OVERVIEW

OPERATION

ADMINISTRATION

HARDWARE

DEVELOPMENT

QUALIFICATION

SyCOS CONTROL PACKAGE

Version: 2.1
Date: January 2003
Author: R. Breitenstein

Information in this document is subject to change without notice and does not represent a commitment on the part of **Syte GmbH**.

No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, or information storage and retrieval systems, for any purpose except in accordance with the terms of any agreement to do so.

© 1999-2003 Syte GmbH Switzerland. All Rights reserved.

1. Table Of Contents

1. Table Of Contents 4
2. Introduction 12
2.1. About This Manual 12
2.2. Who This Manual Is For 12
2.3. To What Does This Manual Apply 12
2.4. How This Manual Is Organized 12
2.5. Related Documents 12
2.6. Conventions Used in This Manual 12
2.7. Request for Comments 13
2.8. Acknowledgments 13
3. INTRODUCTION to CONTROL PACKAGE 14
4. LIST of CONTROL PACKAGE BLOCKS 16
4.1. Introduction 16
4.2. Analogue Blocks 16
4.2.1. Input Blocks 16
4.2.2. Loop Header Blocks (scanned) 17
4.2.3. Control Blocks (scanned, part of Control Loop) 21
4.2.4. Output Blocks (not scanned) 25
4.3. Digital or Discrete Blocks 26
4.3.1. Input Blocks 26
4.3.2. Loop Header Blocks (scanned) 27
4.3.3. Control Blocks (scanned, part of Loop) 32
4.3.4. Output Blocks (not scanned) 33
4.4. Miscellaneous Blocks 33
4.4.1. Cascade Terminal Block (not scanned) 34
4.4.2. Rack Block (not scanned) 34
4.4.3. Analogue Module / Digital Module (not scanned) 35
4.4.4. Timer Block (scanned) 36
4.4.5. Unit Definition Block (not scanned) 36
5. TERMINOLOGY, FIELD TYPES AND ACCESS TO THE FIELDS . . 38
5.1. Field Types 38
6. PROCESSING OF THE DATABASE 42
6.1. ANALOG SIGNAL PROCESSING 42
6.2. PROCESSING OF SENSOR BLOCK 42
6.3. PROCESSING OF SWSENSOR BLOCK 44
6.4. PROCESSING OF CONTR BLOCK 44
7. ALPHABETICAL LIST OF FIELDS 46
7.1. A - Uncompensated flow in MFCALC block 46
7.2. A - Block input in LB block 46
7.3. A - Setpoint STPT in LL block 46
7.4. AABS - State Alarm bit 47
7.5. ACOL - Alarm Colour, Alarm Rank 47
7.6. ACOS - Change of State Alarm bit 47
7.7. ACT - Actual state in digital devices 48
7.8. ACT - Direct or reverse action control 48
7.9. ADDR - Bus Address 49

