string. Same as the Edit – Replace command in Edit mode. |
| F9 | Moves the marked area to the center of the screen. Same as the Edit – Center selection command. The key works in both Edit and Simulation mode. |
| F10 | Resets the program cycle counter to zero and all function block diagram signals to their initial state. Same as clicking on the RST button in Simulation mode. |
| F11 | Toggles between Online and Offline mode. Same as clicking on the Online button in Simulation mode. |
| F12 | Toggles between <i>Edit</i> and <i>Simulation</i> mode. |
| CTRL+I | Zooms in. |

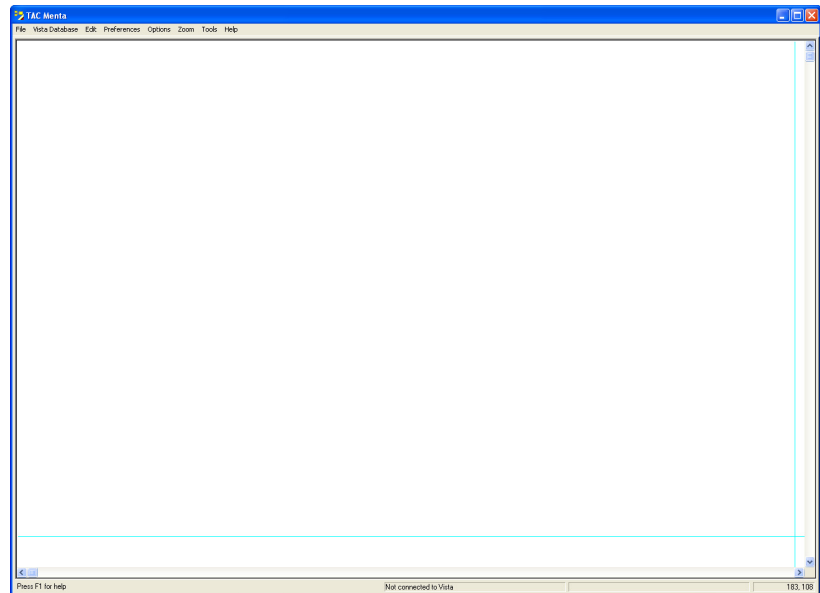
Table 12.2: (Contd.)

| | |
|-----------------|---|
| CTRL+O | Zooms out. |
| CTRL+X | Cuts text in dialogs. |
| CTRL+C | Copies the selected group of blocks or copies text in dialogs (same as <i>Ctrl-Insert</i>). |
| CTRL+V | Pastes the copied group of blocks or pastes text in dialogs (same as <i>Shift-Insert</i>). |
| CTRL+Z | Undoes a deletion. |
| CTRL+Insert | Copies the selected group of blocks or copies text in dialogs. |
| Shift+Insert | Pastes the copied group of blocks or pastes text in dialogs. |
| Tab | Zooms in (same as <i>Ctrl-I</i>). |
| Esc | Cancel the operation (same as Cancel button in dialogs). |
| Arrow key | Moves the FBD window in the direction of the arrow key (Left, right, up or down). Available in both Edit and Simulation mode. |
| Shift+Arrow key | Moves the FBD window one screen page in the direction of the arrow key (Left, right, up or down). Available in both Edit and Simulation mode. |
| CTRL+Arrow key | Moves the cursor in the Trend logging window one sample in the direction of the arrow key (Left or right). |

13 The Edit Mode

When a user starts TAC Menta, it always starts in the edit mode.

The edit mode window has a single work area with scroll bars. The editing window is often called the diagram window. Use the editing window to create the function block diagram (FBD) for the application.



You work in the edit mode to edit files:

- open and save application files.
- specify the devices and the application.
- add function blocks to the function block diagram.
- configure function blocks.
- define constants.
- define alarm texts
- make connections between function blocks.
- add comments to the function block diagram.
- edit associated text files

13.1 The Device Specification

When you start to edit a new application in TAC Menta, you are asked to define the environment for the application.

You define the hardware environment for the application by specifying the type of TAC Xenta device and optional Xenta I/O modules:

- The type of device
- The system software version for the device
- The hardware version for the device.
- The number and types of optional TAC Xenta I/O modules.
- The number and types of optional TAC STR modules.

When you specify the device and Xenta I/O modules, TAC Menta makes it easier to allocate physical I/O points.



Important

- The **Device Specification** command is not available for the Xenta Server 700 devices.

13.1.1 Specifying the type of TAC Xenta Device

The type and version of system software and hardware for the TAC Xenta device in which Menta application shall be executed is configured.

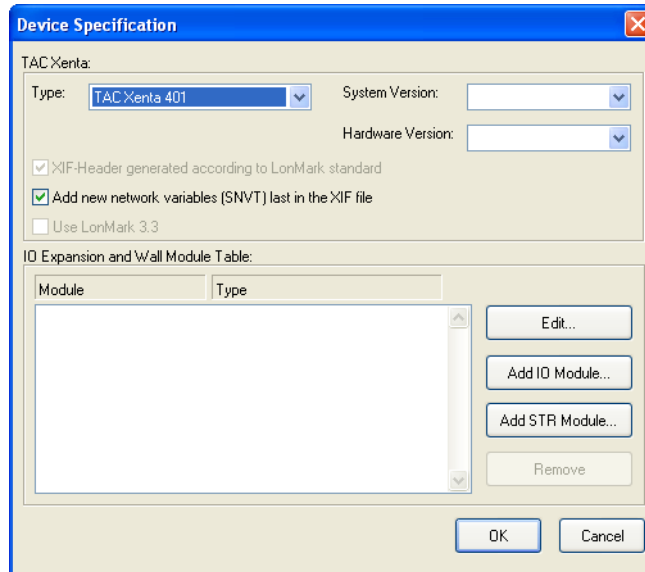
Table 13.1:

| | |
|------------------|--|
| Type | Select in this pre-defined list the type of TAC Xenta device for the current application. |
| System Version | Select in this list the system software version for the TAC Xenta device. The version is used to create downloadable files compatible with the system software in the device. |
| Hardware Version | Select in this list the hardware version of the chosen TAC Xenta device 281/282/301/302/401. To use local trend logs in a TAC Xenta 300 you must use hardware version 2 for the device. |

To specify the type of TAC Xenta device

- 1 In the **Options** menu, click **Device Specification**.

- In the **Device Specification** dialog box, in the **Type** box, select the type of Xenta device.



- In the **System Version** box, select the system version for the Xenta device.
- In the **Hardware Version** box, select the hardware version for the Xenta device.
- Click **OK**.

13.1.2 Setting the Characteristics For the XIF File

When you define the type of TAC Xenta, you can also determine some characteristics for the XIF file that is automatically generated by TAC Menta.

Table 13.2:

| | |
|---|--|
| <p>XIF header generated according to LonMark standard</p> | <p>Select this check box to generate an XIF file where the LonMark approved program ID for the device is included in the first row of parameters. This option applies to TAC Xenta with system software version 3.3 or later.</p> <p>IMPORTANT. If the XIF file for the application is changed by using this option, you must re-make all its bindings to the device using a LonWorks binding tool.</p> |
|---|--|

Table 13.2: (Contd.)

| | |
|--|---|
| <p>Add new network variables (SNVT) last in the XIF file</p> | <p>Select this check box to place new network variables (SNVT) last in the XIF file when an existing Menta application is edited.</p> <p>When you add new network variables (SNVT) last you only need to bind new variables using the LonWorks binding tool. Existing bindings are unaffected.</p> <p>IMPORTANT. If a network variable is deleted, or if an existing I/O block is altered to “SNVT”, you must re-make all bindings. The first time the option is used for an existing application, all existing bindings must be re-made. The option is by default selected in TAC Menta version 3.24 and later.</p> |
| <p>Use LonMark 3.3</p> | <p>Select the Use LonMark 3.3 check box to make a TAC Xenta 283, with system software 3.6 or higher, works as a LONMARK 3.3 approved real time keeper, using UTC time as input and returning local time as output.</p> <p>If the check box is cleared, a LONMARK version 3 XIF file is generated.</p> |

To set the characteristics for the XIF file

- 1 In the **Options** menu, click **Device Specification**.
- 2 Select or clear the **XIF header generated according to LonMark standard** check box, as required.
- 3 Select or clear the **Add new network variables (SNVT) last in the XIF file** check box, as required.
- 4 Select or clear the **Use LonMark 3.3** check box, as required.
- 5 Click **OK**.

13.1.3 Adding a TAC Xenta I/O Module

When you specify the Xenta 280/300/401 device, you can also add optional TAC Xenta I/O modules.

The following parameters are available for the TAC Xenta I/O modules:

Table 13.3:

| | |
|--------------------|--|
| Module | Definition of the number (Module1, Module2 etc.) of each of the I/O modules in the current application. New modules can be added, but only the last module in the list can be removed (deleted). Before an I/O expansion module can be removed from the list, all bindings to its I/O points must be removed (detached). The same applies when an I/O expansion module is changed in type and I/O points are influenced. |
| Type | The type (TAC Xenta 411, TAC Xenta 412 etc.) of each of the I/O modules in the current application, selected from a pre-defined list of supported types. The list is defined in the TATYPE.INI file. |
| Min Send Time | Defines how often the I/O module communicates with the TAC Xenta device. The default and minimum value is 500 ms, but the value can be increased to reduce network traffic. |
| Fast CNT Reporting | Defines whether fast reporting of the status changes of pulse inputs (CNT) is to be used. The function is used when connecting push buttons for lighting control. Note that the function increases network traffic. The box is not checked by default. Reporting a status change takes about 30 s, which can be used for energy pulse metering, for example. |

To add a TAC Xenta I/O Module

- 1 In the **Options** menu, click **Device Specification**.

- 2 Under IO Expansion and Wall Module Table, click the **Add IO Module** button.



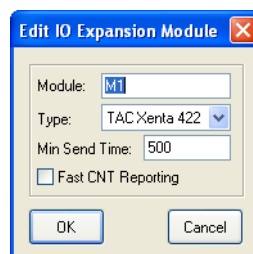
- 3 In the **Edit IO Expansion Module** dialog box, you can type an optional name for the I/O module in the **Module** box.
- 4 In the **Type** box, select the I/O module type.
- 5 In the **Min Send Time** box, enter the a value.
- 6 Select the Fast CNT Reporting check box if required.
- 7 Click **OK**.

13.1.4 Editing a TAC Xenta I/O Module

You can edit Xenta I/O modules.

To edit a TAC Xenta I/O module

- 1 In the **Options** menu, click **Device Specification**.
- 2 In the **IO Expansion and Wall Module Table** list, select an I/O module.
- 3 Click **Edit**.
- 4 In the **Edit IO Expansion Module** dialog box, you can change the options.



- 5 Click **OK**.

13.1.5 Removing a TAC Xenta I/O Module

You can remove Xenta I/O modules.

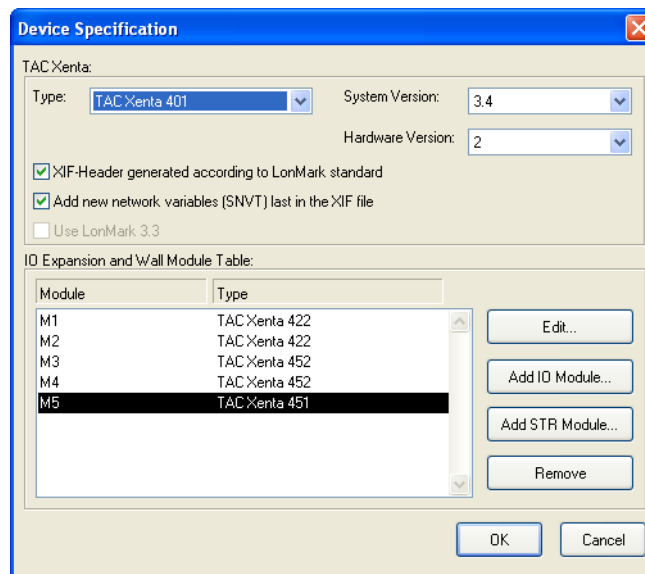


Important

- You can only remove the last item in the **IO Expansion and Wall Module Table** list.

To remove a TAC Xenta I/O module

- In the **Options** menu, click **Device Specification**.



- In the **IO Expansion and Wall Module Table** list, select the last I/O module.
- Click **Remove**.
- Click **OK**.

13.1.6 Adding an STR Wall Module

When you specify the Xenta device, you can also add optional STR Wall modules.

The following parameters are available for the STR Wall modules:

Table 13.4:

| | |
|--------|---|
| Module | Definition of the number (S1, S2 etc.) for each of the STR Wall modules in the current application. New STR Wall modules can be added, but only the last I/O or STR Wall module in the list can be deleted. |
|--------|---|

Table 13.4: (Contd.)

| | |
|--|---|
| Type | Type (STR350/STR351 etc.) for each of the STR Wall modules in the current application, selected from a pre-defined list of supported types. |
| Min send time | Defines how often the STR Wall module is allowed to communicate with the TAC Xenta device. Default and minimum value is 500 ms, but the value can be increased to reduce the network traffic. |
| Minimum change for update (TempMinDelta) | Minimum change in the space temperature value to initiate an update message from the STR Wall module. |
| Space temperature offset (TempOffset) | Offset value to calibrate space temperature sensor. |
| Temperature display resolution (Resolution) | Resolution of the temperature display. |
| Display time-out (DispTimeout) | Time in seconds before the display reverts to the default display. 0 means that a selected function remains on the display indefinitely. |
| Backlight time-out (BackLightOn) | Time in minutes before the display backlight (not available on STR350) turns off. 0 means the disabled backlight is disabled. |
| Temperature setpoint low limit (SetpointLow) | Lower limit for the temperature setpoint. Absolute or Offset value depending on configuration (Options1). When this is an Offset value, a negative value is used, irrespective of the sign of the entered value. |
| Temperature setpoint high limit (SetpointHigh) | Upper limit for the temperature setpoint. Absolute or Offset value depending on configuration (Options1). |
| HVAC Settings (Options1) | Bits are used to set the STR HVAC configuration. Each bit box is described in the Bit(s) description group box when the cursor is put in the bit box. |

Table 13.4: (Contd.)

| | |
|---------------------------------|--|
| Auxiliary Options (Options2) | Bits are used to set the STR auxiliaries configuration. These may include Sunblinds, Lighting, and sensors for CO2, RH, or occupancy. Each bit box is described in the Bit(s) description group box when the cursor is put in the bit box. |
| Spare Options (Options3) | Spare options for future use. |

These configuration settings are realised with a number of PVR - STROUT pairs, put in a separate module, which can be found with the HFB Navigation Tree. Normally, there is no reason to change these settings.

When the device configuration has been made, all binding tables will contain the current configuration information. The Mod Number parameter will show the TAC Xenta and defined I/O or Wall module numbers.

To add a STR Wall module

- 1 In the **Options** menu, click **Device Specification**.
- 2 In the IO Expansion and Wall Module Table, click the **Add STR Module** button. In the **Edit STR Wall Module** dialog box, in the **Module** box, type an optional name for the STR Wall module.
- 3 In the **Min Send Time** box, enter a value.
- 4 In the **Minimum change for update** box, enter a value.
- 5 In the **Space temperature offset** box, enter a value.
- 6 In the **Temperature display resolution** box, enter a value.
- 7 In the **Display timeout** box, enter a value.
- 8 In the **Backlight timeout** box, enter a value.
- 9 In the **Temperature setpoint low limit** box, enter a value.
- 10 In the **Temperature setpoint high limit** box, enter a value.
- 11 In the HVAC Settings (Options1) boxes, enter a Bit pattern.
- 12 In the Auxiliary Options (Options2) boxes, enter a Bit pattern.
- 13 Click **OK**.



Note

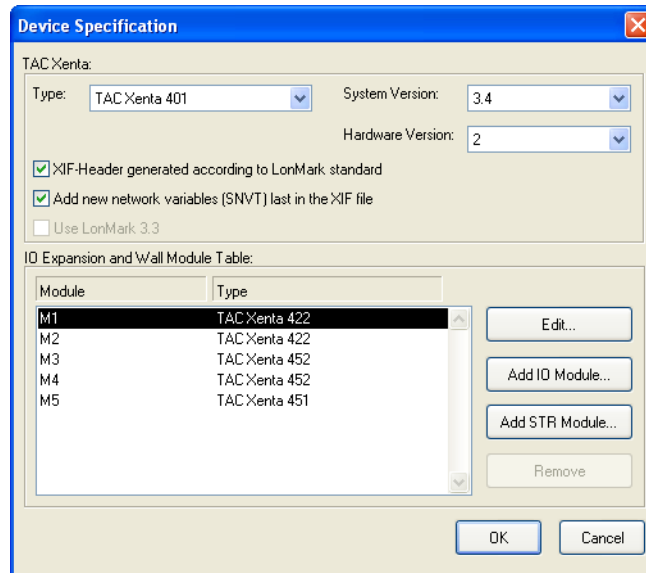
- To view an explanation to each box (bit) you can click the box and read a description in the **Bit Description** box.

13.1.7 Editing an STR Wall Module

You can edit already created STR Wall modules.

To edit an STR Wall module

- 1 In the **Options** menu, click **Device Specification**.



- 2 In the **IO Expansion and Wall Module Table** list, click the required STR wall module.

- 3 Click the **Edit** button.

- 4 In the **Edit STR Wall Module** dialog box, change the options.
- 5 Click **OK**.

13.1.8 Removing an STR wall module

You can remove already created STR Wall modules.



Important

- You can only remove the last item in the **IO Expansion and Wall Module Table** list.

To remove an STR wall module

- 1 In the **Options** menu, click **Device Specification**.
- 2 In the **IO Expansion and Wall Module Table** list, click the required STR Wall module.
- 3 Click the **Remove** button.
- 4 Click **OK**.

13.2 The Program Specification

When you start to edit a new application in TAC Menta you are also asked to define a couple of important details about the Menta application:

- The application name
- The cycle time for the application



Important

- The Program Specification is not available for the Xenta Server 700 devices.

Further information can be added using the dialog box.

Several details, for example number of blocks and I/O signals, are automatically generated and read only.

A useful part of the program specification is the list of all public signals in a table.

The program specification also contains all information about the application, required to install the application in the TAC Xenta device

Table 13.5:

| | |
|--------|---|
| Name | The Name box contains the name of the application, defined by the user as a string with a maximum length of 20 alphanumeric characters. For information on allowed characters in the name string, see Chapter 9.3, “Allowed Characters in Name Strings” |
| Abb | You can use the Abbreviation box to enter a string used to name the application in the OP only. The string length can be up to 4 characters. All characters that can be handled by the OP are legal. |
| Type | You can use the Type box to describe the type of application using a string of arbitrary length. |
| Author | You can use the Author box to name the author using a string of arbitrary length. |
| Date | A read-only field that contains the date when the application last was edited. |

Table 13.5: (Contd.)

| | |
|---------------------|--|
| Cycle time | <p>The execution interval (ms) for the application when executed in the device.</p> <p>The cycle time can be configured from 500 ms up to 65 536 ms.</p> <p>The default value is 1000 ms.</p> <p>The parameter does not affect the execution of the application program during offline simulation.</p> |
| Blocks | <p>The actual number of blocks in the program.</p> <p>The information is only available after a compilation of the program.</p> |
| I/O signals | <p>The number of physical input and output signals, used by the application.</p> <p>The information is only available after a compilation of the program.</p> |
| Public signal table | <p>A read-only list, compiling all public signals and constants in the application.</p> <p>The signals and parameters are sorted in alphabetical order.</p> <p>The list has four columns showing the signal identifier, type, access class and unit.</p> |
| Standard App. | <p>The box indicates that the current application is a standard application when checked. Otherwise the application is a Customer specific application.</p> <p>See also Chapter 9.5, “Marking Standard Applications/Controllers”.</p> |
| Program ID | <p>A read-only, automatically generated string that contains information about the application and it’s signals.</p> |

13.2.1 Naming the Application

The name of the application is used as a header when printing the application and sometimes when addressing a TAC Network variable.

To name the application

- 1 In Edit mode, in the **Options** menu, click **Program Specification**.

| Used Resources: | | I/O Signals | | | |
|-----------------|-----|-------------|------|------|--|
| Blocks | DIs | AIIs | DOIs | ADIs | |
| 0 | 0 | 0 | 0 | 0 | |

XIF file information

Program ID: 80:00:13:52:00:06:04:4D

Public Signal Table:

| Identifier | Type | Access | Units |
|----------------------|------|--------|-----------|
| Cooling\C_Usage | ANA | RW | kWh |
| Cooling\C1_Alarm | DIG | RO | Normal/AL |
| Cooling\C1_Min_Off | ANA | RW | Seconds |
| Cooling\C1_Min_On | ANA | RW | Seconds |
| Cooling\C1_Start_... | PAR | RW | % |
| Cooling\C1_Start_... | DIG | RO | |
| Cooling\C1_Status | DIG | RO | |
| Cooling\C1_Stop_... | PAR | RW | % |
| Cooling\C2_Alarm | DIG | RO | Normal/AL |
| Cooling\C2_Min_Off | ANA | RW | Seconds |
| Cooling\C2_Min_On | ANA | RW | Seconds |

- 2 In the **Program Specification** dialog box, in the **Name** box, type the required name.
- 3 Click **OK**.

13.2.2 Using the Abbreviated Name for the Application

The abbreviated name of the application is used in the menus for the operators panel, (OP).

To use the abbreviated name for the application

- 1 In Edit mode, in the **Options** menu, click **Program Specification**.
- 2 In the **Program Specification** dialog box, in the **Abb** box, type the required abbreviation.
- 3 Click **OK**.

13.2.3 Defining the Cycle Time for the Application

Select a cycle time for the application that is long enough for executing the complete application.

If the execution time is longer than the cycle time, the application is not executed within the designed cycle time.

All application code will be executed with the necessary execution time, but the next execution of the task starts when the application is scheduled to start. The resulting cycle time is then as much as twice the time that was intended.

To define the cycle time for the application

- 1 In the edit mode, in the **Options** menu, click **Program Specification**.
- 2 In the **Program Specification** dialog box, in the **Cycle Time** box, enter the required cycle time.
- 3 Click **OK**.

13.3 Adding Function Blocks to the Diagram

Function blocks in TAC Menta are of the following types:

- Simple blocks
- Operators
- Expressions
- Test probes

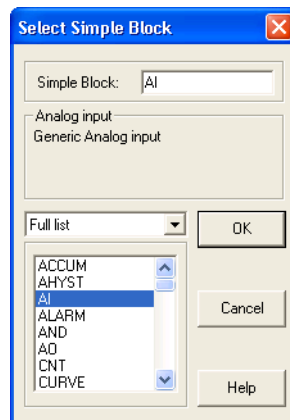
13.3.1 Adding a Simple Block

You can add simple blocks to the function block diagram.

When you add the function block to the diagram window it is placed at position where the cursor was placed when right-clicking. The function block is selected, framed in green.

To add a simple block

- 1 Right-click in the diagram window.
- 2 On the menu, click **Simple Block**.
- 3 In the **Select Simple Block** dialog box, in the function list, select the required function block.



- 4 Click **OK**.
- 5 Move the function block to a suitable position in the diagram window.



Note

- The function block remains as a floating selection in the diagram window if there is not enough free area for the block.

- 6 Click in the diagram window to deselect the function block.



Tips

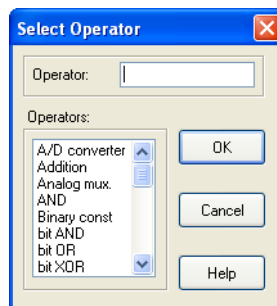
- To choose a function block type you can also do one of the following:
 - double-click a block type in the function block list.
 - Manually enter the type of the block in the **Simple Block** box and then click **OK**.
- You can reduce the function block list by selecting one class only in the classes list.
- You can add a block of the same class (simple block, operator or expression) as the last created block by double-clicking on the work area.

13.3.2 Adding an Operator

When you add an operator to the diagram window it is placed at the cursor position where you started right-clicking. The operator is selected and framed in green.

To add an operator

- 1 Right-click in the diagram window.
- 2 On the menu, click **Operator**.
- 3 In the **Select Operator** dialog box, in the **Operators** list, select the required operator.



- 4 Click **OK**.
- 5 Move the operator to a suitable position in the diagram window.



Note

- The operator remains as a floating selection in the diagram window if there is not enough free area for the block.

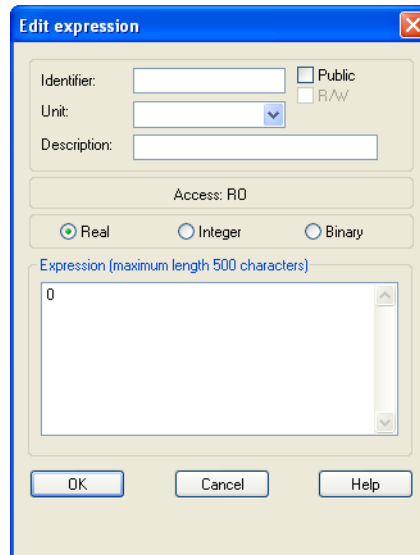
13.3.3 Adding an Expression Block

Expression blocks are blocks with an arithmetic or logical expression, which may be simple or complex.

For more information on expression blocks, see Chapter 22, “Expression Blocks”.

To add an expression block

- 1 Right-click in the diagram window.
- 2 On the menu, click **Expression**.



- 3 Click the required expression type option, **Real**, **Integer** or **Binary**.
- 4 In the **Edit expression** dialog box, in the **Expression** box, enter the required expression.
- 5 Click **OK**.
- 6 Move the expression block to a suitable position in the diagram window.



Important

- The size of the expression block and the number of inputs varies, depending on the defined expression.



Note

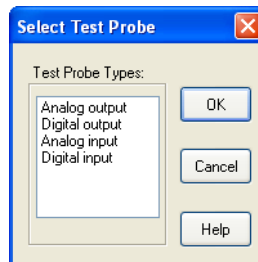
- The expression block remains as a floating selection in the diagram window if there is not enough free area for the block.

13.3.4 Adding a Test Probe

When you add a test probe to the diagram window it is placed at the cursor position where you started right-clicking. The test probe is selected and framed in green.

To add a test probe

- 1 Right-click in the diagram window.
- 2 On the menu, click **Test Probe Block**.
- 3 In the **Select Test Probe** dialog box, in the **Test Probe Types** list, select the required test probe type.



- 4 Click **OK**.
- 5 Move the test probe to a suitable position in the diagram window.



Note

- The test probe remains as a floating selection in the diagram window if there is not enough free area for the block.

13.3.5 Duplicating an Existing Function Block

A simple way to create a block is to duplicate the existing block.

When you duplicated a function block, it is placed in the same position as the original block. The duplicate block is selected and framed in red until it is moved.

To duplicate an existing function block

- 1 In the diagram window, right-click the block you wish to duplicate and then click **Duplicate**.
- 2 Move the block to a suitable position in the diagram window.



Notes

- The block remains as a floating selection in the diagram window if there is not enough free area for the block.
- The duplicate block will have the parameters of the original function block.
- The duplicate block will not have the Name and the Public setting of the original function block.



Important

- You can not duplicate Hierarchical Function Blocks (HFB) using the **Duplicate** command.

13.3.6 Moving a Function Block

You need to move function blocks to position them where you want them in the diagram or to make space for additional function blocks.

You must select the function block before it can be moved in the diagram window.

When you select a function block, the function block is shown with green borders within a green dotted rectangle.

When you click the selection, the dotted rectangle changes from green to red. The cursor changes to a cross of arrows, indicating that you can drag the selected function block to another position.

The selected block becomes floating with dashed border lines and green flexible connection segments.

When you drag the selected function block, it remains floating and follows the cursor. The elastic part of the connections stretch or contracts.

If the selection is moved to the edge of the window, the diagram window scrolls in the direction of the cursor movement.

The function block will remain selected at the new position in the diagram window when the button is released.



Note

- The dotted rectangle remains red and the selected group is floating if there is not enough free space for the selection. You need to re-position the selection.

To move a function block

- 1 Select the function block you want to move by dragging the pointer around the block.
- 2 Click the selection rectangle.
- 3 Drag the selected function block to the new position and then release the mouse button.
- 4 Click outside the selection.

13.3.7 Copying a Function Block

Often, you will find the need to add a set of similar function blocks where only some parameters differ. To do this efficiently, you can copy an existing function block and then change the necessary parameters.

When you select a function block, the block is shown with green borders within a green dotted rectangle.

The pasted function block is placed in the upper left corner of the diagram window.



Note

- If there is not sufficient free area for the block it appears as a floating selection in the diagram window.

To copy a function block

- 1 Select the function block you want to copy by dragging the pointer around the block.
- 2 Right-click the function block and click **Copy**.
- 3 Click outside the selection to deselect the function block.
- 4 Right-click the diagram window and click **Paste**.
- 5 While the function block is selected in a green dotted rectangle, move the block to a new position.



Tip

- To copy a function block you can also click the block and use **Control+C** and **Control+V**.

13.3.8 Deleting a Function Block

When a block is no longer needed in the diagram you can delete it.

To delete a function block

- 1 Right-click the block.
- 2 In the **BLOCK** menu, click **Delete**.



Tip

- You can also delete a function block by selecting the block and press the **Delete** key.

13.4 Configuring a Function Block

The various function blocks are configured using dialog boxes. The options in these dialog boxes varies, depending on the type of block.

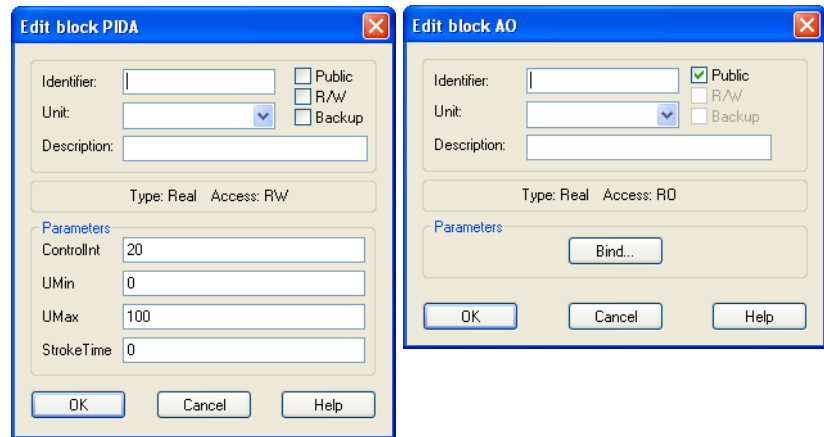


Fig. 13.1: Typical Edit block dialog boxes.

In the **Edit Block** dialog box you can find the following options:

Table 13.6:

| | |
|------------|--|
| Identifier | <p>The identifier is used the name of the signal.</p> <p>The block identifier appears in the function block diagram at the top left hand corner of the block.</p> <p>An identifier is required for all signals defined as public or if the block is an I/O, ALARM, or TSCH block.</p> <p>A maximum of 20 characters can be used. For more information on naming signals, see Chapter 11.2, “Signal Names”.</p> |
| Unit | <p>You can select a unit from the Unit list.</p> <p>For some function block types, you can change the default measurement system by selecting the unit.</p> <p>For more information on the use of units in a function block, see the block of current interest in Chapter 21, “Simple Blocks”.</p> <p>The Unit, used for a public signal, is also transferred to and used in a monitoring and supervising system.</p> |

Table 13.6:



| | |
|-------------|--|
| Description | <p>You can add an optional description of the block by typing any combination of up to 40 characters.</p> <p>The description of a public block is also transferred to and used in a monitoring and supervising system, like TAC Vista.</p> |
| Public | <p>When the Public check box is selected, the output from the block will be a public signal and included in the Public Signal Table.</p> <p>For more information on public signals, see Chapter 11.3, “Public Signals”.</p> |
| Read/Write | <p>When the R/W check box is selected, the block output signal can be modified from other network nodes during execution of the application and manually during offline simulation.</p> <p>The Read/Write option cannot be selected for blocks with RO access.</p> |
| Backup | <p>When the Backup check box is selected, the current value of the block output and internal states in the RAM are used when the device makes a warm start.</p> <p>Otherwise, the block output and internal states are reset to their initial values (same as for a cold start) at a warm start.</p> <p>If a power outage lasts longer than the power failure protection time for the device and the RAM contents are lost, the restart is a cold start. All blocks will be reset to their initial states regardless of the Backup option.</p> |
| Mode 2 | <p>This option, is available for operators with two inputs. This option determines the graphical form of the operator when displayed in the diagram.</p> <p>With the Mode 2 check box selected, the operator is displayed with the inputs entering from the top and the bottom of the block .</p> <p>With the Mode 2 check box cleared, the operator is displayed with the inputs entering from the left and the bottom of the block .</p> |
| Type | <p>The Type shows the data type (real, integer or binary) for the output signal from the block.</p> |

Table 13.6:

| | |
|-----------------|--|
| Access | The Access shows the type of access (RO or RW) for the output signal from the block. |
| Parameters | One or more boxes for block parameters. The set depends on the actual block. Some blocks have no parameter. A special parameter that is used in the function blocks for physical I/Os, is a Bind button. Clicking the Bind button opens a dialog box for binding the I/O point. |
| Expression type | These options, (Real, Integer and Binary), are only found in the dialog for expression blocks. The options define the output signal type of the expression block. |
| Expression | This option is only found in the dialog for expression blocks. The Expression box is where you edit the arithmetic or logical expression. The maximum length of the expression is 500 characters. |
| OK | Clicking the OK button closes the dialog box and introduces the changes. |
| Cancel | Clicking the Cancel button closes the dialog box without introducing edited changes. |
| Help | Clicking the Help button in the dialog box displays a detailed description of the block. |

To configure a function block

- 1 In the diagram window, right-click the required function block.
- 2 In the **BLOCK** menu, click **Edit**.
- 3 In the **Edit block** dialog box, in the **Identifier** box, type the name for the block.
- 4 In the **Unit** list, select an optional unit for the signal.
- 5 In the **Description** box, type an optional description of the signal.
- 6 If applicable for the block, select the **Public** check box when required.
- 7 If applicable for the block, select the **R/W** check box when required.
- 8 If applicable for the block, select the **Backup** check box when required.
- 9 Depending on the block type, configure the required parameters.

10 Click OK.

For details on parameters in the different blocks, see Chapter 21, “Simple Blocks”, Chapter 22, “Expression Blocks”, and Chapter 23, “Operators”.

**Note**

- To open the **Edit block** dialog box, you can also double-click the required function block.

13.4.1 Using Constants for Parameters

To give better access to parameters in function blocks you can use public constants instead of numeric values.

You select the use of a constant by entering a text string, the identifier for the constant, instead of a numerical value for the parameter in a function block.

A dialog, used to define the constant, opens when you finished editing the function block.

The public constants in the application are listed in the program specification.

For more information on constants, see Section 13.10, “Using Constants”.

**Important**

- A constant used as parameter in a function block is not removed from the Constants Table when the function block is deleted.

To use constants for parameters

- 1 Double-click the required block.
- 2 In the **Edit block** dialog box, in the required **Parameter** box, type a text string (identifier) instead of a numeric value.
- 3 Click **OK**.

- 4 In the **New constant** dialog box, in the **Value** box, enter the required value.
- 5 Click the **Public** check box, to get a public constant.

- 6 In the **Unit** list, select the required unit for the constant.
- 7 box, enter the required value.
- 8 In the **Description** box, type an appropriate description for the constant.
- 9 Click **OK**.



Notes

- If you enter more than one constant as parameter in the function block the **New constant** dialog opens to define each of them.
- The new constant is added to the Constants Table after clicking OK.

13.4.2 Configuring TAC Vista Alarm Texts

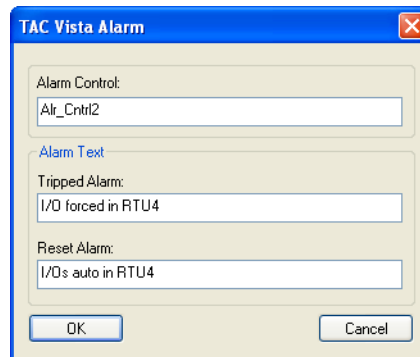
The function block ALARM has a special set of parameters, defining alarm texts for a monitoring and supervising system. You define the different texts in TAC Menta.

For more information on parameters in the ALARM block, see Chapter 21.6, “ALARM – Alarm”.

To configure TAC Vista Alarm texts

- 1 In the diagram window, right-click the required ALARM block.
- 2 In the **BLOCK** menu, click **Edit**.

- 3 In the **Edit block ALARM** dialog box, click the **Edit** button.



- 4 In the **TAC Vista Alarm** dialog box, in the **Alarm Control** box, enter the required alarm control string.
- 5 In the **Tripped Alarm** box, enter the required alarm text.
- 6 In the **Reset Alarm** box, enter the required alarm text.
- 7 Click **OK**.



Note

- To open the **Edit block** dialog box, you can also double-click the ALARM block.

13.5 Connections

Connections in Menta transport signals (data) between the function blocks in the function block diagram. The connection is a link between the output of one block and one or more inputs of other blocks.

The connections have types. They inherit the type from the connected output.

Outputs and inputs can only be connected when they have the same signal data type.

Connections are formed by straight line segments. The points where line segments join are called nodes.



Note

- The corners, formed when connection lines change direction, are also nodes.

In the function block diagram, binary connections end at an input with an un-filled arrowhead, and analog (real or integer) connections end at an input with a filled arrowhead.

A node, joining more than two segments, is displayed as a black circle to differentiate it from connections crossing each other.

A connection normally starts at a block output and ends at a block input but you can sometimes use incomplete connections.

When a connection is connected to an output only, the unconnected end is displayed as a black circle.

When a connection is connected to an input only, the unconnected end is displayed as a black square.



Notes

- You can not compile a function block diagram when there are unconnected inputs.
- Unconnected inputs must be connected before entering the simulation mode

In the editor, you can draw connections from different starting points in the diagram.

- Starting at a block output
- Starting at a connection line
- Starting at a node
- Starting in the work area

13.5.1 Drawing a Connection Line From a Block Output to Input

The most frequent way to draw connections in the diagram is drawing the connection from the output of one function block to the input of another.

When you click the output of a function block, the cursor changes to an angled cross.

When you start moving the cursor, a floating connection line will be started and movements are tracked by an elastic, green line.

When it is necessary to change position or direction of the connection line, you can fixate the current segment end by clicking once. The floating connection line will be changed into a fixed segment and a new floating connection line will be started at the break point.

You can repeat this fixation as many times as necessary, creating a connection made up of various consecutive segments.

When you point to an input where you can connect, the cursor changes to a square around a small angled cross.

You finish the connection by clicking at the required input.



Notes

- You can change the angle for the new segment direction in multiples of 45 degrees by holding down the SHIFT key and move the cursor. The **Orthogonal Connections** option must not be selected. The direction of the segment will only allow angles which are multiples of 45 degrees.
- If the Orthogonal connections function is selected, new segment direction will only allow angles of 90 degrees. For more information on orthogonal connections, see Chapter 13.5.16, “Orthogonal Connections”

To draw a connection line from a block output to input

- 1 Click the output of the block.
- 2 Move the cursor towards the input to where you want to connect.
- 3 Click the point where you need a break point.
- 4 Point to the input to where you want to connect.
- 5 Left-click the input to connect.



Tips

- You can also finish the connection at an arbitrary location by clicking the connection line.
- When drawing a connection you can delete the last segment of a connection by right-clicking.

13.5.2 Creating a Branch on a Connection Line

Sometimes you want to connect the same signal to several inputs in the function block diagram. You can create a node on the connection line and draw a connection line, starting from the node.

To create a branch on a connection line

- 1 Right-click the required connection line, and in the **CONNECTION** menu, click **Create Node**.

A node (filled, black circle) is created on the connection line and a floating connection line will be started at the point where you right-clicked.

13.5.3 Drawing a Connection Line From a Node

You can draw a new connection line, starting from a node on an existing connection line. Corners, or points where a branch is started, are nodes on a connection line.

To draw a connection line from a node

- 1 Click the required node.

A floating connection line will be started, the cursor changes to an angled cross and movements are tracked by an elastic, green line.

13.5.4 Drawing a Connection Line From the Work Area

Sometimes, particularly when you prepare macro blocks, you want to start drawing a connection at an arbitrary place on the work area. To allow for the syntax control when connecting to a block, the connection must be of the required type, real, integer or binary (boolean).

To draw a connection line from the work area

- 1 Right-click in the diagram window.
- 2 In the **NEW** menu, click **Node**
- 3 In the submenu, click the required type of connection (real, integer or boolean).
- 4 Click at a required position to finish the connection line.

13.5.5 Removing all Connections to a Function Block

You can remove all connections to a function block in a single command.

To remove all connections to a function block

- 1 Right-click the block.
- 2 In the **BLOCK** menu, click **Disconnect**.

13.5.6 Removing a Connection from an Input

You can remove a connection to an input of a function block in a single command.

To remove a connection from an input

- 1 Right-click the required connection at the input.
- 2 In the **INPUT** menu, click **Disconnect**.

The complete connection is removed.

13.5.7 Removing a Connection from an Output

You can remove a connection to an output of a function block in a single command.

To remove a connection from an output

- 1 Right-click the required connection at the output.
- 2 In the **OUTPUT** menu, click **Disconnect**.

The complete connection is removed.

13.5.8 Changing a Connection to Another Input

Sometimes you want to change a connection to another input. This can be done without losing the connection lines.

When you detach a connection line from the input, the end is changed to a green flexible line, ready to draw the connection to the alternative input.

For more information on drawing a connection, see Section 13.5.1, “Drawing a Connection Line From a Block Output to Input”.

To change a connection to another input

- 1 Right-click the required input.
- 2 In the **INPUT** menu, click **Detach**.
- 3 Point to the input to where you want to connect and left-click.



Tip

- You can also temporarily finish the disconnected connection line at a suitable location by clicking twice.

13.5.9 Changing a Connection to Another Output

Sometimes you want to change a connection to another output. This can be done without losing the connection lines. As connections are made from outputs to inputs you do this in two steps:

- detaching the connection
- connecting to the output

When you detach a connection line from the output, the end is changed to a green flexible line, ready to change the open end of the connection line to a node.

Clicking creates a node and you can use this node when connecting to the output.

For more information on arranging nodes, see Section 13.5.14, “Moving a Node”.

To change a connection to another output

- 1 Right-click the required output.
- 2 In the **OUTPUT** menu, click **Detach**.
- 3 Click at a suitable location for the node.
- 4 Connect to alternative output to the node.

13.5.10 Deleting a Connection

You can delete a connection or parts of it. The extent of the deletion is determined by the location where you right-click the connection line, the node. When you right-click the connection close to the output of a block, the complete connection including all branches are deleted.

When you right-click the connection after a branch, only the part between the point where you click and the connected input is deleted.

To delete a connection

- 1 Right-click at the required point on the connection line.
- 2 In the **CONNECTION** menu, click **Delete Node**.



Tips

- To delete a connection you can also do one of the following:
 - Right-clicking an input and then clicking **Disconnect** in the **INPUT** menu.
 - Right-clicking an output and then clicking **Disconnect** in the **OUTPUT** menu
- You can also delete a connection by drawing a selection rectangle around a node (a corner or a point where segments are joined) and click **DELETE**.

13.5.11 Breaking a Connection at an Arbitrary Point

To adapt to your need for changing connections, you can break an existing connection at different positions.

Detaching a connection divides it in two pieces and the connection to the output is changed to a floating connection line. The part of the connection to the input is ended in a node.

For more information on how to draw a connection line, see Section 13.5.1, “Drawing a Connection Line From a Block Output to Input”.

To break a connection at an arbitrary point

- 1 Right-click the connection at any intermediate point.
- 2 In the **CONNECTION** menu, click **Detach**.
- 3 Click at a suitable position on the connection line to create a node.

13.5.12 Breaking a Connection at an Input

To adapt to your need for changing connections, you can break an existing connection at different positions.

Clicking a connection at the input of a block will separate the connection from the block and convert the connection line to a floating connection line. The connection is ended in a node.

For more information on how to draw a connection line, see Section 13.5.1, “Drawing a Connection Line From a Block Output to Input”.

To break a connection at an input

- 1 Click the required input.
- 2 Move the cursor to a required position.
- 3 Click to create a node.



Tip

- To break a connection at an input you can also right-click the input and then click **Detach** in the **INPUT** menu.

13.5.13 Breaking a Connection at an Output

To adapt to your need for changing connections, you can break an existing connection at different positions.

When you use the command to detach a connection, it will be separated from the output, converted to a floating connection line to the first node. The connection is ended in a node.

To break a connection at an output

- 1 Right-click the required output.
- 2 In the **OUTPUT** menu, click **Detach**.
- 3 Move the cursor to a required position.
- 4 Click at a suitable location to create a node.

13.5.14 Moving a Node

Sometimes you want to re-arrange function blocks and connections in a function block diagram. You can move a node on an existing connection line.

When you use the command to move a node, the connection lines will be changed to floating connection lines and can be moved, using the cursor.

To move a node

- 1 Right-click at a required node on the connection line.
- 2 In the **CONNECTION** menu, click **Move node**.
- 3 Move the connection line to a required position.
- 4 Click at the required position to fixate the connections.



Tip

- To start moving a node you can also draw a selection rectangle around the node.

13.5.15 Highlighting a Connection

Highlighting a connections (signal) in selected colors is highly valuable when tracing signals in a block diagram. When you mark a signal, all branches of the connection are colored.

You can highlight connections both in the edit and the simulation mode.

To highlight a connection

- 1 Right-click the required connection.
- 2 In the **CONNECTION** menu, click **Mark**.



- 3 In the submenu, click the required color.



Notes

- You can revert to a normal colored connection by right-clicking the connection and then clicking **Unmark** in the **CONNECTION** menu.
- To revert to the normal color for all marked connections, you can right-click a connection and then click **Unmark All** in the **CONNECTION** menu.

13.5.16 Orthogonal Connections

When you draw a connection, the direction of the connection lines can be changed. The direction can be changed in any angle or limited to pre-determined changes. For more information on drawing a connection, see Section 13.5.1, “Drawing a Connection Line From a Block Output to Input”.

Changing the direction in 90 degrees, so called orthogonal connection lines, is the default setting for Menta, but can be changed.

When the orthogonal connections mode is checked, the connection line is created in orthogonal segments when drawing a connection.

To change the orthogonal connection setting

- Click **Preferences** and then check the **Orthogonal Connections** command.



Tips

- You can temporarily change the setting of orthogonal connections by holding down the **CTRL** and **SHIFT** keys simultaneously, while drawing a connection.
- You can lock the vertical position of a segment by holding down the **CTRL** key while drawing the connection.

13.6 Operations on Groups

Sometimes you want to work with a number of function blocks and their connections as a group. This can be useful, for example when copying or defining module name.

13.6.1 Selecting a Group

Before operations on a group can be done, the blocks and connections in the group must be selected.

When you select a group, the cursor is changed to an open hand and the selected function blocks have green borders within a green dotted rectangle.



Notes

- A connection is partially selected if only the starting point or the end point of the connection is selected.
- A connection is completely selected if both starting and end points are selected.

To select a group

- Select all function blocks and connections you want to include in the group by dragging the pointer around the blocks.



Important

- Once a selection has been created, you can add or remove individual blocks or nodes to the selection by clicking the block or node while holding down the **SHIFT** key.
- Clicking, in the same manner, a selected item will unselect it and clicking an unselected item will select it.
- Clicking, in the same manner, an item outside the current selection will select the item and expand the dotted rectangle.



Tip

- You can select the entire function block diagram by clicking **Select all** in the **Edit** menu.

13.6.2 Deselecting a Group

Sometimes you want to cancel the selection, without performing an operation on the selected group.

To de-select a group

- Click outside of the selection rectangle.



Tip

- To de-select a selection you can also right-click the selection and in the **GROUP** menu, click **Deselect**.

13.6.3 Centering the Selection

You may want to center a selected group on the screen.

To center the selection

- In the **Edit** menu, click **Center selection**.



Tip

- To center a selection, you can also use the function key **F9**.

13.6.4 Moving a Group

You may need to move a group of function blocks to position them where you want them in the diagram or to make space for additional function blocks.

You must select the group of function blocks and connections you want to move before you can move them. All function blocks in the selection are indicated by having green borders and are enclosed in a green dotted rectangle.

When you move a selection, the rectangle changes from green to red and the cursor changes to a cross of arrows. The selected group becomes floating, with dashed lines as border and green selected segments of connections. The elastic part of the connections stretch or contracts when you move the selection. The diagram window scrolls in the direction of the cursor movement if the selection is moved to the edge of the window.



Note

- The selection rectangle remains red and the selected group is floating if there is not enough free space for the selection. You need to re-position the selection.

To move a group

- 1 Select all function blocks and connections you want to include in the group by dragging the pointer around the blocks.
- 2 Click the selection rectangle.
- 3 Without releasing the left button, drag the selected group to the required position.
- 4 Release the mouse button when the selection is at the required position.
- 5 Click outside the selection when finished.

13.6.5 Deleting a Group

You can delete all function blocks in a group.

To delete a group

- 1 Right-click the selected group.
- 2 In the **GROUP** menu, click **Delete**.

13.6.6 Copying and Pasting a Group

You can copy and paste groups.

When you use the copy command, all the selected blocks and selected connection segments are copied to the clipboard and are available for a paste command.

To copy and paste a group

- 1 Select the group of function blocks you want to copy by dragging the pointer around the blocks.
- 2 Right-click the selected group.
- 3 In the **GROUP** menu, click **Copy**.
- 4 Click outside the selection.
- 5 Right-click and in the **NEW** menu, click **Paste**.



Tips

- To copy a selected group you can also do the following:
 - In the **Edit** menu, click **Copy**.
 - Click outside the selection.
 - In the **Edit** menu, click **Paste**.
- You can also use CTRL+C, and CTRL+V, to make a copy of a group.

13.6.7 Copying a Selection to the Clipboard

Use this menu option to copy the graphic part of a selection to the clipboard for exporting it to applications in metafile format.

To copy a selection to the clipboard

- 1 Select the group of function blocks you want to copy by dragging the pointer around the blocks.
- 2 In the **Edit** menu, click **Copy To Clipboard**.



Note

- If you want to copy the selection monochromatic, you can click **Black And white** In the **Preferences** menu.

13.6.8 Disconnecting a Group

You can disconnect connections to all function blocks in a group.

To disconnect a group

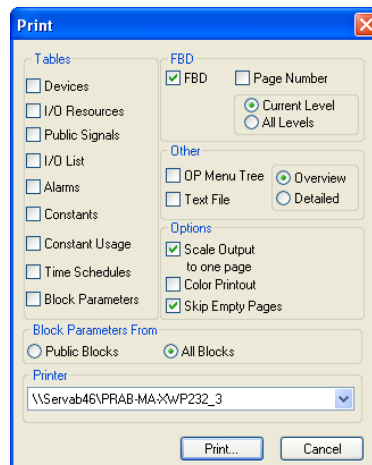
- 1 Right-click the selected group.
- 2 In the **GROUP** menu, click **Disconnect**

13.6.9 Printing the Selection

You can print a selection. To get the correct size of the printout you must also scale the output.

To print a selection

- 1 In the **File** menu, click **Print**.
- 2 In the **Print** dialog box, in the **Options** area, click **Scale output to one page**.



- 3 Click **Print**.

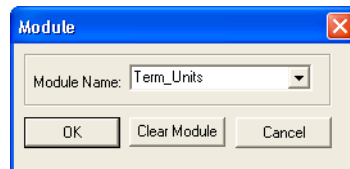
13.6.10 Editing the Module Name

A typical example of the use of groups is when you define module names in Menta.

The module name is shown in each function block in the module. The block type for a block in a module is underlined and purple.

To edit the module name

- 1 Select a block or a group of blocks.
- 2 Right-click the selection.
- 3 In the **GROUP** menu, click **Module**.
- 4 In the **Module** dialog box, in the **Module Name** box, type the name of the module



- 5 Click **OK**.



Note

- If modules are used in the application, you can select a module name from the **Module Name** list in the **Module** dialog.

13.7 Finding and Replacing Text Strings

You can search and also replace text strings the function block diagram. The search for the string can be selective.

13.7.1 Finding a String

You can search the function block diagram for a string in comments, block names and block parameters

In the function blocks you can search for the string in block names, module names or constants. You can choose to search for the string in all types of blocks or one type only.

You can search for the string in alarm texts.

You can choose to search for the string case sensitive and whole words or not.

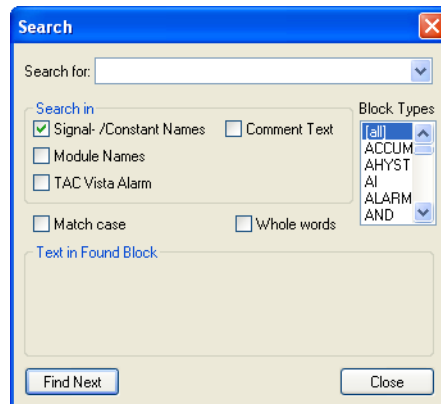
When the searched string is found, the item containing the string is selected, framed in a green rectangle.

Table 13.7: The Find dialog options.

| Option | Description |
|-------------------------|---|
| Search for | The string to search for. When the Search for box is empty, all blocks of the marked block type(s) and comments will be searched for. |
| Signal/- Constant Names | Select this box to search for the string in signal names in block identifiers, constant names, network addresses, and SNVT names. |
| Comment text | Select this box to search for the string in comment texts. |
| Block Types | Select, from the list, a single block type or all types to search in. |
| Module Names | Select this box to search for the string used as a Module name in block identifiers, constant names, network addresses, and SNVT names. |
| TAC Vista Alarm | Select this box to search for the string used in a TAC Vista alarm. |
| Match case | Select this box to make a case sensitive search for the string. |
| Whole words | Select this box for searching whole words. |
| Find Next | Click this button to continue the search. |
| Close | Click this button to close the dialog. |

To find a string

- 1 In the **Edit** menu, click **Find**.



- 2 In the **Search** dialog box, in the **Search for** box, type the string to search for.
- 3 Click the **Signal-/ Constant Names** check box, to search for the string in signal and constant names.
- 4 In the **Block Types** list, select the required block type to search.
- 5 Click the **Comment Text** check box, to search for the string in comment texts.
- 6 Click the **Module Names** check box, to search for the string in module names.
- 7 Click the **TAC Vista Alarm** check box, to search for the string in TAC Vista alarm texts.
- 8 Click the **Match case** check box, to match the letter case in the string.
- 9 Click the **Whole words** check box, to match whole words in the string.
- 10 Click **Find Next** to start the search.



Tip

- You can also start a the search for a string by using the function key **F7**.

13.7.2 Replacing a String

You can search the function block diagram for a string in comments, block names and block parameters and replace it with a required string.

In the function blocks you can search for the string in block names, module names or constants. You can choose to search for the string in all types of blocks or one type only.

You can search for the string in alarm texts. You can choose to search for the string case sensitive and whole words or not.

The replacing of a string is done using a dialog where you can define several options for the search.

When the searched string is found, the item containing the string is selected and framed in green.

Table 13.8: The Replace dialog options.

| Option | Description |
|-------------------------|---|
| Search for | The string to search for. When the Search for box is empty, all blocks of the marked block type(s) and comments will be searched for. |
| Replace with | The string to replace with. |
| Signal/- Constant Names | Select this box to search for the string in signal names in block identifiers, constant names, network addresses, and SNVT names. |
| Comment text | Select this box to search for the string in comment texts. |
| Block Types | Select, from the list, a single block type or all types to search in. |
| Module Names | Select this box to search for the string used as a Module name in block identifiers, constant names, network addresses, and SNVT names. |
| TAC Vista Alarm | Select this box to search for the string used in a TAC Vista alarm. |
| Case sensitive | Select this box to make a case sensitive search for the string. |

Table 13.8: The Replace dialog options.

| Option | Description |
|-------------------------------|---|
| Whole words | Select this box for searching whole words. |
| Replace only first appearance | Select this box to replace only the first match of the string in a block. Otherwise, all matches in the block will be replaced. |
| Find Next | Click this button to continue the search. |
| Replace | Click this button to replace the string. |
| Replace All | Click this button to replace all matches of the string. |
| Close | Click this button to close the dialog. |



Notes

- Checking both the **Module names** and the **Signal and Constant names** boxes allows you to search for a text string containing both the Module name and the Signal name in block identifiers, constant names, network addresses and in SNVT names.
- In this case, the module name part can not be changed.

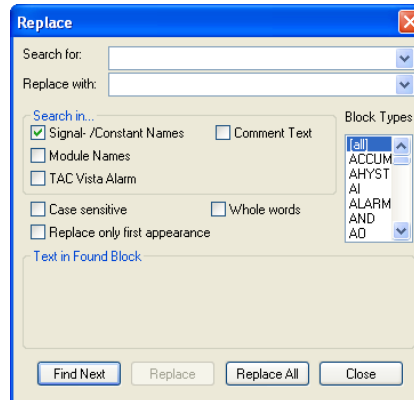


Important

- The program will not always check the length of a name string when executing a **Replace** command.
 - Please check that the Module name does not exceed 12 characters after replacement.
 - Please check that the Signal or Constant names do not exceed 20 characters after replacement.

To replace a string

- 1 In the **Edit** menu, click **Replace**.



- 2 In the **Replace** dialog box, in the **Search for** box, type the string to replace.
- 3 In the **Replace with** box, type the string to replace with.
- 4 Click the **Signal-/ Constant Names** check box, to search for the string in signal and constant names.
- 5 In the **Block Types** list, select the required block type to search.
- 6 Click the **Comment Text** check box, to search for the string in comment texts.
- 7 Click the **Module Names** check box, to search for the string in module names.
- 8 Click the **TAC Vista Alarm** check box, to search for the string in TAC Vista alarm texts.
- 9 Click the **Case sensitive** check box, to match the letter case in the string.
- 10 Click the **Whole words** check box, to match whole words in the string.
- 11 Click **Find Next** to search for the string.
- 12 Click **Replace** to replace the string. Click **Find next** if you want to search for the string again.



Note

- To replace the string in all blocks where it is found, you can click **Replace all**.



Tip

- To open the **Replace** dialog you can also use the function key **F8**.

13.8 Macro Blocks

Necessary time for designing an application can be reduced by reusing existing designs and load them as parts to the function block diagram as so called macro blocks. Existing applications or part of them can be saved as macro blocks.

The default folder for Macros blocks used by Menta is the Library subfolder to Menta, but you can change the path to an optional location. The path to the Library is defined in the preferences for TAC Menta.

For more information on how to set the preferences, see Chapter 9.8, “Defining the TAC Menta Settings”.

13.8.1 Saving a Macro Block

A group of blocks and connections can be saved as a macro block for future re-use.

When a selected group of blocks is saved as an .MTA or .AUT file, only the block definitions and connections are saved.

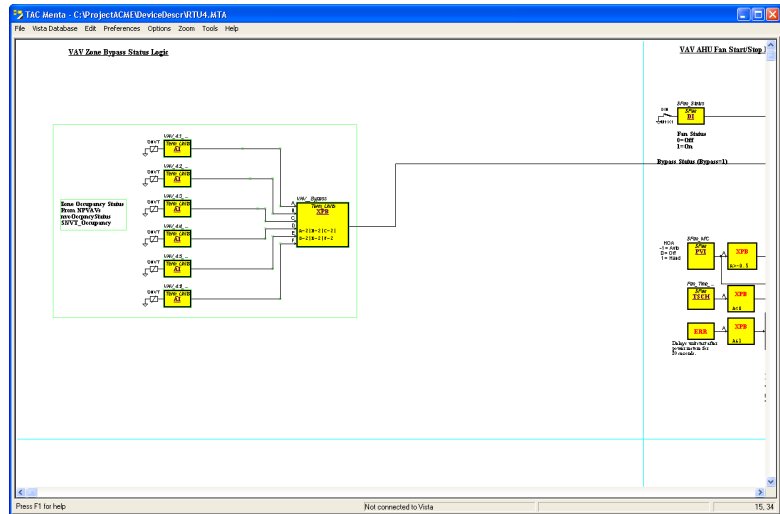


Important

- There is a difference in saving a selection using the **Save** option on the **GROUP** menu and using the **Save** command on the **File** menu.
 - When you use the **Save** command on the **GROUP** menu only information in the selection is included in the saved file.
 - You can not use the **Save** command on the **File** menu to save a selection only. This command saves the complete function block diagram.
 - When you use the **Save** command on the **File** menu more information such as specification data (author, type etc.) is included in the file.

To save a macro block

- 1 Select all function blocks and connections you want to save as a macro block.



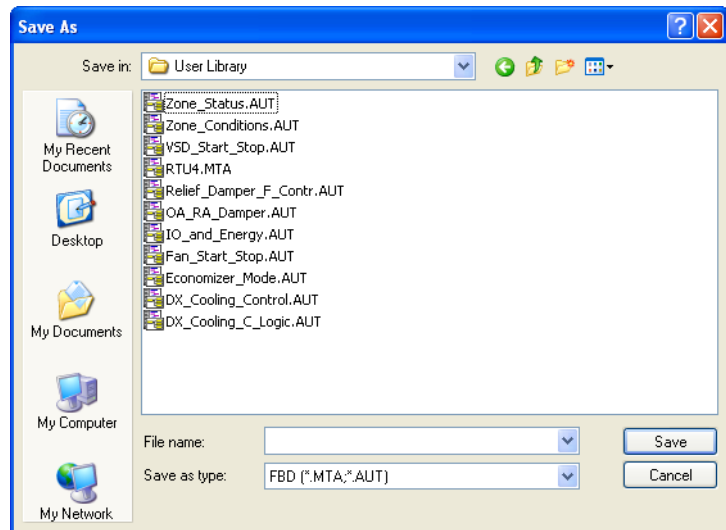
- 2 Right-click the selection, and in the **GROUP** menu click **Save**.



Note

- Only completely selected connections are saved in the macro block, see Section 12.3.1, “Selecting a Group”.

- 3 In the **Save As** dialog box, in the **Save in** box, browse to the location where you want the macro to be stored.



- 4 In the **File name** box, type the file name for the macro block.
- 5 Click **Save**.
- 6 In the diagram window, click outside the selection to de-select.

13.8.2 Loading a Macro Block

Previously saved macro blocks can be added to the function block diagram by loading them to the diagram window.



Important

- Only macro blocks prepared for a Xenta 700 device can be loaded as macros to an application for this type of device.

You can add macro blocks to the diagram window in two ways:

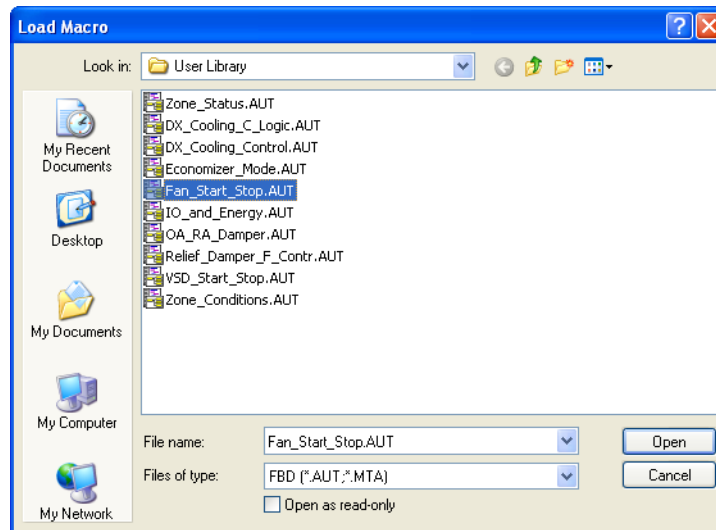
- Using the **Load Macro** command in the **NEW** menu.
- Using Drag and Drop in Windows.

When you add a macro block to the diagram window it will be inserted at the point where the mouse button was clicked when opening the menu.

All inserted items are, by default, selected and framed in green.

To load a macro block

- 1 Click the diagram window to clear any selection.
- 2 Move to a blank area in the diagram window,
- 3 Right-click the diagram window, and then in the **NEW** menu click **Load Macro**.
- 4 In the **Load Macro** dialog box, in the **Look in** box, browse to the folder with the macro block you want to load.



- 5 Select the required macro block file.
- 6 Click **Open**.
- 7 Drag the contents of the macro block to a suitable position in the diagram window.

- 8 Click in the diagram window to de-select the items.



Tips

- The default search path for the **Load Macro** command is the Path for Library, as specified in the **Settings** command in the **Preferences** menu.
- You can also add a macro block to the diagram window by Dragging the file with the macro from Microsoft® Windows Explorer and Drop the file in the diagram window.



Notes

- In the **Load Macro** dialog box you can choose between the file types for an old macro block (.MCB) and the more recent file types (.MTA/.AUT).
- Macro blocks of the .MCB type, includes only names of constants, not their values. Values of constants in a macro block of this type will not be loaded to the application. An error message is shown.
- New constants, public and non public, in a macro block are added to the Constants Table when you load a macro block.
- Duplicate constants will not be added, the old value will be preserved and the message “Constant already defined:” is shown.

13.8.3 Using Macro Commands in Comment Blocks

A saved macro block can contain information on signal names, network addresses or SNVT names.

To make it easier to replace these text strings with site-specific texts, you can add macro commands to the macro block before saving. The macro commands are executed when the macro block is loaded, assisting the user when replacing the texts.

The macro commands are created by entering one or more of the following commands in a comment block:

Table 13.9: Macro Commands.

| | |
|-----------------------------------|--|
| <p>\$MESSAGEBOXExample text\$</p> | <p>The MESSAGEBOX macro command opens the TAM32 message box displaying the message “Example text” when the application is loaded.</p> <p>Clicking OK closes the message box.</p> |
|-----------------------------------|--|

Table 13.9: Macro Commands.

| | |
|--|---|
| <p>\$REPLACEText string\$</p> | <p>The REPLACE macro command opens the Macro Replace dialog box, displaying the "Text string" to replace in the Replace text box.</p> <p>You type the replacement text string in the With text box.</p> <p>Clicking OK replaces all the original strings "Text string" in block names, block parameters and comments.</p> |
| <p>\$REMINDERMemo text\$</p> | <p>The REMINDER macro command opens the TAM32 message box displaying the message "Memo text" when the application is loaded.</p> <p>Clicking OK closes the message box.</p> <p>The message box for the REMINDER macro command will display the "Memo text" every time the application is opened.</p> |
| <p>\$PLAYSOUND SoundFile.waw\$</p> | <p>The PLAYSOUND macro command plays the SoundFile.waw, located in the same folder as the macro file.</p> <p>The sound file will be played every time the application is opened.</p> |



Important

- The macro commands will replace also the command text strings in the comment blocks. Saving the macro again will be with the replacement strings in force.



Notes

- Macro commands are executed in the order from top and down in the comment. The first **Replace** command will replace the text string, in all blocks. The second **Replace** command will replace the second text string and so on.
- You can use macro commands in more than one comment.
- You can mix macro commands with ordinary text in comments.

13.9 Hierarchical Function Blocks (HFB)

To get a better overview of a function block diagram in Menta you can use hierarchical function blocks (HFBs) in the diagram.

A diagram with HFBs is an alternative graphical presentation of the application for printing or viewing on the screen. The application is downloaded to the TAC Xenta device as an ordinary function block diagram, not as hierarchical function blocks.

You can create hierarchical function blocks in two ways:

- You can create an (empty) HFB and then add function blocks and connections to it.
- You can create an **HFB** from a group of existing function blocks and connections.

You can create hierarchical function blocks in more than one level, which means that you can have one or more HFBs within an existing HFB.



Important

- If you create or rename a module that includes an HFB, this HFB and all lower levels will be included in this module.

Special blocks are used as inputs and outputs in a hierarchical function block.

A hierarchical function block with at least one public signal, is shown in the diagram with a red dot in the upper right corner of the HFB symbol. The red dot is also used when the public signal is located in an embedded HFB.

The hierarchical function block is a graphical solution in TAC Menta for grouping function blocks, and can be printed. The program is downloaded to the TAC Xenta device, not as hierarchical function blocks, but as an ordinary function block diagram. The figures below show a hierarchical function block on the top level, as well as on the expanded level.

Running Control

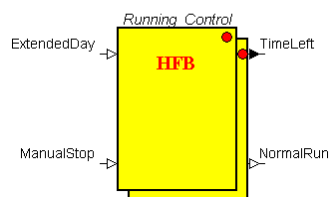


Fig. 13.2: HFB on top level

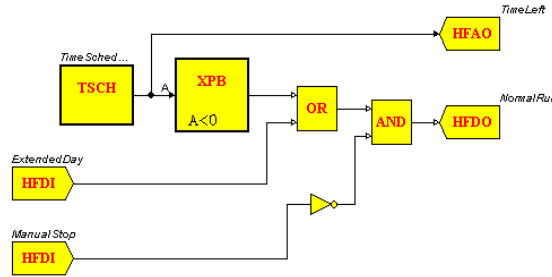


Fig. 13.3: HFB on expanded level

13.9.1 Creating an empty HFB

One way to create an HFB is to start with an new, empty HFB and add function blocks and necessary HFB I/O blocks to the function block diagram.

When you add an hierarchical function block it is placed in the diagram window at the cursor position where you started right-clicking. The function block is selected, framed in green.



Note

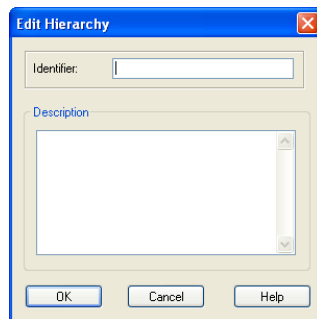
- The HFB block appears as a floating selection in the diagram window if there is not sufficient free area for the block.

Blocks for the application in the HFB function block diagram are added and connected the same way as an ordinary function block diagram.

To connect signals from the HFB to other levels in the hierarchical function block diagram you use HFB I/O blocks.

To create an empty HFB

- 1 Right-click in the diagram window.
- 2 On the **NEW** menu, click **HFB**.



- 3 In the **Edit Hierarchy** dialog box, in the **Identifier** box, type a name for the hierarchical function block.
- 4 In the **Description** box, type a suitable description of the HFB.
- 5 Click **OK**.

13.9.2 Adding an HFB I/O block

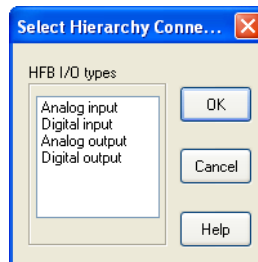
In the HFB you use the block types HFAI, HFAO, HFDI and HFDO to connect signals to the function block diagram higher level.

The signals in these blocks must have unique names within the HFB.

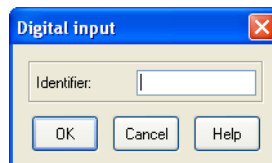
These inputs and output blocks are only displayed when the HFB is expanded and the signals cannot be made public.

To add an HFB I/O block

- 1 In the diagram window for the expanded HFB, right-click and then click **HFB I/O**.



- 2 In the **Select Hierarchy Connection** dialog box, click the required type.
- 3 Click **OK**.



- 4 In the dialog box for the type of HFB I/O, in the **Identifier** box, type a suitable name for the signal.
- 5 Click **OK**.

Remaining blocks for the application in the HFB function block diagram are added and connected the same way as an ordinary function block diagram.

13.9.3 Creating an HFB of Existing Function Blocks

To use hierarchical function blocks (HFBs) to structure a function block diagram, you can select existing blocks and connections and make an HFB of the selection.



Important

- When an hierarchical function block has been created, it can not be reverted to its original (flat) structure.

When you create an HFB of a group of blocks and connections the required input and output block types HFBI, HFAO, HFBI and HFDO are automatically created, using the existing signal names.



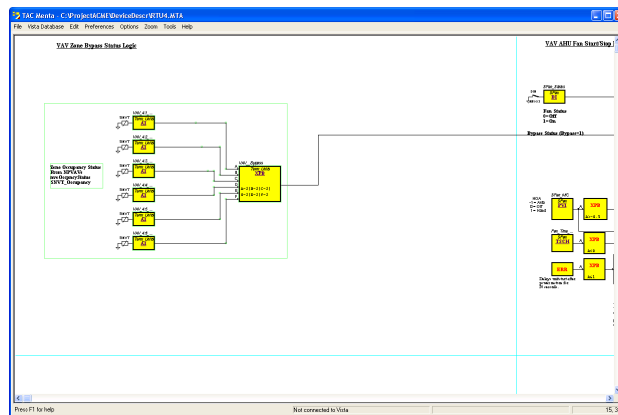
Important

- The names of the automatically created block types HFBI, HFAO, HFBI and HFDO blocks should not be changed.

When a selection of function blocks are changed to a hierarchical function block in the diagram window, the HFB block is selected, and showed framed in green.

To create an HFB of existing function blocks

- 1 Select all function blocks and connections to be saved as a macro block.



- 2 Right-click the selection, and in the **GROUP** menu, click **Create HFB**.
- 3 In the **Edit Hierarchy** dialog box, in the **Identifier** box, type a name for the HFB.
- 4 In the **Description** box, type a suitable description of the HFB.
- 5 Click **OK**.
- 6 Click **Save**.
- 7 Click outside the selection to deselect.

13.9.4 Expanding an HFB

You can expand an HFB To view the structure of the function blocks in the HFB.

In the expanded HFB you find the input and output block types HFAI, HFAO, HFDI, and HFDO. These I/O-blocks are symbols only used to show the connections to the higher level in the function block diagram structure.

To expand an HFB

- Right-click the required HFB function block, and in the **BLOCK** menu, click **Expand HFB**.



Tip

- You can also expand an HFB by double-clicking the HFB function block.

13.9.5 Compressing an HFB

When you compress an HFB you return to show the HFB on the higher level in the structured function block diagram.

To compress an HFB

- Right-click in the required HFB function block diagram window and in the **NEW** menu, click **Compress HFB**.

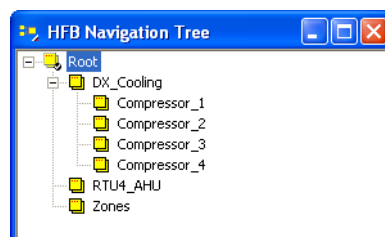
13.9.6 Navigating in a Hierarchical Structure of HFBs

Finding a required part of a function block diagram can be made easier using a navigation tree in Menta.

You can see the name of the current HFB in the status row at the bottom of the diagram window.

To navigate in the hierarchical structure of HFBs

- 1 In the **Options** menu, click **Show HFB Navigation Tree**.



- 2 In the **HFB Navigation Tree** dialog box, click the required HFB.



Notes

- The **HFB Navigation Tree** dialog remains open to allow fast navigation in the diagram.
- You close the dialog by clicking **Close**.

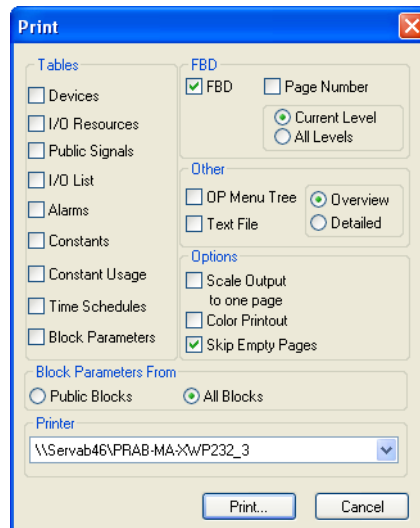
13.9.7 Printing an HFB

When you print a function block diagram with HFB blocks, you can print either a single HFB or all levels of the diagram.

For more information on printing the application, see Chapter 18, “Printing the Application Program Documentation”.

To print an HFB

- 1 In the **File** menu, click **Print**.



- 2 In the **Print** dialog box, select the **FBD** check box.
- 3 Click **All levels**, to get a printout of each HFB level on separate pages, with the name and the page number of the HFB at the bottom of the page.



Note

- You can print only the current level of the HFB by selecting **Current Level** check box, in the **Print** dialog box.

13.10 Using Constants

Constants can be declared as either public or internal. By definition, a constant cannot be modified by the application in which it is declared. By declaring the constant as public, it becomes available to other nodes in the network and can be modified from, for example, an operator panel.

Constants can be created separately or when parameters in a function block are configured. For more information on creating constants as parameters, see Section 13.4.1, “Using Constants for Parameters”.

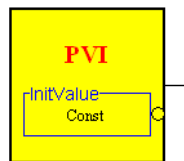
Constant names are alphanumeric strings of up to 20 characters, equal to the length of a line in the display in an operator’s panel (OP) and the maximum length of an ID string in TAC Vista. A constant name must not consist of digits only.

For information on allowed characters in constant names, see Chapter 9.3, “Allowed Characters in Name Strings”

13.10.1 Public Constants

To give better access to parameters in function blocks you can use public constants instead of numeric values. The use of a public constant also allows access to the value from other nodes in the network.

A parameter in a function block, where a public constant is used shows a small circle to the right of the parameter field.



The Module name of a public constant, by default the same as the Module name of the block, will not be displayed in the FBD unless it differs from the Module name of the block.

Public constants in the application are listed in the program specification and are of the following types:

Table 13.10:

| | |
|-----|------------------|
| PAB | Binary constant |
| PAI | Integer constant |
| PAR | Real constant |

Depending on whether other nodes will be allowed to read and/or modify the constant value, constants are also read-only or read/write, see Section 13.10.2, “Accessing Constants”.



Important

- It is important to note that public constants can be modified via the network as well during runtime.
- A public constant in TAC Menta can only be assigned to one function block parameter.
- A Non-public constant may be assigned to any number of function block parameters.
Non-public constant names are substituted with the corresponding numerical values when the FBD is compiled.
- A public constant can not have the same name as a public signal.
- Public constants are not allowed in expression blocks, or in operators.
- Public constants are not allowed inside the binding parameters of I/O blocks.

We recommend that you always name blocks containing public constants, otherwise the constants cannot retrieve/change Module names.

13.10.2 Accessing Constants

Public constants in a TAC Menta function block diagram have the access class RW.

The required access class for constants in an OP display (RW or RO) is specified by the user in the OP configuration tool.

13.10.3 Adding a Constant

All constants in a function block diagram are assembled in the Constants Table. In this table, constants may be viewed, added, modified, or removed.



Important

- A constant used as parameter in a function block is not removed from the Constants Table when the function block is deleted.

To add a Constant

- 1 In the **Options** menu, click **Constants Table**.
- 2 In the **Constants Table** dialog box, click **Add**.
- 3 In the **New constant** dialog box, in the **Identifier** box, type a name for the constant.
- 4 In the **Value** box, type the value of the constant.
- 5 Click the **Public** check box, to have the constant public.
- 6 In the **Unit** list, select the unit for the constant.
- 7 In the **Description** box, type a suitable description for the constant.
- 8 Click **OK**.
- 9 Click **Exit** to close the **Constants Table** dialog box.

13.10.4 Editing a Constant

You can change all properties of a constant with the exception of the name (Identifier) in a dialog.

To edit a constant

- 1 In the **Options** menu, click **Constants Table**.
- 2 In the **Constants Table** dialog box, double-click the required constant.
- 3 In the **Edit constant** dialog box, change the required content.
- 4 Click **OK**.
- 5 Click **Exit** to close the **Constants Table** dialog.



Note

- You can modify the contents of constants using the Constants Table also in the simulation mode.

13.10.5 Removing a Constant

A constant, used as parameter in a function block, is not removed from the Constants Table when the function block is deleted. The constant has to be removed manually when it no longer is needed.



Important

- Ensure that the constant you intend to remove is not used in the function block diagram. Otherwise you will not be able to compile or simulate the application.
- A warning dialog is shown when you compile the application and the block with the missing constant is selected. You can re-create the constant by editing the block.

To remove a constant

- 1 In the **Options** menu, click **Constants Table**.
- 2 In the **Constants Table** dialog box, click the required constant.
- 3 Click the **Delete** button.
- 4 Click **Exit** when finished.

13.10.6 Removing Unused Constants

Sometimes, during revisions of a function block diagram, defined constants are no longer needed. Such constants can easily be removed.

To remove unused constants

- 1 In the **Options** menu, click **Constants Table**.
- 2 In the **Constants Table** dialog box, click the required constant.
- 3 Click the **Remove Unused** button.
- 4 Click **Exit** when finished.

13.11 Using the I/O Configuration Table

The I/O Configuration Table lists all I/O signals in the function block diagram. You can define, modify or view the I/O bindings starting from this table.



Important

- The **I/O Configuration Table** command is not available when Menta is started from XBuilder.

The table has four columns: the signal name, the I/O block type, where the signal is bound to and an override column. The columns can be sorted in alphabetical order.

13.11.1 Opening the I/O Configuration Table

When you want an overview of all I/O bindings in a function block diagram, you can use the I/O Configuration Table.

To open the I/O Configuration Table

- In the **Options** menu, click **I/O Configuration Table**.

13.11.2 Changing I/O Bindings Using the I/O Configuration Table

As the configuration table lists all I/O points and can be sorted, it is a fine starting point for finding and changing I/O point bindings.

To change I/O bindings using the I/O Configuration Table

- 1 In the **Options** menu, click **I/O Configuration Table**.
- 2 In the **I/O Configuration Table** dialog box, double-click the row for required I/O.
- 3 In the **Bind I/O point** dialog box, modify the required binding.



Note

- You can also start changing an I/O binding from the I/O Configuration Table dialog by selecting the I/O point and clicking the **Bind** button.

Depending on the type of I/O block there are different **Bind I/O point** dialogs, each with a specific set of configuration parameters.

13.12 Using the Time Schedule Table

The **Time Schedule Table** table lists all time schedules in the function block diagram, sorted in alphabetic order. You can define, modify or view the time schedules starting from this table.



Important

- The **Time Schedule Table** command is not available when Menta is started from XBuilder.



Notes

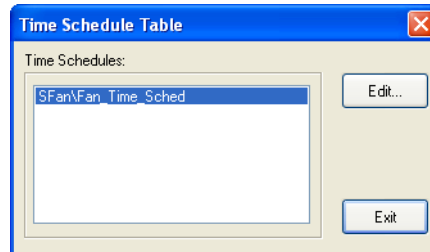
- You can modify the contents of time schedules using the Time Schedule Table also in the simulation mode.
- The number of week and holiday charts can only be modified in Edit mode.

13.12.1 Opening the Time Schedule Table

When you want an overview of all time schedules in a function block diagram, you can use the Time Schedule Table.

To open the time schedule table

- In the **Options** menu, click **Time Schedule Table**.



13.12.2 Changing a Time Schedule using the Time Schedule Table

As the Time Schedule Table lists all time schedules, sorted in alphabetic order, it is a fine starting point for finding and changing time schedules.

To change a time schedule using the time schedule table

- 1 In the **Options** menu, click **Time Schedule Table**.
- 2 In the **Time Schedule Table** dialog box, double-click the row for required time schedule.
- 3 In the **Time Schedule** dialog box, modify the required scheduling.



Note

- You can also start changing a time schedule from the **Time Schedule Table** dialog by selecting the time schedule and clicking the **Edit** button.

13.13 Using the Alarm Text Table

The **Alarm Text Table** table lists all alarms in the function block diagram. You can define, modify or view the alarms from this table.



Important

- The **Alarm Text Table** command is not available when Menta is started from XBuilder.

The table has five columns: the alarm name, the alarm text for OP, the alarm processing, the alarm text when tripped and the alarm text when resetted. The columns can be sorted in alphabetical order.

13.13.1 Opening the Alarm Text Table

When you want an overview of all alarms in a function block diagram, you can use the Alarm Text Table.

To open the alarm text table

- In the **Options** menu, click **Alarm Text Table**.

| Name | OP Alarm Text | TAC Vista Alarm pro... | TAC Vista Tripped a... | TAC Vista Reset ala... |
|--------------------|-------------------------|------------------------|---------------------------|-------------------------|
| Cooling\DAT_High | High DAT | Alr_Cntrl3 | High discharge air te... | Discharge air temper. |
| Cooling\DAT_Low | Low DAT | Alr_Cntrl3 | Low discharge air te... | Discharge air temper. |
| Cooling\DAT_SF | Sensor failure DAT | Alr_Cntrl3 | Sensor failure dischar... | Sensor failure dischai |
| Econ\CO2_Alarm | CO2 Alarm | Alr_Cntrl3 | CO2 2nd Floor South... | CO2 2nd Floor South |
| Econ\MAT_High | High MAT | Alr_Cntrl3 | High mixed air temper... | Mixed air temperature |
| Econ\MAT_Low | Low MAT | Alr_Cntrl3 | Low mixed air temper... | Mixed air temperature |
| Econ\MAT_SF | Sensor failure MAT | Alr_Cntrl3 | Sensor failure mixed ... | Sensor failure mixed . |
| Econ\QAHumidity_SF | Sensor failure QAHu... | Alr_Cntrl3 | Sensor failure QAHu... | Sensor failure QAHu |
| Econ\QAT_SF | Sensor failure QAT | Alr_Cntrl3 | Sensor failure outside... | Sensor failure Outsid. |
| Econ\RAHumidity_SF | Sensor failure RA Hu... | Alr_Cntrl3 | Sensor failure RA Hu... | Sensor failure RA Hu |
| Econ\RAT_High | High RAT | Alr_Cntrl3 | High return air temper... | Return air temperatur |
| Econ\RAT_Low | Low RAT | Alr_Cntrl3 | Low return air temper... | Return air temperatur |
| Econ\RAT_SF | Sensor failure RAT | Alr_Cntrl3 | Sensor failure return ... | Sensor failure return . |
| FD_Alarm | I/O Forced | Alr_Cntrl2 | I/O forced in RTU4 | I/Os auto in RTU4 |



Note

- You can also start changing an alarm from the **Alarm Text Table** dialog by selecting the alarm and clicking the **Change** button.

13.13.2 Changing an Alarm Using the Alarm Text Table

As the Alarm Text Table lists all alarms and can be sorted, it is a fine starting point for finding and changing alarm definitions.

To change an alarm using the time alarm text table

- 1 In the **Options** menu, click **Alarm Text Table**.
- 2 In the click **Alarm Text Table** dialog box, double-click the row for required alarm.
- 3 In the **Edit Alarm Text** dialog box, modify the required alarm.



Note

- You can also start changing an alarm from the **Alarm Text Table** dialog by selecting the alarm and clicking the **Change** button.

13.14 Using Local Trend Logging in TAC Xenta

Trend logging in the TAC Xenta is a function for collecting (logging) values and store them in the TAC Xenta before transferring them to a monitoring and supervising system at a later time.

You can define logs for local trend logging in TAC Xenta 280/300/401 in the Menta application (the .mta file) when you create the application.



Important

- A Menta application designed for a Xenta 700 has no trend log definitions in Menta. The application uses XBuilder trend log objects in the device.

The logged point can be any public signal. Using different log intervals and log space, you can log rapid processes over a short time as well as slower processes at intervals of days or weeks.

The logged values have a date and time stamp.

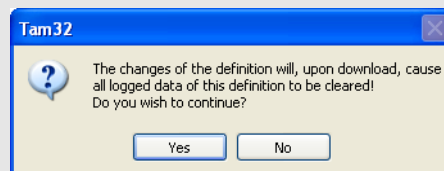
The trend logging in the TAC Xenta can be defined using TAC Menta, but the logged values can only be viewed using TAC Vista.

The number of available log channels is specified in the application. The detailed configuration of each trend log can be configured when the application is created using Menta online or during daily operation.



Important

- If you download changes of certain options in the Trend Log Definition to the TAC Xenta device, the changed log will stop and all logged values in the changed log will be lost.
 - The Logged Signal,
 - The Log Interval,
 - The Log Space,
 - The Non-circular log in TAC Xenta, no storage in TAC Vista option.
- You will receive a warning when closing the Trend Log Definition dialog.



13.14.1 Selecting a Trend Log

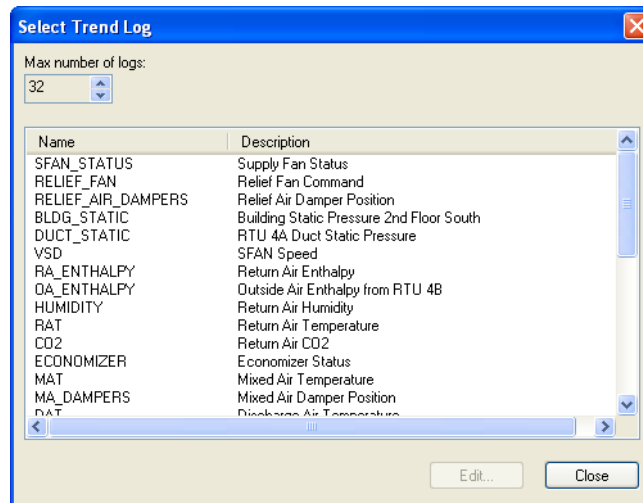
A listing of all local trend logs in the application is available in a select dialog.

Table 13.11: The Select Trend Log options.

| | |
|--------------------|--|
| Max number of logs | The maximum number of trend logs that can be defined in the device. Unconfigured trend logs or trend logs without a name, will be named “LOG<no.>”. |
| Name | The name of each trend log. |
| Descriptions | Description of each trend log. |
| Edit | Click this button to edit a selected trend log. |
| Close | Click this button to close the dialog. |

To select a trend log

- 1 In the **Options** menu, click **Trend Logs**.



- 2 In the **Select Trend Log** dialog box, click the required trend log.

13.14.2 Defining The Number of Trend Logs In a TAC Xenta Device

The maximum number of possible log channels in the application is specified in the application. The number of log channels is limited by the general application size and can only be increased by downloading the application, with an altered value, to the Xenta device.

To define the number of trend logs in a TAC Xenta application

- 1 In the **Options** menu, click **Trend Logs**.
- 2 In the **Select Trend Log** dialog box, in the **Max number of Logs**, enter the required number of logs (log channels). Use the **Up** or **Down** arrow to scroll to the required number.
- 3 Click **Close** when finished.



Tips

- To increase the number of log channels in an application you can do the following:
 - Editing the application file (the .mta file) and increase the number of log channels in TAC Menta.
 - Compiling the application, preferably by simulating, in TAC Menta.
 - Finish by downloading the application to the TAC Xenta device.

13.14.3 Configuring a Trend Log

You configure trend logs in a TAC Xenta 280/300/401 device in a trend log definition dialog.

Trend Log Definition

Name: \$FAN_STATUS

Description: Supply Fan Status

Signal

Logged Signal: \$FAN\SFAN_STATUS

Unit:

Hysteresis: 0.5

Log Interval: 6 Minute(s)

Log Space: 1 Day(s)

Control

Activate: Manual - On

Start Time: 2002-01-31 16:35:48

Start Variable:

No stop if logically activated

Clear log at start

Log Type

Circular log in TAC Xenta, circular storage in TAC Vista

Circular log in TAC Xenta, non-circular storage in TAC Vista

Circular log in TAC Xenta, no storage in TAC Vista

Non-circular log in TAC Xenta, no storage in TAC Vista

Log Space in TAC Vista: 2 Week(s)

Retrieve at % full: 80

Retrieve if connected and %: 60

OK Cancel Help

You can start a trend log manually, by setting a point of time to start the logging.

You can also start the trend log automatically, by defining a logical variable or time schedule to start the logging.

A log type defines the way logged data is stored locally in the Xenta device and optionally in TAC Vista. Storing the data in a circular log means that the logging continues when the log space is full and a new logged value overwrites the oldest value.

The Trend Log Definition dialog box contains the following options:

Table 13.12: The Trend Log Definition dialog options.

| | |
|-------------|--|
| Name | The trend log name. The trend log name can have a length of maximum 20 characters. The name of the trend log (physical resource) cannot be changed in the monitoring and supervising system. |
| Description | The trend log description. The trend log description can have a length of maximum 40 characters. |

Table 13.13: Signal definitions.

| | |
|------------------------|---|
| Logged Signal | The logged point. The point must be a public signal in the current application. |
| Hysteresis | The smallest change in an analog value required to log a value as new. Default value is 0.5. Note: using a hysteresis of 0.0 will store all values, even if the logged value is the same as previous. |
| Log Interval | The required time between loggings, ranging from 10 seconds – 530 weeks. |
| Log Space in TAC Vista | The required log space, expressed as a duration. |

Table 13.14: Logging control definitions.

| | |
|--------------------------------|---|
| Activate | The way to activate the logging. <ul style="list-style-type: none"> The options are: <ul style="list-style-type: none"> Manual-Off Manual-On Automatic |
| Start Time | The choosen date and time for starting a manually activated logging |
| Start Variable | Public variable in the current application, used to control an automatically activated logging. |
| No stop if logically activated | Select this option to inhibit the stopping of a log activated by a logical variable when the variable no longer is true. The logging continues until stopped manually, or the log is full when the option is selected. |
| Clear log at start | Select this option to clear the trend log when it is activated. |

Type of Logging

Table 13.15: Type of loggings

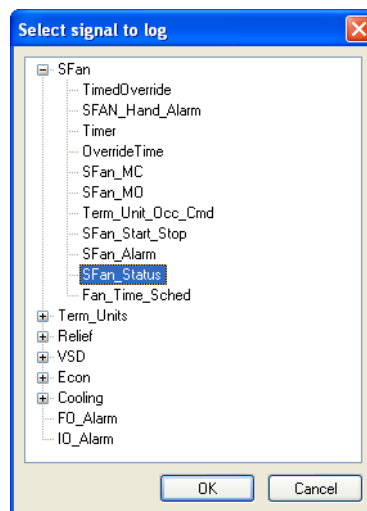
| | |
|---|---|
| Circular log in TAC Xenta, circular storage in TAC Vista. | Circular log in TAC Xenta, circular storage in TAC Vista. |
| Circular log in TAC Xenta, non-circular storage in TAC Vista. | Circular log in TAC Xenta, non-circular storage in TAC Vista. |
| Circular log in TAC Xenta, no storage in TAC Vista. | Circular log in TAC Xenta, no storage in TAC Vista. |
| Non-circular log in TAC Xenta, no storage in TAC Vista. | Non-circular log in TAC Xenta, no storage in TAC Vista. |

Table 13.16: Log space and retrieval definitions.

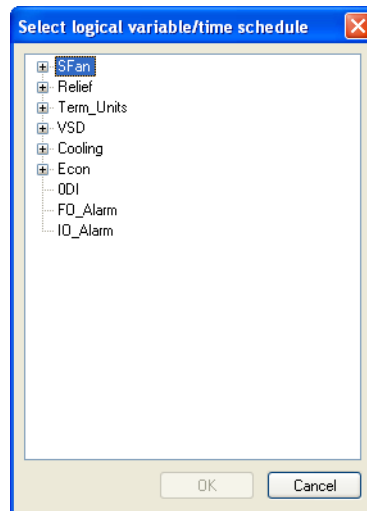
| | |
|-----------------------------|--|
| Log Space in TAC Vista | Specifies the required log space in TAC Vista if a log type with storage in TAC Vista is chosen. The required log space is expressed as a duration. |
| Retrieve at % full | Specifies the level of log space utilization, in percent, at which logged values are transferred from the TAC Xenta device and stored in TAC Vista. Logged values in the TAC Xenta will be transferred to a connected TAC Vista when the specified percentage is reached. |
| Retrieve if connected and % | Specifies the level of log space utilization (in percent) at which logged values are retrieved from the TAC Xenta device and stored in TAC Vista when a connection to TAC Vista is made for any reason. Logged values are always retrieved to TAC Vista during a so called slow poll. |

To configure a trend log

- 1 In the **Options** menu, click **Trend Logs**.
- 2 In the **Select Trend Log** dialog box, double-click the required trend log.
- 3 In the **Trend Log Definition** dialog box, in the **Name** box, type a suitable name for the log.
- 4 In the **Description** box, type a suitable description for the log.
- 5 In the Signal area, click the **Browse** button and browse to the required signal to log.



- 6 In the **Select signal to log** dialog box, select the required point to log.
- 7 Click **OK**.
- 8 In the **Hysteresis** box, enter the required hysteresis.
- 9 In the **Log interval** box, select the required log interval.
- 10 In the **Log space** box, select the required log space.
- 11 In the **Activate** box, select the required activation of the logging.
- 12 In the **Start Time** box, type the required start time for a manually started logging.
- 13 In the Control area, click the **Browse** button and browse to the required start variable for starting the log automatically.
- 14 In the **Select logical variable/time schedule** dialog box, select the variable for automatically starting the log.



- 15 Click **OK**.
- 16 Click **No stop if logically activated** option, if required.
- 17 Click **Clear log at start** option, if required.
- 18 Click the required option for log storage.
- 19 In the **Log Space in TAC Vista** boxes, select the required log space.
- 20 In the **Retrieve at % full** box, select the required percentage.

- 21 In the **Retrieve if connected and %** box, select the required percentage.

**Note**

- You can also start configuring a trend log by clicking **Edit** after selecting the required trend log in the **Select Trend Log** dialog box.

13.15 Setting Date and Time

You can set initial date and time values for offline simulation to make the testing of functions, depending on date & time of day easier.

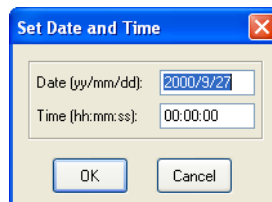
**Note**

- The **Set Date And Time** command is available also in the simulation mode.

If this is done in online-mode, the time will be changed in the TAC Xenta device.

To set date and time

- 1 In the **Options** menu, click **Set Date And Time**.



- 2 In the **Set Date and Time** dialog box, in the **Date** box enter the required date.
- 3 In the **Time** box enter the required time.
- 4 Click **OK**.

13.16 Undo

When you delete one or more function blocks including connected nodes, you can undo the command one step.

The **Undo** command will be unavailable if a subsequent command, except **File/Save**, is executed.

There will be no undo available when you delete a block defined as a SNVT. A warning dialog opens to let the user confirm the deletion.

To undo a command

- In the **Edit** menu, click **Undo**.



Tip

- You can also undo a command by typing **CTRL+Z**.

13.17 Using an Associated Text File

You can associated a text file to each function block diagram (.AUT or .MTA) file. The associated file can be used, for example, as an extended functional description.

The associated file has the same file name as the application but uses the file extension .TXT or .DOC, depending on text editor you choose.



Note

- You choose the text editor and the required file extension using the **Settings** dialog, on the **Preferences** menu.

If a text file with the same name as the application is missing, the editor opens a file with the content from on a file with the name DEFAULT.xxx. This file can be defined by the user and is normally located in the TAC Menta\Projects folder.

By default, the associated text file is saved in the specified library folder.

To use an associated text file

- 1 In the **Tools** menu, click **Text File**.

The text editor opens a file with the name of the application used as file-name ready to edit.

- 2 In the editing tool, edit the associated file
- 3 Save the file and close the editor.