7.10. ADDR - Module Address 49
7.11. ADDR - Card address 49
7.12. ADEV - Deviation Alarm bit 50
7.13. ADHI - DHI alarm bit 50
7.14. ADLO - DLO alarm bit 50
7.15. ADn - n-th alarm destination (AD1 to AD16) 51
7.16. AHI - Setpoint high limit value 51
7.17. AHIGH - HI alarm bit 51
7.18. AHIHI - HIHI alarm bit. 51
7.19. ALE - Loop error alarm flag 52
7.20. ALG - Device algorithm 52
7.21. ALMS - All present alarm bits 53
7.22. ALO - Setpoint low limit value 55
7.23. ALO - LO alarm bit 55
7.24. ALOLO - LOLO alarm bit 56
7.25. ALOP - Loop error alarm bit. 56
7.26. ALRM - Alarm condition (color) code 56
7.27. AM - Auto / Manual status flag for loop body blocks 56
7.28. AM - Enable manual interventions flag for digital devices 58
7.29. AMIH - Alarm message inhibit 58
7.30. AMLK - Auto manual lock 58
7.31. ANCOS - Annunciate change of actual state alarm 59
7.32. ANDCS - Annunciate change of desired state alarm 59
7.33. ANDH - Annunciate deviation high alarm 59
7.34. ANDL - Annunciate deviation low alarm 60
7.35. ANDV - Annunciate difference between actual and desired state alarm . . 60
7.36. ANHH - Annunciate high-high alarm 60
7.37. ANHI - Annunciate high alarm 60
7.38. ANICS - Annunciate change of input value alarm 61
7.39. ANLE - Annunciate loop error alarm for sensor blocks 61
7.40. ANLE - Annunciate loop error alarm for device blocks 61
7.41. ANLL - Annunciate low-low alarm 62
7.42. ANLO - Annunciate low alarm 62
7.43. ANRC - Annunciate measured value rate of change alarm 62
7.44. ANST - Annunciate error status alarm 63
7.45. ANUN - Annunciate undefined state alarm 63
7.46. AOT - Analog output block. 63
7.47. AROC - Setpoint rate of change limit 64
7.48. AROC - Measured value rate of change alarm bit 64
7.49. ASTR - Most critical (most relevant) alarm bit 65
7.50. ATIM - Alarm delay time (in seconds) 65
7.51. AUND - Undefined state alarm bit 65
7.52. B - Gauge pressure in MFCALC block 67
7.53. B - Block input of CALC blocks, LB blocks and PID blocks. 67
7.54. BAFI - Batch file number 67
7.55. BUS - Bus Number 68
7.56. C - Temp deg F in MFCALC block 68

7.57. C - Block input in CALC block and LB block 69
7.58. Cn - Card C01 to C32 69
7.59. CALC - CALCULATION BLOCK and LB - LOGIC BLOCK 69
7.60. CASC - Cascade status definition field 69
7.61. CBn - Type of the n-th control block (CB1 to CB8) 70
7.62. CELT - Controlling ELT 70
7.63. CHAN - Channel number 70
7.64. CLSA - Close cascade action 71
7.65. CNTR - Card address 71
7.66. CRn - Channel Range 1 to 8 (CR1 to CR8) 72
7.67. CONTR - CONTROL-LOOP HEADER BLOCK 73
7.68. CRCOS - Change of state alarm to be critical 73
7.69. CRDCS - Change of desired state alarm to be critical 74
7.70. CRDV - Deviation between actual and desired state alarm to be critical . . 74
7.71. CRICS - Change of input state alarm to be critical 75
7.72. CRLE - Loop error alarm to be critical 75
7.73. CRST - Error State alarm to be critical 75
7.74. CRUN - Undefined state alarm to be critical 76
7.75. CSTS - Cascade status 76
7.76. CTB - CASCADE TERMINAL BLOCK 77
7.77. D - Block input in CALC block and LB block 79
7.78. D - Absolute pressure in MFCALC block 79
7.79. DBND - Alarm Deadband 79
7.80. DDn - Define n-th State (DD1 to DD8) 80
7.81. DEL - Delay time 80
7.82. DELAY - Delay Block 80
7.83. DESn - n-th descriptor tag field (DES1, DES2 and DES3) 80
7.84. DEST - Desired state 81
7.85. DEV - DIGITAL BLOCK FOR HARDWARE I/O 81
7.86. DEVICE - DIGITAL BLOCK FOR DIN/DOT I/O 81
7.87. DEVINP - DIGITAL BLOCK FOR HARDWARE INPUTS 82
7.88. DEVOUT - DIGITAL BLOCK FOR HARDWARE OUTPUTS 83
7.89. DHC - Deviation high alarm to be critical 83
7.90. DHI - Deviation high alarm limit value 83
7.91. DHM - Deviation high alarm to be major 84
7.92. DI - SINGLE BIT TWO STATE BLOCK 84
7.93. DIN - DISCRETE INPUT BLOCK 84
7.94. DIR - Direction 84
7.95. DLC - Deviation low alarm to be critical 85
7.96. DLM - Deviation low alarm to be major 85
7.97. DLO - Deviation low alarm limit value 85
7.98. DOT - DISCRETE OUTPUT BLOCK 86
7.99. DPL - Display decimal point location 86
7.100. DSP - Display the setpoint numerical value 86
7.101. DSPB - Display block in continuous loops 86
7.102. DSPB - Display Block in discrete control blocks 87
7.103. DSPN - Span of displayed values 87