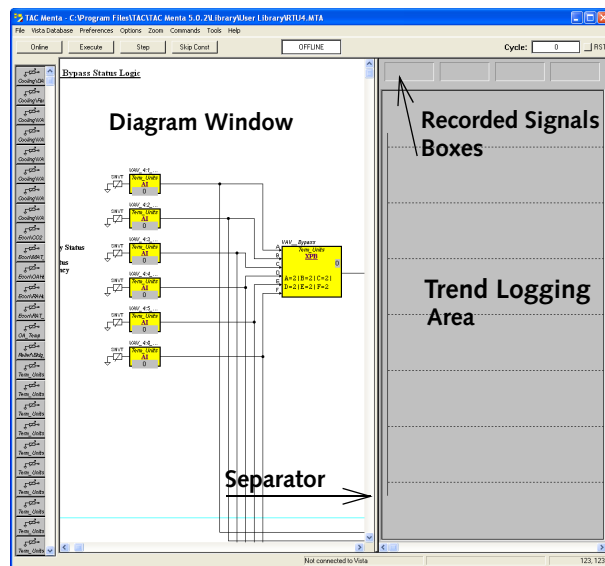
14 The Simulation Mode

The simulation mode in TAC Menta is the mode you can use to run the application and study the behavior of the signals.

When you run the simulation mode with a Xenta 280/300/401 connected directly to the PC, the signals are viewed in real time.

For more information on running the application in a connected device, see Chapter 16, “On-line Mode Functions”.

The application window in the Simulation mode consists of two parts, the diagram window and a trend logging area.



You can change the size of the trend logging area by dragging the separator sideways. Double-clicking the separator switches between a vertical or horizontal trend logging area.

The diagram window normally displays the function block diagram. The window can also display all public signals and parameters in a table, with the signals sorted in alphabetical order. The tabular mode is used when the function block diagram not is available, for example when loading a *.COD file or uploading the application from an online connected device.

You can control the simulation using either commands or buttons at the top of the window.

Buttons to the left in the window provide easy access to physical inputs for changing their values and states.



Note

- Saving a Menta function block diagram will also save the used settings for a simulation.

14.1 Changing to Simulation Mode

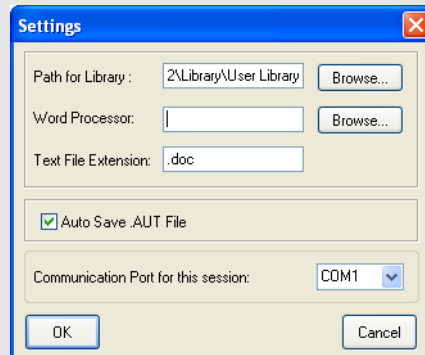
A control of the syntax in the application is automatically done when you change from the edit mode to the simulation mode.

If errors are detected, Menta remains in the edit mode and a dialog box indicates the cause of the error. The block that caused the error is displayed in the center of the diagram window in a green selection rectangle.



Notes

- Menta can automatically save the current function block diagram to a file called AUTOSAVE.AUT before changing to the simulation mode. Select **Auto Save .AUT File** box, in the **Settings** dialog on the **Preferences** menu.



To change to the simulation mode

- On the **Options** menu, click **Simulate**.



Note

- You can also enter the simulation mode by using function key **F12**.

14.2 Executing the Simulation

In the simulation mode you can execute the Menta application continuously or step-by-step to verify the application by observing the development of the signals.

The application program is executed as a cyclic application that runs at constant time intervals (the cycle time). All function block output signals are updated during each program cycle.

In the simulation mode, you can control the simulation in several ways. You can use commands, buttons or function keys.

Table 14.1: Execution control options.

| | |
|-----------------|--|
| Execute | Clicking Execute starts a continuous execution of the application. The execution stops when clicking Execute again or when clicking Step , Restart or RST . The function key F2 is an alternative. |
| Step | Clicking Step executes the application one cycle. The function key F3 is an alternative. |
| Execute n Times | Clicking Execute n Times simulates the application a defined number of cycles. You can view the cycle count in the cycle counter box. The simulation can be interrupted by clicking an alternative command or button. |
| RST | Clicking the RST button resets the cycle counter to zero, and all signals in the application return to their initial state. The function key F10 is an alternative. |
| Reset | Clicking Reset resets the cycle counter to zero, and all signals in the application return to their initial state. The function key F10 is an alternative. |

14.2.1 Starting a Simulation

When you simulate the application you can start the simulation in several ways. You can use either a command, button or function key.

To start a simulation

- In the **Commands** menu, click **Execute**.



Notes

- You can also start the simulation by clicking the **Execute** button.
- Alternatively you can also start the simulation by using the function key F2.

14.2.2 Simulating One Cycle Only

When you simulate the application you can execute the application one step at a time to observe the changes in the applications every cycle. A box shows the cycle number.

To simulate one cycle only

- In the **Commands** menu, click **Step**.

The cycle number in the **Cycle** box upper right side of the window is incremented.



Notes

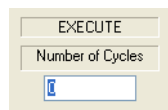
- You can also execute one cycle by clicking the **Step** button.
- Alternatively you can execute one cycle by using the function key F3.

14.2.3 Simulating a Defined Number of Cycles

To observe, for example delays, you can define the number of cycles to simulate.

To simulate a defined number of cycles

- 1 In the **Commands** menu, click **Execute n Times**.



- 2 In the **EXECUTE** dialog, in the **Number of Cycles** box, enter the required number of cycles.
- 3 Click **RETURN**.

14.2.4 Stopping the Simulation at a Limit

You can configure the simulation to stop when value exceeds defined levels (break-points).

For more information on stopping at break-points, see Section 14.11.6, “Stopping at a Limit”

14.2.5 Resetting the Cycle Counter

When you reset the counter to zero, all signals in the application return to their initial state.

To reset the cycle counter

- In the **Commands** menu, click **Reset**.



Notes

- You can also reset the cycle counter by clicking the **RST** button.
- Alternatively you can reset the cycle counter by using the function key F10.

14.2.6 Stopping the Simulation

During simulation you can stop the temporary, change to simulate one step or restart the simulation.

To stop the simulation

- In the **Commands** menu, click **Execute**.



Notes

- You can also stop the simulation by clicking the **Execute** button.
- Alternatively you can also stop the simulation by using the function key F2.
- Returning to the **Edit** mode also stops the simulation.

14.3 Highlighting a Connection

Highlighting a connection (signal) in selected colors is highly valuable when tracing signals in a block diagram. When you mark a signal, all branches of the connection are colored.

You can highlight connections both in the simulation and the edit mode.

To highlight a connection

- 1 Right-click the required connection.
- 2 In the **SIGNAL** menu, click **Mark**.
- 3 In the submenu, click the required color.



Notes

- You can revert to the normal colored connection by right-clicking the connection and clicking **Unmark** in the **SIGNAL** menu.
- To revert to the normal color for all marked connections, you can right-click a connection and then click **Unmark All** in the **SIGNAL** menu.

14.4 Simulating Physical Inputs Manually

When simulating in TAC Menta, you can manage the physical inputs in different ways. One way is to use the input buttons manually. When the value for an input is modified manually, it remains constant until modified again.

There are buttons for each physical input block in the application. The buttons show the name of the corresponding input block. The buttons are placed on the left side of the diagram window. A scroll bar is added if the number of inputs exceeds the available space.

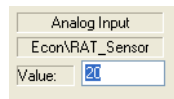
14.4.1 Changing a Physical Analog Input Manually

The simulated values generated by the input buttons are independent of the I/O configuration parameters for the input.

When you click the buttons for the analog inputs, you open a dialog where you define the simulated value. Analog values are entered in the engineering units defined for the input block.

To change a physical analog input manually

- 1 In the left part of the Diagram window, click the required input button.



- 2 In the **Analog Input** dialog, in the **Value** box, type the required value.
- 3 Click **RETURN**.



Tip

- You can identify the signal for each button by pointing the button and read the signal name in the status bar.

14.4.2 Changing a Physical Binary Input Manually



The simulated values generated by the input buttons are independent of the I/O configuration parameters for the input.

The Normally Open parameter in a DI block has no effect during simulation. The simulated state is directly dependent of the input button.

The buttons for physical digital inputs act like switches. The buttons toggle between open and closed state each time you click them.

To change a physical binary input manually

- In the left part of the Diagram window, click the button for the required input.

The symbol on the input button changes between an open , or a closed  switch.



Tip

- You can identify the signal for each button by pointing the button and read the signal name in the status bar.

14.5 Simulating Physical Inputs Automatically

In the simulation mode, you can also generate signal variations for physical inputs automatically. This is called the automatic mode. You define the conditions for the variations in a dialog.

14.5.1 Generating an Analog Input Signal Automatically

There are a number of options to use for creating the automatically generated analog signal. The signal variations you can define for an analog input are determined by the following options:

Table 14.2: Analog Input options.

| | |
|-----------|--|
| Manual | Select the Manual option to manually control the input values. |
| Automatic | Select the Automatic option for automatically generated input values. |

Table 14.2: (Contd.)Analog Input options.

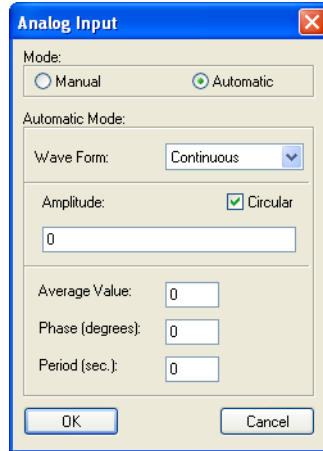
| | |
|---------------|--|
| Wave Form | <p>Defines the shape of the generated signal.</p> <p>The available shapes are:</p> <ul style="list-style-type: none"> Continuous Square Sinusoidal Saw tooth Triangle Pulse |
| Amplitude | <p>In the Amplitude box you can define a sequence of wave periods with different amplitudes for the generated signal.</p> <p>Separate each period of the sequence with a comma sign.</p> <p>The amplitude is generated with the units defined for the analog input block.</p> |
| Circular | <p>Select the Circular option to repeat the sequence, defined in the Amplitude box.</p> <p>When the option not is selected, the input reverts to manual mode after generating the sequence once.</p> |
| Average Value | <p>In the Average Value box you define the average value for the automatically generated signal.</p> <p>For example, an average value of 50 and an amplitude of of 10 generates a wave ranging from 40 to 60.</p> <p>The average value is generated using the units defined for the analog input block.</p> |
| Phase | <p>In the Phase box you can define a time displacement of the generated signal, measured in degrees. A complete period is 360 degrees.</p> <p>For example, a sinusoidal wave with a phase of 90 is equivalent to a cosine.</p> |
| Period | <p>In the Period box you can define the duration of a complete wave period measured in seconds.</p> |
| OK | <p>Clicking the OK button closes the dialog box and introduces the changes.</p> |
| Cancel | <p>Clicking the Cancel button closes the dialog box without introducing the edited changes.</p> |

The simulated values generated by the input buttons are independent of the I/O configuration parameters for the input. The exception to this is the Time Constant parameter in the AI block. An automatically gener-

ated signal will be filtered using the same filter function as defined by the Time Constant parameter in the AI block.

To generate an analog input signal automatically

- 1 In the left part of the Diagram window, right-click the button for the required analog input.



- 2 In the **Analog Input** dialog box, click the **Automatic** option.
- 3 In the **Wave Form** list, select the required wave form.
- 4 Select the **Circular** box if repeated lapses are required.
- 5 In the **Amplitude** box, enter the required amplitude.
- 6 In the **Average Value** box, enter the required value.
- 7 In the **Phase** box, enter the required phase relationship.
- 8 In the **Period** box, enter the required period.
- 9 Click **OK**.

14.5.2 Generating a Binary Input Signal Automatically

There are a number of options to use for creating the automatically generated binary signal. The signal variations you can define for an binary input are determined by the following options:

Table 14.3: Digital Input options.

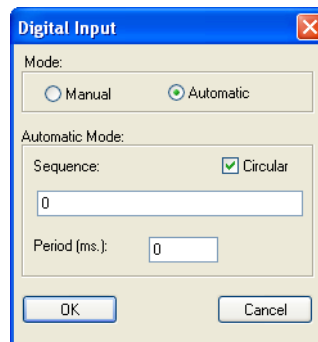
| | |
|-----------|---|
| Binary | Select the Binary option to manually control the input values. |
| Automatic | Select the Automatic option for automatically generated input values. |
| Sequence | In the Sequence box, you can define a sequence of ones and zeroes for generating a square wave. Use no separator. Each digit in the sequence has a duration of the defined period. |

Table 14.3: Digital Input options.

| | |
|----------|---|
| Circular | Select the Circular option to repeat the sequence, defined in the Sequence box. When the option not is selected, the input reverts to manual mode after generating the sequence once. |
| Period | In the Period box you can define the duration of each digit in the defined Sequence . The duration is measured in milliseconds. |
| OK | Clicking the OK button closes the dialog box and introduces the changes. |
| Cancel | Clicking the Cancel button closes the dialog box without introducing the edited changes. |

To generate a binary input signal automatically

- 1 In the left part of the Diagram window, right-click the button for the required analog input.



- 2 In the **Digital Input** dialog box, click the **Automatic** option.
- 3 Select the **Circular** box if repeated lapses are required.
- 4 In the **Sequence** box, type the required sequence.
- 5 In the **Period** box, enter the required period.
- 6 Click **OK**.

14.6 Modifying Signals During Simulation

During the simulation, you can modify the value or state of signals with a RW (read/write) access. Signals with RO (read only) access can not be modified.

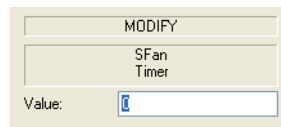
The change of a signal might immediately be overwritten, depending on other application.

14.6.1 Modifying an Analog Signal

You change the value of an analog signal using a dialog.

To modify an analog signal

- 1 In the diagram window, click the required block (signal).



- 2 In the **MODIFY** dialog, in the **Value** box, type the required value.
- 3 Click **RETURN**.



Notes

- To change the value of the signal you can also right-click the connection and then click **Modify** in the **SIGNAL** menu.
- You can also close the **MODIFY** dialog by clicking inside the dialog again.
- You can cancel the change by clicking the **ESCAPE** key.

14.6.2 Modifying a Binary Signal

A binary (logical) signal can have two values. You can change these values using the status option:

- A filled circle in the Status option changes the status to logical one when closing the dialog.
- An empty circle in the Status option changes the status to logical zero when closing the dialog.

To modify a binary signal

- 1 In the diagram window, click the required block (signal).



- 2 In the **ORDER** dialog, ensure the **State** option reflects the required state of the signal:
 - A filled circle to simulate a logical one.
 - An empty circle to simulate a logical zero.
- 3 Click **RETURN**.



Notes

- To change the state of the signal you can also click the connection and then click **Modify** in the **SIGNAL** menu.
- You can also close the ORDER dialog by clicking inside the dialog again.
- You can cancel the change by clicking the **ESCAPE** key.

14.7 Modifying Block Parameters During Simulation

Internal parameters in function blocks cannot be modified during simulation. You can only view their values.

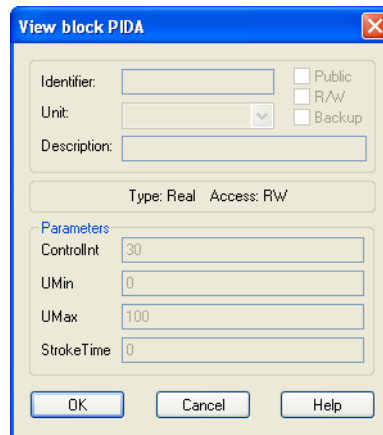
When a public constant is used for a block parameter, you can change the value using the constants table.

14.7.1 Viewing Parameters in a Function Block

When you simulate an application you can only view the internal block parameters as they are shown in the block dialog.

To view parameters in a function block

- 1 In the diagram window, right-click the required block.
- 2 In the signal menu, click **View**.
- 3 In the View block dialog box, view the available values.



- 4 Click **OK** or **CANCEL**.

14.7.2 Changing the Value of a Constant

In the simulation mode, you can change values for public constants. Values of non-public constants can be viewed but not changed.

Values for public constants, modified from the Constants Table, update the source code for the application, except when changes are made during a simulation in tabular mode. For more information on the tabular mode, see Section 14.9, “Simulating Executable Files”

For more information on the Constants Table, see Chapter 13.10, “Using Constants”.

To change the value of a constant

- 1 In the **Options** menu, click **Constants Table**.
- 2 In the **Constants Table** dialog, double-click the required constant.
- 3 In the **Edit constant** dialog, enter the required value.
- 4 Click **OK**.
- 5 Click **Exit**.

14.7.3 Changing a Time Schedule

When you make changes in a TSCH function block in the simulation mode, the changes are retained when you return to the Edit mode.

Changes of week and holiday chart settings in the TSCH, done in online mode, are transferred directly to the device.



Important

- You must generate and download the application to the device if the numbers of week or holiday charts are changed.

To change a time schedule

- 1 In the **Options** menu, click **Time Schedule Table**.
- 2 In the **Time Schedule Table** dialog, double-click the required time schedule.
- 3 In the **Time Schedule** dialog, enter the required change.
- 4 Click **OK**.
- 5 Click **Exit**.

14.7.4 Changing the Binding of an I/O Point

When you make changes to I/O binding data in the simulation mode, the changes are retained when you return to edit mode.



Important

- To transfer the changes to the device, you must generate and download the application after changing I/O binding data.

When you simulate the application in tabular mode, you can only access the I/O binding information through the I/O Binding table.

For more information on the I/O Configuration Table, see Chapter 13.11, “Using the I/O Configuration Table”.

To change the binding of an I/O point

- 1 In the **Options** menu, click **I/O Configuration Table**.
- 2 In the **I/O Configuration Table**, click the row for the required I/O point.
- 3 In the corresponding **Bind** dialog, make the required change.
- 4 Click **OK**.
- 5 Click **Exit**.

14.7.5 Changing an Alarm Text

In the simulation mode you can only access alarm texts starting from the Alarm Text Table.

Changes always update the source code.

For more information on the Alarm Text Table, see Chapter 13.13, “Using the Alarm Text Table”.



Important

- To transfer the changes to the device, you must generate and download the application after changing alarm texts.

To change an alarm text

- 1 In the **Options** menu, click **Alarm Text Table**.
- 2 In the **Alarm Text Table**, click the row for the required alarm.
- 3 In the **Edit Alarm Text** dialog, make the required change.
- 4 Click **OK**.
- 5 Click **Exit**.

14.8 Simulation Using Test Probes

Sometimes testing and debugging Menta applications can be improved when control loops are closed.

A model that describes the dynamics of the controlled system can be programmed using Menta function blocks and be used for closing the control loop.

The function blocks for the model are temporarily included in the function block diagram for the simulated application.



Important

- You can not download applications containing test probes to the TAC Xenta device. They must be removed before downloading.

The model is applied to the application under simulation, using TPAO/TPDO test probe blocks to read the physical outputs in the application. Feedback from the modelled response is entered to the physical inputs in the application, using TPAI/TPDI test probe blocks.

The simulated application reads the physical input blocks and controls the physical outputs.

When you simulate an application, you can not change values for physical inputs using the input buttons if the inputs are connected to test probe blocks. You can only read the values of the inputs.

For more information on the test probe blocks, see Chapter 24, “Test Probe Blocks”.

14.9 Simulating Executable Files

You can use Menta in the simulation mode to simulate executable files, so called .COD files. The file type does not contain the information needed to create the function block diagram for the application. The application can be simulated offline, executed online and configured in its tabular form.

The source code for the application can not be modified.



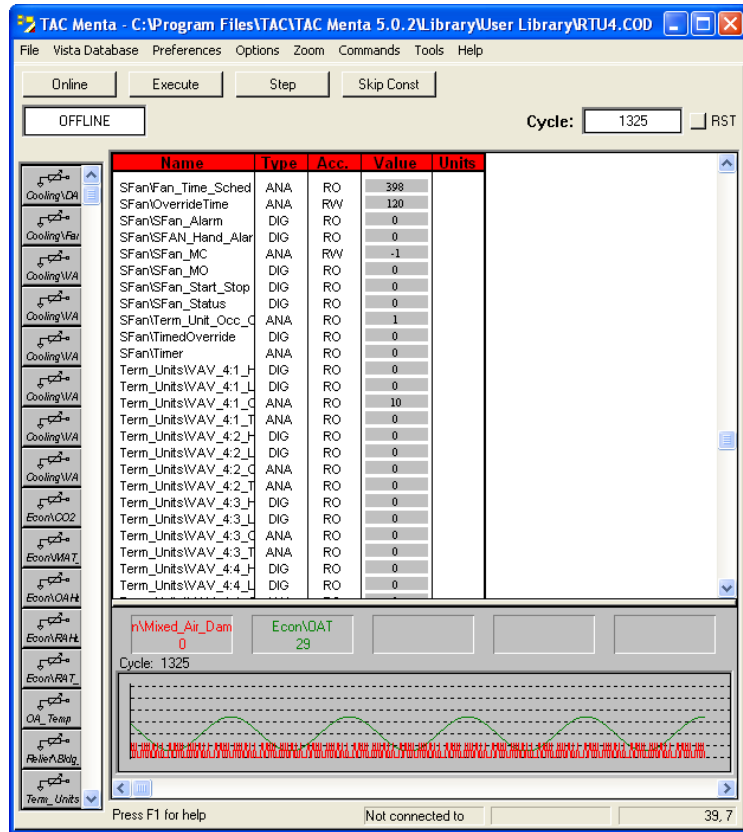
Important

- You can not enter the **Edit** mode for an application loaded as a .COD file.

To simulate an executable file

- 1 In the **File** menu, click **Open**.
- 2 In the **Open** dialog, in the **Files of Type** list, select **FBD (*.COD)**.
- 3 Browse to the required file to load.

4 Select the required file and click **OPEN**.



All signals in the application are alphabetically sorted in the diagram window.



Important

- To revert to the edit mode after loading a executable files (so called .COD files) you must load an application file with a function block diagram and then click function button **F5**.
- Alternatively, after loading an application file with a function block diagram, you can click **View Diagram** in the **Preferences** menu to change between tabular and diagram view.

14.10 Generating Executable Code

You can generate the executable program code for the application when you are working in the simulation mode of TAC Menta.



Note

- The version of the downloadable code files are defined by the System Version setting in the Device Specification

The following files are created, where “*” denotes the file name of the application program source code (.MTA/.AUT) file:

Table 14.4:

| | |
|-------|--|
| *.COD | File with code, in ASCII format, for the TAC Xenta. Executable file that is downloaded to the controller. |
| *.ESP | Specification file: table containing public signals, their attributes. This file is used as the input data file for the OP configuration tool. |
| *.XIF | The external interface file is generated if SNVTs are included in the application program. It contains a standardized description of all the network variables/objects of the application program, which enables binding and communication with LonWorks nodes from other manufacturers. |

The file name and date of the source code (.MTA/.AUT) file is included in the object and executable files. This data is included in the information transferred from the controller to the PC during upload, see Chapter 16.4.2, “Uploading From the Xenta Device”.

When the executable code is generated, new OP files can also be generated, either automatically or user selected.

In doing so, the OP configuration tool is invoked with the .OPC file of the current application, but with “Name” and “Abbreviation” taken from the current program specification. The following files are automatically generated:

Table 14.5:

| | |
|-------|--|
| *.OPC | An OP menu tree file, which consists of a mix of the .OPE (empty menu tree) and the .ESP file. |
| *.BIN | A binary file with menu tree data for downloading to the TAC Xenta controller together with the .COD file. |

Table 14.5: (Contd.)

| | |
|-------|--|
| *.CHR | A national character set file for downloading to the TAC Xenta controller together with the .COD file. |
|-------|--|

To generate application program code

- In the Simulation mode, in the Commands menu, click **Generate**.

14.11 Trend Logging

The trend logging of signals in the simulation mode permits the user to simultaneously record the development over time for six signals in the application.

Analog and binary signals can be shown simultaneously in the trend logging area.



Note

- Saving a Menta function block diagram will also save the used settings for a simulation.

14.11.1 Adding an Analog Signal to the Recorder

When you record an analog signal, you define the plot range (scale) for the signal in a dialog. The scale of the y-axis for the plotted signal is defined using a maximum and a minimum value.

The default plot range is 0–100.

To add an analog signal to the recorder

- In the diagram window, right-click the function block for the required signal.
- In the **SIGNAL** menu, click **Record**.

- In the **RECORD** dialog, in the **Maximum Value** box, enter the required value.
- In the **Minimum Value** box, enter the required value.

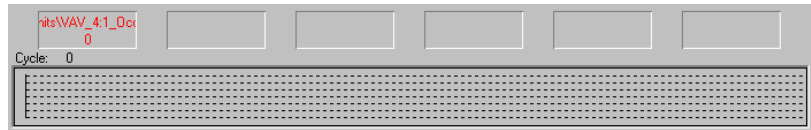
5 Click **RETURN**.



Tip

- You can also close the RECORD dialog by clicking the dialog box once more.

One of the six the channel boxes in the trend logging area shows the name of the signal added to the recorder.



Tip

- To view the full name of the signal you can point to the channel box and view the signal name in the status bar.

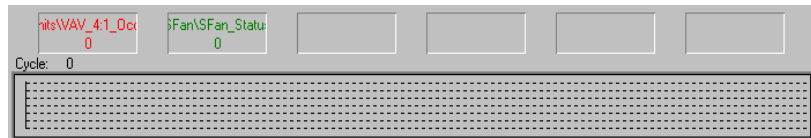
14.11.2 Adding a Binary Signal to the Recorder

When you record a binary signal, the plot range (scale) is always 0 to 1 .

To add a binary signal to the recorder

- In the diagram window, right-click the function block for the required signal.

The binary signal is immediately added to the recorder and the name of the signal is shown in one of the six the channel boxes in the trend logging area.



Tip

- To view the full name of the signal you can point to the channel box and view the signal name in the status bar.

14.11.3 Removing a Signal From the Recorder

When you don't want to record a signal any more, you can remove it from the recorder.

To remove a signal from the recorder

- 1 In the trend logging area, right-click the required recorded signal box.
- 2 In the **RECORD** menu, click **Delete**.



Note

- You can also remove a signal from the recorder by right-clicking the function block for the signal, and then click **Delete** in the **SIGNAL** menu.

14.11.4 Clearing the Recorder

When you clear the recorder, you erase all graphs in the diagram. The definition of signals to record are retained. If the simulation continues, new graphs are drawn starting from the left in the trend logging area.

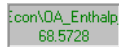
To clear the recorder

- Right-click trend logging area, and in the **TREND** menu, click **Clear**.

14.11.5 Editing the Range of a Recorded value

You can change the range of a recorded signal during the simulation of an application.

To edit the range of a recorded value

- 1 Right-click in the required recorded signal box .
- 2 In the **RECORD** menu, click **Edit Range**.
- 3 In the **RECORD** dialog, in the **Maximum Value** box, enter the required value.
- 4 In the **Minimum Value** box, enter the required value.
- 5 Click **RETURN**.



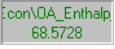
Tip

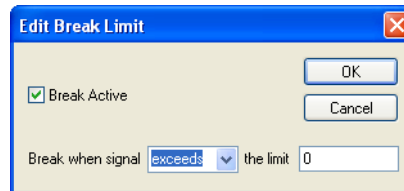
- You can also close the **RECORD** dialog by clicking the dialog box once more.

14.11.6 Stopping at a Limit

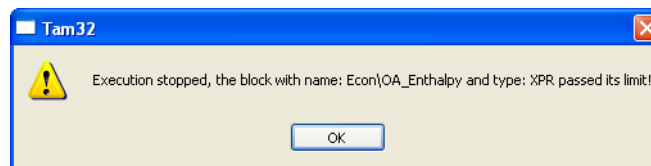
You can configure the logging to stop logging and simulation when logged value exceeds defined levels. You can define limits for more than one signal.

To stop logging at a limit

- 1 Right-click in the required recorded signal box .
- 2 In the **RECORD** dialog, click **Stop at Limit**.



- 3 In the **Edit Break Limit** dialog, click **Break Active**.
- 4 In the **Break when signal** list, select the required condition, (**exceeds** or **is below**).
- 5 In the **limit** box, enter the required value for the limit.
- 6 Click **OK**.
- 7 Execute the simulation.
- 8 The simulation stops when the limit is exceeded.



- 9 In the **TAM 32** dialog, click **OK** to continue the simulation.



Tip

- To continue a simulation without confirming the break, you can also click the **Step** button.

14.11.7 Restarting the Recorder

When you restart the recorder, it continues the graphs, starting from the left in the trend logging area. Previous drawn graphs are unchanged.

To restart the recorder

- Right-click in the trend logging area, and in the **TREND** menu, click **Restart**.

14.11.8 Resetting the Recorder

When you reset the recorder, you remove all defined signals to log. Previous drawn graphs are unchanged.

The user is asked to confirm the operation in a dialog before the **Reset** is done.

To reset the recorder

- 1 Right-click in the trend logging area, and in the **TREND** menu, click **Reset**.
- 2 In the **TAC Menta** dialog box, click **OK**.

14.11.9 Viewing Sampled Values

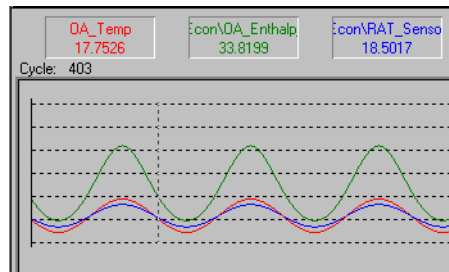
In the trend logging area you can view the logged signals values at any point of time.

Placing a cursor at the required point of time allows you to view the values for each signal in the recorded signal boxes. You can view the sampling cycle in the cycle counter.

To view sampled values

- 1 In the trend logging area, click at the required point of time.

A dotted vertical line, the cursor, appears at the selected point.



- 2 In the **Cycle** counter, view the cycle for the sampling.
- 3 In the recorded signal boxes, view the logged values for the sampling.
- 4 Right-click the trend logging area when finished.

14.11.10 Scanning Sampled Values

You can scan the graph and view each sampled values and the cycle they were sampled.

To scan sampled values

- 1 In the trend logging area, click at the required point of time.
- 2 Click and hold the left button, while moving the cursor.
- 3 In the recorded signal boxes, view the logged values for each sampling.
- 4 Right-click the trend logging area when finished.



Tip

- To achieve a precise control of the cursor movements, you can use the left or right arrow keys while pressing the CTRL key.

14.11.11 Defining the Sampling Period

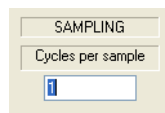
By default, the recorder samples each defined signal every program cycle. This allows you to view values in the trend log for every program cycle. You can define the number of program cycles executed between each sampling.

The sampling rate for trend logging in offline mode is the defined number of program cycles.

When you simulate in online mode, the sampling rate for the trend log is expressed in seconds. However, due to certain limitations, this sampling rate is not very accurate.

To define the sampling period

- 1 In the **Commands** menu, click **Sampling**.



- 2 In the **SAMPLING** dialog, in the **Cycles per sample** box, enter the required number of cycles.
- 3 Click **RETURN** to close the dialog.

15 The Logger Tool

The Logger tool in TAC Menta is an extension to the trend logging during simulation. The Logger tool allows you to make further analysis possible and to save recorded values to a file.



Important

- To record a signal in the Logger tool, you must define the signal to be recorded in the Menta simulation. The data is recorded in the Logger tool when the simulation is executed.

The Logger Tool opens a separate window where you can view the logged signals in a single diagram window or in separate diagram windows. You can define different properties for each diagram.



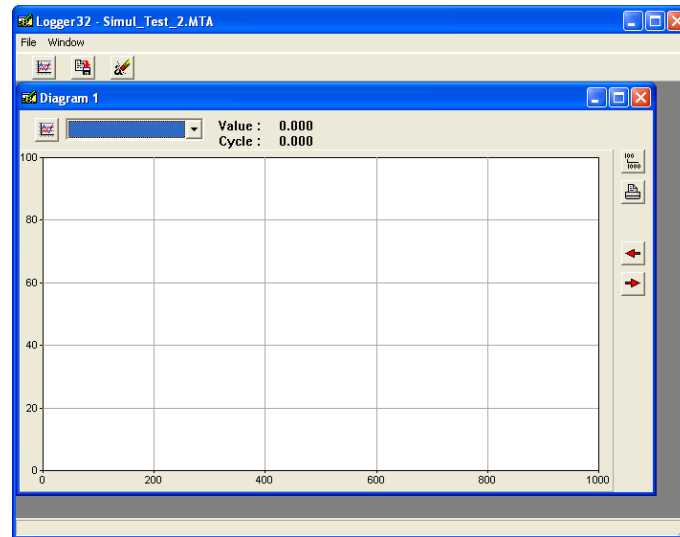
15.1 Opening the Logger

The values are recorded in the Logger when the application is simulated in Menta.

To open the Logger

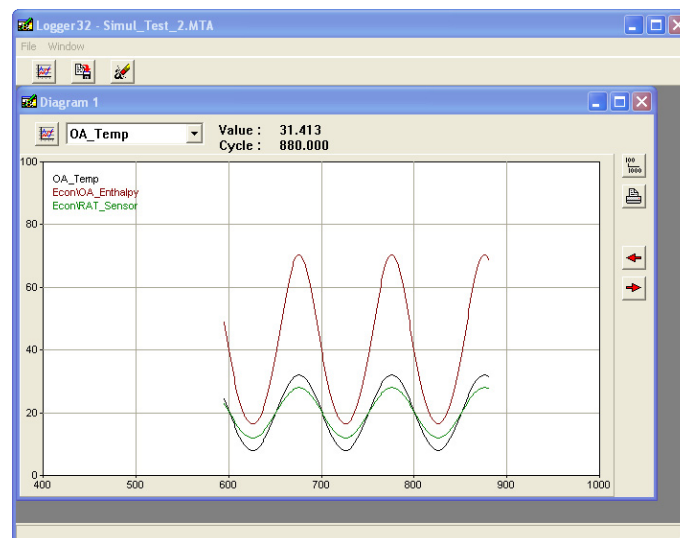
- 1 In the **Tools** menu, click **Logger**.

The Logger 32 window opens and includes the diagram window.



- 2 In Menta, select all signals to be recorded in the Logger tool. See Chapter 14.11.1, "Adding an Analog Signal to the Recorder".
- 3 Start the simulation in Menta.


The logged values and the signal names and program cycle number are shown in the Logger diagram window.

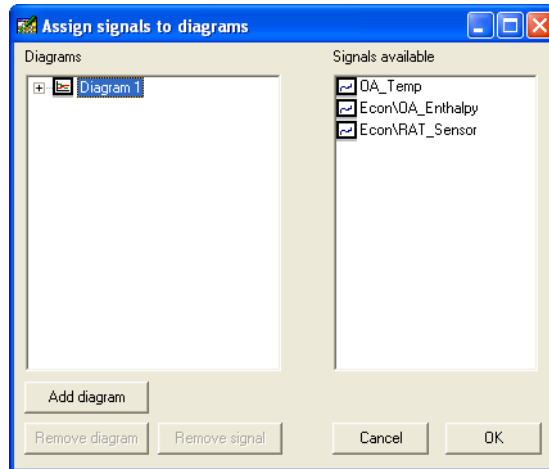


15.2 Adding a Diagram Window

The Logger tool always starts with one diagram window that collects all the logged signals. More diagrams can be added when required.

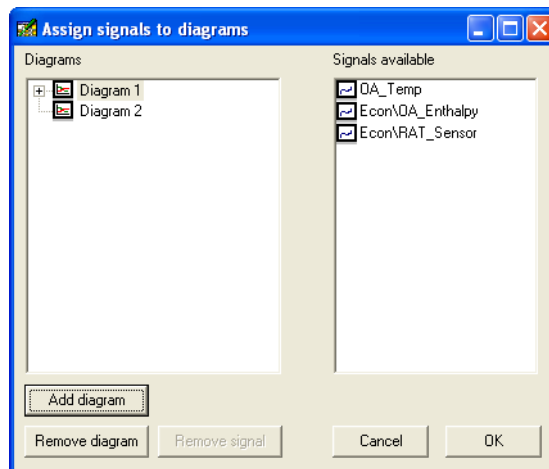
To add a diagram window

- 1 In the **Logger 32** toolbar, click the **Setup** button .



- 2 In the **Assign signals to diagrams** dialog, click the **Add diagram** button.

A new diagram is added to the **Diagrams** box.



- 3 Click **OK**.



Note

- You can remove a diagram from the Logger by selecting the diagram in the **Diagrams** box, and then clicking the **Remove diagram** button.

15.3 Configuring a Diagram Window


To adapt a diagram window to your requirements, you can choose the scaling, the use of grids and other properties for each diagram.

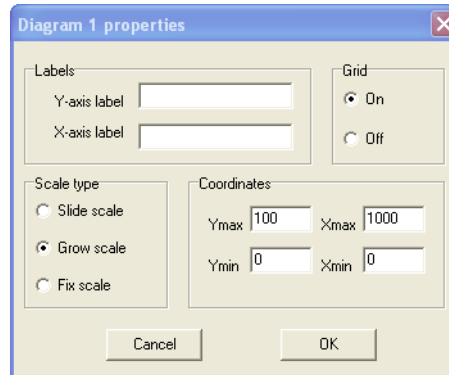
The following properties are available:

Table 15.1: The Diagram window options.

| | |
|--------------|---|
| Labels | |
| Y-axis label | The label for the X axis in the graph. |
| X-axis label | The label for the Y axis in the graph. |
| Grid options | |
| On | Select this option to turn the grid in the graph On. The grid On is the default option. |
| Off | Select this option to turn off the grid in the graph. |
| Scale Types | |
| Slide scale | Select this option to let the X-axis automatically scroll when viewing recorded values. Old values are scrolled out to the left. |
| Grow scale | Select this option to have all recorded values visible, letting the X-axis scale automatically expand when new values are logged. The Grow scale is the default option. |
| Fix scale | Select this option to use the defined Xmax and Xmin values (Coordinates) for the X-axis to view values. Recorded values outside these limits are not visible in the diagram. |
| Coordinates | |
| Ymax | The maximum value for the Y axis. The default value is 100. |
| Ymin | The minimum value for the Y axis. The default value is 0. |
| Xmax | The maximum value for the X axis. The default value is 1000. |
| Xmin | The minimum value for the X axis. The default value is 0. |

To configure a diagram window

- 1 In the Diagram window, click the **Setup** button .



- 2 In the **Diagram properties** dialog, in the **Y-axis label** box, type an appropriate label for the axis, when required.
- 3 In the **X-axis label** box, type an appropriate label for the axis, when required.
- 4 In the Grid area, click the required option.
- 5 In the Scale type area, click the required option.
- 6 In the Coordinates area, type the required value for **Ymax**, when required.
- 7 In the Coordinates area, type the required value for **Ymin**, when required.
- 8 In the Coordinates area, type the required value for **Xmax**, when required.
- 9 In the Coordinates area, type the required value for **Xmin**, when required.
- 10 Click **OK**.

15.4 Adding a Signal to the Diagram Window

The Logger tool starts with all the logged signals collected in one diagram window. You can add recorded signals to diagram windows with a scaling adapted to the actual signal variations.


In the dialog for setting up the diagrams, you see all signals that are recorded in the Signals available box. These are the signals you can choose to view in one or more diagrams.

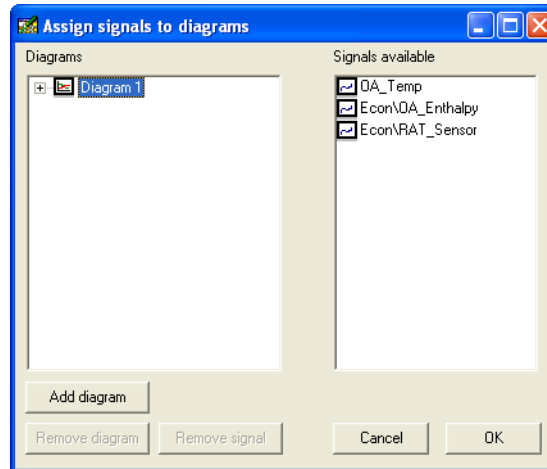


Note

- To view the signals in the **Signals available** box, you have to execute the simulation at least one step.

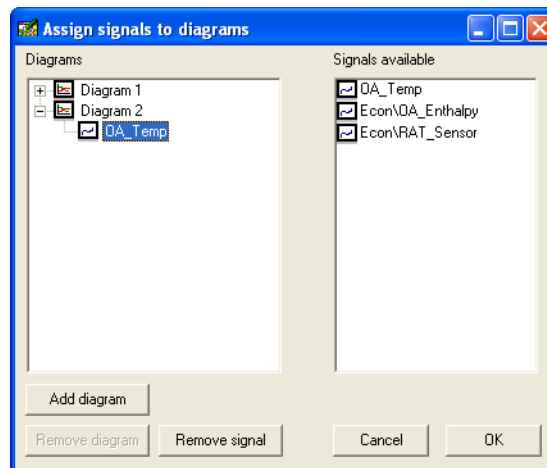
To add a signal to the diagram window

- 1 In the **Logger32** toolbar, click the **Setup** button .



- 2 In the **Assign signals to diagrams** dialog, in the **Signals available** box, drag the signal you want to view to the required diagram in the **Diagrams** box.

The signal is added to the diagram.



- 3 Click **OK**.



Note

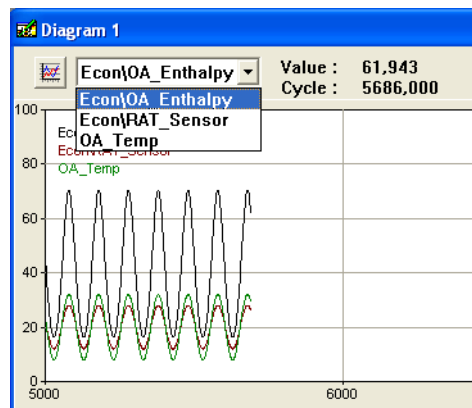
- You can remove a signal from a diagram by selecting the signal in the **Diagrams** box and then clicking the **Remove signal** button.

15.5 Viewing the Diagrams

The logged values are displayed as a graph in a diagram window. The diagram window can be configured to the particular need. For more information on this, see Section 15.3, “Configuring a Diagram Window”.

The names of the signals shown in a diagram are displayed to the left in the graphs area.

The current cycle and recorded signal value is displayed above the graph. If there is more than one recorded signal in a diagram, you can select which signal to display numeric values for.



15.5.1 Viewing the Diagrams Tiled Horizontally

If you have more than one diagram in the Logger, you can view them tiled or cascaded.

To view the diagrams tiled horizontally

- In the menu bar, click **Window** and then click **Tile**.





Note

- You can also view the diagram windows cascaded by clicking **Cascade** in the **Window** menu.

15.5.2 Scrolling a Graph Horizontally

To view logged values that are no longer visible in the diagram, you can scroll the graph along the X axis.

To scroll a graph horizontally

- In the Diagram window, click one of the scroll buttons:
 - the left scroll  button.
 - the right scroll  button.

15.5.3 The Default Scaling of a Graph

The logged values are displayed in the diagram window as a graph using the properties defined for the diagram.

You can use the default scale button to restore the diagram scale to the default settings (X = 0–1000, Y = 0–100).

To default scale a graph

- In the Diagram window, click the Default Scale  button.

15.5.4 Zooming in a part of a diagram

You can zoom in a selected part of a diagram for a detailed analysis.


The scaling of the Y-axis and the X-axis in the diagram are automatically changed so that the selection uses the complete diagram window.

To zoom in a part of a diagram

- In the diagram window, press the left mouse button and make a selection rectangle that covers the part of the graph you want to study in detail.



Tip

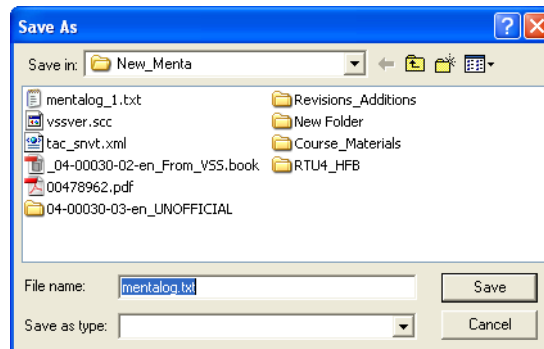
- You can revert to the default scaling by clicking the Default Scale button .

15.6 Saving a Log File

Using the Logger tool, you can save the logged values to a file. The file format is a text file stored in ASCII format. Signal Names, Signal values and Program cycle number are saved in the file.

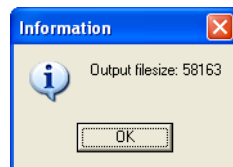
To save a log file

- 1 In the **File** menu, click **Save log**.



- 2 In the **Save As** dialog, browse to the location where you want to save the logged data.
- 3 In the **File name** box, type the required name for the file.
- 4 Click **Save** to continue.


The file is saved and a dialog opens, informing about the size of the saved file.



- 5 Click **OK**.



Note

- You can also start saving the logged data by clicking the  button

15.7 Viewing a Log File

You can use the Logger tool to view a file with logged and saved data.

| | OA_Temp Econl | OA_Enthalpy | EconlRAT_Sensor |
|----|---------------|-------------|-----------------|
| 1 | 19.247646 | 36.927792 | 19.49843 |
| 2 | 18.497149 | 35.35059 | 18.9981 |
| 3 | 17.752583 | 33.819893 | 18.501722 |
| 4 | 17.016884 | 32.339508 | 18.011255 |
| 5 | 16.292957 | 30.9128 | 17.528639 |
| 6 | 15.583662 | 29.542742 | 17.055775 |
| 7 | 14.891795 | 28.231932 | 16.59453 |
| 8 | 14.220086 | 26.982647 | 16.146725 |
| 9 | 13.571189 | 25.796877 | 15.714127 |
| 10 | 12.947655 | 24.676336 | 15.298437 |
| 11 | 12.351955 | 23.622555 | 14.901304 |
| 12 | 11.786439 | 22.636839 | 14.524292 |
| 13 | 11.253336 | 21.720324 | 14.168891 |
| 14 | 10.754748 | 20.873997 | 13.836498 |
| 15 | 10.292645 | 20.098724 | 13.528431 |
| 16 | 9.868851 | 19.39525 | 13.245901 |
| 17 | 9.485038 | 18.764221 | 12.990025 |
| 18 | 9.142719 | 18.206188 | 12.761813 |
| 19 | 8.843247 | 17.721628 | 12.562164 |
| 20 | 8.587804 | 17.31094 | 12.391869 |
| 21 | 8.377396 | 16.974451 | 12.251596 |
| 22 | 8.212852 | 16.712429 | 12.141901 |
| 23 | 8.094827 | 16.52508 | 12.063217 |

Fig. 15.1: Log file example.

To view a log file

- 1 In the Logger 32, in the **File** menu, click **View log**.
- 2 In the **Open** dialog, browse to the saved file.
- 3 Click **Open**.




Important

- The first time you want to open a log file from the Logger, you have to define (browse to) the application, for example NOTE-PAD.EXE, for the Logger tool to use when viewing the file.

15.8 Printing a Logger Diagram

You can print a diagram on the default printer for the PC.

To print a logger diagram

- In the Diagram window, click the Print  button.

15.9 Clearing the Logger Diagrams

When you clear the Logger, you delete the recorded values in all diagrams.

The logging of values and the graphs continues if the application still is simulated. All values logged before clearing a log are lost when you save the log to a file.



Important


- When you use the **Clear log** command, values for the Coordinates Xmax, Xmin, Ymax and Ymin are restored to their default values.

To clear the logger diagrams

- In the **Logger 32** window, in the **File** menu, click **Clear log**.



Note

- You can also clear the diagrams by clicking the **Clear**  button.

15.10 Leaving the Logger Tool

When you leave the Logger tool, all logged data and the configuration of diagram windows are lost.

To leave the Logger tool

- In the **Logger 32** window, in the **File** menu, click **Exit**.

16 On-line Mode Functions

In the online mode of the TAC Menta programming tool, you can download and upload applications for TAC Xenta 280/300/401 devices.



Important

- The Online command is not available for the Xenta Server 700 devices.

You can also test an application by reading and writing public signals in an application, similar to the simulation mode, with the application running in a connected TAC Xenta device.



Tip

- When you view signals of particular interest, you can use the fact that signals recorded in the trend log are updated more frequently than the other signals.

You can use the online mode to change the allocation of I/O points by uploading the application from the controller, changing the allocation, and then downloading the application.

In the online mode, the TAC Xenta device is connected to the PC using a serial port.

16.1 Configuring the Serial Communication

When you work with a TAC Xenta 280/300/401 device in online mode, connected to the PC using serial communication, you have to make sure that the communication settings are correct.



Note

- You can only connect Menta to a Xenta Server 700 via a network and using TCP/IP.

You configure communication using the TAC Menta Setup program module. Normally, you only select the required serial port (COM port).

Table 16.1:

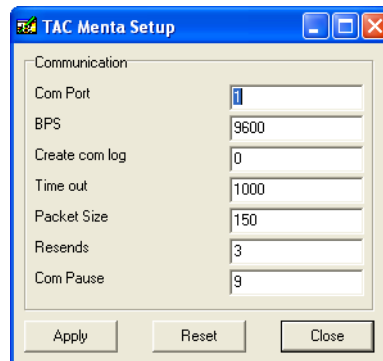
| Com Port | Communication port number on the PC. |
|----------|--------------------------------------|
|----------|--------------------------------------|

Table 16.1: (Contd.)

| | |
|----------------|--|
| BPS | Transfer rate for the Communication port in Bits per second. Must always be 9600. |
| Create Com Log | Debug communication flag (0=log disabled, 1=log enabled). If ComLog=1, all communication to and from Menta is logged in a file named pl_com1.txt. This file is in C:\Windows\Temp. Ensure that this directory exists on your hard disk). Note: This option is for internal debugging purposes only. The size of the log file may become very large. |
| Time Out | Communication parameter. Normally not altered. |
| Packet Size | Communication parameter. Normally not altered. |
| Resends | Communication parameter. Normally not altered. |
| Com Pause | Communication parameter. Normally not altered. |

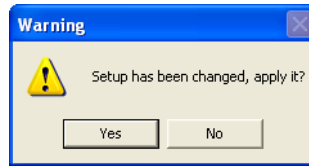
To configure the serial communication

- 1 On the Windows taskbar, click **Start**.
- 2 Point to **All Programs**, point to **TAC**, point to **TAC Tools**, and then click **Menta Setup**.
- 3 In the **TAC Menta Setup** dialog box, in the **Com Port** box, enter the required port number.



- 4 Click **Close** when finished.

- In the **Warning** dialog, click **Yes** if the Com port number is changed.



16.2 Entering the Online Operation Mode

You start the online operation mode in TAC Menta from the simulation mode.



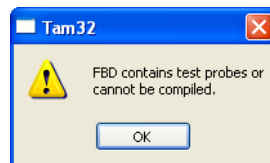
Important

- The **Online** command is not available for the Xenta Server 700 devices.

To use the online operation mode in TAC Menta, the TAC Xenta device must be connected to a serial port in the PC. If the communication fails, the command will be cancelled and an error message is displayed.

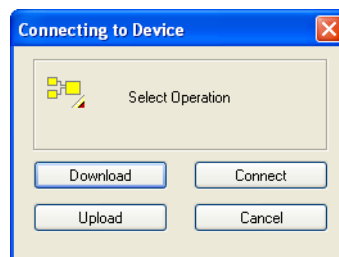


You can not enter the online mode if there are test probe blocks in the function block diagram. If the program contains test probes, the command is cancelled and an error message is displayed.



To enter the online operation mode

- In the Menta simulation mode, in the **Options** menu, click **Online**.



**Note**

- In the simulation mode of TAC Menta, you can also enter the online operation mode using two alternative ways:
 - by clicking the **Online** button.
 - by clicking the function key **F11**.

16.3 Optimizing the Refresh of Signals

You can optimize how signals are refreshed when working online in TAC Menta.

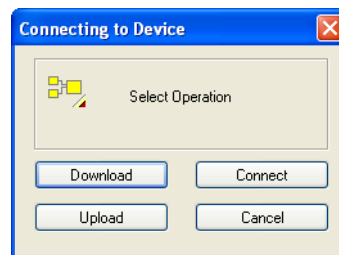
To optimize the refresh of signals

- In the **Simulation** mode, in the **Preferences** menu, click **Optimise Signal Update**.

16.4 The Connected Mode Options

When the connection to the TAC Xenta device is established, you can choose between the following options:

- Download
- Upload
- Connect
- Cancel



The outcome of each option depends on whether there is an application in TAC Menta or in the connected TAC Xenta device.

16.4.1 Downloading to a Xenta Device

You can use the download command to transfer an application from TAC Menta to a connected TAC Xenta device.

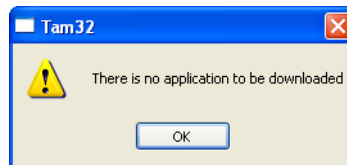


Important

- The Download command in online mode is not available for the Xenta Server 700 devices.

A download will be cancelled and a message box is opened in the following cases:

- There is no function block diagram in TAC Menta.



- A standard application is to be loaded into a custom device
- A custom application is to be loaded into a standard device

If no errors are detected in the function block diagram, the values in the application in TAC Menta are reset to their initial state and the code to download, the .COD file, is generated.

If errors are detected in the application, message boxes showing each error condition open and the download is cancelled.

One component in the downloaded data is the information about the network and its neighbourhood for the TAC Xenta device. This data is retrieved from and included in the download in one of the following two ways:

- retrieved from the connected TAC Xenta device
- retrieved from the TAC Vista database

Retrieving from the connected TAC Xenta device

When you use TAC Menta, connected to the serial interface port, to download an application, the download process begins with retrieving the network neighbourhood information from the connected TAC Xenta device.

The retrieved information contains the network addresses and other properties for the TAC Xenta device and its I/O modules. The network neighbourhood, also defines what group of controllers it belongs to, the master and the names of the other controllers in the group and the names of the other groups in the network.

The network information is then downloaded together with the application.

Retrieving from the TAC Vista database

When you download an application using TAC Vista and the Lon Network, the network neighbourhood information is collected from the TAC Vista database.

Before downloading to the Xenta device, the .COD file saved on the hard disk (in TAC Vista) is updated with all changes made in online mode (from the Vista database).

The text for an operators panel (OP) menu tree is an optional component in the downloaded data. This menu tree is included in the download the following way.

Unless the **Automatic Generation of Menu Tree** option, in the **Preferences** menu is selected, the user will be asked if new files for an optional OP menu shall be generated.



Important

- When you generate and the **Automatic Generation of Menu Tree** option, in the **Preferences** menu, a previous customized menu tree is destroyed.

The structure of the application (.OPE file) and the specification file for public signals (.ESP file) are used to generate the OP menu file (.BIN file) The “Name” and “Abbreviation” from the Program specification are used when generating the files.

Finally, three other files are downloaded to the TAC Xenta device:

The .COD file which is the code for the application

The CHR file, that is the file describing national characters

The .BIN file that is a binary data file

For more information on the different files, see Chapter 25.1, “Data Files”.

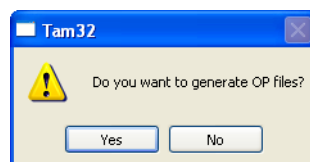


Note

- You define the version (1 or 3) for the downloadable code files in the **System Version** setting in the **Device Specification** command on the **Options** menu.

To download to a TAC Xenta device

- 1 In the simulation mode of TAC Menta, click **Online**.
- 2 In the **Connecting to Device** dialog, click **Download**.



- 3 In the **TAM 32** dialog, click **Yes** if you want to generate files for an OP menu.

16.4.2 Uploading From the Xenta Device

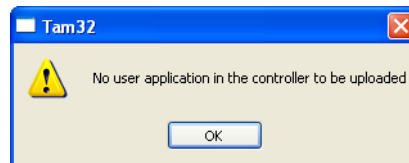
You can use the upload command to transfer an application from a connected TAC Xenta device to TAC Menta.



Important

- The Upload command in online mode is not available for the Xenta Server 700 devices.

The upload command is cancelled and a message box is opened if there is no application in the connected TAC Xenta device.



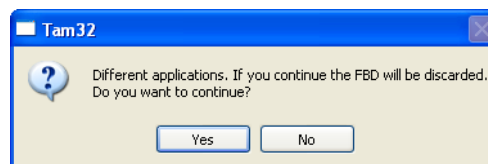
If the program IDs of the function block diagram in TAC Menta and the application in the connected device are the same, the application in the Xenta device is loaded into TAC Menta and you can view all signals in the function block diagram.



Tip

- You can use the fact that signals recorded in the trend log are updated more frequently than the other signals, when viewing signals of particular interest.

If the program IDs of the function block diagram in TAC Menta and the application in the connected device differs the existing function block diagram in TAC Menta will be discarded.



You are asked to enter a name and a location for saving the uploaded data from the TAC Xenta device in a file, the .COD file.

If there is no function block diagram in TAC Menta, you are asked to enter a name and a location for saving the file.



Note

- When the application is uploaded, you can view the file name and date for the application in the **Program Specification**. You can identify and find the corresponding function block diagram for the application.

Optional files, defining an OP menu in the TAC Xenta device, are also uploaded.

Finally, TAC Menta opens the uploaded file, the .COD file.

When an application is uploaded from a Xenta device without the matching function block diagram in TAC Menta, you can only work in the online mode using tabular mode. The graphical information in the function block diagram is not stored in the TAC Xenta device.

During an upload to TAC Menta the following files are loaded:

- the .COD file
- the .BIN file, optional
- the .CHR file, optional
- the .BPR file

For more information on the different files, see Chapter 25.1, “Data Files”.

To upload from a TAC Xenta device

- 1 In the simulation mode of TAC Menta, click **Online**.
- 2 In the **Connecting to Device** dialog, click **Upload**.

16.4.3 Connecting To the Xenta Device

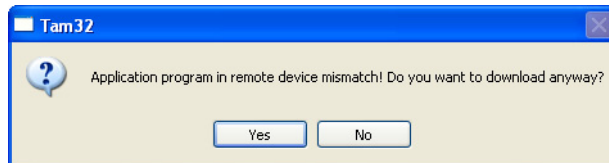
Use the Connect command when you have the same application in both TAC Menta and the connected TAC Xenta device.

If the program IDs of the current function block diagram in TAC Menta and the application in the device match, the TAC Xenta device is immediately connected. Otherwise a download will be performed before connecting.

A TAC Menta without an application represents an application with a differing program ID and TAC Menta fails when trying to download.

Connecting to a Xenta device when there is no application in TAC Menta must be preceded by an upload and the continued work can be done in tabular mode only.

If the applications in TAC Menta and the connected Xenta device differs, a message box opens where you are asked to confirm the download.



To connect to a TAC Xenta device

- In the **Connecting to Device** dialog, click **Connect**.

16.5 Execution Control in the Online Mode

In the online mode, you can execute the application in the TAC Xenta device in three ways:

- Continuous steps, with the cycle time specified in the **Program Specification**
- Single steps
- Multiple steps

For more information on controlling the execution, see Chapter 14.2, “Executing the Simulation”.



Important

- An application stopped in the online mode remains stopped until the stop is cleared or the Xenta device is restarted. Leaving the online mode does not return to normal execution in the Xenta device.

16.6 Overriding a Physical I/O Signal

For commissioning purposes, all I/O blocks except the CNT block can be forced to a required state or level when you work in the online mode.



Important

- Overriding I/O signals in online mode are not possible in the Xenta Server 700 devices.

The setting of an I/O signal to a forced value is intended for commissioning purposes, where you can use it via communication. The override function can be considered as the online equivalent to the manual setting of I/O input values during offline simulation.



Important

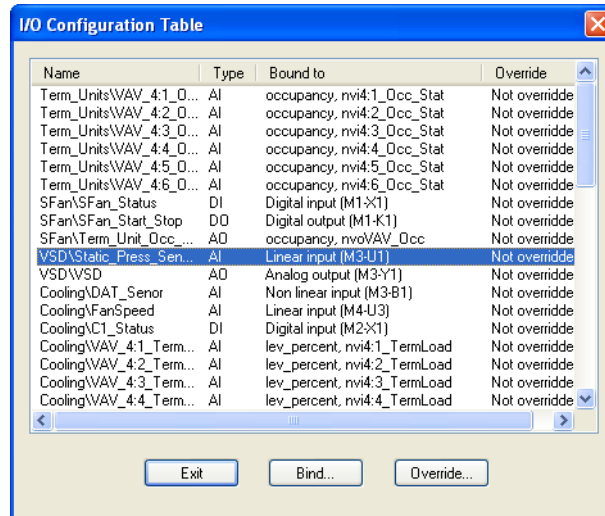
- A forced I/O signal remains in the forced state until the override is cleared or the Xenta device is restarted. Leaving the online mode does not clear the override function.

Table 16.2:

| | |
|----------------|---|
| Identifier | Name of the selected physical I/O signal. |
| Override | Check box for selecting the forced state. Default is unselected. |
| Override Value | Parameter of the type for the block output signal. A forced value directly affects the value of the I/O block. Any filtering of an analog I/O signal is not active. The override value is usually defined in the same engineering unit as the block output. The exception is an analog output with an override value given in % (percent). |

To override a physical I/O signal

- 1 In the simulation mode of TAC Menta, in the **Options** menu, click **I/O Configuration Table**.

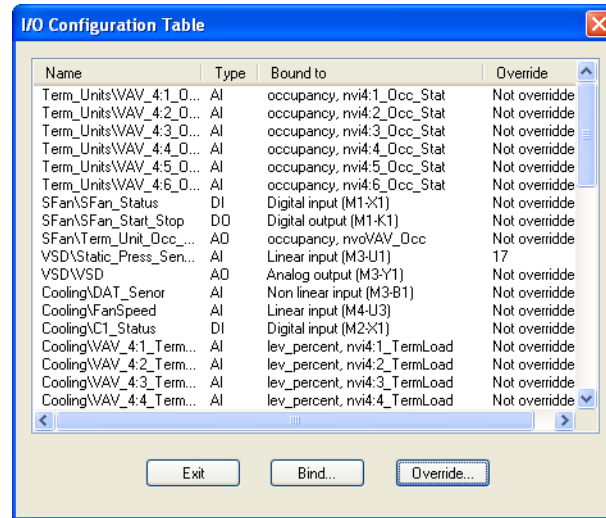


- 2 In the **I/O Configuration Table** dialog box, select the required I/O.
- 3 Click the **Override** button.



- 4 In the **Override** dialog box, select the **Override** check box.
- 5 In the **Override Value** box, enter the required override value.
- 6 Click **OK**.

You can view the override status for the I/O signals in the **Override** column.



7 Click **Exit**, to leave the I/O Configuration dialog.

16.7 Modifying Parameter Value Blocks in the Online Mode

In the online mode, you can also modify the output state for the parameter value blocks PVB, PVI, and PVR.



Important

- The ability to modify parameter value blocks in online mode is not available for the Xenta Server 700 devices.

A modification of values in parameter value blocks, made in the online mode, differs depending on whether the **Backup** option for the function block is selected or not.

If the **Backup** check box for the function block is selected, the output of the parameter value block will be modified and the new value is copied to the **InitValue** parameter. This parameter is stored in the non-volatile memory. The changed **InitValue** will then be included in an uploaded application file, the .COD file.

If the **Backup** check box for the function block is cleared, the modified output value of the parameter value block will not be copied to the **InitValue** parameter. The block output will be reset to **InitValue** after a restart. Modified block output values are not included in an uploaded application file, the .COD file.

16.8 Modifying Public Constants in Online Mode

You can modify public constants in the online mode of TAC Menta.



Important

- Modifying public constants in online mode is not available for the Xenta Server 700 devices.

When a value for a constant in the TAC Xenta device is modified from an operator's panel (OP), the changed value will not be included in the TAC Menta application, until the application is uploaded.

16.8.1 Turning Off the Updating of Public Constants

When there are many public constants in an application, you can turn off the dynamic updating of these values to increase the updating speed for signals in the display.

To turn off the online updating of public constants

- In the online mode, click the **Skip Const** button.

16.9 Modifying Binding Parameters in Online Mode

You can modify binding parameters for physical I/O blocks in the online mode of TAC Menta.



Important

- Modifying binding parameters in online mode is not available for the Xenta Server 700 devices.

When you modify a binding parameter for a physical I/O block, the AI, AO, CNT, DI, DO, and DOPU blocks, in online mode of TAC Menta, the new binding parameter value will not be transferred to the TAC Xenta device until a download is done.

17 Memory Usage

When you design an application in TAC Menta, you can view the calculated memory usage for the application.



Important

- The Memory usage command is not available for the Xenta Server 700 devices.

You can also view an estimation of whether the application will fit into the specified TAC Xenta device or not.

There are also facilities for:

- finding the number of unused network variables in the device
- estimating the size of an unknown network neighbourhood file downloaded to the TAC Xenta device.

17.1 Viewing the Calculated Memory Usage

The memory usage values are calculated for a TAC Xenta device with a system version 3, and may differ slightly from the true memory usage in the TAC Xenta.

The memory usage dialog is accessed in the Edit mode of TAC Menta.



Notes

- To get the best estimation, you can enter the simulation mode and then click the **Generate** command in the **Commands** menu.
- Sometimes you can start a download of an application that is estimated to fit into the TAC Xenta device, but the application is does not to fit. A message box is opened, warning for a parser code error. In this case, the network configuration in the device will be lost and needs to be downloaded again.

There are additional limitations in the TAC Xenta device, which may stop a download. For more information on the error codes, see Chapter 26.6, “Download”.

The available general storage depends on the type and system version of the TAC Xenta device.

The four memory areas in the TAC Xenta 401 are:

- Application
- Application files
- Parameters
- Work Area

Memory Usage in Target

Estimation

Application will fit in TAC Xenta
Note: To get the best estimate generation must be done

| | Application: | Application Files: | Parameters: | Work Area: | Number of Objects: |
|-------------|--------------|--------------------|-------------|------------|--------------------|
| Total Used: | 14656 | 37848 | 33629 | 3506 | 459 |
| Available: | 57344 | 240144 | 240144 | 14000 | 5500 |
| Free: | 42688 | 202296 | 206515 | 10494 | 5041 |

Application Files Details

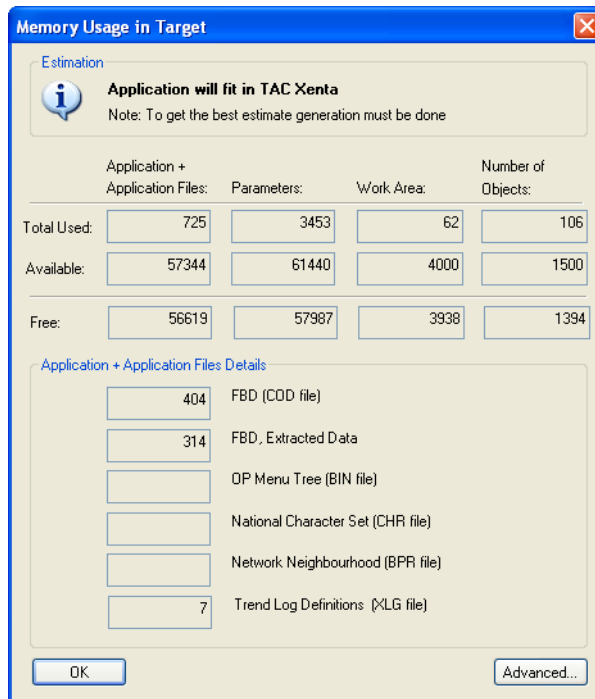
| | |
|-------|-----------------------------------|
| 21449 | FBD (COD file) |
| 14526 | OP Menu Tree (BIN file) |
| 17 | National Character Set (CHR file) |
| 2 | Network Neighbourhood (BPR file) |
| 1854 | Trend Log Definitions (XLG file) |

OK Advanced...

In the TAC Xenta 300, both the application files and the application are stored in one memory area resulting in the three areas:

- Application and the Application files
- Parameters

- Work Area



The dialogs show the total memory used, available memory, and the free memory for each memory area in the TAC Xenta device.

Table 17.1:

| | |
|-------------------|---|
| Application files | The downloaded files (*.COD, *.BIN, *.CHR and *.BPR). |
| Application | Data extracted (parsed) from these files. For example, tables for public signals, time schedules and alarm texts. |
| Parameters | The parameters used by the function blocks. For example, initial values in PV blocks and coordinates in CURVE blocks. |
| Work area | Data created when executing the application program. For example, output status and time schedules. |
| Number of objects | The function blocks and definitions of TAC Menta in the TAC Xenta are transformed into objects. |

The sizes of each file to download, the application files, are shown in detail.

For more information on the file types, see Chapter 25.1, “Data Files”

If the amount of free memory in the areas is too small, you may have to reduce the application size:

- The usage of the application memory area can be reduced by reducing the size of the application program files. In particular, the OP menu, the expression blocks, and operators (are compiled as expression blocks) use a lot of memory.
- The memory space for the free Work area can be increased by reducing the number of week and holiday charts.

To view the calculated memory usage

- 1 In the **Edit** mode of TAC Menta, in the **Options** menu, click **Memory Usage**.
- 2 In the **Memory Usage in Target** dialog box, click **OK**.

17.1.1 Viewing the Advanced Memory Usage

Each type of TAC Xenta device allows a fixed number of Network variables (TACNV) and SNVTs. The number of defined and free variables in the current application are displayed in a dialog.

If there is a risk of running out of application memory, an estimated size of an unknown network neighborhood file (.BPR) can be calculated after entering some data:

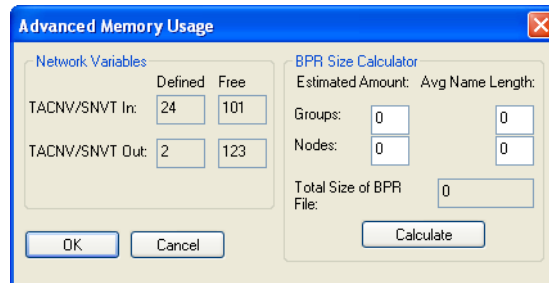
- the estimated number of TAC groups
- The average number of TAC Xenta devices (nodes) in each group
- the average number of characters in the Group and Node names.

Table 17.2:

| | |
|---------------------|--|
| TACN/SNVT in | The number of defined and free input network variables in the current application. |
| TACN/SNVT out | The number of defined and free Output network variables in the current application. |
| BPR size calculator | <p>A calculator for estimating the size of the network neighborhood file (.BPR) by entering:</p> <ul style="list-style-type: none"> • the number of TAC groups. • the number of TAC Xenta devices (Nodes) in each group. • average number of characters in the Group and Node names. <p>After clicking the Calculate button, the calculated .BPR size is displayed in the Total Size of BPR File box.</p> |

To view the advanced memory usage

- 1 In the **Edit** mode of TAC Menta, in the **Options** menu, click **Memory Usage**.
- 2 In the **Memory Usage in Target** dialog box, click the **Advanced** button.



- 3 In the **Advanced Memory Usage** dialog box, click **OK**.

18 Printing the Application Program Documentation

You can print the application designed in TAC Menta. A few settings for the printout are defined in TAC Menta and you select the details of the application to print in pre-formatted lists.

18.1 Setting up the TAC Menta Printout

Before you print an application designed in TAC Menta you can adjust some settings.

When the function block diagram is printed, the paper orientation for the pages is defined in settings in TAC Menta. Other printer settings are in accordance with the settings for the selected Microsoft® Windows printer.

To bring about the required overview or detail in the printed function block diagram, you can select the size of the graphical symbol for the block symbols.

You can define the number of horizontal and vertical pages including suitable margins.

You can customize the header for the printed pages.

Table 18.1:

| | |
|------------------|---|
| Variables | Select an item in the list for optional use in a second line of the page header. The available list items are: #abbrev , #author , #date , and #name (originating from the program specification) #prdate and #prtime (printout time and date) #tab (tab character) |
| Margins | |
| Left | In this box you can enter a value, in millimeter, for a left margin. |
| Top | In this box you can enter a value, in millimeter, for a top margin. |

Table 18.1: (Contd.)

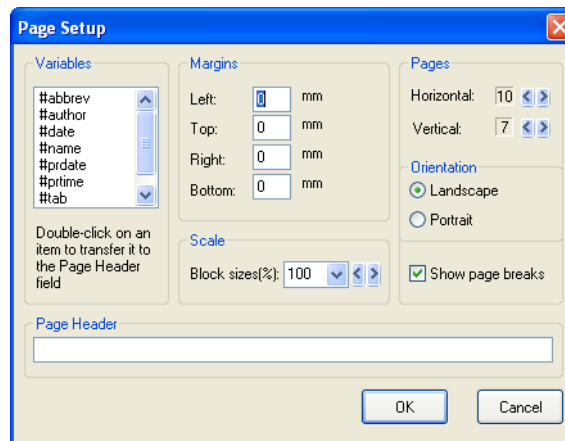
| | |
|------------------------|--|
| Right | In this box you can enter a value, in millimeter, for a right margin. |
| Bottom | In this box you can enter a value, in millimeter, for a left margin. |
| Scale | |
| Block sizes (%) | Use this box to enter a value for the size of blocks in a printout. Smaller block sizes can be used for better overview. Larger block sizes to make the printed pages easier to read. The range for the Block sizes is 25 to 500%. Default value is 100%. |
| Pages | |
| Horizontal | Use this box to choose the number of horizontal pages used to print the function block diagram. You can not set the number of pages lesser than required to display the complete function block diagram when maximally zoomed out. Changing the Block size percentage reduces the minimum number of pages. |
| Vertical | Use this box to choose the number of vertical pages used to print the function block diagram. You can not set the number of pages lesser than required to display the complete function block diagram when maximally zoomed out. Changing the Block size percentage reduces the minimum number of pages. |
| Orientation | |
| Landscape | Select this option to print the pages in landscape orientation. Note: selecting this option also influences the orientation of page breaks in the diagram window. |
| Portrait | Select this option to print the pages in portrait orientation. Note: selecting this option also influences the orientation of page breaks in the diagram window. |

Table 18.1: (Contd.)

| | |
|-------------------------|--|
| Show Page breaks | Select this option to show the page breaks in the diagram window. |
| Page Header | <p>You can use the Page Header box to create a second line in the page header. You can enter either free text or add items from the Vari-ables list.</p> <p>The first line in the page header always contains the name of the application.</p> |

To set up the TAC Menta printout

- 1 In the **Edit** mode of TAC Menta, in the **Preferences** menu, click **Page Setup**.



- 2 In the **Page Setup** dialog box, select the required options.
- 3 Click **OK**.

18.2 Using Boundary Ties

A useful facility when preparing a printout of the function block diagram on multiple pages is to add cross reference boxes to both ends of connections that crosses pages. These boundary ties are automatically created but can be manually moved and deleted.



Important

- All manually moved and deleted cross reference boxes are restored when you add boundary ties.

The boundary ties indicate where the signal continues in a three positions, separated by a dot, format:

- First position:
 - page number, vertically continued.
- Second position:
 - page number, horizontally continued.
- Third position:
 - signal number on page.

To use boundary ties

- In the **Edit** mode of TAC Menta, in the **Edit** menu, click **Add Boundary Ties**.



Note

- You remove boundary ties by clicking **Remove Boundary Ties** in the **Edit** menu.

18.3 Printing the TAC Menta documentation

When you document the application in TAC Menta, you can select what information to print. You can, for example, select to list the specified devices, the I/O resources, public signals, constants and their usage, alarm texts, and more.

Some of the lists can be printed with varied details.

An optional, separate text file, containing information about the application can be added to the printout.

Table 18.2:

| |
|--------|
| Tables |
|--------|

Table 18.2: (Contd.)

| | |
|-----------------------|--|
| Devices | <p>Select this check box to print a list of the specified device and I/O modules for the application.</p> <p>The listed properties are:</p> <ul style="list-style-type: none"> Unit Name Device Type. |
| I/O Resources | <p>Select this check box to print a listing of the number of used I/O points in application.</p> <p>The listed properties are:</p> <ul style="list-style-type: none"> DIs (digital inputs) AIs (analog inputs) DOs (digital outputs) AOs (analog outputs). |
| Public Signals | <p>Select this check box to print a list of all public signals in the application.</p> <p>The listed properties are:</p> <ul style="list-style-type: none"> Name Type Acc (access class) Unit Description. |
| I/O Lists | <p>Select this check box to print a list of all I/O points in the application.</p> <p>The listed properties for I/O points are:</p> <ul style="list-style-type: none"> Module number Terminal Ref. Name Bound to Parameter or Value Parameter or Value. <p>A complete listing of all I/O signals in the application is also included.</p> <p>The listed properties for I/O signals are:</p> <ul style="list-style-type: none"> Name Block Type Parameter or Value Parameter or Value. |
| Alarms | <p>Select this check box to include a listing of all alarms in the application.</p> <p>The listed properties are:</p> <ul style="list-style-type: none"> Alarm name Parameter name (OP alarm text) Alarm Text. |

Table 18.2: (Contd.)

| | |
|-------------------------|--|
| Constants | Select this check box to print a listing of all defined constants in the application. The listed properties are: Public Name Value. |
| Constant usage | Select this check box to print a listing of used constants in the application. The listed properties are: Constant Block where it is used. |
| Time Schedules | Select this check box to print a listing of the time schedules in the application. The listed properties are: Name W (max. number of weekly charts) H (max. number of holiday charts) Start date Stop Date Start Time Stop Time Weekdays. |
| Block Parameters | Select this check box to print a listing of all function block parameters in the application. The listed properties are: Type Name Parameter or Value Parameter or Value. Note: the content of the list depends on the selected All Blocks or Public Blocks options. |
| FBD | |
| FBD | Select this check box to print the function block diagram graphic. |
| Page Number | Select this check box to number the function block diagram. The listed properties are: Page number (Total number of pages). |
| Current Level | Select this check box to print only the current level of a function block diagram when hierarchical function blocks (HFBS) are used. |

Table 18.2: (Contd.)

| | |
|---------------------------------|--|
| All Levels | Select this check box to print all levels of a function block diagram when hierarchical function blocks (HFBs) are used. |
| Other | |
| OP Menu Tree | Select this check box to include an OP menu tree in the printout. |
| Overview | Select this check box to print the OP menu tree as an overview. |
| Detailed | Select this check box to print the OP menu tree detailed, including a fully expanded menu tree. |
| Text File | Select this check box to add an optional associated text file to the TAC Menta printout. |
| Options | |
| Scale Output to one page | Select this check box to print each level of the function block diagram on a single page. |
| Color Printout | Select this check box when printing on a color printer. |
| Skip Empty Pages | Select this check box to not print completely empty pages in a function block diagram. |
| Block Parameters From | |
| Public Blocks | Select this option to list parameters from public function blocks only. |
| All Blocks | Select this option to list parameters from all function blocks. |



Notes

- When you select several tables, each listed table is appended to the printout.
- If a group of blocks is selected when the **Print** command is executed, only the selected group will be printed.

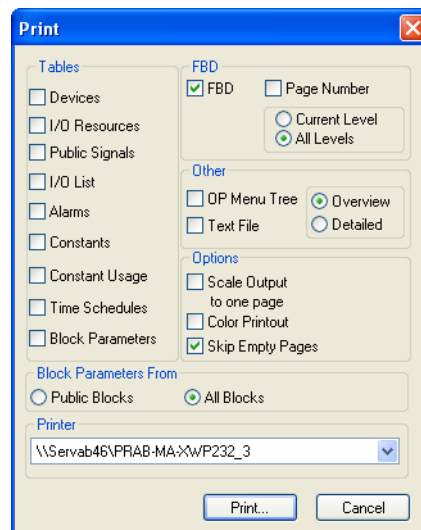


Tips

- An alternative way to document the I/Os in the application is to use the **Export I/O List** command in the **File** menu.
- When you export the I/O list, you save an ASCII text file (.TXT), listing following:
 - the device and I/O modules
 - the terminal references
 - the signal names

To print the TAC Menta documentation

- 1 In the **Edit** mode of TAC Menta, in the **File** menu, click **Print**.



- 2 In the **Print** dialog box, select the required option(s).
- 3 In the **Printer** list, select the required printer.
- 4 Click **Print** when ready.

19 The OP Configuration Tool

The OP configuration tool is accessed through the **OP configuration** command in the **Tools** menu.



Important

- The OP configuration tool is not available for the Xenta Server 700 devices.

When this command is selected, the specification (.ESP) file is automatically generated and the OP configuration tool is invoked with the .OPC file of the current application. If this file does not exist, a warning message will appear and the user will be prompted to select either the default file (DEFAULT.OPE) or an existing .OPE file to be used as the default tree. Clicking *OK* without selecting an .OPE file will invoke the tool with a completely empty menu tree.



Fig. 19.1:

The screen consists of four main areas:

Table 19.1:

| | |
|----------------------------|---|
| <p>Menu Structure Tree</p> | <p>Shows the complete menu structure of the created dialog. The operator can navigate in this tree by highlighting the different menu items. Depending on the selected item type, the three different parts on the right side of the screen will show different types of information.</p> |
|----------------------------|---|



Table 19.1: (Contd.)

| | |
|------------------------------|---|
| Change Menu Item Name | <p>The menu item name can be edited using the Change Name command button. For the root menu item (application), both the name and the abbreviation are shown. Neither the name nor the abbreviation can be changed, since they are defined in the Program specification in the FBD programming part of TAC Menta.</p> <p>The abbreviation is shown in all dialogs so that the operator can always see the device on the network that he is presently working with. The abbreviation is also used to indicate a summary alarm in the TAC Xenta. The text will flash to indicate that there are unacknowledged alarms in the alarm list.</p> |
| Operator Panel Display | <p>By locating the cursor in the display area, texts can be edited or added. Depending on the menu item type, different default texts will be copied to the display. The application abbreviation text will always be copied to the upper left corner of all displays. Also, the menu item name will automatically be copied to the top row of the screen display when a screen item is created.</p> <p>For some menu items, this area also includes a number of command buttons for inserting references and names. There are also command buttons for setting the properties of inserted signal references.</p> |
| Menu Item List (Signal List) | <p>This area will contain the list of menu items (sub menu) for the selected menu item, if that is a sub menu or the root item type. This area can also display a list of signals used when inserting references, names etc. into the OP display.</p> |

19.1 The Menu Structure Display









The different menu item types are represented by different symbols in the menu structure tree. The first two item types can be compared with folders in Microsoft® Windows Explorer:

Table 19.2:

| | |
|---|--|
|  | <i>Menu root item</i> , also the name of the application to be displayed in the TAC Xenta OP display when the network devices are shown. |
|  | <i>Sub menu item</i> , this is a menu item type that can contain a number of other menu items forming a sub menu. It is used to build the hierarchical levels of the menu structure. |



The screen items, can be compared to files in Microsoft® Windows Explorer, with the “file” contents shown in the OP display:

Table 19.3:

| | |
|---|---|
|  | <i>Status</i> , this type contains a number of pages/screens displaying traditional signals and parameters such as set point values and measured value. |
|  | <i>Alarm</i> , below this menu item the alarm list of the device is displayed. |
|  | <i>Access code</i> , this menu item will lead to a screen where an access code can be entered. |
|  | <i>Edit access code</i> , this menu item will lead to a screen where the access code can be edited. |
|  | <i>Date and time</i> , this menu item will lead to a screen where the date and time can be set. |
|  | <i>Daylight saving</i> , this menu item will lead to a screen where the daylight saving settings can be entered. |
|  | <i>Week chart</i> , this menu item contains a number of week charts related to one time schedule. |
|  | <i>Holiday chart</i> , this menu item contains a number of holiday charts related to one time schedule. |

19.2 Creating the Menu Structure

19.2.1 Adding Menu Items

If no previously saved menu structure is selected, the OP configuration tool starts displaying only a default menu root item in the menu structure display. This root item has a name that identifies the TAC Xenta device (application) used when the OP displays all the devices in the network. A menu structure is created by adding menu items to the root menu () and to sub menus (). You can add a menu item by selecting the root menu or a sub menu in the menu structure display and then selecting the **Add** button in the menu item list part of the screen. This will open a dialog where a name can be entered and a menu item type selected. Depending on the type selected, different default panel displays will be created. When there are menu items below a created sub menu item, this is indicated by a “+” sign in the sub menu item icon in the menu structure display. Double-clicking on this icon will also make the menu items in the sub menu appear in the menu structure display (see also *Tree – Expand* menu option).




Note

The OP menu tree can only contain a maximum of 255 OP screens.

To change the order in which menu items appear, these can be moved using the drag-and-drop technique in the menu item list window. To delete a menu item, select (highlight) it in the menu item list and then click the **Delete** button.


19.2.2 Sub Menu

The simplest menu item type is a sub menu (). It has only two properties; name and access level. A sub menu can have any type of menu item in its menu list, including a new sub menu. However, a maximum of 15 sub menu levels is allowed.

A sub menu can contain menu items with different access levels. When you increase the access level for a sub menu, the underlying sub menus may also change, resulting in the same or higher access levels for these sub menus.

When you decrease the access level for a sub menu, you may also choose to change the access levels for the underlying sub menus, resulting in the same or higher access levels for these sub menus.

19.2.3 Status

The menu item type *Status* () is used to create dialogs in which parameters and measured values are displayed and edited.

In the *Operator Panel display* window, the actual editing of the screen layout is done. Text strings and references to signals (parameters, measured values) can be inserted into a certain position indicated by the cursor by double clicking in the signal list. You can also click the *Insert of the Insert Module* (including the selected separator), *Insert Name*, *Insert Signal* and **Insert Unit** command buttons. These insert buttons will only work if a specification file has been loaded. With the cursor correctly positioned in the *Operator Panel display* window, a signal is selected in the list in the bottom right corner of the screen. Used signals in the signal list are shown in normal characters, while unused ones are shown in bold characters. If a signal reference is used in more than one place within the menu tree structure, this signal will be shown in normal characters.



Note

- The displayed list only contains signals belonging to the same Module, so you will first you have to select Module via the box above the signals list. Using the different “insert” buttons names, unit or signal value references can be inserted into the dialog.

The **Property** command button is used to invoke a dialog for the signal highlighted in the *Operator Panel display* window. The dialog contains the following settings:

Table 19.4:

| | |
|-----------------------------|---|
| Operator Panel Attributes | Used to define whether the signal should be read only or possible to read and write from the OP. |
| Signal Reference Properties | <p>This option only applies to real or integer values. The setting No. of decimals is used to decide the signal's display format. This number must be set between -4 and 4. A negative number means that the value will be altered in steps of 10 (No. of decimals = -1), 100, 1000 or 10000 (No. of decimals = -4) when altered by the operator.</p> <p>If a real value requires more digits than have been reserved, the following characters on the display row will be overwritten. If the value won't fit in the available space at the end of the row, it will be shown in scientific format (e.g. 14.1E6).</p> |

The settings *Minimum* and *Maximum* are used to set the limits, which must be within the range (-32768, 32767), for changes using the “+” or

“–” key in the OP. These limits are only in use when the operator alters the value via the OP.

Default values for *Minimum*, *Maximum* and *No. of decimals*, for which all new signal reference property dialogs, are set via the **Signal Properties** command in the **Formats** menu.

A highlighted signal reference in the *Operator Panel display* window can be deleted using the **Delete signal** button.


A *Status* menu item can consist of several pages. Use the **Add page** command button to create the additional pages with parameters/values.



Note



- If a *Status* menu item is deleted, the corresponding signals will not be indicated as unused (bold) in the signal list until the list is updated.

19.2.4 Alarm

When a menu item of the *Alarm type* () is created, a default screen is created, with references to the alarm list. Note that even if several instances of Alarm type can be created, they will all refer to the same alarm list. No other properties than name and access level should be edited for this menu item type. The alarm text displayed in the OP is defined in TAC Menta.



19.2.5 Access Code

When the root menu item is active in the left part of the screen, you can edit the access levels for the items in the menu item list. One or more menu items can be selected at a time. When you click the **Level** button a dialog to define the access level (*low(green)*, *medium(yellow)* or *high(red)*) will appear. The access level of each item in the menu structure display is indicated by a color code and a letter (L, M or H). A menu item of the type *Access Code* must always be defined with the access level low.



The Access Code () and *Edit Access Code* () menu item types are used to create the dialogs for changing the access level and editing the access code. For these types, the explanatory text in the display screen may be edited. This is done by placing the cursor (using the mouse) in the *Operator Panel display* window and overwriting the text to be changed.

In each TAC Xenta device, there will be default codes (*medium* = 1111 and *high* = 2222) that can be edited via the OP. The TAC Xenta will revert to the default codes whenever the application program is downloaded.

19.2.6 Date and Time/Daylight Saving

The *Date and Time* () and *Daylight Saving* () menu item types are used to create the dialogs for date and time settings and daylight saving settings. The default dialogs may be changed or extended by manually adding text in the *Operator Panel display* window.

19.2.7 Week/Holiday Chart

The *Week chart* () and *Holiday chart* () menu item types are used to create the dialogs for viewing and editing time schedules. A time schedule consists of a number of week charts and holiday charts. The time schedule has been divided into two dialogs (week and holiday chart) to simplify the operator interface in the OP. When the reference is assigned for the *Week and Holiday chart*, it is set to the same time schedule. The reference is assigned by selecting a time schedule object name in the signals list (cf. *Insert signal* in *Status* dialogs) and clicking the **Assign time sch.** button. Used signals in the signal list are shown in normal characters, and the unused ones are shown in bold characters. If a signal only is selected in a week or holiday chart, this will be shown in normal characters. The displayed list only contains signals belonging to the same Module, so you will first have to select Module via the box above the signals list.

In order to reduce the TAC Xenta's memory usage, you can choose to use Time schedule templates. These templates have a fixed layout for the Week and Holiday chart dialog boxes. Otherwise, the default dialogs may be changed or extended by manually adding text in the *Operator Panel display* window.

19.2.8 TAC Service Menu

The **TAC service** menu is a fixed menu. Through the OP, you can use this menu to read the TAC Xenta system program version, TAC Xenta name and network address, configure the I/O modules and restart the TAC Xenta.

The service menu has no menu item, since it is automatically created when generating the OP tree, if the OPDOCADD.GOP file is present in the TAC Menta directory (can be set in *Formats – Settings*).

The service menu, which is always located at the end of the menu tree, is only accessible via a fixed access code.

19.3 Editing an Existing Menu Structure Tree

When a menu structure is reused there may be a need to translate/change texts or rearrange menu items.

To reuse all or part of an OP menu tree file, save the file with a new name." Next, exit the OP configuration tool, and create a new file (.AUT/.MTA) with the same name (if it does not already exist)When you re-enter the OP configuration tool, you can edit the new menu tree file. All references to signals that still exist (with the same name) in the new file are left intact, other references will be empty and can be pasted in from the signal list of the new file. New signals which did not exist in the original signal list are available in the signal list of the revised file and can also be pasted in.

19.3.1 Moving Menu Items

In the "Menu item list", the different items can be sorted using drag-and-drop (select a menu item by pointing at it with the mouse, click the left mouse button and keep it depressed while you drag the menu item to the required position).It is also possible to move a menu item in the menu structure tree display from one sub menu node to another.

19.3.2 Changing the Operator Panel Display Layout

When you are translating or editing the texts in the OP display, you can drag and drop references to signals in the OP display.

19.3.3 Copying and Pasting

The Copy and paste functions can be used on all items in the menu root item, or under a sub menu node. The items are copied/pasted with all sub menus and signals. The copied items are stored on the clipboard – both as text and as graphics – and so can also be used for copying between two operator dialog description files.

19.4 Automatic Generation of an OP Menu Tree

The OP configuration tool contains a function for automatically generating a complete OP menu tree. The user can import all public signals from the FBD for standard format OP presentation, rather than manually insert every signal.

The function is invoked using a **Generate** or **Download** command in TAC Menta's Simulation mode if *Auto generation of menu tree* has been set in the **Preferences** menu. The menu tree will then be built from the empty standard OP menu tree file as specified (Standard OP tree = Path\FILENAME.OPE) in *Formats – Settings*. If no standard OP menu tree file has been specified, the STD_AUT.OPE file will be used. Auto-

matic generation can also be started from the OP configuration tool by selecting **Build** in the **Tree** menu.

The automatic generation function first erases the old data in the status menu items and then it does the following:

- All RO *binary* signals will be sorted in alphabetical order in the first available low access level *Status* menu item, with one signal per row. The signal name is placed in positions 1–14 (longer names are truncated), the signal reference is placed in positions 15–17 and the unit is placed in positions 18–20. New pages are automatically added, if necessary.
- All RO *analog* signals will be sorted in alphabetical order in the second low access level *Status* menu item, with one signal per row. Consequently, there must be at least two status menu items in the tree for the analog signals to be displayed. The signal name is placed in positions 1–9 (longer names are truncated), the signal reference (with default properties) is placed in positions 10–17 and the engineering unit is placed in positions 18–20. New pages are automatically added, if necessary.
- All *binary* signals (Both RO and R/W) will be sorted in alphabetical order in the first medium or high access level *Status* menu item. Layout handling as above.
- All *analog* signals (Both RO and R/W) will be sorted in alphabetical order in the second medium or high access level *Status* menu. Layout handling as above.
- The first available *Time schedule* will be assigned to the first *Week chart* menu item and the first *Holiday chart* menu item. The second Time schedule will be assigned to the second Week chart menu item and the second Holiday chart menu item, etc. If there are more Week/Holiday charts in the menu tree than Time schedules in the FBD, there will be an error message, and generation will be interrupted .

If the FBD contains two or more *Modules*, the Status menu item will automatically be replaced by a *Sub menu* item containing the corresponding number of Status menu items that have the same names as the Modules.

The OP menu tree often contains an information menu displaying the application type, software version etc. This type of menu is created as a *Status* menu item only containing text. To prevent the information menu from being overwritten by automatic generation, the information menu can be defined as an Information node (Name “Info” menu = MENU_NAME) in *Formats – Settings*. This menu will then be excluded when the program searches for status menus.

The generated OP menu tree can, of course, be manually edited in the OP configuration tool. Please note, however, that all manual alterations will be deleted if you auto generate the menu tree again. Therefore,

always try to prepare the standard OP menu tree file, from which the menu tree will be built, in such a way that no manual editing will be needed. For example, it is possible to change the standard of the signal features and/or to save them.

19.5 OP Description Files

One alternative way to create an OP menu tree is by using OP description files (.DOP). The user describes the OP tree in a text file, imports this file into the OP configuration tool, and the .OPC and the .BIN files are automatically created and saved to disk. The .DOP file can be edited by any Windows based text editor or word processor.

19.5.1 Data and Declarations Syntax in OP Description Files (.DOP)

The first position in a row is reserved for one of three key words:

Table 19.5:

| | |
|------|--|
| OPC | Full pathname\filename for the OP menu tree to be created, normally an .MTA file. The .BIN file will also be stored in the same place. |
| ESP | Full pathname\filename to the signals list, normally an .MTA file, that is used as the input file for the .DOP file. |
| MENU | Information about Menu name, Menu item type, access level and other attributes describing the layout of the specific OP display. |

Comments can be added by starting a row with the * character.

Each Menu item in the OP menu tree must be defined separately. To place a Menu item on the “root” level, enter a hyphen (-) character in front of the Menu name. Menu items (without a hyphen), defined after a “Sub menu” will be built under the “Sub menu” level in the OP tree. To place the Menu item under another “Sub menu” than the previous one, enter the full path name e.g. -PARAMETERS- HEATING_COIL-FROST_PROTECTION.

Key words and attributes are separated by blanks or tabs. Consequently, all attribute texts that contain blanks must be typed between single quotes, e.g. “Sub menu”. The following Menu item types may follow the MENU keyword:

Table 19.6:

| | |
|----------|--|
| Sub Menu | MENU 'MENU NAME' 'Sub menu' High A standard layout display will be created. |
|----------|--|

Table 19.6: (Contd.)

| | |
|---------------|---|
| Week Chart | MENU 'MENU NAME' 'Week chart' Low AHU01\TC_SF01 A standard layout display will be created, but note that a reference to a specific time schedule (e.g. AHU01\TC_SF01) must be included. |
| Holiday Chart | MENU 'MENU NAME' 'Holiday chart' Medium AHU01\TC_SF01 A standard layout display will be created, but note that a reference to a specific time schedule (e.g. AHU01\TC_SF01) must be included. |
| Status | MENU 'MENU NAME' Status High 'Text' 'T11 ~##.#°C' T11 0 100 1 RO |

Row by row, the OP display texts are entered between single quotes, exactly as they will appear on the screen. The texts are truncated if they are longer than 20 characters. An empty line can be defined as a space between double quotes (" "). Do not use tabs. New pages are automatically created, when necessary.

Digital signals are defined with a “~#” in the text, followed by two attributes: the signal reference (including module) and the read/write attribute (RO/RW), all written on one line.

Analog signals are defined with a “~#.#” in the text, followed by five attributes: the signal reference (including module), the *Minimum* setting, the *Maximum* setting, the *No. of decimals* setting and the read/write attribute (RO/RW), all written on one line. To display more than one signal in an OP display row, separate the next signal reference with a tab/space and define the signal and its attributes. Max. 10 signals can be defined in one row, and all the data for one specific row of the display has to be on one row in the .DOP file.

Table 19.7:

| | | | | |
|------------------|------|-------------|--------------------|--------|
| Access Code | MENU | 'MENU NAME' | 'Access Code' | Low |
| Edit Access Code | MENU | 'MENU NAME' | 'Edit Access Code' | Medium |
| Alarm | MENU | 'MENU NAME' | Alarm High | |
| Date and Time | MENU | 'MENU NAME' | 'Date and Time' | Low |
| Daylight Saving | MENU | 'MENU NAME' | 'Daylight Saving' | Medium |

A standard layout display will be created for all these menu items.