7.104. DTMS - Delta Measurement 88
7.105. DZER - Lowest displayed value 88
7.106. E - Block input in CALC block and LB block 89
7.107. E - Ref. abs. temperature divided by the ref. abs. pressure in MFCALC block89
7.108. EFNC - EXTOL function to be triggered 89
7.109. EGUT - Engineering units. 90
7.110. EI - Derivative Action 90
7.111. ENA - Target setpoint 91
7.112. ENAn - Enable state n (ENA1 to ENA8) 91
7.113. ENCOS - Enable EXTOL action on events 92
7.114. ENDH - Enable EXTOL action on deviation high alarm 92
7.115. ENDL - Enable EXTOL action on deviation low alarm 92
7.116. ENDSC - Enable EXTOL action on change of desired state alarm 92
7.117. ENDV - Enable EXTOL action on diff. between desired and actual state alarm92
7.118. ENHH - Enable EXTOL action on high-high alarm. 93
7.119. ENHI - Enable EXTOL action on high alarm. 93
7.120. ENISC - Enable EXTOL action on change of input state alarm. 93
7.121. ENLE - Enable EXTOL action on loop error alarm 93
7.122. ENLL - Enable EXTOL action on low-low alarm 93
7.123. ENLO - Enable EXTOL action on low alarm. 94
7.124. ENRC - Enable EXTOL action on measured value rate of change alarm . 94
7.125. ENSM - Enable simulation flag 94
7.126. ENST - Enable EXTOL action on error state alarm 94
7.127. ENTD - Enable EXTOL action on threshold down 95
7.128. ENTU - Enable EXTOL action on threshold up 95
7.129. ENUN - Enable EXTOL action on undefined state alarm 95
7.130. ENWA - Enable EXTOL Action on Loop 95
7.131. EOVR - Enable override 95
7.132. EQN - General control equation 96
7.133. ERES - Enable reset 98
7.134. F - Temperature offset in MFCALC block 99
7.135. F - Block input in CALC and LB block 99
7.136. FFW - Feed forward term 99
7.137. FILT - Filtering Factor 100
7.138. FLER - Force to loop error 100
7.139. FSLE - Loop FS enable. 101
7.140. HHC - High-high alarm to be critical 101
7.141. HHM - High-high alarm to be major. 101
7.142. HI - High alarm limit value. 102
7.143. HIC - High alarm to be critical 102
7.144. HIHI - High-high alarm limit value 102
7.145. HIM - High limit alarm to be major 102
7.146. IBIT - Input bit number. 103
7.147. IDHA - Inhibit deviation high alarm. 103
7.148. IDHI - Inside Deadband High (Gap action control) 103
7.149. IDLA - Inhibit deviation low alarm 104
7.150. IDLO - Inside Deadband Low (Gap action control) 105

7.151. IHHA - Inhibit high-high alarm 105
7.152. IHIA - Inhibit high alarm 105
7.153. ILC - Integral limiting continuous. 106
7.154. ILEA - Inhibit loop error alarm. 106
7.155. ILLA - Inhibit low-low alarm 106
7.156. ILOA - Inhibit low alarm 106
7.157. ILT - Integral limiting action type. 107
7.158. ILT - Integral limiting action type (PID block). 108
7.159. ILT1 - Halfband for integration 112
7.160. ILT1 - Halfband for integration in PID algorithm 112
7.161. ILT2 - Halfband for integration in PID algorithm 112
7.162. INA - Digital input address 112
7.163. INC - Increment. 113
7.164. INCOS - Inhibit change of state alarm 113
7.165. INDCS - Inhibit desired state change alarm 113
7.166. INDV - Inhibit difference between actual and desired state alarm 113
7.167. INICS - Inhibit input change of state alarm 113
7.168. INIT - Block initialization flag 114
7.169. INLE - Inhibit loop error alarm. 115
7.170. INLK - Interlock state 115
7.171. INM - Input mask 115
7.172. INP - Input field of the CONTR block 116
7.173. INP - Input field of the SPLIT block. 116
7.174. INP - Input field of the SWSENSOR block 116
7.175. INPn - n-th Input of the DEVICE block (INP1 to INP8) 117
7.176. INST - Inhibit error status alarm 117
7.177. INUN - Inhibit undefined state alarm 118
7.178. INV - Inversion flag in digital inputs (DIN) 118
7.179. INV - Inversion flag in digital outputs (DOT) 118
7.180. INV - Output inversion in AOT blocks. 118
7.181. IPNT - Input point 118
7.182. IRCA - Inhibit measured value rate of change alarm 119
7.183. ITYP - Initialization type for control blocks 119
7.184. ITYP - Initialization type control blocks. 120
7.185. IVAL - Input value in DIN 121
7.186. IVAL - Simulated input value in digital devices. 121
7.187. K - Gain of Lead Lag Block 122
7.188. KD - Derivative (rate) tuning (constant) 122
7.189. KI - Integral (reset) tuning constant 122
7.190. KNL - Nonlinear Scalar 123
7.191. KP - Proportional gain (constant) 123
7.192. LACOS - Change of state alarm to be latched 124
7.193. LADCS - Change of desired state alarm to be latched 124
7.194. LADH - Deviation High alarm to be latched 125
7.195. LADL - Deviation Low alarm to be latched 125
7.196. LADV - Difference between actual and desired state alarm to be latched 125
7.197. LAG - Lag 126

7.198. LAHH - High High alarm to be latched 126
7.199. LAHI - High alarm to be latched. 126
7.200. LAICS - Change of input state alarm to be latched 126
7.201. LALE - Loop error alarm to be latched. 127
7.202. LALL - Low Low alarm to be latched 127
7.203. LALO - Low alarm to be latched 128
7.204. LARC - Rate of Change alarm to be latched. 128
7.205. LAST - Error status alarm to be latched 128
7.206. LAUN - Undefined state alarm to be latched 129
7.207. LBn - Type of the n-th loop body block (LB1-LB8) 129
7.208. LEAD - Lead (sec) 129
7.209. LEC - Loop error alarm to be critical 129
7.210. LEM - Loop error alarm to be major. 130
7.211. LINC - Linearization. 130
7.212. LL - LEAD/LAG BLOCK. 134
7.213. LLC - Low-low alarm to be critical 135
7.214. LLM - Low-low alarm to be major. 135
7.215. LO - Low alarm limit value. 135
7.216. LOC - Low alarm to be critical. 136
7.217. LOLO - Low-low alarm limit value 136
7.218. LOM - Low alarm to be major 136
7.219. MEAS - Measurement in analogue Loops 138
7.220. MEAS - Measurement in discrete or digital Loops. 139
7.221. MFCALC - COMPENSATED GAS MASS FLOW BLOCK 139
7.222. MHI - Mux high clamp 140
7.223. MLO - Mux low input clamp 141
7.224. MODEL- Rack-, Module- or Card-Type 141
7.225. MOUT - Manual output position. 142
7.226. MRCOS- Change of state alarm to be major 143
7.227. MRDCS - Change of desired state alarm to be major. 143
7.228. MRDV - Difference between actual and desired state alarm to be major. 143
7.229. MRICS - Change of input state alarm to be major 144
7.230. MRLE - Loop error alarm to be major 144
7.231. MRST - Error status alarm to be major. 144
7.232. MRUN - Undefined state alarm to be major. 144
7.233. MSBL - Measurement display block. 145
7.234. NLP - NONLINEAR PID BLOCK. 145
7.235. NLPI - Error squared option 146
7.236. NMFL - Normal / Maintenance flag 146
7.237. NMLK - Normal / Maintenance lock flag. 147
7.238. NPAR - Convert from normalized parameters (PB, TR, TD). 147
7.239. NPOS - Neutral output position 147
7.240. OBIT - Output bit 149
7.241. OCB - Output control block 149
7.242. ODHI - Outside Deadband High (Gap action control) 150
7.243. ODLO - Outside Deadband Low (Gap action control). 150
7.244. ODSP - Display output of the display block. 151