19.5.2 OP Description File Example

```

*****
*      TAC XENTA AHU11      FILE TYPE: TAC XENTA OP
*      PROJECT:            Hospital west
*      AUTHOR:             U. N. Owen
*      DATE:               980401
*****
***** FILE NAME *****
*      Name
OPC    C:\PROGRAM\TAC\PROJECT\HOSPITAL_WEST\AHU11\AHU11
ESP    C:\PROGRAM\TAC\PROJECT\HOSPITAL_WEST\AHU11\AHU11
***** OP INFORMATION *****
*      Name                Type                Access Level
*      'OP text row       ' Signal ref          Min  Max  Dec.  RO/RW
MENU  -INFO                Status                Low
      'Appl: 123456-789'
      'Version: 1.0B'
MENU  '-PLANT STATUS'      Status                Low
      'T11 MV ~###.#"C ' AHU11\T11_MV          0   100  1   RO
      'T11 SP ~###.#"C ' AHU11\T11_SP          0   50   1   RO
      'Outdoor ~###.#"C ' T31_MV                -50  50   1   RO
      'Heating pump '
      'Ind./Man. ~# / ~# ' AHU11\P11_Op  RO AHU11\P11_M  RO
      'Run time ~##### h' AHU11\P11_RT          0   32500 1   RO
MENU  -ALARM                Alarm                Low
MENU  '-ACCESS CODE'       'Access Code'        Low
MENU  '-TIME PROGRAM'     'Sub menu'           Medium
MENU  'Week program'      'Week chart'         Medium
      AHU11\TC_SF11
MENU  'Holiday program'   'Holiday chart'      Medium
      AHU11\TC_SF11
MENU  -DATE/TIME           'Date and Time'      Medium
MENU  '-CONTROL PAR'      'Sub menu'           Medium
MENU  Room                 Status                Medium
      'Gain Room ~###"C ' AHU11\ROOM_G          0   99   0   RW
      'Itime Room ~### Min' AHU11\ROOM_ITIME      0   99   0   RW
MENU  Heating              Status                Medium
      'Heat PBand ~###"C ' AHU11\HEAT_PBAND      0   99   0   RW
      'Heat Itime ~### Min' AHU11\HEAT_ITIME      0   99   0   RW
MENU  '-CHANGE CODE'     'Edit Access Code'   Medium
***** END *****

```

Fig. 19.2:

19.5.3 Importing the OP Description File

There are two alternative ways to import the .DOP file:

- Use *File – Import* in the OP configuration tool. If there is a menu tree already, the imported tree will automatically be placed after the existing one. Normally, a “Save?” question will be displayed when exiting the tool. If this question is not displayed the import has failed.
- Use the **Run** command (or a shortcut) in the Windows start menu. Syntax: *Run pathname\GraphOP.exe /d pathname\filename.dop*

Note that the **Run** command uses DOS; so the *pathname\filename* must be entered between quotes if it contains a space or more than eight characters.

The program will create a new .BIN file, but no .OPC file. If there is already an .OPC file with the same name, this will not be altered. To avoid mistakes, you should either move the old .OPC file or start the OP configuration tool and save the correct .OPC file.

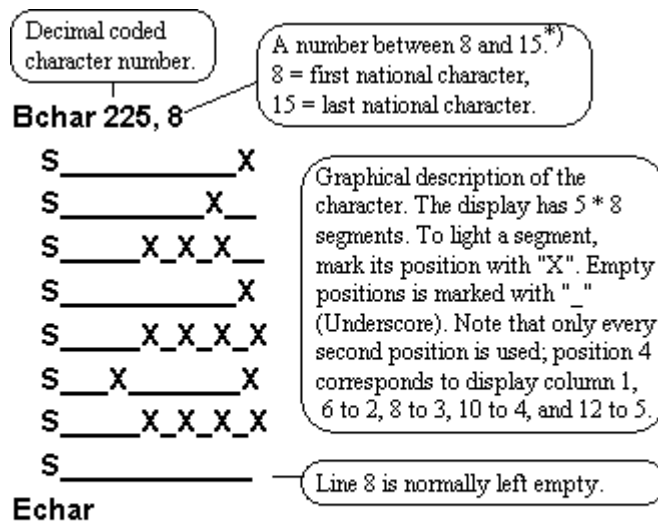
19.5.4 Exporting the OP Description File

To export an open menu tree to a .DOP file, use *File – Export* in the OP configuration tool. When exporting to a .DOP-file, the attributes belonging to the keyword will be in English, e.g. MENU Week chart 'Week chart' Medium. However, if your TAC Menta program is not in English, the attributes will not have to be edited, since the program handles both English and your national language when importing.

19.6 Defining a Character Set File

The OP's LCD display shows decimal coded characters 32-125 in accordance with the standard ASCII set. In addition, 8 characters with codes higher than 127 can be defined in a national character set file.

To define a new character set file, start by saving the existing file CHARSET under a new name in Windows™ Explorer. Then edit the new character set file by means of the Notepad editor, for instance. The following format is used in CHARSET:



*) Note that national character number 14 and 15 are reserved for special "AM"/"PM" characters, when using the AM/PM Time Format.

Fig. 19.3:

A simple way to find the decimal coded character number, is to start the Character mapping program (CHARMAP.EXE) in Windows™ Explorer (normally located in the Accessories program group), and look for the character in the Arial True Type mapping, for instance. Here all characters are shown in order, starting with the "space" (decimal code 32) in the upper left corner.

Finally, open the *Formats – Character set file* dialog in the OP configuration tool, and select the new character set file. From now on, the new characters will be used when the **Generate** command is executed.

19.7 Menu Options

19.7.1 Menu Bar

On the menu bar, there are a number of menus with various entries:

FILE

Table 19.8:

| | |
|---------------|---|
| New | Closes the present OP menu tree file and opens a new file. If the present file has not been saved, the user will be asked if he/she wants to save the existing file. |
| Open | Loads an OP menu tree file from the disk. The files on the disk must have the extension .OPC (menu tree + specification list) or OPE. (Only menu tree). The present OP dialog will close before the new file opens. If the present file has not been saved, the user will be asked if he/she wants to save the existing file. |
| Save | Saves the OP dialog with the same name as. |
| Save As | Saves the present OP dialog but allows the user to enter a new file name. It is possible to save it either as an .OPC file or an .OPE file. To save as an .OPE file, the loaded signal list must first be deleted (Tree – Remove Specification). |
| Print | Prints an overview or a detailed (a fully expanded menu tree) documentation of the OP dialog on the active printer. |
| Print Preview | Displays a preview of the printout on the screen. |
| Print Setup | Used to select the active printer and change its settings. |
| Import | Used to import an OP menu tree from an OP description (.DOP) file. |
| Export | Used to export an OP menu tree to an OP description (.DOP) file. |
| Exit | Leaves the program and returns to the program manager, or to TAC Menta if it was started from there. If the present file has not been saved, the user will be asked if he/she wants to save before this. |

FORMATS

Table 19.9:

| | |
|-------------------------|--|
| Signal Properties | Sets the default values for Minimum, Maximum and No. of decimals, for use in all new signal reference properties dialogs. |
| Alarms | A dialog where the texts indicating the alarm status in the alarm list can be edited. Abbreviated localized texts for “tripped”, “reset” and “acknowledged” can be defined. |
| Date and Time | A dialog where the date and time display format can be set. The selected format is used in alarms, time schedule displays etc. National texts for weekday names are also edited in this dialog. |
| National Week Days Text | In this dialog, the national texts for weekdays are defined. These text are used in the date and time display on the operator panel. |
| National Months Text | In this dialog, the national texts for months are defined. These text are used for displaying some of the date formats. Only the first three characters in each month name are used in the supported date formats. |
| Character Set File | When using national characters outside the standard ASCII set (0-127), a character set file including definitions of the national character set mapping must be used. The LCD display of the OP supports eight user defined characters, defined in the national character set file. The file is used by the tool when the Generate command is executed. The tool will then check the defined texts for ASCII codes higher than 127, and replace them with characters defined in the character set file. This will enable the OP to display the correct national characters. |

Table 19.9: (Contd.)

| | |
|----------|---|
| Settings | <p>Used for setup of:</p> <ul style="list-style-type: none"> • <i>Temporary directory</i>; defines the directory for all temporary menu tree files, created and used during the <i>Generate</i> process. • <i>Separator</i>; one character, e. g. “-” or “/”, to be used to separate module and signal names. • <i>Use Time Channel Templates</i>; defines whether Time Schedule Templates are to be used. • <i>Use Service menu</i>; defines whether the TAC Service menu will automatically be included when generating the OP menu tree. • <i>Standard OP tree</i>; path and name of the empty OP menu tree file, from which the OP menu tree will be built upon an automatic generation of the menu tree. • <i>Name “Info” menu</i>; defines the entered Status menu name as an Information node, which means that it will be prevented from being overwritten upon automatic generation of the menu tree. • <i>OP</i>; select <i>Custom</i> if an OP using Russian character set is to be used, otherwise <i>Default</i>. |
|----------|---|



Note

If you want to permanently use other default texts or values for *Signal properties*, *Alarms*, *Date and time*, *National Weekdays* and *National Months* than those stored in the OP configuration tool, you will be able to save these altered texts and values via *Save as* an empty menu tree file (.OPE). You use this file as a template file whenever you are creating a new menu tree.

TREE

Table 19.10:

| | |
|--------------------|---|
| Expand | To display all levels of the menu tree. This is equivalent to double clicking on all menu tree icons with a “+” sign. |
| Test | <p>Will perform a form of pre-compilation check to ensure that the defined OP dialog can be successfully downloaded. This test will check for the following errors:</p> <ul style="list-style-type: none"> • Menu items not arranged by increasing access level. • Specification file including signal list not loaded. • National character set not selected. • Root menu item empty. • Time schedule reference in week chart display not defined. • Time schedule reference in holiday chart display not defined. <p>This test will also check for the following warnings:</p> <ul style="list-style-type: none"> • All signals in specification file not used in OP dialogs. • Time schedule in specification file not used in OP dialogs. |
| Generate | First, the complete OP dialog is tested (see above). Then, if no errors are found, pressing <i>OK</i> will compile the menu structure to a file that can be downloaded together with the application from TAC Menta. The result of the compilation is presented dynamically in a window on the screen. Close this window before continuing your work. |
| Build | Starts automatic generation of an OP menu tree. |
| Load Specification | Used to load a new signal specification list. |

Table 19.10:

| | |
|----------------------|--|
| Update Specification | Will update the active signal specification list. Used when working with FBD programming and the OP configuration tool simultaneously. However, a new specification file (.ESP) version must first be created via the Simulation mode command Generate before the specification list can be updated. |
| Remove Specification | Will delete the loaded signal list from memory. |
| View Specification | Used to display the signal specification list |
| Copy to clipboard | The menu structure tree is copied to the clipboard in Microsoft® Windows meta file format and text format. This may in turn be exported from the clipboard to other Microsoft® Windows applications. It can be pasted as text or a picture in, for instance, Microsoft® Word. |

HELP

Table 19.11:

| | |
|----------|--|
| Contents | Will invoke a help file containing the basic information in this manual. |
| About | Will display information on the current software revision. |

20 The Download Wizard

The Download Wizard is a guided download procedure for upgrading the System program of one or several TAC Xenta 280/300/401 devices.



Important

- The Download Wizard is not applicable for the Xenta Server 700 devices.

Open the **Start** menu in Microsoft® Windows. Select TAC and TAC Tools and you will find the Download Wizard.



Notes

- For TAC Vista, the Download Wizard should only be used to replace system programs in TAC Xenta devices.
- Use of the Download Wizard for applications and network configuration only applies for TAC Vista 3.x and TAC Menta 3.x.
- The possibility to reuse/update a program module during a system program upgrade depends on the program version compatibility. Also, different extra preparations or actions may be required. For example, a TAC Xenta 300 using System program v 1.01 cannot be updated to version 1.13 (or later), unless both the application source code (.AUT) and the network configuration (.NWC) are available when the procedure is run. So, always read the enclosed upgrading information before installing a new program version.



Warning

- NEVER download a system program for a TAC Xenta 300 device into a TAC Xenta 280 device, as this will render the TAC Xenta 280 device useless.

20.1 How to Use the Download Wizard

20.1.1 The Dialog

The Download Wizard dialog consists of two pages, displayed via the **Next** and **Previous** buttons. Page two contains the following sections:

Table 20.1:

| | |
|----------------|--|
| System | First check the TAC Xenta Unit type . Select to <i>Retain current</i> System program, or to <i>Download new</i> . If downloading a new program, the name and path of the <i>System file</i> (.MOT) must be entered. |
| Application | Selects either Retain current Application program , or Download new . Selecting the Clear memory option results in a second dialog where two alternatives are offered. Pressing the Yes button results in clearing the memory totally and shall only be selected if there shall be <i>no application program at all</i> in the TAC Xenta device. This clearing causes problems when the device is used in some types of networks. Pressing the No button results in restoring the device to its factory contents. The No alternative is the recommended alternative. If downloading a new program, name and path to the Application file must be entered. |
| Configuration | Select to <i>Retain current</i> Network configuration, or to <i>Download new</i> . Tick <i>Check Neuron ID</i> if the Download Wizard is to look for missing and mismatched Neuron IDs in the database. If downloading a new configuration, the name and path of the <i>Database file</i> must be entered. |
| Connected unit | The current node name of the connected TAC Xenta device is displayed. Select the desired new node name (normally the same as <i>Connected unit</i>) from the <i>Node to download</i> list, which displays all the nodes in the selected <i>Database file</i> . |

The dialog also contains the following buttons:

Table 20.2:

| | |
|-------------|--|
| Information | Displays the Device type, Hardware version, System program version and Boot PROM version of the connected TAC Xenta device, and informs you if the selected Database file is marked as dirty (i. e. the network database contains data which has not been downloaded to the TAC Xenta device). |
| Apply | Starts the download procedure. The result of the download is shown in a download progress window. |
| Close | Closes the dialog. |

20.1.2 The General Download Procedure

If the download includes a new Application program, we recommend that you open the corresponding .AUT file and then immediately close it again. In this way, you will give the file a new date and avoid an optional question about generating a .BIN file from TAC Menta during the download procedure.

- 1 From the TAC Menta Group, start the “Download Wizard”.
- 2 Connect the RS232 cable between the communication port you have specified and the TAC Xenta controller.
- 3 Follow the “Wizard” and specify the software to download.
- 4 To have the Wizard compare the ID of the device with that of the database, click Check the Neuron ID in the Configuration area. If the ID is missing from the database, it will be added. If the IDs differ, you will be asked whether the database is to be updated.
- 5 When the appropriate boxes have been checked, click “Apply”.

The result is shown in a download progress window. If the message “Remote device does not reconnect!” appears, this can normally be ignored.



Warning

- The TAC Xenta System download phase is critical; no other activity is to take place at this time! If the procedure is interrupted, it may be impossible to reload the System software.

- 6 When the download has been acknowledged, the procedure can be repeated from step 2.

The accumulated results of all the downloads are listed in the file “dwiz.log”, normally residing in the TAC Menta program directory. The log contains information about

- start time and affected node
- performed operations with time stamp
- total time required
- result of operations
- delimiter
- at the end: the Neuron ID, if updated in the database.

20.2 TAC Menta v3 Compatibility

TAC Menta v4 is fully compatible with the TAC Xenta 280/300/3000/400/901 and TAC Vista v3. It is also possible to connect to and generate code for the TAC Xenta 280/300/400 v1.1 (Cf. the Device Specification dialog), provided that only functions compatible with TAC Menta v1.2 and TAC Xenta v1.1 are used.

20.2.1 .AUT Files

Earlier versions of TAC Menta application source code files may be opened and edited. When the old file is opened, it is automatically converted to the latest version. If the file is then saved using a manual **Save** command, the file is saved in the new format. You cannot save new files in older format(s).

Due to modifications to the print-out page size and page breaking functions in TAC Menta v3, the upgrade of an application from v1 may require manual adjustment of the graphical layout of the FBD.

20.2.2 .COD Files

After upgrading the TAC Xenta system program from v1.x to v3, the .COD file must be regenerated from the .AUT file before download, due to changes in the downloaded (.COD) file format between TAC Menta v1.2 and v3. Thus, the application source code must be available when a controller system program is upgraded.

When connected to a TAC Xenta v1.x device, TAC Menta will by default generate and download .COD files in v1.x format. Before communication starts, a special message is sent to the TAC Xenta device, the response to which decides what type of protocol to use.

When connected to a TAC Xenta v1.x device, it is also possible to upload the application. If, before the upload, you open the corresponding .AUT file, you may then regenerate the .COD file (with the uploaded parameters) in TAC Menta v3 format. *This will not, however, be possible if you do not have the .AUT file.* There is currently no way

of converting the .COD file from v1.x to v3 without having access to the .AUT file.

20.2.3 OP Menu Tree Files

V1.x OP menu tree (.BIN) files using the old, more space-consuming time schedule format will be accepted by TAC Xenta v3. However, if application memory space is a problem (which is often the case), the user is strongly recommended to regenerate the OP file to v3 format using the **Time Schedule Templates** option to save space.

20.3 Upgrading a TAC Xenta 300 Device to v3

When upgrading a TAC Xenta 300 device from system program v1.x to v3, the procedure below should be followed. All TAC Xenta 280/301/302/401s in the network must have the same System software version (v1 or v3).

20.3.1 Updating the .AUT File with Application Data (Optional)

- 1 Start TAC Menta and open the .AUT file for the application in the TAC Xenta device to be upgraded.
- 1 Upload the .COD file from the TAC Xenta device.
- 1 Save the application program file in Simulation mode. It will then automatically be converted to TAC Menta v3 format, and updated with the uploaded parameters.

These steps may be omitted if the .AUT file saved on disk has up-to-date parameters. This procedure will be necessary, if you believe that the parameter settings have been modified from the OP or from TAC Vista, and these settings must be kept after the upgrade.

20.3.2 Upgrading System and Application Programs

Either Alt. 1 or 2 below will apply, depending on the circumstances.

- 1 No Change in the Network Configuration
 - Start the Download Wizard, and select the following options: Download new system, Download new application, and Retain current configuration.
 - Select the appropriate system program file and application program file (select the file created in step 1 if applicable).
 - Click the **Apply** button to complete the upgrade procedure. After the system program has been downloaded, a new .COD file in v3 format will be generated from the selected application program file and downloaded together with the .BIN file (if applicable) and the uploaded .BPR file.

2 Alt. 2 – New Or Modified Network Configuration

- Start the Download Wizard, and select the following options: *Download new system, Download new application, and Download new configuration.*
- Select the appropriate system program, application program (select the file created in step 1, if applicable), and network database files.
- Click the **Apply** button to complete the upgrade procedure. After the system program has been downloaded, a .BPR file for the connected device will be generated from the selected network database file, a .COD file in v3 format will be generated from the selected application program file and, if applicable, a .BIN file as well. Finally, the .COD file will be downloaded together with the .BIN file and the .BPR file.

20.3.3 Verifying Correct Operation (Optional But Recommended)

- Start TAC Menta, and open the application program file with the application source code.
- Connect to the device, and verify that the application program in the controller is operating correctly.

21 Simple Blocks

One of the four classes of function blocks in TAC Menta are the simple blocks. Each simple block have a fixed number of inputs and parameters. Simple blocks generates a single output signal.

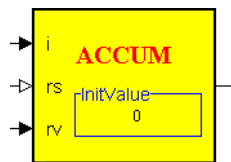


Fig. 21.1: A typical simple block.

The parameters in a simple block are always a numeric value or a list of numeric values within a pre-determined range for each type of parameter.

There are two ways to specify the numeric value:

Table 21.1:

| | |
|----------|--|
| Number | A decimal number in exponential format. |
| Constant | An identifier of 20 significant characters which must be defined in the <i>Constants table</i> . (see Chapter 13.10, “Using Constants”). |

21.1 Simple Blocks, Overview

Simple blocks can be divided in the following groups:

- Connection Blocks
- Physical I/O Blocks
- Signal Sources
- Logical Functions
- Non-linear Functions
- Delay Blocks
- Controllers and Filters
- Accumulators
- System Variables
- Time Schedules and Alarms

- Transformation Functions

21.1.1 Connection Blocks

Table 21.2: Connection blocks.

| Acronym | Short description |
|---------|--|
| RI | Real (analog) input connection block. |
| RO | Real (analog) output connection block. |
| PI | Pulse counting, digital input connection block |
| BI | Boolean (digital) input connection block. |
| BO | Boolean (digital) output connection block. |
| PU | Pulse width modulated output connection block. |

21.1.2 Physical I/O Blocks

Physical inputs and outputs (I/O) in TAC Xenta 280/300 and Xenta I/O modules are connected to the application using four I/O blocks: the AI, DI, AO and DO blocks.

Configuring the type of connection for the I/O blocks is called binding. The I/O Configuration Table lists all bindings for the physical I/O blocks in the application.

The I/O blocks can be configured to use either a physical connection or a signal from the network.

Two I/O blocks are special:

- one for counting pulses on digital inputs (CNT),
- one for pulse width modulating digital outputs (DOPU).

To avoid timing and synchronization problems, you can not bind these two blocks to network addresses.

Two special blocks are used for inputs and outputs in STR Wall Modules. These STRIN and STROUT blocks can only be associated with certain STR variables.

Table 21.3:

| Acronym | Short description |
|---------|------------------------------|
| AI | Physical analog input. |
| AO | Physical analog output. |
| CNT | Pulse counting digital input |

Table 21.3:

| Acronym | Short description |
|---------|--------------------------|
| DI | Physical digital input. |
| DO | Physical digital output. |
| DOPU | Digital pulse output. |
| STRIN | STR Wall Module input. |
| STROUT | STR Wall Module output. |

21.1.3 Signal Sources

Table 21.4:

| Acronym | Short description |
|---------|----------------------------------|
| NCYC | Program cycle counter. |
| OSC | Oscillator, gives a pulse train. |
| PVB | Binary value parameter. |
| PVI | Integer value parameter. |
| PVR | Real value parameter. |

21.1.4 Logical Functions

Table 21.5:

| Acronym | Short description |
|---------|--|
| AND | AND function on 2 Binary signals. |
| NOT | Inverts a Binary signal. |
| OR | OR function on 2 Binary signals. |
| PULSE | Monostable pulse generator (signal). |
| SR | Set-reset flip-flop |
| TRIG | Trigger, true on input change of state |
| XOR | XOR function on 2 Binary signals. |

21.1.5 Non-linear Functions

Table 21.6:

| Acronym | Short description |
|---------|------------------------------------|
| AHYST | Analog hysteresis |
| HYST | Binary hysteresis (relay function) |
| LIMIT | Max/min limit |
| MAX | Maximum value of 2 signals |
| MIN | Minimum value of 2 signals |

21.1.6 Delay Blocks

Table 21.7:

| Acronym | Short description |
|---------|-----------------------------------|
| DELAY | Delay on or off. |
| DELB | Delays a binary value 1 cycle. |
| DELI | Delays an integer value 1 cycle. |
| DELR | Delays a real value 1 cycle. |
| SHB | Sample and hold a binary value. |
| SHI | Sample and hold an integer value. |
| SHR | Sample and hold a real value. |

21.1.7 Controllers and Filters

Table 21.8:

| Acronym | Short description |
|---------|---|
| FILT | First order filter. |
| OPT | Start/stop time optimization |
| PIDA | PID controller (analog output) |
| PIDI | PID controller (increase/decrease output) |
| RAMP | Ramp filter (rate limit) |
| SEQ | Sequencer. |

21.1.8 Accumulators

Table 21.9:

| Acronym | Short description |
|---------|--------------------------|
| ACCUM | Real accumulator |
| INTEG | Integrator |
| RT | Running time measurement |

21.1.9 System Variables

This group of function blocks have no inputs. The blocks have their values updated by the system software in the TAC Xenta device.

Table 21.10:

| Acronym | Short description |
|---------|--|
| DATE | Actual date. |
| ERR | System error in a Xenta 280/300/400 device. |
| ERROR | System error in a Xenta 700 device |
| HOUR | Actual hour. |
| MINUTE | Actual minute. |
| MONTH | Actual month. |
| RST | Restart. The output is activated the first execution |
| SECOND | Actual second. |
| TCYC | Cycle time for the application program module. |
| WDAY | Actual day of week. 1 = Monday. |

21.1.10 Time Schedules and Alarms

Table 21.11:

| Acronym | Short description |
|---------|------------------------------------|
| ALARM | Initiates alarm message |
| TSCH | Time schedule |
| TSCHI | Time schedule block in a Xenta 700 |

21.1.11 Transformation Functions

Table 21.12:

| Acronym | Short description |
|---------|----------------------------------|
| CURVE | Partially linear curve function. |
| ENTH | Enthalpy calculation. |
| POLY | Polynomial transform function. |
| PRCNT | Percentages transformation |
| VECTOR | Vectorial transform function. |

21.2 Simple Block Concepts

Description of the function block

In the description of a block the following information is provided:



Fig. 21.2:

Table 21.13:

| | | | |
|--------------|------|------|-------------|
| Input | Name | Type | explanation |
| Parameters | Name | Type | explanation |
| Output types | Type | | |
| Access | RO | | |

Description.

Table 21.14:

| | |
|--------------|--|
| Inputs | Defines the input types and briefly explains their function. |
| Parameters | Defines the FB parameters and briefly explains their meaning. |
| Output Types | Defines the data type of the block output. When two blocks are connected, the output type should coincide with the input type. |

Table 21.14:

| | |
|-------------|---|
| Access | Defines whether the signal exported by the block is read only (RO) or read/write (R/W). Only blocks defined as R/W may be modified by an action from a superior node. |
| Description | A brief functional description of the block. |

21.3 ACCUM – Accumulator

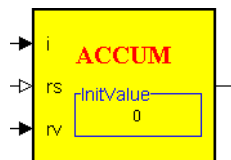


Fig. 21.3:

Table 21.15:

| | | | |
|--------------|-----------------|--------|--|
| Parameters | Increment (i) | REAL | Amount accumulated in each program cycle |
| | Reset (rs) | BINARY | Reset input (1 = reset) |
| | ResetValue (rv) | REAL | Value assigned to the output when the reset input is activated |
| Parameters | InitValue | REAL | Initial accumulator value |
| Output types | REAL | | |
| Access | R/W | | |

Description

This block is used to accumulate the total consumption from an incremental input (the consumption during one program cycle). It calculates the sum of the variable *Increment* over time. The summation is done using numerical precision to avoid underflow when the increment is small.



Note

- The accumulated value is delayed by one program cycle before it appears on the output.

The accumulator output is set to *InitValue* at the initial state. When the Reset input is activated, the accumulator output is reset to a value specified by the *ResetValue* input signal. When the Reset input is deacti-

vated, the accumulation resumes from the last value of the *ResetValue* input.

The upper and lower output limits are set by the maximum floating-point number that can be represented in the controller.

21.4 AHYST – Analog Hysteresis

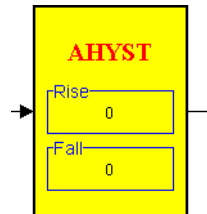


Fig. 21.4:

Table 21.16:

| Inputs | variable | REAL | input signal |
|--------------|----------|------|--|
| Parameters | Rise | REAL | value of the input signal for the ascent in the hysteresis loop |
| | Fall | | value of the input signal for the descent in the hysteresis loop |
| Output types | REAL | | |
| Access | R/W | | |

Description

This implements an Analog hysteresis function. The output follows the input as long as the input is outside the limits of the hysteresis loop. When the input variable enters into the hysteresis loop, the output takes the limiting values of the hysteresis, cf. the figure below:

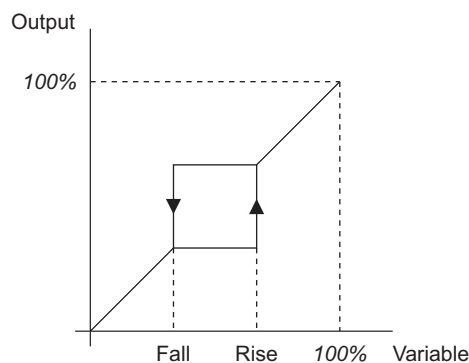


Fig. 21.5:

If $Rise > Fall$, the loop will be counter-clockwise as indicated in the figure. If $Rise < Fall$, the hysteresis loop will be clockwise.

If a complex function containing several hysteresis loops is required, the best method of achieving this is to form the function using other blocks (POLY, CURVE, VECTOR, etc.) and append in cascade as many AHYST blocks as there are hysteresis loops in the desired transfer function.

21.5 AI – Analog Input

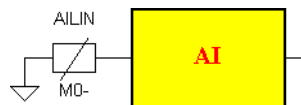


Fig. 21.6:

The AI block has five different sets of parameters depending on the selected **I/O Configuration** option:

- Linear Analog input
- Non-linear Analog input
- Network variable
- SNVT
- Constant value

The block output is updated only once during each program cycle. Changes in the physical inputs with a duration of less than one program cycle will not be noted by the application program.

The AI function block is not available for programming Xenta 700 devices in the Menta programming tool.



Note

- If you are using TAC Menta version 4.0 and system version 3.5 of the TAC Xenta device, it is possible to get the AI block to use the SI or I-P system of units for specified parameters.

Linear Analog Input

Table 21.17:

| | | | |
|-------------|---------------|---------|---|
| Parameters | Mod Number | INTEGER | Number of I/O module(s). |
| | Terminal Ref | STRING | Terminal reference. |
| | Sensor Type | ENUM | Sensor type |
| | Min Value | REAL | Lower input range limit |
| | Max Value | REAL | Upper input range limit |
| | Time Const | REAL | Filter time constant (sec). |
| | Initial Value | REAL | Initial output value (Only I/O module). Default value = 0. |
| Output type | REAL | | |
| Access | RO | | |

Description

The Linear Analog Input configuration option reads the value of an Analog input using a linear sensor. The block output value is the physical Analog input value converted to engineering units.

The conversion for different system of units (SI or I-P) is defined when the fields *Min Value* and *Max Value* are determined in relation to the input signal range of the selected type of Sensor.

The *Mod Number* parameter specifies the I/O module number (0 = TAC Xenta 280/300/400). *Terminal Ref* indicates the type and number of the input terminal (e.g. B1 – B4, U1 – U4).

Select the *Sensor Type* from a predefined list for the selected type of TAC Xenta device or I/O module, e.g. 0–10 V or 4–20 mA. The conversion to engineering units is done with the *Min Value* and *Max Value* parameters, which represent the lower and upper sensor range limits in engineering units, e.g. 0–40 °C, corresponding to the minimum and maximum electrical input signal range limits.

After conversion, the sensor reading can be filtered in a discrete time first order software filter included in the function block. The filter time constant is specified in seconds with the *Time Const* parameter.

The filter algorithm is

$$y(k) = y(k-1) + \frac{1}{1 + \frac{T}{h}}(u(k) - y(k-1))$$

where $y(k)$ and $u(k)$ are the filtered and unfiltered values at time k , respectively, h is the sampling interval (i.e. the application program cycle time), and T is the filter time constant. If the time constant is less

than or equal to zero, no filtering is done. To obtain a good filter function, the time constant should be set significantly greater than the application program cycle time.

This filter algorithm is based on a straightforward *backward difference* approximation of the continuous time derivative operator.

The initial block output value for an I/O module block (before a signal value has been received from the I/O module) is specified by the Initial Value parameter. If the I/O module goes offline, the block output will keep the last value that was received from the I/O module.

Non-Linear Analog Input

Table 21.18:

| | | | |
|-------------|---------------|---------|---|
| Parameters | Mod Number | INTEGER | Number of I/O module. |
| | Terminal Ref | STRING | Terminal reference. |
| | Sensor Type | ENUM | Sensor type |
| | Time Const | REAL | Filter time constant (sec). |
| | Initial Value | REAL | Initial output value (Only I/O module). Default value = 0. |
| Output type | REAL | | |
| Access | RO | | |

Description

The Non-linear Analog Input configuration option reads the value of an Analog input using a non-linear sensor, for instance a thermistor.

The block output value is the physical Analog input value converted to engineering units. The conversion is done through a predefined curve defining the sensor characteristics.

The selection is made when the Sensor: parameter is selected in the Drop-down Combo box in the Bind Analog Input dialog. Selecting a sensor in the SI unit system (metric) will generate a block where the block value is in the SI unit system (degree Celsius). Selecting a unit in the I-P (inch-pound) unit system will generate a block where the block value is converted to the IP unit system (degrees Fahrenheit).

For this type of analog input, the national setting of the PC does not influence the unit system.

The Unit: parameter for this type is only a text, used to describe the block and does not influence the unit system.

An edited application Function Block Diagram (FBD) will retain the selected unit system for this type of block whether the FBD is opened in a PC with an SI or an I-P unit system.

The *Mod Number* parameter specifies the I/O module number (0 = TAC Xenta 280/300/400). *Terminal Ref* indicates the type and number of the input terminal (e.g. B1 – B4, U1-U4).

The type of sensor characteristic used in the conversion to engineering units is selected from a predefined set of sensor types with the *Sensor Type* parameter. The choice *SP adjust* is used when connecting the potentiometer signal from the set point offset dial of a ZS 100 range wall module.

After conversion, the sensor reading can be filtered in a discrete time first order software filter included in the function block. The filter time constant is specified in seconds with the parameter *Time Const.*

The filter algorithm is the same as in the Linear Analog Input configuration option.

The initial block output value for an I/O module block (before a signal value has been received from the I/O module) is specified by the parameter *Initial Value*. If the I/O module goes offline, the block output will keep the last value that was received from the I/O module.

Network Variable

Table 21.19:

| | | | |
|-------------|-----------------|---------|---|
| Parameters | Network Address | STRING | Reference to external TAC network signal |
| | Initial Value | REAL | Initial output value. Default value = 0. |
| | Delta | REAL | The smallest change of value in the external signal which will initiate an update of the imported value. Default value = 0.5 |
| | Period | INTEGER | Maximum time interval (in seconds) between two updates of the imported value. Default value = 60 |
| Output type | REAL | | |
| Access | RO | | |

Description

The Network Variable configuration option is used to import signals from other TAC devices into the application program via the network.

The *Network Address* is a public Real or Integer signal in the other TAC device, entered as a character string e.g. \RPU1\AHU1\Outdoor-Temp (maximum string length \20\12\20). You cannot change the network address reference during runtime.

The initial block output value (before an external signal value has been received via the network) is specified by the *Initial Value* parameter. In case of a communications error, the block output will not be updated (i.e. it will always keep the last signal value that was received via the network). After a cold start, when the controller RAM is cleared, the block output will be reset to *Initial Value*.

The imported value will be updated each time the external signal value differs from the last sent value by more than *Delta* units (A *negative Delta* value means that the function is *not* used). Regardless of whether the external signal has changed or not, the imported value will also be updated when the time interval *Period* has elapsed since the last update (*Period* values less than 1 s will give *Period* = 1 s).



Note

- Do not use the same network address in several AI blocks in the same FBD, because it would impose an unnecessary load on network communications to have multiple import declarations of the same external signal. Only one block for each signal should be used, and connected to all blocks in the FBD that are using the imported signal.

SNVT

Table 21.20:

| | | | |
|-------------|---------------|---------|--|
| Parameters | Type | STRING | Reference to SNVT type |
| | Member | STRING | Reference to SNVT member (Only applicable when a structured SNVT type is selected). |
| | SNVT Name | STRING | Network identifier for the signal, max. 16 characters. |
| | Initial Value | REAL | Initial output value. Default value = 0. |
| | Poll | BINARY | Defines whether the bound external output variable is to be polled or not. |
| | Period | INTEGER | Maximum time interval (in seconds) between two updates of the imported value. Default value = 60 |
| Output type | REAL | | |
| Access | RO | | |

Description

The SNVT option is used to import external signals into the application program via the network. The input block must also be bound to an

external output variable of the same SNVT type via network management software. Note that the signal can be given different names within the application (the block name, e.g. OutdoorTemp) and on the network (the SNVT *name*, e.g. nvioutdoortemp).

A SNVT is, by design, represented in the SI unit system, but you can set it to be represented in the I-P unit system. The default unit selection is determined by the national setting of the Windows© operating system for the PC in which the application program is created.

The default Unit: parameter is determined when the Type and Member of the SNVT is selected in the Bind Analog Input dialog. Selecting the SNVT in a PC with a SI (metric) unit system will generate a block where the unit remains in the SI unit system. Selecting the SNVT in a PC with an I-P (inch-pound) unit system will generate a block where the value is converted to the I-P unit system.

The Unit: parameter can be redefined using the Drop-down Combo box in the Bind dialog.

An edited application Function Block Diagram (FBD) will retain the selected unit system for this type of block whether the FBD is opened in a PC with an SI or an I-P unit system setting.

Select *Type* and *Member* from predefined lists of supported SNVT types.

The initial block output value (before an external signal value has been received via the network) is specified by the *Initial Value* parameter. After a cold start, when the controller RAM is cleared, the block output will be reset to *Initial Value*.

In case of a communications error, the block output will not be updated, i.e. it will keep the last signal value that was received via the network.

After external binding, the external output variable will be polled if *Poll* is checked and no updates arrive within *Period*.

Constant Value

Table 21.21:

| | | | |
|-------------|---------------|------|------------------------------|
| Parameters | Initial Value | REAL | initial value of the output. |
| Output type | REAL | | |
| Access | RO | | |

Description

The input block is not connected to an external physical or network address. Instead the option assigns a constant value specified by the *Initial Value* parameter to the block output.

21.6 ALARM – Alarm

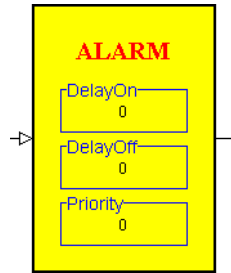


Fig. 21.7:

Table 21.22:

| Inputs | Input | BINARY | Input signal |
|-------------|-----------|---------|-----------------------------------|
| Parameters | DelayOn | REAL | Delay before alarm is set (sec) |
| | DelayOff | REAL | Delay before alarm is reset (sec) |
| | Priority | INTEGER | Alarm priority level |
| | AlarmText | STRING | Optional alarm text. |
| Output type | BINARY | | |
| Access | RO | | |

Description

The ALARM block monitors the state of the Binary input signal. A rising edge (transition from 0 to 1) starts a timer which measures how long the input is true (1). If the measured time exceeds the specified alarm activation (set) delay time, an alarm record containing information about the status, time, signal name, priority and an optional alarm text will be generated by the system software. When an alarm is set, the alarm block waits for the input to become false (0). The falling edge of the input signal triggers a timer in the same manner as the rising edge. After the alarm reset delay time has expired, the alarm is reset by the system software.

The ALARM block output signal indicates the current alarm status: 1 = alarm set, 0 = no alarm (reset).

The alarm set and reset delay times are specified in seconds with the *DelayOn* and *DelayOff* parameters, respectively. Please refer to the timing diagram below, which shows the setting and resetting of an alarm. If a power outage occurs during the delay, the elapsed delay time will be saved, if the *Backup* check box is set.

The alarm priority is specified with the *Priority* parameter. There are 10 levels, 1 to 9 and 0, where 1 is the highest. Events with priority 0 (Infor-

mation messages) have the lowest priority. They are not placed in the alarm database of TAC Vista and do not appear in the alarm overview.

The priority value has no function in the Xenta application program itself; it is used for presentation in, for example, TAC Vista or the TAC Operator Panel.

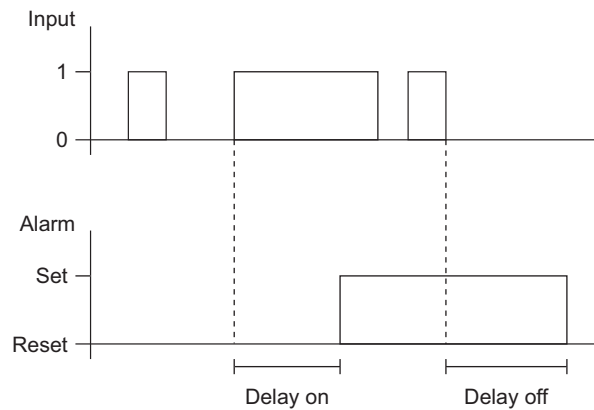


Fig. 21.8: Relation between Input signal and Alarm Output signal.



Note

- The input signal must be true for a time interval longer than *DelayOn* in order to generate a new alarm. The input must also be false for a time interval longer than *DelayOff* in order to reset the alarm.

21.7 AND – Logical AND Gate

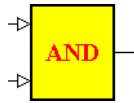


Fig. 21.9:

Table 21.23:

| | | | |
|-------------|--------|--------|--|
| Inputs | state1 | BINARY | |
| | state2 | BINARY | |
| Output type | BINARY | | |
| Access | RO | | |

Description

Calculates the boolean AND function of state1 and state2 according to the following truth table:

Table 21.24:

| state 1 | state 2 | output |
|---------|---------|--------|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

21.8 AO – Analog Output



Fig. 21.10:

The AO block has three sets of parameters depending on the selected **I/O Configuration** option:

- Physical output
- SNVT
- Not connected

The AO function block is not available for programming Xenta 700 devices in the Menta programming tool.



Note

- If you are using TAC Menta version 4.0 and system version 3.5 of the TAC Xenta device, it is possible to get the AO block to use the SI or I-P system of units for specified parameters.

Physical Output

Table 21.25:

| | | | |
|-------------|---------------------|---------|--|
| Input | Input | REAL | Input signal, 0-100% |
| Parameters | Mod Number | INTEGER | Number of I/O module. |
| | Terminal Ref | STRING | Terminal reference. |
| | Voltage 0% | REAL | Output signal at 0% input (Volt) |
| | Voltage 100% | REAL | Output signal at 100% input (Volt) |
| | Initial Value | REAL | Initial output value (Only I/O module). Default value = 0 (Volt). |
| Output type | Block has no output | | |
| Access | RO | | |

Description

The Analog output block converts a 0–100% signal from the application program to a voltage signal on a physical Analog output.

The *Mod Number* parameter specifies the I/O module number (0 = TAC Xenta 280/300/400). *Terminal Ref* indicates the type and number of the output terminal (e.g. Y1–Y4).

The output voltages corresponding to 0% and 100% input signal are specified (in Volts) with the *Voltage 0%* and *Voltage 100%* parameters, respectively. Note that the voltages may be reversed, e.g. *Voltage 0%* = 10 V and *Voltage 100%* = 2 V.

The *Initial Value* parameter specifies the output signal value (in V) for an I/O module block when the I/O module goes offline, e.g. following a restart immediately after a power outage.

SNVT

Table 21.26:

| | | | |
|-------------|---------------------|---------|---|
| Parameters | Type | STRING | Reference to SNVT type |
| | Member | STRING | Reference to SNVT member (Only applicable when a structured SNVT type is selected). |
| | SNVT Name | STRING | Network identifier for the signal, max. 16 characters. |
| | Initial Value | REAL | Initial output value. Default value = 0. |
| | Send | BINARY | Defines whether the bound external input variable is to be automatically updated or not. |
| | Delta | REAL | The smallest change of value in the input signal which will be exported to the external input variable. Default value = 0.5 |
| | Period | INTEGER | Maximum time interval (in seconds) between two updates of the exported value. Default value = 60 |
| Output type | Block has no output | | |
| Access | RO | | |

Description

The SNVT option is used to export signals from the application program via the network to an external device. The output block must also be bound to an external input variable of the same SNVT type via network management software. Note that the signal can be given different names within the application (the block name, e.g. OutdoorTemp) and on the network (the *SNVT Name*, e.g. nvooutdoortemp).

A SNVT is, by design, represented in the SI unit system, but you can set it to be represented in the I-P unit system. The default unit selection is

determined by the national setting of the Windows© operating system in the PC, in which the application program is created.

The default Unit: parameter is determined when the Type and Member of the SNVT is selected in the Bind Analog Input dialog. Selecting the SNVT in a PC with a SI (metric) unit system will generate a block where the unit remains in the SI unit system. Selecting the SNVT in a PC with an I-P (inch-pound) unit system will generate a block where the value is converted to the I-P unit system.

The Unit: parameter can be redefined using the Drop-down Combo box in the Bind dialog.

An edited application Function Block Diagram (FBD) will retain the selected unit system for this type of block whether the FBD is opened in a PC with an SI or an I-P unit system setting.

Select *Type* and *Member* from predefined lists of supported SNVT types.

The initial output value (before the first program cycle is executed) is specified by the *Initial Value* parameter.

If *Send* is checked, the external variable will be updated each time the block input signal value deviates from the last sent value by more than Delta units. Regardless of whether the input signal has changed or not, the external variable will also be updated when the time interval Period has elapsed since the last update. If *Period* is set to 0, the external variable will only be updated on *Delta* deviation.

If *Send* is not checked, the external device will have to poll to get updated signal values. The exported value will only be altered if the block input signal value deviates from the last exported value by more than *Delta* units.

Not Connected

The Not connected binding means that the block is not connected to a physical or network address. It has no configuration parameters.

21.9 BI – Binary Input

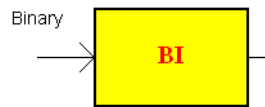


Fig. 21.11:

The BI connection block is used only in a Menta application for the Xenta 700 range of devices. The block is only available when the Menta programming tool is started from XBuilder.

The block is used to connect signals between Menta objects and signals in Xenta I/O modules, SNVTs and other signals, using XBuilder.

Connection blocks are not connected to physical I/O blocks when the Menta application is edited.

The block is updated once during each program execution. This means that changes to the physical inputs with a duration of less than one program execution will not be noted by the application program.

Table 21.27:

| | | | |
|------------|--------|--|--|
| Parameter | None | | |
| Input type | BINARY | | |
| Access | RO | | |

21.10 BO – Binary Output

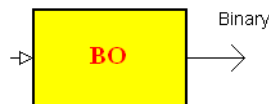


Fig. 21.12:

The BO connection block is used only in a Menta application for the Xenta 700 devices. The block is only available when the Menta programming tool is started from XBuilder.

The block is used to connect signals between Menta objects and to signals in Xenta I/O modules or SNVTs, using XBuilder.

Connection blocks are not connected to physical I/O blocks when the Menta application is edited.

Table 21.28:

| | | | |
|-------------|--------|--|--|
| Parameter | None | | |
| Output type | BINARY | | |
| Access | RO | | |

21.11 CNT – Digital Input – Pulse Counter

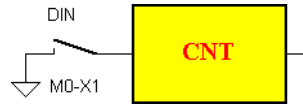


Fig. 21.13:

The CNT block can, via the **I/O Configuration** option, be bound as either Pulse counter (Physical input) or Not connected.

The CNT function block is not available for programming Xenta 700 devices in the Menta programming tool.

Table 21.29:

| | | | |
|-------------|---------------|---------|---|
| Parameters | Mod Number | INTEGER | Number of I/O module. |
| | Terminal Ref | STRING | Terminal reference. |
| | Normally Open | BINARY | Check box set = trig on rising edge i.e. open → closed (Default). Check box not set = trig on falling edge i.e. closed → open. |
| | Multiplier | REAL | Scale factor. Default value = 1.0 |
| Output type | REAL | | |
| Access | RO | | |

Description

This block counts the number of pulses on a digital input since the last execution of the application program. The counter will stop, *not* reset to zero, if the number of pulses reaches the maximum Integer value (32767) before the next program cycle or (if an I/O module is used) before the value has been sent to the TAC Xenta 280/300/400. The minimum pulse length depends on the hardware used.

The number of pulses is converted to engineering units by multiplying the pulse counter value with the scaling factor *Multiplier*.

The *Mod Number* parameter specifies the number of I/O module (0 = TAC Xenta 280/300/400), and *Terminal Ref* indicates the type and number of the input terminal (e.g. U1–U4, X1–X4).

Depending on the value of the *Normally Open* parameter, counting is triggered by the rising or falling edge of the digital input signal:

Check box set = trig on rising edge (open → closed) Check box not set = trig on falling edge (closed → open).

Not Connected

The *Not connected* binding means that the block is not connected to a physical or network address. It has no configuration parameters.

21.12 CURVE – Curve Function

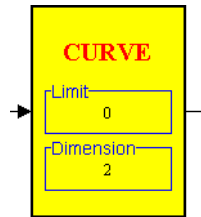


Fig. 21.14:

Table 21.30:

| Inputs | Input | REAL | |
|-------------|------------------------------|--------|---|
| Parameters | Limit | BINARY | Selector for limit function (limit = 1) or linear extrapolation (limit = 0). |
| | Dimension (Pair list x,y) | REAL | A list of co-ordinate pairs x,y defining the breakpoints of the curve function. |
| Output type | REAL | | |
| Access | RO | | |

Description

This block implements an arbitrary partially linear curve defined by a number of curve breakpoints (x_i, y_i) , i.e. the function values $y = f(x)$ for a set of input x values. The number of breakpoints must not exceed 127. The number of specified breakpoints (N) is indicated as Dimension in the graphical function block symbol. The breakpoints are entered as a list of Real values x,y separated by commas, with one co-ordinate pair on each row. The x values in the list should be entered in order, i. e. $x_{i-1} < x_i < x_{i+1}$.

The Input signal is compared to the x -components of the breakpoints in order. The first coordinate pair with an x -component greater than the Input signal is chosen as the end point of a line segment starting in the previous point. The Output signal is then calculated by linear interpolation on this line segment.

If the limit function selector is set to 1, the output will be set to y_1 when the input is less than x_1 , and to y_N when the input is greater than x_N . When the limit selector is disabled, the output is computed by linear extrapolation whenever the input signal exceeds the range (x_1, x_N) .

The following is a graphical representation of a “simple” case:

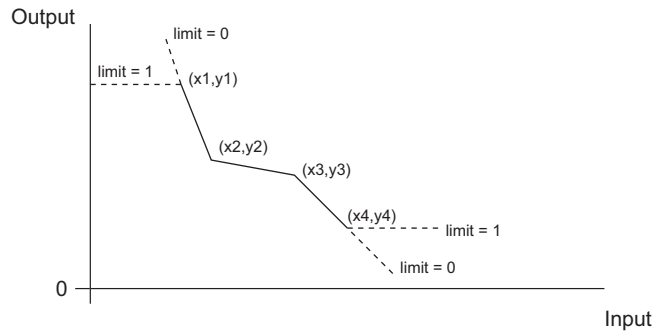


Fig. 21.15:

If two coordinates are equal, the curve will look like this:

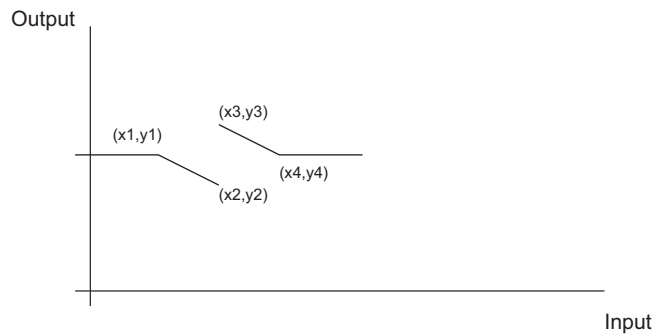


Fig. 21.16:

At the discontinuity, i.e. when the input = x_2 , the function output will be y_2 .

21.13 DATE – Day



Fig. 21.17:

Table 21.31:

| | | | |
|-------------|---------|--|--|
| Output type | INTEGER | | |
| Access | RO | | |

Description

Provides the day of month (1-31) according to the internal time clock.

21.14 DELAY – Delayed On/Off

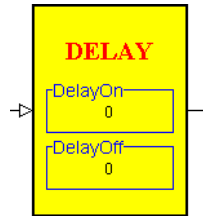


Fig. 21.18:

Table 21.32:

| | | | |
|-------------|----------|--------|-------------------------------|
| Inputs | state | BINARY | input signal |
| Parameters | DelayOn | REAL | activation delay in seconds |
| | DelayOff | REAL | deactivation delay in seconds |
| Output type | BINARY | | |
| Access | RO | | |

Description

This block delays the transitions of an input signal (*state*) by the time specified in seconds as defined by the *DelayOn* (transition 0 to 1) and *DelayOff* (transition 1 to 0) parameters. Note that the input signal must be true for a time interval longer than *DelayOn* in order to generate a pulse on the block output, cf. the timing diagram below. The input must also be false for a time interval longer than *DelayOff* in order to reset the output to false.

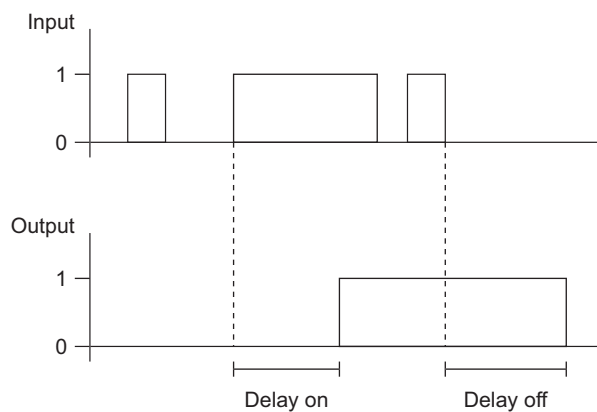


Fig. 21.19:

If a power outage occurs during the delay, the elapsed delay time will be saved, if the Backup check box is set.

21.15 DELB – Binary Value Delay

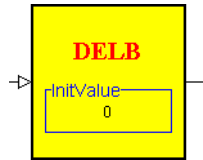


Fig. 21.20:

Table 21.33:

| | | | |
|-------------|-----------|--------|-------------------------------------|
| Inputs | state | BINARY | Binary input signal |
| Parameters | InitValue | BINARY | initial value for the output signal |
| Output type | BINARY | | |
| Access | R/W | | |

Description

This block introduces a delay of one program cycle in the propagation of a Binary signal. For each program cycle, the output value is updated with the input value from the previous cycle.

21.16 DELI – Integer Value Delay

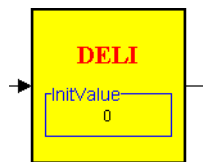


Fig. 21.21:

Table 21.34:

| | | | |
|-------------|-----------|---------|--------------------------------------|
| Inputs | variable | INTEGER | input signal. |
| Parameters | InitValue | INTEGER | initial value for the output signal. |
| Output type | INTEGER | | |
| Access | R/W | | |

Description

This block introduces a delay of one program cycle in the propagation of an Integer signal. For each program cycle, the value of the output is updated with the input value from the previous cycle.

21.17 DELR – Real Value Delay

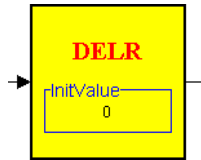


Fig. 21.22:

Table 21.35:

| | | | |
|-------------|-----------|------|--------------------------------------|
| Inputs | variable | REAL | Analog input signal. |
| Parameters | InitValue | REAL | Initial value for the output signal. |
| Output type | REAL | | |
| Access | R/W | | |

Description

This block introduces a delay of one program cycle in the propagation of a Real signal. For each program cycle, the value of the output is updated with the input value from the previous cycle.

21.18 DI – Digital Input

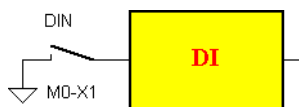


Fig. 21.23:

The DI block has five different sets of parameters depending on the selected **I/O Configuration** option:

- Network Variable
- Physical Input
- Online Device
- SNVT
- Constant Value

The block output is updated only once during each program cycle. This means that changes to the physical inputs that have a duration of less than one program cycle will not be noted by the application program.

The DI function block is not available for programming Xenta 700 devices in the Menta programming tool.

Network Variable

Table 21.36:

| | | | |
|-------------|-----------------|---------|---|
| Parameters | Network Address | STRING | Reference to external TAC network signal. |
| | Initial Value | BINARY | Initial block output value; On or Off. Default value = Off. |
| | Period | INTEGER | Maximum time interval (in seconds) between two updates of the imported value. Default value = 60. |
| Output type | BINARY | | |
| Access | RO | | |

Description

The Network Variable configuration option is used to import signals from other TAC devices into the application program via the network. The *Network Address* must be a public Binary signal in the other TAC device, entered as a character string e.g. \RPU1\AHU1\ExtendedOp (maximum string length \20\12\20). It is not possible to change the network address reference during runtime.

The initial block output value (before an external signal value has been received via the network) is specified by the *Initial Value* parameter. In case of a communications error, the block output will not be updated (i.e. it will always keep the last signal value that was received via the network). When cold starting, after the controller RAM is cleared, the network input block output will be reset to *Initial Value*.

The imported value will be updated each time the state of the external signal changes. Regardless of whether the external signal has changed or not, the imported value will also be updated when the time interval *Period* has elapsed since the last update (*Period* values less than 1 s will give *Period* = 1 s). There is no point in using the same network address in several DI blocks in the same FBD, because it would impose an unnecessary load on network communications to have multiple import declarations of the same signal. Only one block for each signal should be used and connected to all blocks in the FBD that are using the imported signal.

Physical Input

Table 21.37:

| | | | |
|-------------|---------------------------|---------|---|
| Parameters | Mod Number | INTEGER | Number of I/O module. |
| | Terminal Ref | STRING | Terminal reference. |
| | Initial Value | BINARY | Initial block output value (Only I/O module); On or Off. Default value = Off. |
| | Normal Polarity | BINARY | Open or Closed. Open (NO): open input contact interpreted as false (0); default. Closed (NC): open input contact interpreted as true (1). |
| | LED Color (422A, 452A) | BINARY | Green or Red. Color of LED when activated. |
| | LED Polarity (422A, 452A) | BINARY | Condition for LED activation; Non-Inverted or Inverted. Non-Inverted: Lighted when Closed Inverted: Lighted when Open. |
| Output type | BINARY | | |
| Access | RO | | |

Description

The Physical input configuration option reads the state of a digital input and translates it into a Binary value.

The parameter, *Mod Number*, specifies the I/O module number (0 = TAC Xenta 280/300/400), and *Terminal Ref* indicates the type and number of the input terminal (e.g. U1–U4, X1–X4).

Interpretation of the physical digital input state is defined by the *Normal Polarity* parameter. *Normal Polarity = Open* means that an open input contact gives a block output value of 0 (false), and *Normal Polarity = Closed* entails that an open input contact gives a block output value of 1 (true). The initial block output value for an I/O module block (before a signal value has been received from the I/O module) is specified by the *Initial Value* parameter. If the I/O module goes offline, the block output will keep the last value that was received from the I/O module.

The TAC Xenta 422A and 452A have universal inputs that can be used as digital inputs; their status are displayed using LED indicators. *LED Color* determines the color (Green or Red) when the Universal Input functioning as a Digital Input is activated. (If used for pulse counting, only Green is possible.)

LED Polarity determines if the LED should be lit at closed contact (Non-Inverted) or at open contact (Inverted).

Online Device

Table 21.38:

| | | | |
|-------------|----------------|--------|---|
| Parameters | Device Address | STRING | Reference to supervised device |
| | | BINARY | Initial block output value; On or Off. Default value = Off. |
| Output type | BINARY | | |
| Access | RO | | |

Description

The Online Device option is used to monitor the communication status of an I/O module or another device in the network. This can be used, for instance, in applications where signals are imported from other devices in the network, and where alternative actions must be taken if communication breaks down. The device name (or the I/O module number: 1, 2, etc.) is specified in the *Device Address* parameter, which is entered as a character string. To monitor the communication status of a supervisory TAC Vista system, use the *LonWorks Network* name as the device name. Note that the Online Device check can only be used for I/O modules which are defined in the TAC Xenta device where the application is running.

The block output will be true (1) if the device referenced by *Device Address* is online. Otherwise, the output will be false (0). If *Device Address* is the network address of the device itself, the block output will be false (0) when no other node in the network can be found. Otherwise, it will be true (1).

SNVT

Table 21.39:

| | | | |
|-------------|---------------|---------|--|
| Parameters | Type | STRING | Reference to SNVT type |
| | Member | STRING | Reference to SNVT member (Only applicable when a structured SNVT type is selected). |
| | Name | STRING | Network identifier for the signal, max. 16 characters. |
| | Initial Value | BINARY | Initial output value. Default value = Off. |
| | Poll | BINARY | Defines whether the bound external output variable is to be polled or not. |
| | Period | INTEGER | Maximum time interval (in seconds) between two updates of the imported value. Default value = 60 |
| Output type | BINARY | | |
| Access | RO | | |

Description

The SNVT option is used to import external signals into the application program via the network. The input block must also be bound to an external output variable of the same SNVT type via network management software. Note that the signal can be given different names within the application (the block name, e.g. StartButton) and on the network (the SNVT *Name*, e.g. nvistartbutton).

Select *Type* and *Member* from the predefined lists of supported SNVT types. For a Binary signal, SNVT_switch may be used.

The initial block output value (before an external signal value has been received via the network) is specified by the *Initial Value* parameter.

In case of a communications error, the block output will not be updated, i.e. it will keep the last signal value that was received via the network.

After external binding, the external output variable will be polled if *Poll* is checked and no updates arrive within *Period*.

Constant Value

Table 21.40:

| | | | |
|-------------|---------------|--------|---|
| Parameters | Initial Value | BINARY | Initial block output value; On or Off. Default value = Off. |
| Output type | BINARY | | |

Table 21.40:

| | | | |
|--------|----|--|--|
| Access | RO | | |
|--------|----|--|--|

Description

This input block is not connected to an external physical or network address. Instead the option assigns a constant value specified by the *Initial Value* parameter to the block output.

21.19 DO – Digital Output

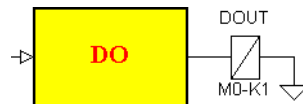


Fig. 21.24:

The DO block has three different sets of parameters depending on the selected **I/O Configuration** option:

- Physical output
- SNVT
- Not connected

The DO function block is not available for programming Xenta 700 devices in the Menta programming tool.

Physical Digital Output

Tabell 7:

| | | | |
|-------------|---------------------|---------|---|
| Inputs | Input | BINARY | Block input signal |
| Parameters | Mod Number | INTEGER | Number of I/O module. |
| | Terminal Ref | STRING | Terminal reference. |
| | Initial Value | BINARY | Initial output value (Only I/O module). Default value = Off. |
| Output type | Block has no output | | |
| Access | RO | | |

Description

This block sets the state of a physical digital output.

The *Mod Number* parameter specifies the I/O module number (0 = TAC Xenta 280/300/400), and *Terminal Ref* indicates the type and number of the output terminal (e.g. K1–K4).

The *Initial Value* parameter specifies the output signal value for an I/O module block when the I/O module goes offline, e.g. following a restart immediately after a power outage.

SNVT

Table 21.41:

| | | | |
|-------------|---------------------|---------|--|
| Parameters | Type | STRING | Reference to SNVT type |
| | Member | STRING | Reference to SNVT member (Only applicable when a structured SNVT type is selected). |
| | SNVT Name | STRING | Network identifier for the signal, max. 16 characters. |
| | Initial Value | BINARY | Initial output value. Default value = Off |
| | Send | BINARY | Defines whether the bound external input variable is to be automatically updated or not. |
| | Period | INTEGER | Maximum time interval (in seconds) between two updates of the exported value. Default value = 60 |
| Output type | Block has no output | | |
| Access | RO | | |

Description

This SNVT option is used to export signals from the application program via the network to an external device. The output block must also be bound to an external input variable of the same SNVT type via network management software. Note that the signal can be given different names within the application (the block name, e.g. StartButton) and on the network (the *SNVT Name*, e.g. nvostartbutton).

Select *Type* and *Member* from the predefined lists of supported SNVT types. For a Binary signal SNVT_switch may be used.

The initial output value (before the first program cycle is executed) is specified by the *Initial Value* parameter.

If Send is checked, the external variable will be updated each time the block input signal changes its state. Regardless of whether the input signal has changed or not, the external variable will be updated when the time interval *Period* has elapsed since the last update. If *Period* is set to 0, the external variable will only be updated on status changes.

If *Send* is not checked, the external device will have to poll to get updated signal values.

Not Connected

The *Not connected* binding means that the block is not connected to a physical or network address. It has no configuration parameters.

21.20 DOPU – Digital Pulse Output

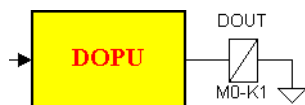


Fig. 21.25:

The DOPU block can, via the **I/O Configuration** option be bound as either Digital pulse output (Physical output) or Not connected.

The DOPU function block is not available for programming Xenta 700 devices in the Menta programming tool.

Digital Pulse Output

Table 21.42:

| | | | |
|-------------|---------------------|---------|---|
| Inputs | Input | REAL | Pulse length (sec). |
| Parameters | Mod Number | INTEGER | Number of I/O module. |
| | Terminal Ref | STRING | Terminal reference. |
| | Min Pulse | REAL | Minimum output pulse length (sec). Default = 0.5 sec. |
| | Initial Value | REAL | Initial output value (Only I/O module). Default value = 0 sec. |
| Output type | Block has no output | | |
| Access | RO | | |

Description

This block activates a pulse on a physical digital output. The output pulse length is given by the block input signal value (in seconds). The DOPU block is designed to be used together with the increase/decrease PID block (PIDI). Negative block input values are ignored.

The *Mod Number* parameter specifies the I/O module number (0 = TAC Xenta 280/300/400), and *Terminal Ref* indicates the type and number of the output terminal (e.g. K1–K4).

The *Min Pulse* parameter specifies the minimum output pulse length in seconds. Input signals less than this value are accumulated for the next application program cycle. Input pulse lengths longer than the application program cycle time are truncated to a length equal to the program cycle.

The *Initial Value* parameter specifies the output signal value (in sec) for an I/O module block when the I/O module goes offline, e.g. following a restart immediately after a power failure.

Not Connected

The *Not connected* binding means that the block is not connected to a physical or network address. It has no configuration parameters.

21.21 ENTH – Enthalpy

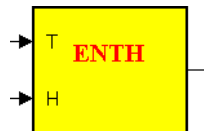


Fig. 21.26:

Table 21.43:

| | | | |
|-------------|-----------------|------|------------------------------|
| Inputs | Temperature (T) | REAL | Dry-bulb temperature °C (°F) |
| | Humidity (H) | REAL | Relative humidity (%) |
| Output type | REAL | | |
| Access | RO | | |

Description

The enthalpy, kJ/kg (BTU/pond) of moist air at normal atmospheric pressure is calculated as a function of the dry-bulb temperature °C, (°F) and the relative humidity (%).



Note

- If you are using TAC Menta version 4.0 and system version 3.5 of the TAC Xenta device, it is possible to get the ENTH block to use the SI or I-P system of units for specified parameters.

The ENTH block will use the T input (temperature) in SI unit system (metric) or I-P (inch-pound) unit system, depending on the national setting of the Windows© operating system of the PC in which the application is generated. Using the metric setting will produce a block where

the T input is in degrees Celsius, a US setting will produce a block where the T input is in degrees Fahrenheit.

The above-mentioned setting will also determine the units for the calculated enthalpy. An SI (metric) setting will give the result in kJ/kg. An I-P (inch-pound) setting will give a result in BTU/pound. The calculation of enthalpy in each unit system differs, and cannot be converted from one unit system to another.

Algorithm, SI measurement units

The water vapor saturation pressure over liquid water for the temperature range 0 to 200 °C is approximated by the following function:

$$pws(T) = \exp(c8/T + c9 + c10*T + c11*T^2 + c12*T^3 + c13*\ln(T)) \quad (6)$$

where

T= Absolute temperature (K),

pws = saturation pressure (Pa)

and

$$c8 = -5.8002206E+03;$$

$$c9 = 1.3914993E+00;$$

$$c10 = -4.8640239E-02;$$

$$c11 = 4.1764768E-05;$$

$$c12 = -1.4452093E-08;$$

$$c13 = 6.5459673E+00;$$

The enthalpy of moist air at normal atmospheric pressure as a function of the dry-bulb temperature and the relative humidity is then given by the following equations:

$$pw_{sat} = pws(t+273.15)$$

$$pw = \Phi * pw_{sat} / 100 \quad (24)$$

$$W = 0.62198 * pw / (p - pw) \quad (22)$$

$$h = 1.006 * t + W * (2501 + 1.805 * t) \quad (32)$$

where

pw_{sat} = water vapor saturation pressure (Pa)

t = dry-bulb temperature (°C)

Φ = relative humidity (%)

pw = partial pressure of water vapor in moist air (Pa)

p = ambient pressure (101325 Pa)

W = humidity ratio, mass of water per unit mass of dry air

h = enthalpy (kJ/kg)



Note

- Reference: ASHRAE Handbook 2001 – Fundamentals SI System of units. Chapter 6, Psychrometrics. The equation numbers within parenthesis refer to the corresponding equations in the ASHRAE Handbook.

Algorithm, I-P measurement units

The water vapor saturation pressure over liquid water for the temperature range -80 to 300 °F is approximated by the following function:

$$pws(T) = \exp(c8/T + c9 + c10*T + c11*T^2 + c12*T^3 + c13*\ln(T)) \quad (6)$$

where

T = Absolute temperature (°R),

pws = saturation pressure (psia)

and

$$c8 = -1.0440397E+04;$$

$$c9 = -1.1294650E+01;$$

$$c10 = -2.7022355E-02;$$

$$c11 = 1.2890360E-05;$$

$$c12 = -2.4780681E-09;$$

$c13 = 6.5459673E+00$; The enthalpy of moist air at normal atmospheric pressure as a function of the dry-bulb temperature and the relative humidity is then given by the following equations:

$$pw_{sat} = pws(t+459.67)$$

$$pw = \Phi * pw_{sat} / 100 \quad (24)$$

$$W = 0.62198 * pw / (p - pw) \quad (22)$$

$$h = 0.240 * t + W * (1061 + 0.444 * t) \quad (32)$$

where

pw_{sat} = water vapor saturation pressure (psi)

t = dry-bulb temperature (°F)

Φ = relative humidity (%)

pw = partial pressure of water vapor in moist air (psi)

p = ambient pressure (14.696 psi)

W = humidity ratio, mass of water per unit mass of dry air

h = enthalpy (BTU/pound)



Note

- Reference: ASHRAE Handbook 2001 – Fundamentals I-P System of units. Chapter 6, Psychrometrics. The equation numbers within parenthesis refer to the corresponding equations in the ASHRAE Handbook.

21.22 ERR – System Error



Fig. 21.27:

Table 21.44:

| | | | |
|-------------|---------|--|--|
| Output type | INTEGER | | |
| Access | RO | | |

The ERR function block is not available for programming Xenta 700 devices in the Menta programming tool.

Description

The output of this block is an Integer value where each bit represents an internal signal or error from the system program (see the following section). Each output bit will be set as long as the error condition remains. In the case of a restart after a power failure, during the first program cycle following the restart. The output value is zero when no error conditions are set.

21.22.1 Error Codes for the ERR Function Block

Table 21.45:

| Bit No. | Output Value | Description |
|---------|--------------|---|
| 0 | 1 | Restart after power failure |
| 1 | 2 | Integer underflow/overflow |
| 2 | 4 | CNT block overflow |
| 3 | 8 | RT block overflow |
| 4 | 16 | Mainloop overrun. The application program cannot execute within the specified cycle time |
| 5 | 32 | At least one of the switches on one of the I/O modules associated with the application is in the position for manually overriding (ON, OFF, MAN) the output |
| 6 | 64 | OP logged in on yellow or red access level (applicable from Xenta system v3.51) |
| 7 | 128 | |
| 8 | 256 | Dial-up modem line to TAC Xenta device disabled |
| 9 | 512 | Update of a Network variable value was not received within the set Period |
| 10 | 1024 | There are tripped, unacknowledged, priority 1 alarms in the alarm list |
| 11 | 2048 | One or several I/O modules offline |
| 12 | 4096 | I/O in forced mode |
| 13 | 8192 | There are tripped and unacknowledged alarms in the alarm list |
| 14 | 16384 | There are tripped and not reset alarms in the alarm list |

Use only one ERR block in the application, otherwise the load on the TAC Xenta device can be high.

21.23 ERROR – System Error in Xenta 700



Fig. 21.28:

Table 21.46:

| | | | |
|-------------|---------|--|--|
| Output type | INTEGER | | |
| Access | RO | | |

The ERROR function block is used only in a Menta application for the Xenta 700 devices.

The block is available only in the Menta programming tool when it is started from XBuilder.

Description

The output of this block is an integer value where each bit represents an internal signal or error from the system.



Important

- The ERROR block does not adjust all the fifteen bits like the ERR function block.

Each output bit will be set as long as the error condition remains.

For a restart after a power failure, the value of bit 0 is set during the first program cycle after a restart.



Important

- The bit 4 is set when an Overrun in any control task in the device has occurred. To reset the bit 4, the system variable **Clear all** for the actual task, or the system variable **Control Task Overrun Any** must be set to zero, manually or programmatically.

The output value is zero when there is no error condition.

21.23.1 Error Codes for the ERROR Function Block

Table 21.47:

| Bit No. | Output Value | Description |
|---------|--------------|--|
| 0 | 1 | Set during the first execution when restarted after a power failure. |
| 1 | 2 | Integer underflow/overflow when the II function block is used in 16 bit mode (Mode 0). |
| 2 | 4 | Not adjusted. |
| 3 | 8 | RT block overflow. |
| 4 | 16 | Overrun in any control task in the device. The code for a Menta object cannot execute within the specified cycle time. The bit is not automatically reset. |
| 5 | 32 | One or several switches for manually overriding the output on an I/O module associated with the application is in the ON, OFF, or MAN position. |
| 6 | 64 | Not adjusted. |
| 7 | 128 | Not adjusted. |
| 8 | 256 | Not adjusted. |
| 9 | 512 | Not adjusted. |
| 10 | 1024 | There are tripped and not reset priority 1 alarms in the alarm list. |
| 11 | 2048 | One or several I/O modules associated with the application is offline. |
| 12 | 4096 | Not adjusted. |
| 13 | 8192 | There are tripped and unacknowledged alarms in the alarm list. |
| 14 | 16384 | There are tripped and not reset alarms in the alarm list. |

21.24 FILT – First Order Filter

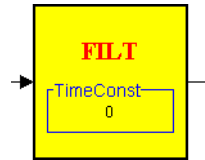


Fig. 21.29:

Table 21.48:

| | | | |
|-------------|-----------|------|---|
| Inputs | variable | REAL | Input signal |
| Parameters | TimeConst | REAL | Time constant of the filter measured in seconds |
| Output type | REAL | | |
| Access | R/W | | |

Description

The function block is a discrete time first order software filter. The filter time constant is specified in seconds with the *Time Const* parameter. The initial block output value is zero.

The filter algorithm is

$$y(k) = y(k-1) + \frac{1}{1 + \frac{T}{h}}(u(k) - y(k-1))$$

where $y(k)$ and $u(k)$ are the filtered and unfiltered values at time k , respectively, h is the sampling interval (i.e. the application program cycle time), and T is the filter time constant. If the time constant is less than or equal to zero, no filtering is done. To obtain a good filter function, the time constant should be set significantly greater than the application program cycle time.

This filter algorithm is based on a straightforward *backward difference* approximation of the continuous time derivative operator.

21.25 HOUR – Hour



Fig. 21.30:

Table 21.49:

| | | | |
|-------------|---------|--|--|
| Output type | INTEGER | | |
| Access | RO | | |

Description

Provides the current hour (0–23) according to the internal time clock.

21.26 HYST – Binary Hysteresis

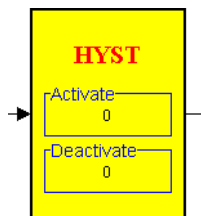


Fig. 21.31:

Table 21.50:

| | | | |
|-------------|------------|------|---|
| Inputs | variable | REAL | input signal |
| Parameters | activate | REAL | threshold value of the input signal for activation of the output. |
| | deactivate | REAL | threshold value of the input signal for deactivation of the output. |
| Output type | REAL | | |
| Access | RO | | |

Description

This block implements a relay function with hysteresis.

When *activate* is greater than *deactivate*, the block has the following function:

- If the output is false (0) and the input signal exceeds the activation threshold, the output will change to true (1).
- If the output is true (1) and the input signal drops below the deactivation threshold, the output will change to false (0).
- When the input signal is in the zone between the two thresholds, the output will remain in its previous state.

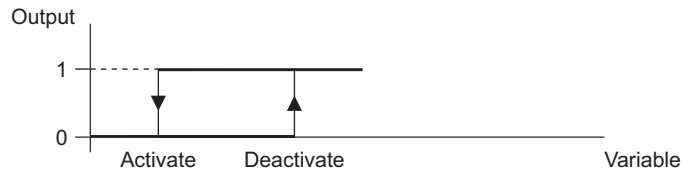


Fig. 21.32:

When activate is less than deactivate, the block has the following function:

- If the output is true (1) and the input signal exceeds the deactivation threshold, the output will change to false (0).
- If the output is false (0) and the input signal drops below the activation threshold, the output will change to true (1).
- When the input signal is in the zone between the two thresholds, the output will remain in its previous state.

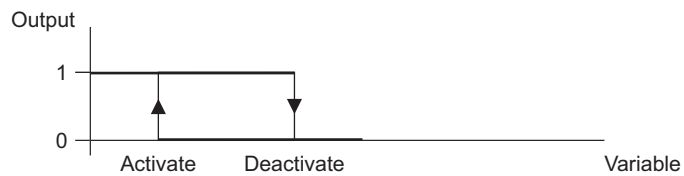


Fig. 21.33:

21.27 INTEG – Integrator

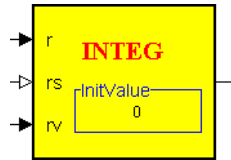


Fig. 21.34:

Table 21.51:

| | | | |
|-------------|-----------------|--------|--|
| Inputs | Rate (r) | REAL | rate input variable |
| | Reset (rs) | BINARY | reset input (1 = reset) |
| | ResetValue (rv) | REAL | value assigned to the output when the reset input is activated |
| Parameters | InitValue | REAL | Initial integrator value |
| Output type | REAL | | |
| Access | R/W | | |

Description

This block allows the integration of a rate or flow over time. The integral is computed as the sum of the rate multiplied by the time increment, i.e. by the program cycle time. The summation is done using extended numerical precision to avoid underflow when the rate is small.

The integrator output is set to *InitValue* at the initial state. When the *Reset* input is activated, the integrator output is reset to a value specified by the *ResetValue* input signal. When the *Reset* input is deactivated, the integration is resumed starting from the last value of the *ResetValue* input.

The upper and lower output limits are set by the maximum floating-point number that can be represented in the controller.

21.28 II – Integer Input

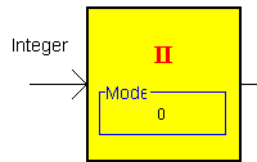


Fig. 21.35:

The II connection block is used only in a Menta application for the Xenta 700 devices. The block is only available when the Menta programming tool is started from XBuilder.

The block is used to connect signals between Menta objects and signals in Xenta I/O modules, SNVTs and other signals, using XBuilder.

Connection blocks are not connected to physical I/O blocks when the Menta application is edited.

Description

The value of the II connection block is a 16 bit integer value. The block handles inputs of 32 bit integer values controlled by the *Mode* parameter value:

- *Mode* parameter value 0.
The block integer value is limited to the range -32768 and 32767. An input value outside this range generates a system error called integer underflow/overflow and can be detected by the ERROR block.
- *Mode* parameter value 1.
The block integer value is the most significant word (16 bits) of the 32 bits input value.
- *Mode* parameter value 2.
The block integer value is the least significant word (16 bits) of the 32 bits input value.

Table 21.52:

| Parameter | Mode | | 16 Bit output control |
|------------|---------|--|-----------------------|
| Input type | INTEGER | | |
| Access | RO | | |

21.29 IO – Integer Output



Fig. 21.36:

The IO connection block is used only in a Menta application for the Xenta 700 devices. The block is only available when the Menta programming tool is started from XBuilder.

The block is used to connect signals between Menta objects and signals in Xenta I/O modules, SNVTs and other signals, using XBuilder.

Connection blocks are not connected to physical I/O blocks when the Menta application is edited.

Table 21.53:

| | | | |
|-------------|---------|--|--|
| Parameter | None | | |
| Output type | INTEGER | | |
| Access | RO | | |

21.30 LIMIT – High/Low Signal Limit

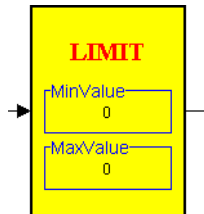


Fig. 21.37:

Table 21.54:

| | | | |
|-------------|----------|------|---------------------|
| Inputs | Input | REAL | input signal |
| Parameters | MinValue | REAL | minimum value limit |
| | MaxValue | REAL | maximum value limit |
| Output type | REAL | | |
| Access | RO | | |

Description

The Input signal is limited to the range [MinValue, MaxValue].

21.31 MAX – Maximum Signal Selector



Fig. 21.38:

Table 21.55:

| | | | |
|-------------|--------|------|---------------------|
| Inputs | Input1 | REAL | first input signal |
| | Input2 | REAL | second input signal |
| Output type | REAL | | |
| Access | RO | | |

Description

The output of the MAX block is set to the maximum value of *Input1* and *Input2*.

21.32 MIN – Minimum Signal Selector



Fig. 21.39:

Table 21.56:

| | | | |
|-------------|--------|------|---------------------|
| Inputs | Input1 | REAL | first input signal |
| | Input2 | REAL | second input signal |
| Output type | .. | | |
| Access | RO | | |

Description

The output of the MIN block is set to the minimum value of *Input1* and *Input2*.

21.33 MINUTE – Minute



Fig. 21.40:

Table 21.57:

| | | | |
|-------------|---------|--|--|
| Output type | INTEGER | | |
| Access | RO | | |

Descriptipn

Provides the current minute (0–59), according to the internal time clock.

21.34 MONTH – Month



Fig. 21.41:

Table 21.58:

| | | | |
|-------------|---------|--|--|
| Output type | INTEGER | | |
| Access | RO | | |

Descriptipn

Provides the month of the year according to the internal time clock. Month 1 corresponds to January and month 12 to December.

21.35 NCYC – Program Cycle Counter

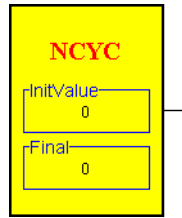


Fig. 21.42:

Description

This block increments the output value by one unit each program cycle if *IniValue* is less than *Final*, and decreases it if the opposite applies. When the count reaches the final value, the counter restarts the count at the initial value in the next cycle.

21.36 NOT – NOT Gate

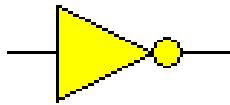


Fig. 21.43:

Table 21.59:

| | | | |
|-------------|--------|--------|--------------|
| Inputs | State | BINARY | Input signal |
| Output type | BINARY | | |
| Access | RO | | |

Description

This block inverts a Binary signal.

21.37 OPT – Optimization

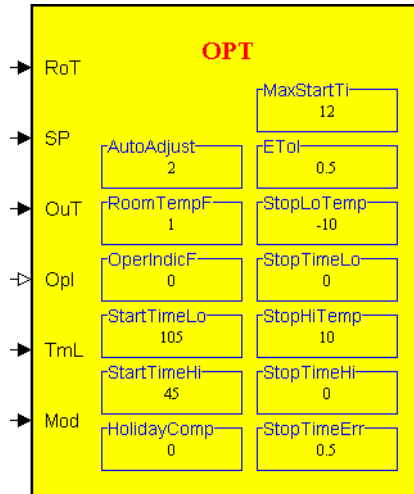


Fig. 21.44:

| | | | |
|--------|------|---------|--|
| Inputs | RoT | REAL | Room temperature °C (°F). |
| | SP | REAL | Set point, i.e. the desired room temperature during day time °C (°F). |
| | OuT | REAL | Outdoor temperature °C (°F). |
| | OpI | BINARY | Operation indication, i.e. an external signal that is used for detecting whether the controlled equipment is in operation. OpI = 0 => not in operation. OpI = 1 => in operation. |
| | TmL | INTEGER | Time left (minutes) until the plant goes into operation. If TmL is negative, OPT will interpret the size of TmL as the time until the plant d shuts down. |
| | Mode | INTEGER | Switch for media mode and disabling optimisation. Mode = 0 => heating. Mode = 1 => cooling. Mode = -1 => no optimisation. Default = 0, i.e. heating. |

Table 21.60:

| | | | |
|------------|--------------|---------|---|
| Parameters | AutoAdjust | INTEGER | Automatic adjustment switch. AutoAdjust = 0 => no adjustment. AutoAdjust = 1 => adjustment of curve points. AutoAdjust = 2 => adjustment of curve points and holiday compensation. Default = 2. |
| | RoomTempF | BINARY | Room sensor flag. RoomTempF = 0 => Sensor not present. RoomTempF = 1 => Sensor present. Default = 1, i.e. sensor present. |
| | OperIndicF | BINARY | Flag for selecting whether external operation indication is present. OperIndicF = 0 => Not present. OperIndicF = 1 => Present. Default = 0, i.e. not present. |
| | StartTimeLo | REAL | Start-time (minutes) at low outdoor temperature-10 °C (14°F), 10 °C (50 °F). Default = 105. |
| | StartTimeHi | REAL | Start-time (minutes) at high outdoor temperature 10 °C (50 °F), 30 °C (86°F). Default = 45. |
| | Holiday-Comp | REAL | Holiday compensation (%) when the plant has been shut down > 48 h. Default = 0. |
| | MaxStartTi | REAL | Max start-time (hours). Default = 12. |
| | ETol | REAL | Temperature error when switching from optimisation to normal operation. Default = 0.5. |
| | StopLoTemp | REAL | Low outdoor temperature point in stop-time optimisation °C (°F). Default = -10. |
| | StopTimeLo | REAL | Stop-time (minutes) when outdoor temperature = StopLoTemp. Default = 0. |
| | StopHiTemp | REAL | High outdoor temperature point in stop-time optimisation °C(°F). Default = 10. |
| | StopTimeHi | REAL | Stop-time (minutes) when outdoor temperature = StopHiTemp. Default = 0. |

Table 21.60:

| | | | |
|-------------|---------|--|--|
| Output type | INTEGER | | |
| Access | R/W | | |

Description

General

OPT provides optimum start and stop. OPT may be used in heating as well as cooling applications. OPT is executed as often as the other function blocks in the same application. However, the main part of the algorithm is only executed once every minute. The purpose of the start-time optimisation is to start the heating/cooling system in advance in order to obtain the correct temperature at the beginning of normal operation. The purpose of the optimum stop is to shut down the heating/cooling system before the end of normal operation, without the temperature falling outside the given limits during normal operation.

The optimisation function may be blocked by setting the parameter *Mode* = -1. The output of OPT will then follow the time schedule status, i.e. the output = 0 when *TmL* > 0, and output = 1 when *TmL* < 0. Curve points and *HolidayComp* will not be adjusted when *Mode* = -1.



Note

- If you are using TAC Menta version 4.0 and system version 3.5 for TAC Xenta devices, it will be possible to get the OPT block to use the SI or I-P system of units for specified parameters.

The block will use the *RoT*, *SP*, and *Out* inputs in the SI unit system (metric) or I-P (inch-pound) unit system, depending on the national setting of the Windows© operating system of the PC in which the application is generated. Using the metric setting will produce a block where the inputs are in degrees Celsius, and a US setting will produce a block where the inputs are in degrees Fahrenheit.

The block parameters *StartTimeLo*, *StartTimeHi*, defines parts of coordinates for breakpoints which are using fixed temperatures of -10 °C (14 °F), 10 °C (50 °F), and 30 °C (86 °F). Entering values for the start times completes definition of the coordinates.

Start-Time Optimization

The start-time is calculated once every minute on condition that *TmL* > 0 and optimisation has not already started. If the calculated start-time is longer than the time left until the plant goes into normal operation, the output of OPT will be set to 2 (optimisation state).

When the room temperature has reached the set point (*SP*) minus an adjustable tolerance, *ETol* (default = 0.5 °C), the output of the function block is set to 1 (normal operation) in order to indicate that the start-time optimisation is completed and that normal operation may begin. If the

room temperature has not reached the temperature required for normal operation to be set before TmL turns negative, the output will be set to 1 at this point. During cooling, the output is instead set to 1 when the room temperature falls below the set point (SP), plus the same tolerance ($ETol$). The principle of start-time optimisation is illustrated in Figure 1. Note that we have assumed in the figure that the correct temperature is reached at the same time as TmL turns negative.

The start-time is obtained from a curve that defines the relationship between the outdoor temperature ($OutT$) and the start-time. If a room sensor is not used, the start-time will be calculated directly from the curve with a possible contribution from holiday compensation, see below. When a room sensor is connected, the curve instead yields the start-time per °C of deviation between the room temperature (RoT) and its set point (SP). For example, the start-time will be 120 min if the curve yields the value 60 min at the outdoor temperature in question and the room temperature is 2 °C too low.

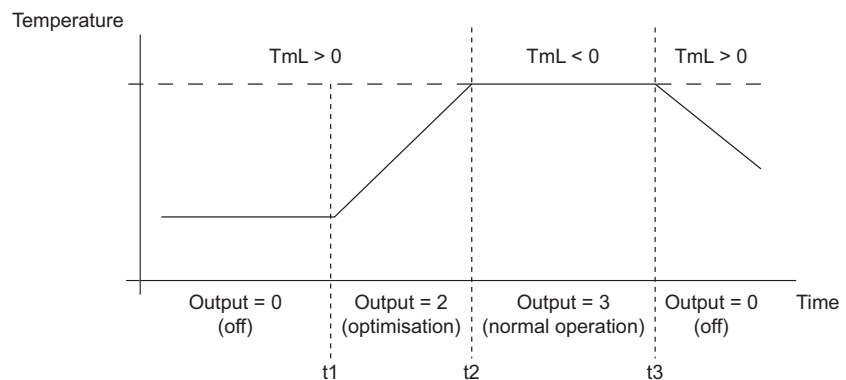


Fig. 21.45: Principle for start-time optimisation.

To obtain a correct start-time after longer shut-down periods, e.g. after holidays or weekends, an extra contribution may be added to the start-time. Holiday compensation will contribute to the start-time when the building has not been in use for a long period, see *Holiday compensation (Monday effect)* on the next page. If the parameter $OperIndicF$ is set to 1, OPT will use the input OpI to decide whether the plant is in operation. If $OperIndicF$ is set to 0, OPT will ignore the input OpI and instead assume that the plant is in operation if the output of OPT is 1 (normal operation) or 2 (optimisation state).

Finally, the start-time is limited to a maximum value ($MaxStartTi$).

OPT Output Values

As described above and in Figure 1, the OPT block output value is used to set the plant's operating mode, as described in the table below:

- Output = 0 => Off.
- Output = 1 => Normal operation.
- Output = 2 => Optimisation state.

Curve, Start-Time as a Function of Outdoor Temperature

The curve that describes the relationship between the outdoor temperature and the start-time, or alternatively the start-time per °C of room deviation if the room sensor is connected, is defined by two points, see Fig. 2. The outdoor temperature at these points will be defined by the selection of the Mode (cooling or heating), while the corresponding start-times (*StartTimeLo* and *StartTimeHi*) are freely selectable. The outdoor temperatures of the curve points are -10 °C and 10 °C during heating, respectively 10 °C and 30 °C during cooling. The default values for the start-times are 105 minutes (*StartTimeLo*) and 45 minutes (*StartTimeHi*). The start-times at the curve points may be adjusted automatically, see below.

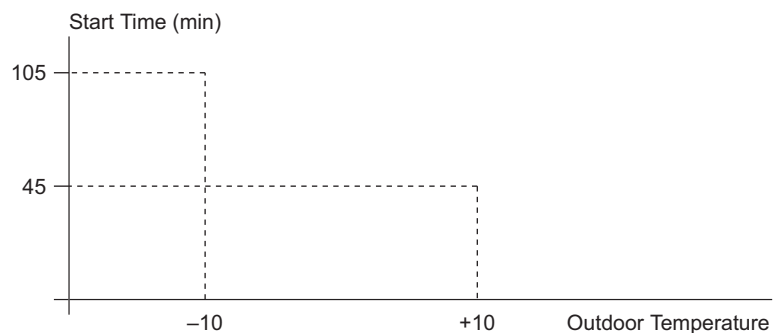


Fig. 21.46: Curve describing the relationship between outdoor temperature and start-time. The figure shows the default-curve during heating.

Automatic Adjustment of Curve Points

To be able to adjust the curve to the thermal properties of a specific building, there is a mechanism for automatically adjusting the curve points. If the desired room temperature has not been reached when the optimisation time expires (or, if it is reached too early), the adjustment mechanism will compute an estimate of the optimisation time that should have been used and will adjust the curve points accordingly. Automatic adjustment of the curve points may only, of course, be used in systems with a room sensor.

Adjustment of the curve points is performed when normal operation begins following an optimisation phase and adjustment is selected (*AutoAdjust* >0). Adjustment of the curve points will not be performed when the plant has been shut down for more than 20 hours.

You can alter the start-times in the curve points via the Operators Interface. If this is done while optimisation is active, no adjustment of the curve points will be performed when normal operation begins following this optimisation phase. After the next optimisation phase, the adjustment mechanism will work as normal again).

Holiday Compensation (Monday Effect)

When the building has been shut down for a long period of time, a longer heating time is required to reach the desired temperature, due to the fact that the building has been cooled down more effectively. To compensate for this, we add an extra percentage of the start-time calculated above when the plant has been shut down for more than 20 hours, see Fig. 3. 48 hours after shut down, the extra contribution is at maximum (*HolidayComp*), since we then assume that the building has reached a stationary state, i.e. it will not get any colder even if the shut down lasts longer. The percentage is increased linearly from 0 % at 20 h of shut down time to *HolidayComp* % at 48 h of shut down time.

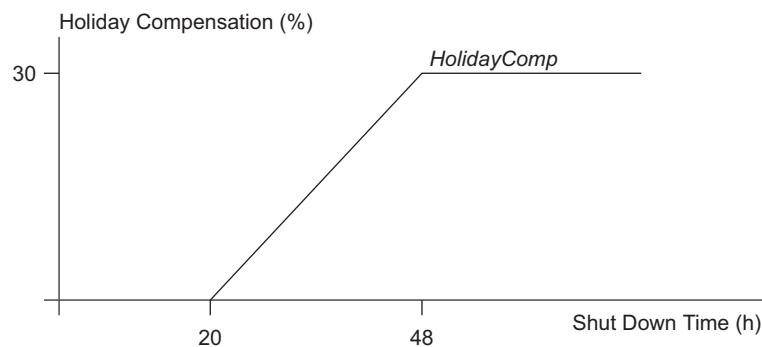


Fig. 21.47: Curve describing the relationship between holiday compensation and shut down time. *HolidayComp* is assumed to be 30% in the figure.

It should be noted that the influence of holiday compensation is more important in systems without a room sensor. The reason for this is that systems with a room sensor will automatically start earlier following a longer shut down, since the temperature in the room will in general have fallen more than during a normal shut down.

Automatic Adjustment of Holiday Compensation

The adjustment of holiday compensation is performed when normal operation begins following an optimisation phase, on the condition that the plant has been shut down for more than 30 h. A necessary precondition is that adjustment has been selected (*AutoAdjust* = 2). It should be noted that the curve points and *HolidayComp* are never adjusted simultaneously following an optimisation phase. The curve points may only be adjusted if the plant has been shut down for less than 20 hours while *HolidayComp* may only be adjusted if the plant has been shut down for more than 30 hours. This also means that no adjustment at all will be performed if the shut down time is between 20 h and 30 h.

Stop-Time Optimization

The reason for using stop time optimisation is to save energy by stopping heating/cooling before the end of the occupancy time. It is of course important that the room temperature does not fall outside an acceptable temperature range during the occupancy time. Since the stop time that may be used without the temperature falling outside the acceptable temperature range depends on the room temperature, optimised stop is nor-

mally only used in systems with a room sensor. The stop time is obtained from a curve that describes the relationship between outdoor temperature and stop-time. The curve gives the stop-time per °C of deviation between the room temperature and the lowest permissible temperature in the room at the end of normal operation, when a room sensor is used. The lowest permissible temperature is set to the set point (*SP*) minus *StopTimeErr* during heating. *StopTimeErr* is by default = 0.5 °C. During cooling, the highest permissible room temperature in the room at the end of normal operation is set to the set point plus *StopTimeErr*. If a room sensor is not used, the optimisation function will assume that the room temperature exceeds the set point by 1 °C in heating operation and is 1 °C below the set point during cooling, cf. start-time optimisation. Both x (*StopLoTemp* and *StopHiTemp*) and y-values (*StopTimeLo* and *StopTimeHi*) for both curve points are freely selectable. The default values are (-10 °C, 0 min) and (10 °C, 0 min) respectively. This implies that stop-time optimisation will not have any effect before these parameters are changed. The reason for this is that stop time optimization is used relatively seldomly used.

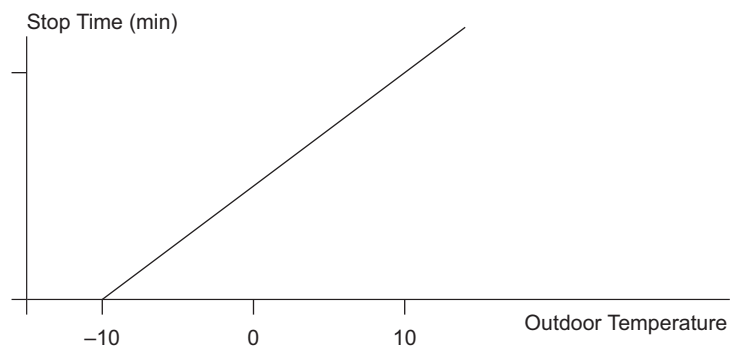


Fig. 21.48: Curve describing the relationship between outdoor temperature and stop-time. The curve does not illustrate default values conditions.

21.38 OR – OR Gate

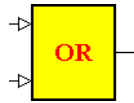


Fig. 21.49:

Table 21.61:

| | | | |
|-------------|--------|--------|----------------|
| Inputs | state1 | BINARY | Input signal 1 |
| | state2 | BINARY | Input signal 2 |
| Output type | BINARY | | |
| Access | RO | | |

Description

Calculates the boolean OR function of state1 and state2, according to the following truth table:

Table 21.62:

| state 1 | state 2 | output |
|---------|---------|--------|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

21.39 OSC – Oscillator

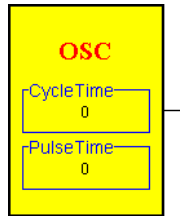


Fig. 21.50:

Table 21.63:

| | | | |
|-------------|-----------|------|------------------------------|
| Parameters | CycleTime | REAL | Oscillation period (seconds) |
| | PulseTime | REAL | Pulse duration (seconds) |
| Output type | BINARY | | |
| Access | RO | | |

Description

This block generates a train of pulses of duration *PulseTime* and a period of *CycleTime*.

The train of pulses is always a multiple of the program cycle time. For example if *CycleTime* is 7.4 seconds and *PulseTime* is 5.2 seconds and the cycle time is 1 second, a 6 second pulse will fire every eight seconds.

21.40 PI – Pulse-Counter Binary Input

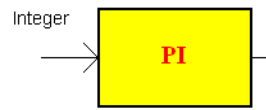


Fig. 21.51:

The PI connection block is used only in a Menta application for the Xenta 700 devices. The block is only available when the Menta programming tool is started from XBuilder.

The block is used to connect signals between Menta objects and pulse counting signals in Xenta I/Os, using XBuilder.

Connection blocks are not connected to physical I/O blocks when the Menta application is edited.

Description

The PI function block counts the number of pulses on the input during one execution of the application program. The ACCUM block can be used to accumulate the number of pulses during execution.