7.245. OHC - Output Hi clamp 151
7.246. OIDR - Direction for displaying the display block output 151
7.247. OLC - Output Low clamp 152
7.248. OPNA - Open cascade action 152
7.249. OPNT - Output point 152
7.250. OPTI - Stroke time (in seconds) 152
7.251. OPWK - Name of the EXTOL operation to be triggered 153
7.252. OROC - Output rate of change limit 153
7.253. OUT - Card address in the AOT block 153
7.254. OUT - Output from the control blocks 154
7.255. OUT - Output in DELAY block 155
7.256. OUTA - Output address 155
7.257. OUTM - Output mask 155
7.258. OUTn - n-th output (OUT1 and OUT2) for SPLIT block 156
7.259. OUTn - n-th output (OUT1 to OUT8) for digital devices 156
7.260. OVAL - Output value 157
7.261. OVR - Override state of device 157
7.262. PB - Proportional Band 157
7.263. PCI - Pulse Counter Input block 157
7.264. PHAS - Poll phase 158
7.265. PID - TCS COMPATIBLE PID BLOCK 158
7.266. POLL - Poll time 160
7.267. PUNI - Parent unit 160
7.268. PXn - Split range parameters (PX1 and PX2) 160
7.269. RCC - Measured value rate of change alarm to be critical 162
7.270. RCM - Measured value rate of change alarm to be major 162
7.271. RES - Result 162
7.272. RESE - Reset Flag 163
7.273. ROC - Measured value rate of change limit 163
7.274. SAFE - Loop failsafe output 163
7.275. SAFE - Input Safe Value 163
7.276. SCAN - On / Off scan 164
7.277. SDSn - Name of the n-th state (SDS1 - SDS2) 164
7.278. SENSOR - Analogue Input Block 164
7.279. SFOP - Safety Operation 165
7.280. SLAVE - Block to be a slave block 165
7.281. SMFL - Simulation Flag 165
7.282. SMOO - Smoothing 166
7.283. SMOUT - Set Manual Output 166
7.284. SnI - Input corresponding to n-th state (S1I to S8I) 167
7.285. SnIE - State-n is an error state (S1IE to S8IE) 167
7.286. SnO - State-n output (S1O to S8O) 167
7.287. SNPN - Set neutral position 167
7.288. SPAN - Engineering units conversion constant 168
7.289. SPAN - Instrument span of AOT 168
7.290. SPID - SPID BLOCK 169
7.291. SPLIT - SPLIT RANGE BLOCK 170