Table 21.64:

| | | | |
|-------------|---------|--|--|
| Parameter | None | | |
| Input type | INTEGER | | |
| Output type | REAL | | |
| Access | RO | | |

21.41 PO – Pulsed Binary Output

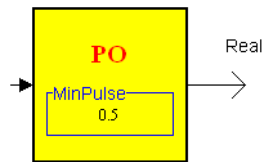


Fig. 21.52:

The PO connection block is used only in a Menta application for the Xenta 700 devices. The block is only available when the Menta programming tool is started from XBuilder.

The block is used to connect signals in Xenta I/O modules, using XBuilder.

Connection blocks are not connected to physical I/O blocks when the Menta application is edited.

The function block is used in XBuilder to generate a pulse width modulated signals on a connected digital pulse output signal in a Xenta I/O module. The PO block is designed to be used together with the PID function block (PIDI).



Important

- The data type for the **Value** signal in a digital physical output, is normally declared as BOOL.
- When the physical output is configured as pulse type, the data type for the **Value** signal is automatically declared as REAL, and allows you to connect the output signal of the PO block to the physical output in XBuilder.

Description

The pulse duration is determined by the block input signal value (in seconds). Negative values are ignored.

The PO function block has a *MinPulse* parameter to specify a minimum duration of a pulse on a connected physical I/O signal.

Block input signals with duration shorter than the *MinPulse* value are accumulated until the *MinPulse* value is reached. Short input signals can only be accumulated to a duration of the program cycle.

Input pulses with durations longer than the program cycle time can create output pulses longer than the program cycle time.

Table 21.65:

| Parameter | Min Pulse | REAL | Minimum output pulse duration (seconds). |
|-------------|-----------|------|--|
| Output type | REAL | | |

Table 21.65:

| | | | |
|--------|----|--|--|
| Access | RO | | |
|--------|----|--|--|

21.42 PIDA – PID Controller – Analog Output

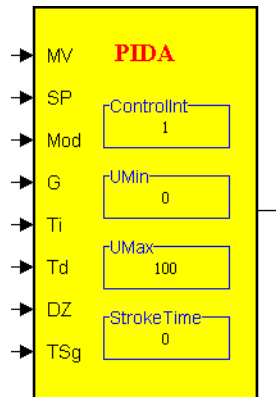


Fig. 21.53:

Table 21.66:

| | | | |
|-------------|------------|---------|--|
| Inputs | MV | REAL | Measured value. |
| | SP | REAL | Set point. |
| | Mode | INTEGER | Controller operating mode. |
| | G | REAL | Proportional gain. |
| | Ti | REAL | Integral time (sec). |
| | Td | REAL | Derivative time (sec). |
| | DZ | REAL | Dead zone |
| | TSg | REAL | Tracking signal (actual value of the previous control signal). |
| Parameters | ControlInt | REAL | Control interval (sec) |
| | UMin | REAL | Minimum permissible control signal. |
| | UMax | REAL | Maximum permissible control signal |
| | StrokeTime | REAL | Actuator full stroke travel time (sec) |
| Output type | REAL | | |

Table 21.66:

| | | | |
|--------|-----|--|--|
| Access | R/W | | |
|--------|-----|--|--|

Description

Control Algorithm

The PIDA block is designed to be used in control loops where the controller output is either connected to an analog physical output or used as a set point for another control loop (cascade control). The control algorithm is a discrete time- incremental PID algorithm, where the change in control signal $du(t)$ is calculated as

$$du(t) = G \cdot \left(e(t) - e(t-h) + \frac{h}{T_i} \cdot e(t) - T_d \cdot \frac{y(t) - 2 \cdot y(t-h) + y(t-2h)}{h} \right)$$

where e is the control error, y is the measured value (MV), G is the controller gain, T_i is the integral time, T_d is the derivative time and h is the control interval (*ControlInt*), i.e. the time between two successive updates of the controller output signal. If *ControlInt* is set to 0, the control interval will automatically be set equal to the program cycle time. Time index t represents the present value of a variable, $t-h$ represents the value at the previous evaluation of the control algorithm, and so on.

The PID-module is executed at the interval hx seconds, where hx seconds is the application program execution interval, even if the selected control interval is longer. The control interval, h , must be a multiple of hx . If this is not the case, the PID algorithm will automatically select the closest multiple smaller than h as the control interval. By default the control interval is set equal to 1.

The control error e is defined as $e = SP - MV$. Thus, if the measured value is below the set point and the gain G is positive, the controller output will increase (heating control). With a negative G value, the controller output will decrease instead (cooling control). When the control error is smaller than the dead zone, i.e. $\text{abs}(e) < \text{DZ}$, the change in the control output, $du(t)$, is set to zero. The dead zone is given in the same units as the measurement value and the set point.

The controller output signal is calculated as

$$(2) \quad u(t) = u(t-h) + du(t)$$

where $u(t)$ is the present control signal and $u(t-h)$ is the previous value of the control signal. The value of $u(t-h)$ is taken from the input *TSg*, which represents the actual *value of the previous control signal*, taking into account any external limitations and/or override functions in the application program. Normally, the *TSg* input would be connected directly to the controller output.

The *proportional band* corresponding to a certain proportional gain value G can be calculated as

$$(3) \quad P_{\text{band}} = \frac{U_{\text{Max}} - U_{\text{Min}}}{G}$$

P and PD Controller

The algorithm described above is used when the controller has integral action. If a controller without I- or D-action is desired, T_i or T_d respectively will be set to 0. In accordance with this, a PI-controller is obtained by setting $T_i = 0$ and $T_d = 0$. If the gain G is set to 0, the program will not fail to execute, but the control signal will not change, regardless of the size of the error.

If the incremental control algorithm in equation (1) is used without integral action, an arbitrarily large stationary error may be obtained which will not necessarily decrease if the controller gain is increased. For this reason, we use a special algorithm for P and PD control, where the control signal is calculated according to eq. (4):

$$(4) \quad u(t) = G \cdot e(t) - T_d \cdot \frac{(y(t) - y(t-h))}{h} + \frac{(U_{\text{Max}} + U_{\text{Min}})}{h}$$

where U_{Max} is the biggest permissible control signal and U_{Min} is the smallest permissible control signal.

Limitation of the Control Signal

The maximum rate of change of the controller output during one control interval, Du_{Max} , depends on the actuator stroke time and can be calculated as

$$(5) \quad Du_{\text{Max}} = \frac{(U_{\text{max}} - U_{\text{Min}}) \cdot h}{\text{StrokeTime}}$$

The calculated change in the control output, $du(t)$, is limited to the range $\pm Du_{\text{Max}}$ before the absolute level of the control signal is calculated. The calculated new control signal $u(t)$ is limited to the interval $(U_{\text{Min}}, U_{\text{Max}})$. If U_{Min} and/or U_{Max} are not defined, the corresponding limitation will not be performed. The parameters U_{Min} and U_{Max} should be given in engineering units. Default values are 0 and 100 (%), respectively.

The parameter *StrokeTime* is used to define the actual full stroke travel time of the actuator. Note that *StrokeTime* may be used to limit the change of the control signal even if the output is not connected to an actuator. The *StrokeTime* is then the minimum permissible time for the control signal to change from U_{Min} to U_{Max} . If the controller output is used as a set point for another controller and there is no special reason for limiting the change in the control signal, *StrokeTime* should be set to 0.

Operating Mode

The controller operating mode depends on the input signal *Mode*, as described in the table below:

- Mode = 0 => Off, controller stopped (du = 0)
- Mode = 1 => Normal control.
- Mode = 2 => Controller output forced to UMax.
- Mode = 3 => Controller output forced to UMin.

If *Mode* = 0, the controller output will track the signal on the tracking signal (*TSg*) input. If *Mode* < 0 or *Mode* > 3, the controller operating mode will be Off (same as *Mode* = 0).

21.43 PIDI – PID Controller – Incremental Output

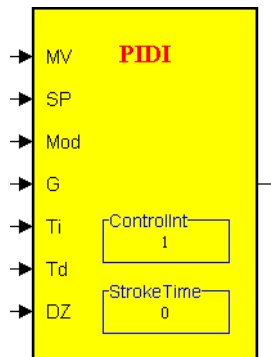


Fig. 21.54:

Table 21.67:

| | | | |
|-------------|------------|---------|--|
| Inputs | MV | REAL | Measured value. |
| | SP | REAL | Set point. |
| | Mode | INTEGER | Controller operating mode. |
| | G | REAL | Proportional gain. |
| | Ti | REAL | Integral time (sec). |
| | Td | REAL | Derivative time (sec). |
| | DZ | REAL | Dead zone |
| Parameters | ControlInt | REAL | Control interval (sec) |
| | StrokeTime | REAL | Actuator full stroke travel time (sec) |
| Output type | REAL | | |
| Access | R/W | | |

Description

Control Algorithm

The PIDI block is designed to be used together with two digital pulse output (DOPU) blocks in control loops with increase/decrease actuators. The control algorithm is a discrete time incremental PID algorithm, where the calculated change (increment) in the control signal is converted to the corresponding travel time of the actuator. The change in the control signal $du(t)$ is calculated using the same formula as in the PIDA block (cf. equation (1) in the PIDA block description).

The calculated change in the control signal, $du(t)$, is converted to the corresponding actuator travel time dt (in seconds) using the following equation:

$$(2) \quad dt = \frac{du(\%)}{100\%} \cdot \text{StrokeTime}$$

where *StrokeTime* is the full stroke actuator travel time (in seconds). Note that it is assumed that *StrokeTime* corresponds to a 100% change in the actuator position. The PIDI block output is the computed dt value.

If *StrokeTime* is set to 0, a stroke time of 60 seconds will automatically be used.

If *ControlInt* is set to 0, the control interval will automatically be set equal to the application program cycle time. However, if the *ControlInt* is longer than the cycle time (hx), we will have to take into consideration that the DOPU block cannot be active more than hx seconds before a new output from the PIDI block is calculated. This means that the output time dt , calculated in equation (2), has to be propagated in pieces that are not longer than hx .

Example: if $ControlInt = 10$ s, $hx = 1$ s and $dt = 5.5$ s, the output from the PIDI block will be 1 s for 5 cycles, then 0.5 s during the sixth cycle and finally 0 s during the following 4 cycles, until it is time to calculate dt again.

The *proportional band* corresponding to a certain proportional gain value can be calculated as

$$(3) \quad \text{Pband} = \frac{100\%}{G}$$

P- and PD Control

A controller with an incremental output does not work very well without I-action. Arbitrarily large steady state errors may be obtained, which do not necessarily decrease when the controller gain is increased. For this reason, a controller with an incremental output without I-action should not be used. However, if a controller without I or D-action is desired, Ti or Td respectively will be to 0. In accordance with this, a PI-controller is obtained by setting $Ti \neq 0$ and $Td = 0$. If the gain G is set to 0, the program will not fail to execute, but the control signal will not change, regardless of the size of the error.

Limitation of the Control Signal

The calculated travel time is limited to the range $\pm ControlInt$ (sec), since that is the maximum amount of actuator travel time that can be obtained during one control interval.

Operating Mode

The controller operating mode depends on the input signal *Mode*, as described in the table below:

- *Mode* = 0 => Off, controller stopped ($dt = 0$)
- *Mode* = 1 => Normal control.
- *Mode* = 2 => Actuator forced to max ($dt = ControlInt$ sec).
- *Mode* = 3 => Actuator forced to min, ($dt = -ControlInt$ sec).

Note that when *Mode* = 2 or 3, the travel time is set to the smallest value that makes the actuator move constantly in the desired direction. If *Mode* < 0 or *Mode* > 3, the controller operating mode will be Off (same as *Mode* = 0).

21.44 PIDP – PID Controller – Analog Output

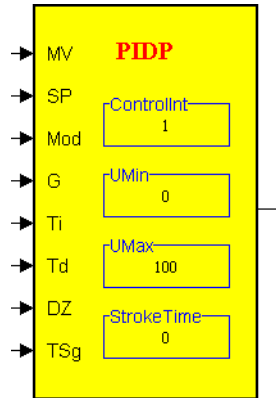


Fig. 21.55:

Table 21.68:

| | | | |
|-------------|-------------|---------|--|
| Inputs | MV | REAL | Measured value. |
| | SP | REAL | Set point. |
| | Mode | INTEGER | Controller operating mode. |
| | G | REAL | Proportional gain. |
| | Ti | REAL | Integral time (sec). |
| | Td | REAL | Derivative time (sec). |
| | DZ | REAL | Dead zone |
| | TSg | REAL | Tracking signal (actual value of the previous control signal). |
| PARAMETERS | ControllInt | REAL | Control interval (sec) |
| | UMin | REAL | Minimum permissible control signal. |
| | UMax | REAL | Maximum permissible control signal |
| | StrokeTime | REAL | Actuator full stroke travel time (sec) |
| OUTPUT TYPE | REAL | | |
| ACCESS | R/W | | |

Description

Control Algorithm

The **PIDP** block is designed to be used in control loops where the controller output is either connected to an analog physical output or used as a set point for another control loop (cascade control).

The control algorithm is a discrete time positional PID algorithm, where the output signal $U(k)$ is calculated according to the diagram below.

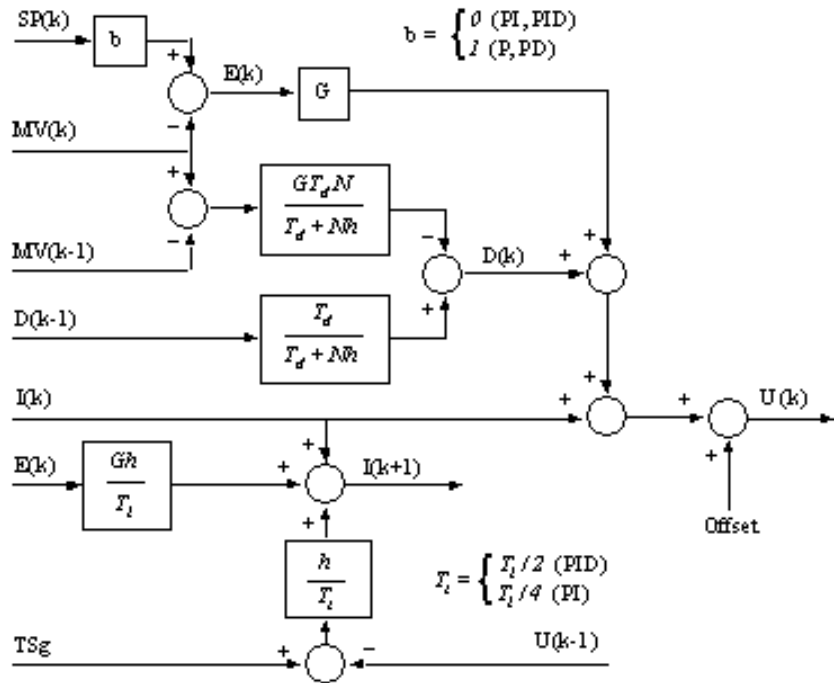


Fig. 21.56: PIDP Block diagram

Here SP is the setpoint value, MV is the measured value, G is the controller gain, T_i is the integral time, T_d is the derivative time and h is the control interval (*Control Int*), i.e. the time between two successive updates of the controller output signal. If *Control Int* is set to 0, the control interval will automatically be set equal to the program cycle time.

Time index k represents the present value of a variable, $k-1$ represents the value at the previous evaluation of the control algorithm, and so on.

The PID-module is executed at the interval hx seconds, where hx seconds is the application program execution interval, even if the selected control interval is longer. The control interval, h , must be a multiple of hx . If this is not the case, the PID algorithm will automatically select the closest multiple smaller than h as the control interval. By default the control interval is set equal to 1.

When the control error is smaller than the dead zone, i.e. $\text{abs}(e) < DZ$, the control signal is set to the same value as the previous output, that is, the output is not changed. The dead zone is given in the same units as the measurement value and the set point.

Other Controller Types

The algorithm described above can be used for all types of controllers. At PI or PID control the P-part does not depend on the difference between the setpoint and the measured value, but only on the measured value, MV .

If P or PD control is used, the P-part will automatically be changed to depend on the error (SP-MV). When these controllers are used, an offset value is also added to the output signal:

$$\text{Offset} = \frac{(U_{max} + U_{min})}{2}$$

where U_{max} is the biggest permissible control signal and U_{min} is the smallest permissible control signal.

If a controller without I- or D-action is desired, T_i or T_d respectively is set to 0. In accordance with this a PI-controller is obtained by setting $T_i > 0$ and $T_d = 0$. If the gain G is set to 0, the program will not fail to execute, but the control signal will not change regardless of the size of error.

Please note that when any of the G , T_i and T_d parameters have been changed, an automatic internal update of the controller is performed. No measures have to be taken in the application to avoid bumps in the control signal.

Limitation of the Control Signal

The parameter *StrokeTime* is used to define the actual full stroke travel time of the actuator. Note that *StrokeTime* may be used to limit the change of the control signal even if the output is not connected to an actuator.

If the controller output is used as a set point for another controller and there is no special reason for limiting the change in the control signal, *StrokeTime* should be set to 0, which is also the default value.

The maximum rate of change of the controller output during one control interval, *DuMax*, depends on the actuator stroke time and can be calculated as

$$DuMax = \frac{(U_{max} - U_{min}) \times h}{StrokeTime}$$

This corresponds to the maximum change in two successive output signals from the PIDP block.

The calculated new control signal $u(t)$ is limited to the interval (U_{min} , U_{max}). The parameters *UMin* and *UMax* should be given in engineering units. Default values are 0 and 100 (%), respectively.

Operating Mode

The controller operating mode depends on the input signal *Mode*, as described in the table below:

Mode = 0 => Off, controller stopped

Mode = 1 => Normal control.

Mode = 2 => Controller output forced to UMax.

Mode = 3 => Controller output forced to UMin.

If *Mode* = 0, the controller output will track the signal on the tracking signal (*TSg*) input. If *Mode* < 0 or *Mode* > 3, the controller operating mode will be Off (same as *Mode* = 0).

21.45 POLY – Polynomial Function

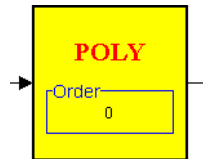


Fig. 21.57:

Table 21.69:

| | | | |
|-------------|----------------------|------|---|
| Inputs | variable | REAL | |
| Parameters | list of coefficients | REAL | List of polynomial coefficients a_0, \dots, a_n |
| Output type | REAL | | |
| Access | RO | | |

Description

This block calculates the polynomial function defined by the expression:

$$p(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \dots + a_1 \cdot x + a_0$$

The order of the polynomial (*n*) is equal to the number of coefficients minus one, and this is indicated in the graphical function block symbol. The maximum number of coefficients is 255.

The polynomial value $p(x)$ is calculated using an algorithm known as Horner's scheme [Reference: Fröberg, Carl-Erik: “*Numerical Mathematics – Theory and Computer Applications*”, Addison-Wesley (1985)] to improve the numerical accuracy and reduce the number of floating

point operations. The algorithm can be described using the following equations, where $p(x) = b_n$:

$$b_0 = a_n$$

$$b_1 = b_0 \cdot x + a_{n-1}$$

$$b_2 = b_1 \cdot x + a_{n-2}$$

....

$$b_n = b_{n-1} \cdot x + a_0$$

21.46 PRCNT – Percentage

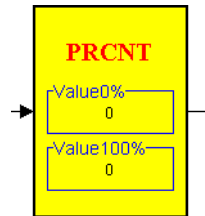


Fig. 21.58:

Table 21.70:

| | | | |
|-------------|-----------|------|--|
| Inputs | variable | REAL | input signal |
| Parameters | value0% | REAL | value of the input variable for 0% at the output |
| | value100% | REAL | value of the input variable for 100% at the output |
| Output type | REAL | | |
| Access | RO | | |

Description

Performs a simple linear transformation of the input signal applying the conversion:

$$\text{output} = 100 * (\text{variable} - \text{value0\%}) / (\text{value100\%} - \text{value0\%})$$

The output value is always limited to be between 0 and 100.

21.47 PULSE – Pulse Generator



Fig. 21.59:

Table 21.71:

| | | | |
|-------------|------------------|--------|------------------------------------|
| Inputs | Trig (t) | BINARY | Trigger signal |
| | PulseLength (pl) | REAL | Duration of output pulse (seconds) |
| Output type | BINARY | | |
| Access | RO | | |

Description

A function block with a mono-stable Binary output of variable pulse length (pulse generator). The pulse length (in seconds) is given by the input signal *PulseLength*. The PULSE block only has one stable output state (0). When the input signal *Trig* switches from 0 to 1, the output switches to the unstable state (1) and remains in this state for *PulseLength* seconds, after which it returns to 0. The output pulse is triggered by the transition of the input signal, from zero to one, and not on its state. Thus, the output pulse length is independent of the duration of the trig input pulse length.

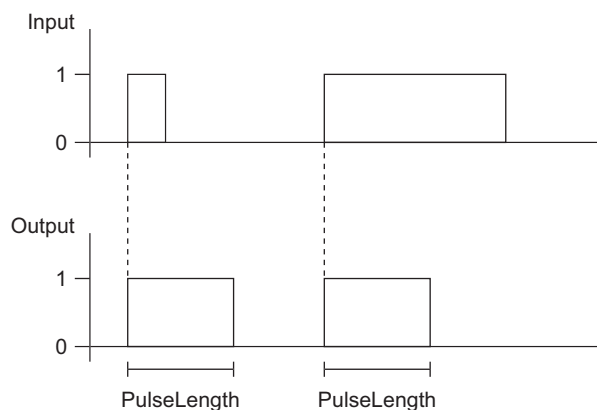


Fig. 21.60:

The output pulse length is always a multiple of the program cycle time e.g. if the *PulseLength* input is set to 5.2 seconds and the cycle time is 1 second, the output pulse duration will be 6 seconds.

21.48 PVB – Binary Value Parameter

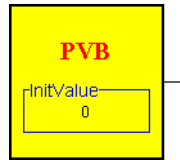


Fig. 21.61:

Table 21.72:

| | | | |
|-------------|-----------|--------|----------------------|
| Parameters | InitValue | BINARY | Initial output state |
| Output type | BINARY | | |
| Access | R/W | | |

Description

This block is used to assign a user-selectable Binary parameter value to the input signal of another block. To be accessible from the OP or via the network, the parameter block output must be declared as Public. The initial state of the output is determined by the *InitValue* parameter.

If the output signal is not public, this block will act as a constant parameter value for another block input, since it is never modified during execution of the application program.

21.49 PVI – Integer Value Parameter

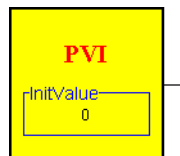


Fig. 21.62:

Table 21.73:

| | | | |
|-------------|-----------|---------|-------------------------------------|
| Parameters | InitValue | INTEGER | Initial value of the output signal. |
| Output type | INTEGER | | |
| Access | R/W | | |

Description

This block is used to assign a user selectable Integer parameter value to the input signal of another block. To be accessible from the OP or via the network, the parameter block output must be declared as Public. The initial value of the output is determined by the *InitValue* parameter.

If the output signal is not public, this block will act as a constant parameter value for another block input, since it is never modified during execution of the application program.

PVI – XENTASYSREG

A special variant of the PVI is available (starting with TAC Xenta system program v 3.61), adding three new functions. These are activated by setting one or several of the three least significant bits in the InitValue of a public PVI block named XENTASYSREG:

- Menu tree without 30 minute reset: Normally the menu tree in the OP panel is reset to display the root Menu root after 30 minutes. By activating this option, the OP panel will not be reset to display the Menu root after 30 minutes.
- Australian daylight saving time option: This will adjust the daylight saving time according to Australian conditions, which are the reverse of European conditions.
- For TAC Xenta 401 the normal propagation speed of two SNVTs per second can be increased to 15 SNVTs/second.



Note

To be able to activate these functions, version 3.61 (or higher) of the system program and the OP Panel program is required.

To activate one or more of these functions, the PVI block should be given the name “XENTASYSREG”. The bits will be set if one or several of the following non-zero values are added and assigned to the InitValue of the “XENTASYSREG” PVI block:

Value = 0. None of the functions are active.

Value = 1. Menu tree without 30 minute reset is activated.

Value = 2. Australian daylight saving time option is activated.

Value = 4. Fast SNVT propagation: up to 15 SNVTs/sec.

Note! This is allowed only in TAC Xenta 401!

Example: InitValue=5 will activate Menu tree without reset and Fast SNVT propagation.

21.50 PVR – Real Value Parameter

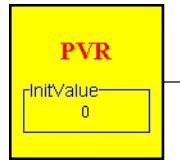


Fig. 21.63:

Table 21.74:

| | | | |
|-------------|-----------|------|------------------------------|
| Parameters | InitValue | REAL | Initial value of the output. |
| Output type | REAL | | |
| Access | R/W | | |

Description

This block is used to assign a user selectable Real parameter value to the input signal of another block. To be accessible from the OP or via the network, the parameter block output must be declared as Public. The initial value of the output is determined by the *InitValue* parameter.

If the output signal is not public, this block will act as a constant parameter value for another block input, since it is never modified during execution of the application program.

21.51 RAMP – Ramp Filter

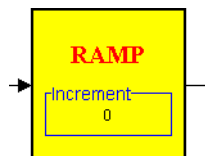


Fig. 21.64:

Table 21.75:

| | | | |
|-------------|-----------|------|---|
| Inputs | variable | REAL | input signal |
| Parameters | increment | REAL | maximum increment in the signal per second. |
| Output type | REAL | | |
| Access | R/W | | |

Description

This filter acts as a rate limit, i.e. it limits the rate of change of the input variable. The maximum increment in the output signal per second is given by the parameter increment. The sign of the increment parameter is ignored and the absolute value is used as a rate limit.

21.52 RI – Real Input

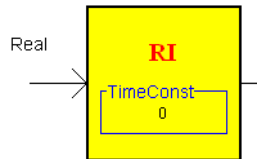


Fig. 21.65:

The RI connection block is used only in a Menta application for the Xenta 700 devices. The block is only available when the Menta programming tool is started from XBuilder.

The block is used to connect signals between Menta objects and to signals in Xenta I/O modules or SNVTs, using XBuilder.

Connection blocks are not connected to physical I/O blocks when the Menta application is edited.

Description

The RI function block has a discrete time first order software filter for filtering a sensor reading. The filter time constant is specified in seconds with the parameter *TimeConst*. The filter algorithm is the same as used for the Linear Analog Input configuration option for the AI function block.

Table 21.76:

| Parameters | Time Const | REAL | Filter time constant (seconds). |
|------------|------------|------|---------------------------------|
| Input type | REAL | | |
| Access | RO | | |

21.53 RO – Real Output



Fig. 21.66:

The RO connection block is used only in a Menta application for the Xenta 700 devices. The block is only available when the Menta programming tool is started from XBuilder.

The block is used to connect signals between Menta objects and to signals in Xenta I/O modules or SNVTs, using XBuilder.

Connection blocks are not connected to physical I/O blocks when the Menta application is edited.

Table 21.77:

| | | | |
|-------------|------|--|--|
| Parameter | None | | |
| Output type | REAL | | |
| Access | RO | | |

21.54 RST – Restart



Fig. 21.67:

Table 21.78:

| | | | |
|-------------|--------|--|--|
| Output type | BINARY | | |
| Access | RO | | |

Description

The output is activated during first program cycle following a warm start.

21.55 RT – Run-Time Measurement

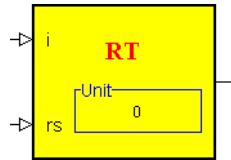


Fig. 21.68:

Table 21.79:

| | | | |
|-------------|-------------------|---------|---|
| Inputs | RunIndication (i) | BINARY | Running indication to be measured |
| | Reset (rs) | BINARY | Reset input (1 = reset) |
| Parameters | Unit | INTEGER | Output time unit (0 = hours, 1 = minutes, 2 = seconds). Default value = 0 (hours) |
| Output type | INTEGER | | |
| Access | R/W | | |

Description

This block is used to accumulate the period of time during which the Binary signal *RunIndication* is activated (true). The output time unit (hours, minutes or seconds) is selected using the *Unit* parameter. The output and all internal block states are set to zero at the initial state or when the *Reset* input is activated. When the *Reset* input is deactivated, accumulation resumes.

When the output reaches the maximum Integer limit (32767), accumulation stops, but the output is *not* reset to zero.

21.56 SECOND – Second



Fig. 21.69:

Table 21.80:

| | | | |
|-------------|---------|--|--|
| Output type | INTEGER | | |
| Access | RO | | |

Description

Provides the current second (0-59), according to the internal time clock.



Note

- The output signal does not change its value during execution of the application program module where it is being used. This is important if longer cycle times than one second are used.

21.57 SEQ – Sequencer

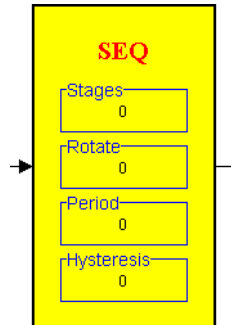


Fig. 21.70:

Table 21.81:

| Inputs | Input | REAL | Input signal (%) |
|-------------|------------|---------|---|
| Parameters | Stages | REAL | Number of output stages (1 to 16). |
| | Rotation | BINARY | Defines whether or not there is rotation of the stages. |
| | Period | INTEGER | Delay of simultaneous activation of the stages in milliseconds. |
| | Hysteresis | REAL | Width of the hysteresis (%). |
| Output type | INTEGER | | |
| Access | RO | | |

Description

This block is used to start N of M stages, where M is the total number of output stages as defined by the *Stages* parameter (up to a maximum of 16) and N is the Integer part of the result of the following calculation:

$$N = \frac{(M + 1) \cdot \text{Input}}{100}$$

where *Input* is a value between 0% and 100%. The M stages are represented by the first M bits of the Integer block output value. The first N of these bits will be true (1) and the rest false (0).

Example: If *Stages* is 4, and neither *Hysteresis* nor *Rotation* is used, stage one will start (Output = 1; 0001) at 20% *Input* signal, stage two (Output = 3; 0011) at 40%, stage three (Output = 7; 0111) at 60% and stage four (Output = 15; 1111) at 80%. Note that TAC Menta uses a two-complement representation of signed integers, i.e. the output signal for starting 16 stages would be -1 (1111111111111111).

If the *Hysteresis* parameter is zero, the previous formula will give the number of activated stages as a function of the input signal. If *Hysteresis* has a non-zero value, a hysteresis loop will exist to the left or to the right (depending on whether *Hysteresis* is a negative or a positive number) of the points calculated in the previous equation. In this case, the activation values of the stages will be displaced with respect to the deactivation values. For example, if we define a 4 stage sequencer, the stages started according to the previous formula will be one stage at 20%, two stages at 40%, three stages at 60% and four stages at 80%. If a positive hysteresis is defined, e.g. equal to 10, activation of the stages will be displaced to the right by 10%, i.e. activation of the stages will be produced at 30%, 50%, 70% and 90%, but deactivation will be maintained at the previous values.

On the other hand, if the hysteresis were negative, for example -15% , deactivation would be displaced to the left and so, the stages would activate at 20%, 40%, 60% and 80% and deactivate at 5%, 25%, 45% and 65%.

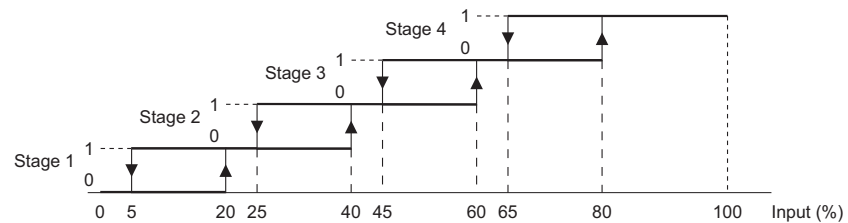


Fig. 21.71: Stages = 4, Hysteresis = -15 .

The *Period* parameter is used to prevent two or more stages from being activated simultaneously. If the value is non-zero, the stages will always be activated sequentially with an interval equal to *Period* (milliseconds), even if the input were to vary sharply, obliging simultaneous activation of the stages. However, if this value is *zero*, the stages may be activated simultaneously.

The *Rotation* parameter determines whether the active stages should rotate or not. The difference between *Rotation* = 1 (with rotation) and *Rotation* = 0 (without rotation) is that, in the first case, the stages will deactivate in the same order in which they activated. The stage which has been active longest will always deactivate first, while in the second case, the deactivation order is the opposite of the activation order. When the sequencer is defined with rotation, you can assume that the time during which each stage remains active will, in the long term, be approximately the same for all stages.

21.58 SHB – Sample and Hold Binary Value

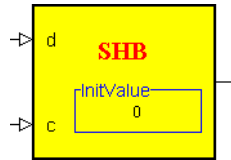


Fig. 21.72:

Table 21.82:

| | | | |
|-------------|-------------|--------|------------------------------------|
| Inputs | state (d) | BINARY | Binary input signal. |
| | control (c) | BINARY | control signal. |
| Parameters | InitValue | BINARY | initial value of the input signal. |
| Output type | BINARY | | |
| Access | R/W | | |

Description

This is a sample and hold function with the following transition table:

Table 21.83:

| state (t) | control (t) | output (t+1) |
|-----------|-------------|--------------|
| 0 | 0 | output (t) |
| 1 | 0 | output (t) |
| 0 | 1 | 0 |
| 1 | 1 | 1 |

The block will copy the input state to the output, if the control signal is active. Whereas, if the control signal is inactive, the output will remain in the same state.

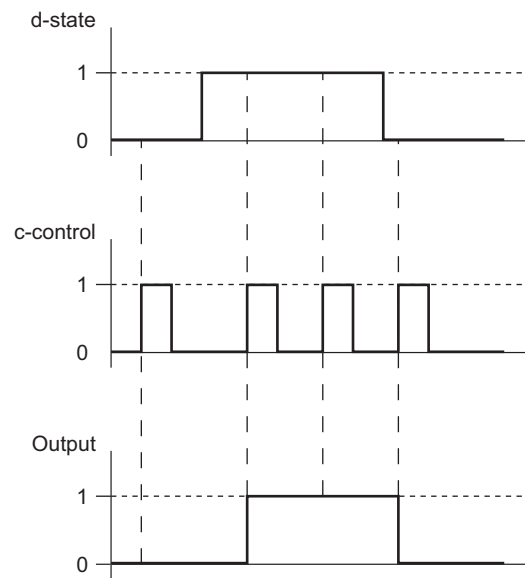


Fig. 21.73:

21.59 SHI – Sample and Hold Integer Value

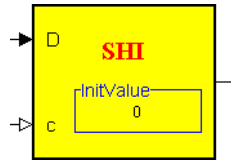


Fig. 21.74:

Table 21.84:

| | | | |
|-------------|--------------|---------|------------------------------|
| Inputs | variable (D) | INTEGER | Analog input signal. |
| | control (c) | BINARY | control signal. |
| Parameters | InitValue | INTEGER | initial value of the output. |
| Output type | INTEGER | | |
| Access | R/W | | |

Description

This block functions in an identical manner to the DELI block while the control input remains active; i.e. delaying the propagation of the input signal for one program cycle. However, while the control signal remains inactive, the output retains the value of the input during the last cycle when the control signal was active.

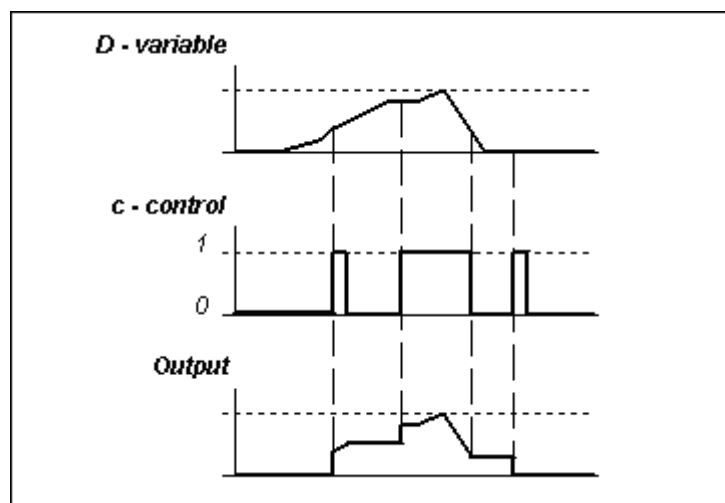


Fig. 21.75:

21.60 SHR – Sample and Hold Real Value

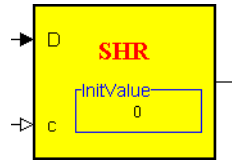


Fig. 21.76:

Table 21.85:

| | | | |
|-------------|--------------|---------|------------------------------|
| Inputs | variable (D) | NTEGER | Analog input signal. |
| | control (c) | BINARY | control signal. |
| Parameters | InitValue | INTEGER | initial value of the output. |
| Output type | REAL | | |
| Access | R/W | | |

Description

This block functions in an identical manner to the DELR block while the control input remains active; i.e. is delaying the propagation of the input signal for one program cycle. However, while the control signal remains inactive, the output retains the value of the input during the last cycle when the control signal was active.

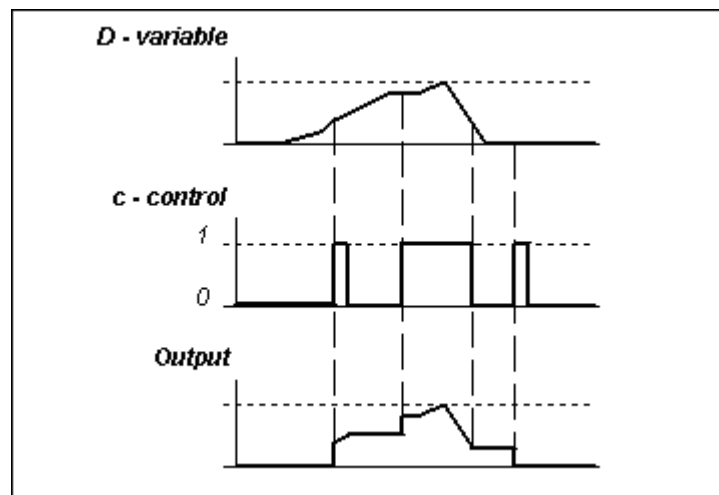


Fig. 21.77:

21.61 SR – Set-Reset Flip-Flop

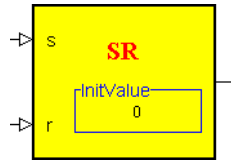


Fig. 21.78:

Table 21.86:

| | | | |
|-------------|-------------|--------|------------------------------------|
| Inputs | Set (s) (D) | BINARY | activates the output state |
| | Reset (r) | BINARY | deactivates the output state |
| Parameters | InitValue | BINARY | initial value of the output signal |
| Output type | BINARY | | |
| Access | R/W | | |

Description

The SR flip-flop is a bi-stable block with two inputs: set and reset. Depending on the value of these inputs at any given moment, the output of the block will be given a value during the next program cycle in accordance with the following transition table:

Table 21.87:

| state (t) | reset (t) | output (t+1) |
|-----------|-----------|---------------------|
| 0 | 0 | output (t) |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | inverse (output(t)) |

Thus, the output at time t+1 will be equal to the input at time t if the two inputs are inactive. If the two inputs are active, the output changes state every program cycle. If only one of them is active, the output will be

activated or deactivated depending on whether the input variable set or reset is active respectively.

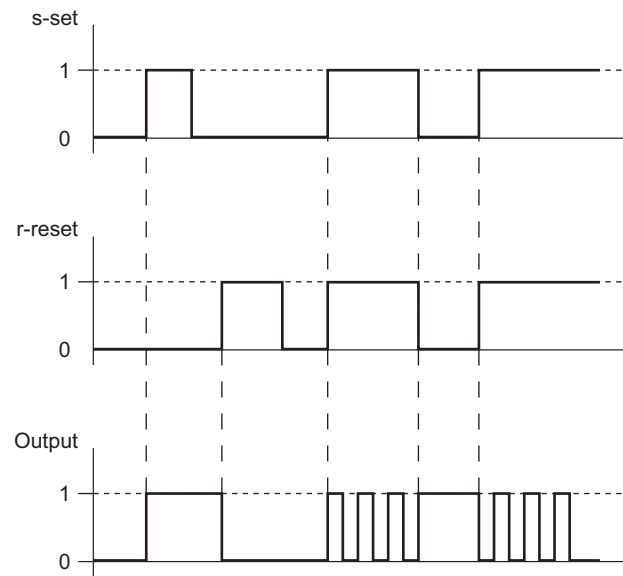


Fig. 21.79:

21.62 STRIN – STR Input

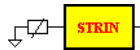


Fig. 21.80:

Table 21.88:

| | | | |
|-------------|-------------------|--------|---|
| Parameters | Wall Module | STRING | Select name of the STR Wall module defined in Device Specification. |
| | Reference | STRING | Select the relevant input signal from the STR. |
| | Initial Value | REAL | Initial output value. Default value = 0. |
| | Application Value | ENUM | For temperature input signals: unit (°C or °F) of value delivered to application. Default value = °C. |
| Output type | | REAL | Output signal to application. |
| Access | RO | | |

Description

The STRIN block handles an input from the STR wall module. Depending on how the STR is configured, different types of signals may be of interest. An application may require several STRIN blocks from one STR wall module.

The block output is updated only once during each application program cycle. Changes in the physical inputs, with a duration less than one program cycle will not be noted by the application program.

In the **Edit block STRIN** window *Unit* is used to determine which temperature unit (when applicable) will be used for presentation in TAC Vista.

The Analog input block transfers a signal from the STR Wall module to the application.

In the **Bind STR Input** window the *Wall Module* parameter you specify the STR module.

Reference: Available input signals from the STR are:

- SpaceTemp (temperature, °C or °F)
- Setpoint (temperature, °C or °F)
- SetptOffset (temperature °C, °F; or °F, no offset)
- OccManCmd (Manual occupancy command)
- AnalogValue (e.g. from a CO₂ or RH sensor, 0-10V=0-100%; occupancy sensor, <10%=closed, >25%=open)

- FanSpeedCmdValue (Fan speed 0 or 1-3)
- FanSpeedCmdState (Fan on/off)

The temperature unit delivered to the application is defined in the field *Application Value*. The input block will produce a numeric value corresponding to the selected unit.

The initial block output value for an STR block (before a signal value has been received from the STR module) is specified by the parameter *Initial Value*. If the STR module goes offline, the block output will keep the last value that was received from the STR module.

21.63 STROUT – STR Output

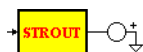


Fig. 21.81:

Table 21.89:

| | | | |
|-------------|---------------------|--------|---|
| Input | Input | REAL | Input signal from application. |
| Parameters | Wall Module | STRING | Select name of the STR Wall module defined in Device Specification. |
| | Reference | STRING | Select the relevant output signal to the STR Wall module. |
| | Initial Value | REAL | Initial output value. Default value = 0. |
| Output type | Block has no output | | |
| Access | RO | | |

Description

The STROUT block handles an output to the STR Wall module. Depending on how the STR is configured, different types of signals may be of interest. An application may require several STROUT blocks.

In the **Edit block STROUT** window, *Unit* is used to determine which temperature unit (when applicable) will be sent from the application to the Wall module *and* be used for presentation in TAC Vista.

The Analog output block transfers a signal from the application program to the STR Wall module.

In the **Bind STR Output** window, the *Wall Module* parameter specifies the STR.

Reference: Available output signals to the STR are:

- SpaceTemp (temperature, °C or °F)

- UserLockout
- EffectSetpt (temperature, °C or °F)
- EffectOccup
- UnitStatus
- OutdoorTemp (temperature, °C or °F)
- SpaceRH
- SpaceCO2
- FanSpeedValue
- TempMinDelta (temperature, °C or °F)
- TempOffset (temperature, °C or °F)
- Resolution
- DispTimeout
- BackLightOn
- Options1
- Options2
- Options3
- SetpointLow (temperature, °C or °F)
- SetpointHigh (temperature, °C or °F)

The *Initial Value* parameter specifies the output signal value (in the relevant engineering unit), which is transmitted to the STR by the base unit at startup, e.g. following a restart immediately after a power outage.

21.64 TCYC – Cycle Time



Fig. 21.82:

Table 21.90:

| | | | |
|-------------|------|--|--|
| Output type | REAL | | |
| Access | RO | | |

Description

Gives the duration in seconds of one program cycle as specified in the Specification Table.

21.65 TRIG – Trigger

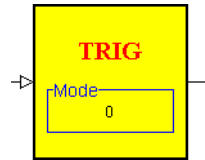


Fig. 21.83:

Table 21.91:

| | | | |
|-------------|--------|---------|---------------------------|
| Inputs | state | BINARY | trigger signal |
| Parameters | mode | INTEGER | operation modes 1, 2 or 3 |
| Output type | BINARY | | |
| Access | RO | | |

Description

The trigger is a mono-stable one which fires when it detects a transition in its input signal and generates a pulse which is equal to the duration of one program cycle. Depending on the mode of operation selected, the trigger may fire:

- Mode 0 and 1: on the low to high transitions.
- Mode 2: on the high to low transitions.
- Mode 3: on any transition.

The mode may be entered as an integer between 0 and 255, but any mode higher than mode 3 will work as mode 3 e.g. the trigger will fire on any transition.

21.66 TSCH – Time Schedule



Fig. 21.84:

Table 21.92:

| | | | |
|------------|----------------------|---------|--------------------------|
| Parameters | Week charts, Max. | INTEGER | Number of week charts |
| | Holiday charts, Max. | INTEGER | Number of holiday charts |

Table 21.92:

| | | | |
|-------------|---------|--|--|
| Output type | INTEGER | | |
| Access | RO | | |

Description

Time scheduling can be used in the Menta function block diagram designed for Xenta 280/300/401 devices, using the TSCH function block.



Note

- An application program for TAC Xenta 280 devices may only contain one TSCH block.

The TSCH function block is not available for programming Xenta 700 devices in the Menta programming tool.

The time scheduling function is used to configure week charts to start and stop, e.g. an AHU at different hours, depending on the day of the week.

It is also possible to define exception schedules, called holiday charts, during which the normal weekly charts are overridden with different operating hours.

The output of the TSCH block is a signed Integer where the sign indicates the time schedule status (negative = true, positive = false). The value of the TSCH block indicates the time left until the next status change (negative = time in minutes to false, positive = time in minutes to true).



Note

- Please note that the *Backup* check box only applies to the block output status, i.e. the time left to the next status change. The week and holiday chart settings are *always* saved in the controller's Flash memory regardless of the Backup setting.

Each *time schedule* can include several *weekly* and *holiday charts*. The *Week charts Max.* parameter defines the number of week charts in the time schedule. Not all week charts need to be defined during the application programming phase. The parameter is used to decide how many week charts should be available in the TAC Xenta OP. Undefined week charts may be defined using the OP at runtime.

The parameter *Holiday charts, Max.* defines the number of holiday charts. Not all holiday charts need to be defined during the application programming phase. The parameter is used to decide how many holiday charts should be available in the TAC Xenta OP. Undefined holiday charts may be defined using the OP at runtime.

The total number of Week and Holiday charts is only limited by the available memory space in the TAC Xenta controller. But, as each Week and Holiday chart uses a relatively large memory space, you should carefully plan how you will use the Time schedules before increasing the Max. parameter value.

Holiday charts can also be defined in TAC Vista via the central time schedule function.

Example

Let's assume that an AHU is to have the following operating hours:

08:00 – 12:00 and 13:00 – 17:30 Monday through Friday

09:00 – 14:00 Saturday

10:00 – 12:00 Sunday

On Christmas Eve, the AHU will be in operation between 15:00 and 16:00, if Christmas Eve is a Monday, Tuesday, Wednesday, Thursday or Friday.

It should also be possible to define 2 extra holidays via the TAC Xenta OP.

The time schedule definition would then be as follows:

Week charts, Max. = 4

Table 21.93:

| Start time | Stop time | Weekdays |
|------------|-----------|-------------------------|
| 08:00 | 12:00 | Mon, Tue, Wed, Thu, Fri |
| 13:00 | 17:30 | Mon, Tue, Wed, Thu, Fri |
| 09:00 | 14:00 | Sat |
| 10:00 | 12:00 | Sun |

Holiday charts, Max. = 3:

Table 21.94:

| Start date | Stop date | Start time | Stop time | Weekdays |
|------------|-----------|------------|-----------|-------------------------|
| *-12-24 | *-12-24 | 15:00 | 16:00 | Mon, Tue, Wed, Thu, Fri |



Notes

- Wild cards in the start date and stop date fields of a holiday chart makes it possible to define a certain day every year. Wild cards are only allowed in holiday charts.
- It is permissible to enter the time 24:00. Operating hours of 00:00-24:00 mean that the controlled equipment will be on for a full twenty-four hour period.
- It is permissible to define operating hours of 23:00-04:00 Mon. This means the controlled equipment will run from 23:00 to 24:00 for the defined weekday (Mon), and continue to run from 00:00 to 04:00 on the next weekday (Tue).
- 00:00-00:00 could be used in holiday definitions to turn the controlled object off for a full twenty-four hour period. Operating hours defined as 03:34-03:34 will be handled as 00:00-00:00, i.e. the controlled equipment will be turned off.

21.67 TSCHI – Time Schedule Block in Xenta 700



Fig. 21.85:

The TSCHI function block is used only in a Menta application for the Xenta 700 devices. The block is only available when the Menta programming tool is started from XBuilder.