7.292. SPWK - EXTOL step the action on which is to be triggered 171
7.293. STAT - State 171
7.294. STLK - Setpoint lock 171
7.295. STPT - Input for DELAY block 172
7.296. STPT - Setpoint in CALC Block (input A) 172
7.297. STPT - Setpoint in control blocks 172
7.298. SWSENSOR - SOFTWARE SENSOR BLOCK 173
7.299. TAG - Tag name 174
7.300. TBCC - CTB close condition 174
7.301. TD - Derivative Time 174
7.302. THLD - Threshold value 174
7.303. TIME - Timer count 175
7.304. TIMER - Timer Block 175
7.305. TLVL - Delayed state 175
7.306. TMAN - To manual flag 175
7.307. TR - Reset Time 177
7.308. TUNE - Tuning 177
7.309. TYPE - Split type 178
7.310. UBTn - User bytes (USBT1 to USBT4) 178
7.311. UFLn - User float (UFL1 to UFL4) 178
7.312. UNIT - Unit on which EXTOL action will be triggered 179
7.313. UNIT - EXTOL Unit Block 179
7.314. USHn - User short (USH1 to USH4) 179
7.315. VC - Velocity compensation factor 179
7.316. VP - Velocity / Position mode 180
7.317. ZERO - Engineering units conversion constant 180
7.318. ZERO - Engineering units conversion constant for AOT 181
8. Index 182

2. Introduction

2.1. About This Manual

This manual is the reference to the ***Control Package*** functionality used to implement:

- the internal representation of the plants sensors and actuators
- control loops to control individual sections of the process
- interlocks between different actuators and sensors
- higher level logical devices and control loops
- standard alarms about errors in the process configured thruogh alarm limits, causing messages to be sent to the Operator Consoles and into batch Protocol Files

The document has four major sections:

1. The preface you are currently reading,
2. An introduction to the Control Package concepts,
3. An overview of the Database Blocks
4. An introduction to the terminology and detailed concepts of the database for the control package functions
5. A detailed description of the processing of the database blocks
6. An alphabetical listing of all Database Block Fields

2.2. Who This Manual Is For

This manual has been written for the application engineers who will be designing and configuring process control databases

2.3. To What Does This Manual Apply

This manual is applicable to SyCOS Version 9.2.

2.4. How This Manual Is Organized

See “About This Manual” on page 12

2.5. Related Documents

For information on the EXTOL application programming refer to the SyCOS EXTOL Reference Manual.

For information on the database builder utility (DBB) refer to the SyCOS DBB Reference Manual.

2.6. Conventions Used in This Manual

The `Courier` font is used to highlight user inputs, file and directory names, the name of the system utilities that are included from the operating system com-

mand line.

2.7. Request for Comments

If you have any problems with this text contained in this manual or you wish to make suggestions for improvements then contact the SyCOS Product Support Group, namely:

Mr. Rolf Breitenstein
Syte GmbH
Seewenweg 5
CH-4153 Reinach, Switzerland

Please provide the full name of the document and its version number.

2.8. Acknowledgments

This manual has been assembled in a raw form by M. Kulka. The final form of this manual was written by R. Breitenstein.

3. INTRODUCTION to CONTROL PACKAGE

GENERAL:

The SyCOS system provides three different control functionalities:

- The analogue control package part
- The discrete or digital control package part
- The timer control package part

The analogue control package and the discrete control package work in a very similar way, although there are a few very distinct differences, which the user should be aware of.

The timer control package part is only supporting simple timing block functionality.

Additional blocks exist in the process database, which are not scanned and processed by a control package part. They just serve as a place-holder for data, such as the UNIT block.

The functionality performed by the control packages is defined through structures configured by the user, forming so called control loops. The control loops are configured using standard blocks whose input-fields and output-fields can be softwired to form a loop, which is handled as an entity.

Softwiring of fields is also possible between ddifferent loops.

All configured data, i.e. the blocks are part of the process database. One database, consisting of all blocks for one MFC, is required for each MFC. It is possible to use cross-references between the individual databases forming the database of the same SyCOS system.

Each Database Block consists of a list of fields, where each field has a short field name and is of a particular type. There are fields, which the user can configure and change through the database builder utility (DBB) and fields, which are for internal use or are not configurable.