Table 21.95:

| | | | |
|-------------|---------|--|--|
| Parameters | None | | |
| Access | RO | | |
| Output type | INTEGER | | |

21.68 VECTOR – Vectorial Curve Function

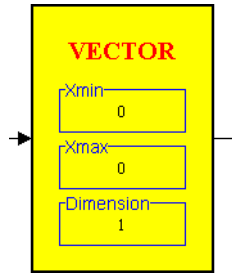


Fig. 21.86:

Table 21.96:

| | | | |
|-------------|------------------|------|--|
| Inputs | Input | REAL | Input signal |
| Parameters | Xmin | REAL | Lower limit of the input range. |
| | Xmax | REAL | Upper limit of the input range. |
| | Dimension (Y(X)) | REAL | List of function values (minimum of 2 and maximum of 255). Each value in the list is entered on a separate line. |
| Output type | REAL | | |
| Access | RO | | |

Description

The VECTOR block permits the definition of any piece-wise linear function for an input signal within the limits Xmax and Xmin. The function is defined by specifying the output function values $y = f(x)$ for N equally spaced input (x) values between the two limits. For input values between two points, the function value is calculated by linear interpolation. If the value of the input signal is less than Xmin, the value defined at that point will be taken, and likewise if the input is greater than Xmax.



Note

- This block can act as a ranged array of Real values for use as a look-up table as in the example below:

Table 21.97:

| Input | Output |
|-------|--------|
| 5 | 34.5 |
| 6 | 28.0 |

Table 21.97: (Contd.)

| Input | Output |
|-------|--------|
| 7 | 42.5 |
| 8 | 33.9 |

To do this, the input must always be an Integer between Xmin and Xmax (Xmin = 5, Xmax = 8 in the example), and the number of elements in the list must be equal to Xmax – Xmin + 1 (8 – 5 + 1 = 4 in the example).

21.69 WDAY – Week Day



Fig. 21.87:

Table 21.98:

| | | | |
|-------------|---------|--|--|
| Output type | INTEGER | | |
| Access | RO | | |

Description

Provides the day of the week, according to the internal time clock. The output value 1 corresponds to Monday and 7 to Sunday.

21.70 XOR – Exclusive OR Gate

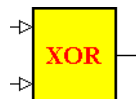


Fig. 21.88:

Table 21.99:

| | | | |
|-------------|--------|--------|--|
| Inputs | state1 | BINARY | |
| | state2 | BINARY | |
| Output type | BINARY | | |
| Access | RO | | |

Description

Calculates the boolean exclusive OR function of state1 and state2, according to the following truth table:

Table 21.100:

| state1 | stat2 | output |
|---------------|--------------|---------------|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

22 Expression Blocks

The expression block is one of the four classes of function blocks in TAC Menta.

Expressions are special blocks with one parameter, which is an arithmetic or logical expression. The parameter can be simple or complex.

There is only one output signal from an expression block.

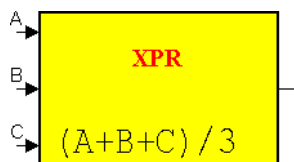


Fig. 22.1:

Depending on the expression, the block can have many inputs or only a few. The graphic representation of the blocks varies in size, depending on the expression and the number of inputs.

You choose the type of expression block according to the required type of output, binary (logical), real, or integer.

Table 22.1:

| Acronym | Short description | Comments |
|---------|-----------------------------------|----------|
| XPB | Expression block. Binary output. | |
| XPI | Expression block. Integer output. | |
| XPR | Expression block. Real output. | |

Expression blocks are always read only and the memory they occupy when compiled depends on the complexity of the expression.

The expression may contain:

Table 22.2:

| | |
|----------------|---|
| Input Variable | You write an input variable as a single letter. A capitalized (“X”) letter represents an analog variable, a lower case (“x”) letter represents a binary variable. The input variables are sorted in alphabetical order on the left side of the block. |
|----------------|---|

Table 22.2: (Contd.)

| | |
|-----------|--|
| Constants | <p>Constants in an expression can be either numericals: “5”, “-13”, “0.03”, “1.25E12”. or alphanumeric: “PI”, “ENERGY”.</p> <p>Alphanumeric constants are identifiers with maximum 20 significant characters and must be defined in the Constants Table. The constants can be entered within quotes.</p> <p>Public constants are not allowed in expression blocks.</p> |
| Operators | <p>Operations such as “-” (change of sign), “*” (multiplication), “+” (addition), “>>” (right shift), “<” (less than) and “a ? b : c” (IF-THEN-ELSE statement: “if a then b else c”).</p> |

22.1 Operands

The Operands in the expressions can be:

Table 22.3:

| | |
|------------------------|---|
| Input Variables | <p>These are declared in the expression using a single letter. An uppercase letter represents an Analog input and a lowercase letter represents a Binary input. It is not permissible to use the same letter in both upper and lower case, e.g. “A” and “a”, in the same expression. The input variables are sorted in alphabetical order on the left side of the expression block.</p> |
| Numeric Constants | <p>The constant is an Integer portion which may be preceded by a symbol: + or -. The Integer portion may be followed by a decimal point (.) and a decimal portion. Finally, an exponential portion may be added: the letters e or E, followed by an Integer with two digits.</p> |
| Alphanumeric Constants | <p>These are alphanumeric identifiers of 20 significant characters which must be defined in the table of constants. The constant name must be between two " characters if a / or : character is included in the name. Public constants are not allowed in expression blocks.</p> |

22.2 Operators

The operands are combined with the operators, described below in order of precedence.

Table 22.4:

| | | |
|-------------------|----------|--|
| Unitary Operators | – , ! | change of sign, logical negation |
| Binary Operators | *, / , % | multiplication, division and modulus |
| | + , – | addition, subtraction |
| | << , >> | left shift, right shift |
| | < , > | less than, greater than |
| | <= , >= | less than or equal to, greater than or equal to |
| | = , != | equal to, not equal to |
| | & | logical AND (bit wise) |
| | ^ | logical Exclusive OR (bit wise) |
| Ternary Operators | | logical OR (bit wise) |
| | ? : | IF-THEN-ELSE statement. “a ? b : c” means “if a then b else c”. |

22.3 Aritmethical Functions

Table 22.5:

| | |
|-----------|--|
| $x^{**}y$ | x raised to the power of y |
| LN (x) | Natural logarithm of x |
| LOG (x) | Base 10 logarithm of x |
| EXP (x) | Exponential e to the power of x |
| COS (x) | Cosine of x (radians), defined for all x |
| SIN (x) | Sine of x (radians), defined for all x |
| TAN (x) | Tangent of x (radians), defined for all x |
| ACOS (x) | Arc cosine of x, where $-1 < x < 1$, gives a result between 0 and $\pi/2$ radians |

Table 22.5:

| | |
|----------|--|
| ASIN (x) | Arc sine of x, where $-1 < x < 1$, gives a result between $-\pi/2$ and $\pi/2$ radians |
| ATAN (x) | Arc tangent of x, defined for all x, gives a result between $-\pi/2$ and $\pi/2$ radians |
| SQRT (x) | Square root of x |
| ABS (x) | Absolute value of x |
| INT (x) | Conversion to Integer (truncation) |

You can use parentheses in an expression to force an evaluation order to differ from the precedence order of the operators.

The three types of variables may always be combined with an operator without problem, since the necessary type conversions are automatically performed according to the rules described below:

- For the operations $+$, $-$, $*$ and $/$, all values are converted to Real values.
- For the operations $\%$, \ll , \gg , $\&$, \wedge and $|$, all values are converted to Integer values (Real values are truncated).
- For the operation $!$, a Real or Integer value is converted to Binary using the following rule: if the value is zero, it will be converted to zero, and if it is non-zero, it will be converted to one before logical negation is carried out. A “true” bitwise Not operation on an integer A can be obtained with the operation $A \wedge (-1)$, i.e. A XOR (-1).
- The operations $>$, $<$, \geq , \leq , $=$, \neq convert the operands to Real values, perform the comparison and return a Binary result.

22.4 Output

The output signal from the expression blocks are:

Table 22.6:

| | |
|-----------|---|
| XPB Block | A binary output. If the result is zero, the output from the block will be zero and if it is any value other than zero, the output will be one. |
| XPI Block | The result is converted to a 16 bit signed integer. |
| XPR Block | The output is a real number, obtained from the evaluation of the expression. |

23 Operators

The operators is also one of the function block classes in TAC Menta.

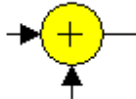


Fig. 23.1: The Addition operator.

Standard symbols are used for their graphic representation.

Several operators in TAC Menta are equivalent to those available in expression blocks.



Important

- You can not use public constants in the operators.

You can design many expressions, used in expression blocks, using operators and connections.

An advantage of designing the equivalent expression using operators is that you can view intermediate results (signals) in the function block diagram.

An operator's analog inputs can be connected to a Real or an Integer output.

Analog outputs from operators can be connected to both Real and Integer inputs. When an analog output of an operator is connected to an integer input, the operator output is converted to an integer.



Note

- If the same operator output is connected to more function blocks, these inputs will receive an integer signal. An additional conversion might be needed.

Binary inputs may only be connected to Binary outputs.

You can use operators to convert signals between real and integer, and to convert between analog and binary signals.

In this chapter, the available operators are divided into groups, similar to the simple blocks:

- Constants
- Logical operators
- Math operators
- Comparison operators
- Bit operation operators
- Other operators

23.1 Constants

Table 23.1:

| Name | Comments |
|---------------|---------------------------|
| Binary const | No input → Binary output |
| Integer const | No input → Integer output |
| Real const | No input → Real output |

Public constants are not allowed in constant operators.

23.2 Logical Operators

Table 23.2:

| Name | Comments |
|------|-------------------------------|
| NOT | Binary input → Binary output |
| AND | Binary inputs → Binary output |
| OR | Binary inputs → Binary output |
| XOR | Binary inputs → Binary output |

23.3 Math Operators

Table 23.3:

| Name | Comments |
|-------------|-------------------------------|
| Negate | Analog input → Analog output |
| Addition | Analog inputs → Analog output |
| Subtraction | Analog inputs → Analog output |
| Product | Analog inputs → Analog output |
| Division | Analog inputs → Analog output |
| Module | Analog inputs → Analog output |

23.4 Comparison Operators

Table 23.4:

| Name | Comments |
|------------------|-------------------------------|
| Less than | Analog inputs → Binary output |
| Greater than | Analog inputs → Binary output |
| Equal | Analog inputs → Binary output |
| Not equal | Analog inputs → Binary output |
| Greater or equal | Analog inputs → Binary output |
| Less or equal | Analog inputs → Binary output |

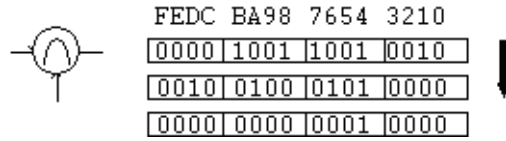
23.5 Bit Operation Operators

Table 23.5:

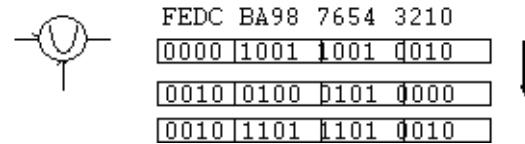
| Name | Comments |
|-------------|-------------------------------|
| bit AND | Analog inputs → Analog output |
| bit OR | Analog inputs → Analog output |
| bit XOR | Analog inputs → Analog output |
| Shift right | Analog inputs → Analog output |
| Shift left | Analog inputs → Analog output |

Examples

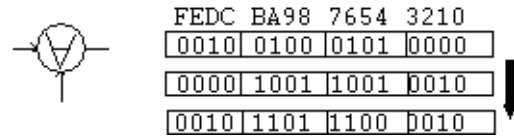
Operator bit AND



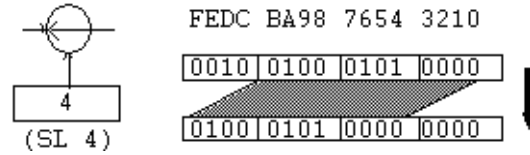
Operator bit OR



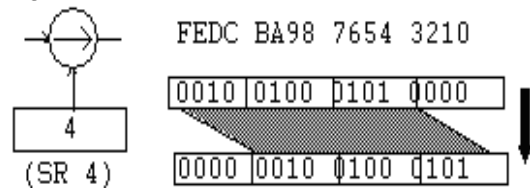
Operator bit XOR



Operator <<



Operator >>



Note

All 16 positions are shifted, and 0 is shifted into empty positions. Any shift count 16 always gives the output value 0.

23.6 Other Operators

Table 23.6:

| Name | Comments |
|--------------------|---|
| D/A converter | Binary input → Analog output |
| A/D converter | Analog input → Binary output |
| Analog multiplexer | Analog inputs → Analog output |
| Binary multiplexer | Binary inputs → Binary output |
| Conversion AA | Integer signal → Real signal or Real signal → Integer signal. |

In the expression block, operands of different types may be combined using any operator. However, when using operator blocks, conversions should be performed explicitly.

The D/A and A/D operators are simply operators that convert from Binary to Analog and vice versa. Those operators are needed because the rest of the operators have typed inputs so, without them, you would not, for example, be able to sum a Binary and an Analog signal. In expression blocks, conversions are done implicitly so you won't need these operators in expressions. The conversion rules are as follows:

- A Real or Integer number is converted to a Binary value giving a logical 0 if the input is exactly 0, or 1 if it is different.
- A Binary signal is converted to Analog as expected: logical 0 gives the number 0 and logical 1 gives the number 1.

The multiplexer operators are switches which select one of two input signals, depending on the Binary switch value.

The Conversion AA operator is used when connecting an integer output to a real input or vice versa. Since operators do not distinguish between Real and Integer values, the Conversion AA operator does not carry out an explicit type conversion with rounding or truncation. Instead, this type conversion is done in the block using the Conversion AA output signal as the input. We recommended using expression blocks when explicit type conversions with rounding or truncation are needed.

24 Test Probe Blocks

A model that describes the dynamics of the controlled system can be programmed using Menta function blocks. The model can be used for closing a control loop.

Test probe blocks are used for sampling the state of the physical outputs. These blocks are also used to write the calculated values to the physical inputs when using a model of the controlled system during a simulation.

24.1 Overview

The function blocks for the model are temporary included in the function block diagram for the simulated application.



Fig. 24.1: A TPAO – Analog output test probe.

The model is applied to the application under simulation, using TPAO/TPDO test probe blocks to read the physical outputs in the application. Feedback from the modelled response is entered to the physical inputs in the application using TPAI/TPDI test probe blocks.



Important

- You can not download applications containing test probes to the TAC Xenta device. The test probes must be removed before downloading.

The simulated application reads the physical input blocks and controls the physical outputs.



Important

- A test probe block samples or gives a value to a specific I/O block by having same identifier (name) as the I/O block.

When you simulate an application, you can not change values for physical inputs using the input buttons if the inputs are connected to test probe blocks. You can only read the values of the inputs.

There are four different test probe blocks, one for each of the physical I/O block types:

Table 24.1:

| Acronym | Short description |
|---------|-------------------------------|
| TPAO | Test probe for Analog output |
| TPDO | Test probe for Digital output |
| TPAI | Test probe for Analog input |
| TPDI | Test probe for Digital input |

24.2 TPAI – Test Probe for Analog Input



Fig. 24.2:

Table 24.2:

| | | | |
|-------------|------------------------------------|------|--------------|
| Inputs | Input | REAL | Input signal |
| Output type | Block has no output | | |
| Access | Block used only during simulation. | | |

Description

The TPAI block writes a value to the Analog input block which has the same name (identifier) as the test probe block. The TPAI block may be used with all types of Analog input blocks. The value of the Analog input block will be set to the value of the input to the corresponding TPAI block, i.e. no scaling is carried out.

24.3 TPAO – Test Probe for Analog Output



Fig. 24.3:

Table 24.3:

| | | | |
|-------------|------------------------------------|--|--|
| Inputs | Block has no inputs | | |
| Output type | REAL | | |
| Access | Block used only during simulation. | | |

Description

The TPAO block reads the value of the Analog output block which has the same name (identifier) as the test probe block. The output of the TPAO block is set to the same value as the input value to the Analog output block.

24.4 TPDI – Test Probe for Digital Input



Fig. 24.4:

Table 24.4:

| | | | |
|-------------|------------------------------------|--------|--------------|
| Inputs | Input | BINARY | input signal |
| Output type | Block has no output | | |
| Access | Block used only during simulation. | | |

Description

The TPDI block writes the value of Input to the Digital input block which has the same name (identifier) as the test probe block.

TPDI can also be linked to a CNT block. In this case, only one pulse per application program cycle can be counted. The pulses are counted on high to low transitions.

24.5 TPDO – Test Probe for Digital Output



Fig. 24.5:

Table 24.5:

| | | | |
|-------------|--|--|------------------------------------|
| Inputs | | | Block has no inputs |
| Output type | | | Block has no output |
| Access | | | Block used only during simulation. |

The TPDO block reads the status of the Digital output block which has the same name (identifier) as the test probe block.

TPDO can also be linked to a DOPU block. In this case, the DOPU parameter MinPulse must be set to the same value as the application program cycle time (sec), otherwise simulation may give incorrect values.

25 The Menta Application File

During creation of a complete application program for Xenta 280/300/401 devices, a number of files containing data are involved.



Important

- A control application for a Xenta 700 is finalized using an XBuilder project.
- The complete application for an XBuilder project is normally saved into the TAC Vista database.

25.1 Data Files

The following data files are involved when creating a Menta application for a Xenta 280/300/401 device.



Note

- All files related to a TAC Menta application must be located in the same directory.

Table 25.1:

| | |
|------|---|
| .AUT | Text file with application program source code and data including graphical block diagram. |
| .TLG | The trendlog definition. |
| .MCB | Macro block file (Group of blocks). |
| .COD | “Machine code” file in ASCII format for downloading to the TAC Xenta controller. |
| .XLG | The compiled trendlog definition. |
| .ESP | Specification file with public signals, their attributes etc. generated by TAC Menta. This file contains input data to the OP configuration tool. |

Table 25.1: (Contd.)

| | |
|-----------|--|
| .OPE | A file with an empty tree, i.e. an OP tree without a specification file. This is the kind of file that is created when the OP configuration tool is run from the program manager. |
| .OPC | A mix of an .OPE and an .ESP file. This file is created by the OP configuration tool when it has been invoked from TAC Menta. |
| .BIN | Binary file with OP menu tree data for downloading to the TAC Xenta controller together with the .COD file. |
| .CHR | National character set file for downloading to the TAC Xenta controller together with the .COD file. |
| .XIF | External interface file containing a standardized description of all network variables/objects of the application program. The .XIF enables binding and communication with LonWorks nodes from other manufacturers. |
| .MTA | <p>Menta project file, which can include all files (all types mentioned above except for .MCB and .OPE) related to a certain application. Once a file (for example .AUT or .OPC) has been inserted (saved) in the project file, it can be extracted (opened) anytime as an .MTA file.</p> <p>The .MTA file can be stored directly in the TAC Vista database.</p> <p>Note</p> <p>The .MTA file can be stored in the TAC Vista database even though it can not be compiled. An .MTA file which is not compilable will not, of course, be possible to download into a TAC Xenta device. It must be correct and compilable before download is possible.</p> <p>In addition to the files listed above, the following data files are involved in the creation of the .BIN file. These files are placed in a specific directory for temporary files.</p> |
| OPDOC.GOP | An ASCII file created by the OP configuration tool when the Generate command is executed. This file is converted to an .SPT file, which in turn is converted to a .BIN file. |

Table 25.1: (Contd.)

| | |
|-----------|---|
| OPDOC.SPT | Intermediate ASCII file generated from the .GOP file. |
| OPDOC.TMP | Intermediate ASCII file generated from the .GOP file. |

Upon downloading, the following file is temporarily created as well:

Table 25.2:

| | |
|-----|--|
| BPR | The Network Neighborhood file, describing the TAC Xenta device's surrounding network nodes and groups. |
|-----|--|

25.2 Renaming a TAC Menta application File

The TAC Menta Project file (the .MTA file) contains a number of files where all the included files use the name of the .MTA file.

You can not rename the .MTA file using a file administering tool. Doing so will cause the .MTA file to be unusable.

To rename the application file, open the file in TAC Menta and save the file using another name.

26 Error Messages

TAC Menta displays a number of information, warning and error messages. In this section, the possible causes of some of these messages are listed as well as some hints about how to find and correct the errors.

26.1 System Errors

System out of memory, Out of memory, Compiler out of memory, API error, No timer allocated, Printing error

All of these error messages can occur if system resources are too low, although other errors may cause some of them. Close all other programs and/or restart Windows.

Type description file TATYPE.INI not found

Program files have been deleted or moved from the TAC Menta directory. Re-install TAC Menta.

26.2 FBD Compilation

During FBD compilation, i. e. when going from Edit mode to Simulation mode in the TAC Menta main program, there is a syntax error check. The faulty block will be marked and centered.

Sensor type 0..1V may give poor resolution

Not an error, just a warning that the default setting of the Sensor field in the AI – Linear analog input parameter hasn't been altered. In most cases, 0..10V or 2..10V are better choices.

Unconnected input

The inputs to blocks may not be left unconnected. When this error appears, find the block in the selection rectangle and connect the input(s) that are unconnected.

Illegal closed loop has been detected

Feedback loops are permitted in the programming language. However, at least one RW block, i.e. a block with an internal delay between the input and the output signal, must exist in the loop. If all of the blocks that constitute the feedback loop are RO, this error message will appear. The selection rectangle will be positioned over one of the blocks in the

feedback loop. The solution to this type of error is to introduce an RW block, typically a delay block, into the feedback loop.

Output is connected to incompatible inputs

This error only occurs when an operator with an analog output is simultaneously connected to a real input and an integer input. To resolve this error, we recommend duplicating the operator and connecting each of the operators to a single input.

The physical IO requirements exceed the capabilities of the chosen device.

This error occurs when the same connection has been used for several blocks.

Redefined identifier

This error occurs when two blocks whose output have been defined as a public signal have the same identifier. To correct this error, change the name of one of the blocks.

Undefined constant

This error occurs when a constant identifier is used in a block or blocks without having been defined in the Constants table.

Block must have a name

This error occurs when a public block output signal has not been given a name (identifier).

Public constant not supported in expression blocks

Public constants are not permitted in expression blocks. The solution is to move the constant to a PVB/PVI/PVR block and connect it to the expression block via an input (or use a non-public constant).

Public constant used more than once

Public constants may not be used in several block parameters at the same time. The solution is to define different constant names for each block parameter (if they have to be public), or use a non-public constant.

Public constant identifier is in use already

A signal identifier must not exist simultaneously as a public constant and public signal. The constant must either be renamed or made non-public.

Public constant not used

You are not allowed to define a public constant and then not use it as a block parameter. The public constant must either be renamed or made non-public.

I/O terminal reused in I/O block

Either two I/O blocks use the same terminal reference, or one I/O block has no terminal reference.

Node name too long

The name of the node is too long. For information on syntax requirements, see Chapter 11.2, “Signal Names”.

The number of backslashes must be 2 or 3

The number of backslashes is wrong. Only 2 or 3 backslashes are acceptable. For information on syntax requirements see Chapter 11.2, “Signal Names”.

26.3 Saving and Loading the Application Program in the Database

Could not save .mta file

The mta file could not be saved in the TAC Vista database.

Failed to update the database (reserve after inject)

The application program could not be saved in the TAC Vista database.

Could not load mta file

File could not be opened from the TAC Vista database

The application could not be opened. Check that TAC Vista Server is working properly.

File object is locked by another application

Another application (or user) is currently working on the file and it is not available.

Could not read temporary file

When working with an application program, a temporary file is used. This error indicates that the temporary file is damaged or corrupted.

File object contains no actual MTA file

This error can occur when saving an empty application program in the TAC Vista database.

Unable to save file. The application contains <no> blocks, but only <no> are allowed

This error occurs when trying to save an application program which contains more than 4,000 blocks.

26.4 Simulation

Division by zero, Numerical overflow

Logical application program error. Reconstruct your FBD.

Execution stopped, the block named: <name> and type: <type> passed its limit

Error message displayed when the limit *Stop at a limit* (breakpoint) is reached during a simulation.

26.5 Code Generation

Symbol table full, Value doesn't fit in byte

In some cases, an application contains too many signals which disables generation of the OP menu tree. Start the *OP Configuration Tool* and remove parts of the menu tree.



Note

If selected, remove the “Automatic generation of menu tree” selection in the **Preferences** menu, or the new menu tree will be overwritten during the next OP tree generation.

Software error parsing output file, Software error parsing generated file

TAC Menta was unable to run through the code generation process completely, although in most cases there are no FBD errors. Try again!

An FBD with test probe cannot be executed online, Can't generate a COD file with test probes

An application program containing test probes has not been allowed to be generated and downloaded. This error message may also be shown if an error that has no specific error message, occurs during code generation.

Parser code error

The application may be using too much memory. Please contact the TAC Helpdesk.

The maximum number of Alarm blocks is exceeded in the application. The problem must be solved before the application can be generated

The application contains more than 127 Alarm Blocks. The number of Alarm Blocks must not exceed 127.

The TACSnvtCfg.dll was not instantiated successfully! It will not be possible to use SNVTs in the applications.

The file TACSnvtCfg.dll has been removed from C:\Program Files\Common Files\TAC Shared. This is probably due to an uninstallation. To have access to SNVTs in the application, reinstall TAC Menta.

This SNVT could not be used because all engineering units are not available for it.

The SNVT in question is not available in Taxif.ini. Please contact the TAC helpdesk for information.

26.6 Download

If an error occurs during download to the TAC Xenta device, TAC Menta will display an error message and open the .COD file with the suspected error marked. The error code number indicates the section in the .COD file where the download was interrupted. This information may be useful when trouble-shooting.

Update of description of network neighbourhood from TAC Vista Server is not performed

This error occurs when performing a download using the serial interface (RS-232) on a TAC Xenta device. The error indicates that network neighborhood is not updated in the same way as when downloading using TAC Vista.

26.6.1 Error Codes

Table 26.1:

| No. | Error in section | Comment |
|-----|------------------|---------|
| 1 | Create object | |
| 2 | Allocate flash | |
| 3 | I/O module save | |
| 4 | Version | |
| 5 | Application name | |
| 6 | Abbreviation | |
| 7 | Type | |
| 8 | Source file | |
| 9 | Signals | |

Table 26.1: (Contd.)

| No. | Error in section | Comment |
|-------------------|------------------|--|
| 10 | Alarm texts | |
| 11 | Time schedule | |
| 12 | I/O module | |
| 13 | I/O | The terminal may not exist in this unit type |
| 14 | Cycle time | |
| 15 | Code | |
| 16 | DST | |
| 17 | Checksum | |
| 18 | EOF | |
| Memory allocation | | The application is too big |

26.7 TAC Xenta Communication

Communication time-out, Time-out, Unknown error from remote unit

Communication problems; the TAC Xenta unit is not answering. Try again.

Communication failed, Upload failed, Download failed

The TAC Xenta unit may have stopped communicating. Try again or reset the TAC Xenta unit manually. Before this error message, there might have been another one, further explaining the problem.

Error restarting unit

The TAC Xenta unit was too busy to answer a question. Try again.

27 Programming Hints

27.1 Program Cycle Time

The cycle time for the application program is variable. Take this into account when drawing your application program.

Example: The INTEG block can be used for calculating Energy consumption from the input Power. When resetting the block at the start of a new calculation interval, the reset value must be the input signal multiplied by the program cycle time.

27.2 Time Counter

The time counter counts in seconds, which takes care of the actual application program cycle time. The counter is reset by a Binary signal.

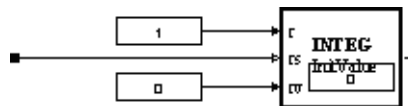


Fig. 27.1:

Time counters can also be made by means of the PULSE and RT blocks. Designs using RT and INTEG can be reset during the count. The INTEG counter can be used generally, whenever counting (time, power etc.) over time.

27.3 Equality

If testing whether two values are equal, ensure that the tested values are of the Binary or Integer type. Be sure to not carry out an equal test on Real values.

27.4 Reset Counter

To reset, for example, an energy counter without losing any pulses during the reset:

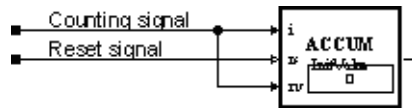


Fig. 27.2:

27.5 Bitwise Logical Operators

The output of, for instance, the bitwise AND (&) operator is an Integer which represents the result of the AND operation on two Analog signals. By using this operation inside an expression block with a Binary output and assigning the result of the AND operation to the output, you obtain a Binary signal which will be zero if there is no match and one as soon as there is a match in at least one bit position.

27.6 Multiplier Parameter in CNT Block

The internal multiplier parameter in the CNT block cannot be assigned to a public constant since it is a configuration parameter. If you want to be able to change this value from the operator's panel, you will have to make a special program to take care of the conversion from the number of pulses to engineering units by other blocks.

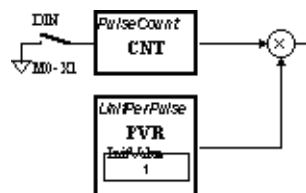


Fig. 27.3:

27.7 Sliding Average Value

Programming example of the sliding average of three values:

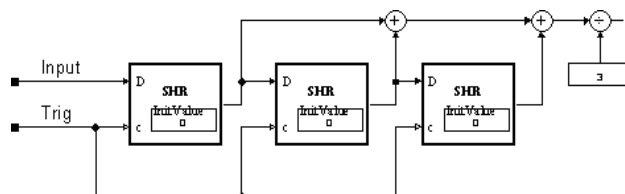


Fig. 27.4:

27.8 TSCH Output

The output from the time schedule block is an integer, showing the remaining number of minutes to the next change of state of the time schedule. For making a Binary signal, there are different possibilities:



Fig. 27.5:

27.9 PIDI – DOPU

Normal use of the PIDI and the DOPU block;

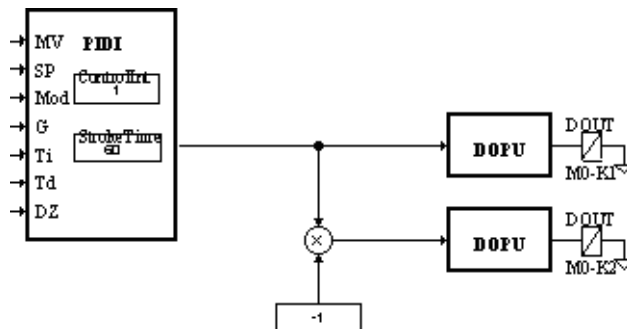


Fig. 27.6:

27.10 Day Shift

It is possible to detect a day shift, if the new hour has a lower value than the previous hour.

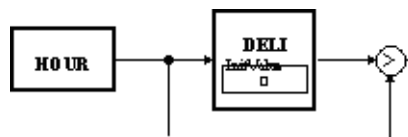


Fig. 27.7:

27.11 Expression Blocks

You can not change the output type of an expression block when the output is connected to a node.

Expression blocks can be used to convert from a Real signal to an Integer signal and vice versa:

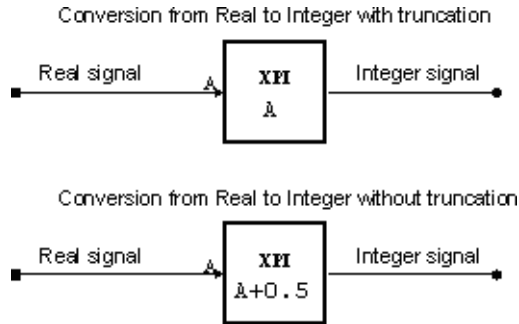


Fig. 27.8:

27.12 Start-Up Delay

Use the RST block to create a start-up delay following a warm start of the TAC Xenta device. Set different delays for different AHUs etc. by altering the DelayOff time (60 s in the example):

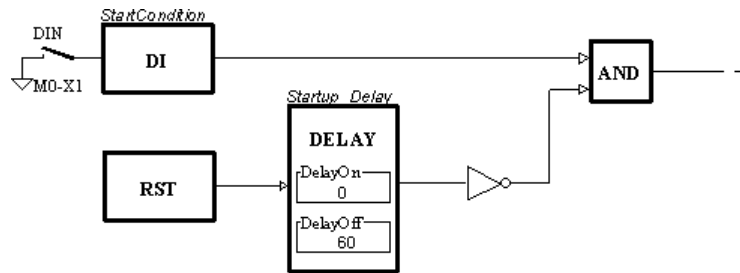


Fig. 27.9:

27.13 Using the SNVT_reg_val in a TAC Xenta 700 Device

The new Menta Object in the Xenta700 series differs regarding SNVT's. All SNVTs are connected outside the Menta Object using XBuilder connection rules.

For example, the SNVT reg_val and SNVT reg_val_ts must be treated specially, since they include a 32-bit data field "raw". Normally Menta integers are signed 16-bit. To enable input of 32-bit data we use the function block II in *Mode 1* to read the most significant word MSW (most significant 16 bits) of the connected input signal, and the function block II in *Mode 2* to read the least significant word LSW (least significant 16 bits) of the connected input signal.

A Menta application example of how to convert the two words to a real (floating point) value is shown in Fig. 27.10.

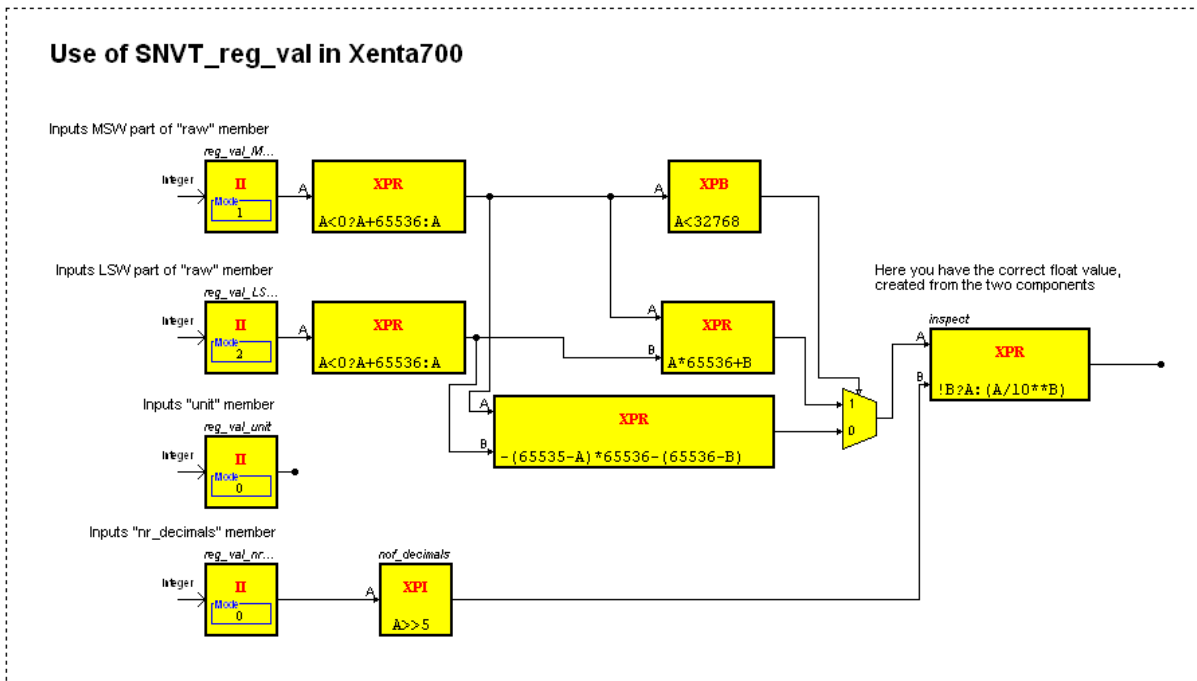


Fig. 27.10:

The application as a Menta object in XBuilder looks like Fig. 27.11.

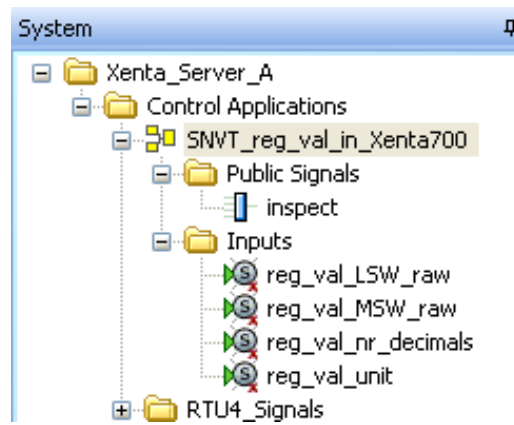


Fig. 27.11:

The SNVT reg_val in XBuilder looks like Fig. 27.12.

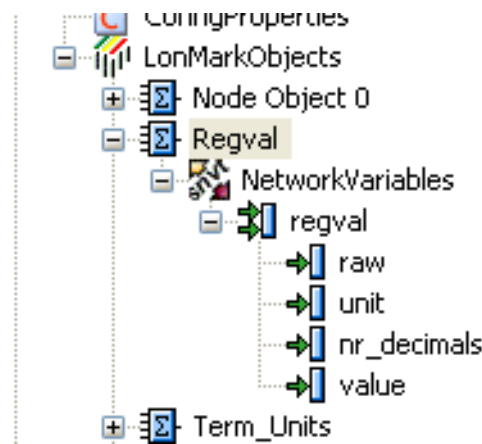


Fig. 27.12:

To make the application to work, connect to the SNVT to the signals in the Menta object as shown in Fig. 27.13.

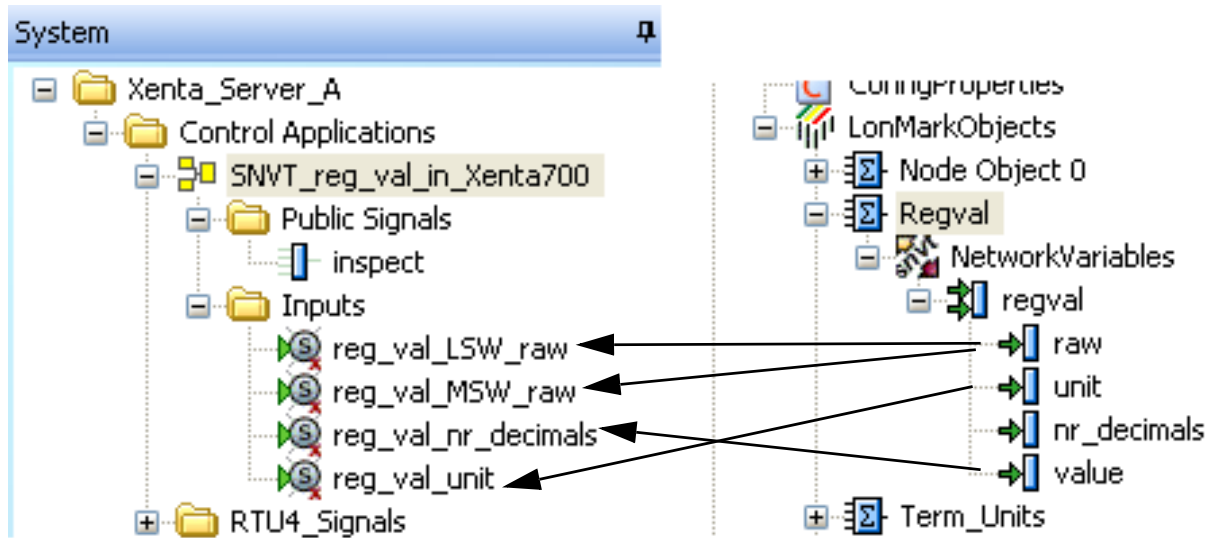


Fig. 27.13:

The required floating point value is available in the public signal named "inspect".

Index

A

- Accessing Constants 174
- Accessing Signals 98
- ACCUM 285
- accumulator 285
- Adding a Constant 175
- AHYST 286
- AI 287
- ALARM 293
- alarm 293
- analog hysteresis 286
- analog input 287
- analog output 296
- analog waves
 - amplitude 197
 - average value 197
 - circular 197
 - period 197
 - phase 197
 - wave form 197
- AND 295
- AO 296
- application program documentation, print 247
- application program ID 88
- associated text file 90
- associated text files 188

B

- B 20
- binary hysteresis 321
- binary sequences
 - circular 199
 - period 199
 - sequence 198
- binary value delay 304
- binding parameters 174
- binding parameters, modifying in online mode 239
- bits per second 228
- bitwise logical operators 402
- block parameters, modifying 202
- BPS 228

C

- character set file, defining 267
- CNT 300
- cold start 20
- COM pause 228
- COM port 227
- compatibility
 - AUT files 276

- COD files 276
- OP menu tree files 277
- TAC Menta v3 276
- Configuring a Trend Log 182
- connection
 - highlight 194
- connections 93, 142
 - break 148
 - delete 147
 - highlight 150
- constants 93
- correct operation, verify 278
- create COM log 228
- Creating a Hierarchical Function Block 72
- Creating Hierarchical Function Block in Levels 73
- CURVE 301
- curve function 301
- cycle time 369

D

- DATE 302
- date and time 187
- day 302
- day shift 403
- defining character set file 267
- DELAY 303
- delayed On/Off 303
- DELB 304
- DELI 304
- DELR 305
- destination blocks 93
- Device Configuration
 - Use LonMark 3.3 118
- device configuration
 - add new network variables (SNVT) last in the XIF file 118
 - base unit type 116
 - hardware version 116
 - system version 116
 - XIF header generated according to LonMark 117
- DeviceDescr folder 27
- DI 305
- digital input 305
- digital output 310
- digital pulse output 312
- disconnect block 145
- DO 310
- Documentation folder 27
- DOPU 312
- download wizard 273
 - dialog 274
 - general downloading procedure 275

E

- edit 112
 - block 137
 - center selection 112
 - find 112
 - replace 112
- edit mode 87, 111, 115
- Editing a Constant 175
- ENTH 313
- enthalpy 313
- equality 401
- ERR 316
- Error Block 316
- Error Block codes 317, 319
- exclusive OR 375
- executable code
 - BIN 207
 - CHR 208
 - COD 207
 - ESP 207
 - generate 207
 - OPC 207
 - XIF 207
- executable files, simulation 205
- Expanding an Hierarchical Function Block 73
- expression blocks 174, 404

F

- FB 20
- FBD 20, 93, 112
- feedback loop 93
- File menu
 - Exit 268
 - Export 268
 - Import 268
 - New 268
 - Open 268
 - Print 268
 - Print Preview 268
 - Print Setup 268
 - Save 268
 - Save As 268
- FILT 320
- first order filter 320
- Formats menu
 - Alarms 269
 - Character Set File 269
 - Date and Time 269
 - National Months Text 269
 - National Week Days Text 269
 - Settings 270
 - Signal Properties 269
- function block 20, 93, 130

- Function Block Diagram 20
- function keys 111–112

G

- generate executable code 207

H

- Help menu
 - About 272
 - Contents 272
- HFB 167
- hierarchical function blocks 167
 - create 168, 170
 - printout 172
- high/low signal limit 325
- holiday charts 371
- HOURL 321
- HYST 321

I

- I/O configuration table 99, 280
- Inch-Pound unit system 87
- INTEG 323
- integer value delay 304
- integer value parameter 352
- integrator 323
- internal constants 173
- IO expansion module table
 - fast CNT reporting 119
 - Min send time 119
 - module 119
 - type 119

K

- K 20

L

- LIMIT 325
- Loading a Macro Block 60
- Local Trend Logging 180
- logger tool
 - clear 225
 - start 216
- logical AND gate 295
- LonWorks 20

M

- MAX 326
- Maximum Signal Selector 326
- memory usage
 - application 243
 - application files 243

- BPR size calculator 244
- calculate 241
- number of objects 243
- parameters 243
- TACN/SNVT in/out 244
- work area 243
- menu options 268
- menu structure
 - access code 260
 - adding Items 258
 - alarm 260
 - changing OP display layout 262
 - copy and paste 262
 - creating 258
 - date and time/daylight saving 261
 - editing an existing tree 262
 - moving menu items 262
 - status 259
 - sub menu 258
 - TAC service menu 261
- menu structure display 257
- MIN 326
- minimum signal selector 326
- MINUTE 327
- minute 327
- modifying binding parameters 239
- modifying block parameters 202
 - alarm texts 204
 - constants 203
 - function block parameters 202
 - I/O binding data 204
 - time schedules 203
- modifying parameter blocks 238
- modifying public constants 239
- module 95
- modules 96
- MONTH 327
- month 327
- mouse 111
- move a node 149
- multiple instances 89
- multiplier parameter in CNT block 402

N

- Naming the Application 37
- Navigating in the Hierarchical Structure 78
- NCYC 328
- network address 98
- network variable 100
- ninary value parameter 352
- nodes 142
- NOT 328

O

- online 112
- online device 105
- Online mode functions 227
- OP 20
- OP configuration tool 20
 - formats 269
 - menu bar 268
 - tree 271
- OP configuration, signals list 256
- OP Description Files
 - ESP 264
 - MENU 264
 - OPC 264
- OP description files 264
 - access code 265
 - alarm 265
 - data and declaration syntax 264
 - date and time 265
 - daylight saving 265
 - DOP 264
 - edit access code 265
 - example 266
 - exporting 267
 - importing 266
- OP menu tree
 - automatic generation 262
- operation modes 87
- operations on groups 152
 - center the selection rectangle 153
 - copy a selection rectangle to the clipboard 155
 - copy and paste 154
 - delete 154
 - deselect 153
 - disconnect 155
 - enter or edit the module name 156
 - load a macro block 164
 - macro commands in comment blocks 165
 - move 153
 - print the selection rectangle 155
 - save a macro block 162
 - select 152
- OPT 329
- optimization 329
- OR 336
- OSC 337
- oscillator 337
- output signal 279
- override of physical I/O signals 236

P

- packet size 228
- parameter blocks, modifying in on-line mode 238

parameters 93
percentage 350
physical terminal 100
PID Controller – Incremental Output 343
PIDA 340
PIDI 343
PIDI – DOPU 403
PIDP 346
POLY 349
polynomial function 349
power failure 20
PRCNT 350
priority 1 alarms 317
program cycle counter 328
program cycle time 401
program licenses 89
Program Specification 126
 Author 126
 Blocks 127
 Cycle time 127
 Date 126
 I/O signals 127
 Name 126
 Public signal table 127
 Standard App. 127
 Type 126
project folder
 DeviceDescr 27
 Documentation 27
Public Constants 173
public constants 173
 modifying in on-line mode 239
PULSE 351
pulse generator 351
PVB 352
PVI 352
PVR 354

R

RAMP 354
ramp filter 354
read/write 96, 174
read-only 96, 174
real value delay 305
real value parameter 354
Recorder
 Clear 210
 Place signal 208
 Remove signal 210
 Reset 212
 Restart 211
remove signal from recorder 210
Removing a Constant 176

Removing Unused Constants 176
Resends 228
reset counter 402
restart 20, 356
RST 112, 356
RT 357
run-time measurement 357

S

sample and hold binary value 361
sample and hold real value 364
sample time, define 213
Saving the Application 39
SECOND 358
segments 142
SEQ 359
sequencer 359
SHB 361
SHR 364
SI (metric) unit system 87
signal, place in recorder 208
simulation
 executable files 205
 external inputs 195
 modifying block parameters 202
 test probes 205
simulation mode 87, 111–112
sliding average value 402
SNVT 20, 106
source blocks 93
Specifying 35
Specifying a TAC Xenta Device 33
Specifying an I/O Module 35
SR 365
standard applications/controllers
 marking 89
Standard Network Variable Type 20
Starting TAC Menta 89
start-up delay 404
STR Wall module 121, 367–368
STRIN 367
STROUT 368
structured type 106
system error 316
System of Units 87

T

tabular 112
TAC Menta 20, 87
TAC Menta v3 compatibility 276
TAC Xenta 20
TAC Xenta, upgrading 277
TCYC 369

test probe for analog input 388
test probe for analog output 389
test probe for digital input 389
test probe for digital output 390
text file 90
The Device Specification 116
time counter 401
time out 228
time schedule 370–371, 373
TPAI 388
TPAO 389
TPDI 389
TPDO 390
TREE
 Build 271
 Copy to clipboard 272
 Expand 271
 Generate 271
 Load Specification 271
 Remove Specification 272
 Test 271
 Update Specification 272
 View Specification 272
trend logging 208
 select trend log 181
 signal 183
TRIG 370
trigger 370
TSCH 370, 373
TSCH output 403

U

U 20
Undo 188
upgrading
 AUT file 277
 system and application programs 277
upgrading a TAC Xenta 300 to v3 277
Using Constants 173
Using the Alarm Text Table 178
Using the Time Schedule Table 177

V

VECTOR 374
vectorial curve function 374
verify correct operation 278

W

Wall module table
 Auxiliary Options 123
 Backlight timeout 122
 Display timeout 122
 HVAC Settings 122

Min send time 122
Minimum change for update 122
module 121
Space temperature offset 122
Temperature display resolution 122
Temperature setpoint high limit 122
Temperature setpoint low limit 122
type 122
warm start 20
WDAY 375
week day 375

X

X 20
XOR 375

Y

Y 20

Copyright © 2008, TAC AB

All brand names, trademarks and registered trademarks are the property of their respective owners. Information contained within this document is subject to change without notice. All rights reserved.

04-00030-03-en

Europe / Headquarters

Malmö, Sweden
+46 40 38 68 50

Americas

Dallas, TX
+1 972-323-1111

Asia-Pacific

Sydney, Australia
+61 2 9700 1555

www.tac.com

t.a.c. 
by **Schneider Electric**