LOOP STRUCTURE:

A Control Loop always starts with a **LOOP HEADER** Block, designated to be block number 0 of the loop. The Loop Header Block is assigned a **TAG** name. Within the Loop Header Block's data definition there are fields, which allow to define up to 8 **CONTROL BLOCKS**. When the Loop Header block is due for being processed, the following Control Blocks are also processed in ascending order, i.e. Control Block 1, 2, 3 up to 8.

There are further Block types used for input and output, which are not part of a Control Loop, such as the DIN, DOT, AIN, AOT blocks. They all have their own TAG name.

A special type of Block is the CTB Block (Cascade Terminal Block), which is used to interconnect cascaded loops. It also has it's own TAG name.

Furthermore there are blocks which serve to define the hardware inputs and outputs: RACK, DIGMODULE and ANAMODULE.

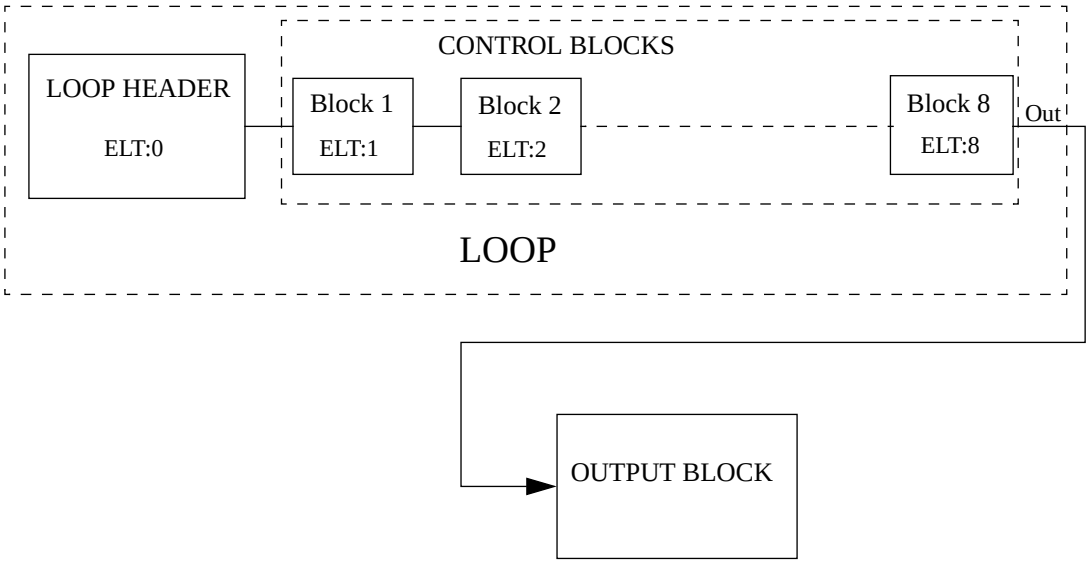


Figure 1: Principle of a Control Loop Structure

4. LIST of CONTROL PACKAGE BLOCKS

4.1. Introduction

Restrictions: T**he following chapters list those blocks, which are available in the** SyCOS **system and only in the** SyCOS **system .**

Conventions: The following sections list the available Blocks of SyCOS. The tables list the names of the fields and their short descriptors, as seen in the database-builder utility.

Those fields with a light shading are hidden fields, i.e. they are not visible when using the database-builder utility. Nevertheless they can be accessed from EXTOL application programs and referenced within database-loops.

Fields with a dark shading are not present anymore in SyCOS.

Similar Block types, using the same internal data representation are shown in the same table. The table giving details about the fields available for the different block types (marked with a 'x'). Fields in a Multi-Block Table without an 'x' are either not available or meaningless.

4.2. Analogue Blocks

All scanned analogue blocks are processed by one system task on the MFC: called **ccex.**

The analogue Control Loops use the concept of having:

- a Display Block
- a Measurement Block
- a Output Block

defined. They are selected from one of the Control Blocks or the Loop Header Block in case of the Measurement Block Definition.

The Control Loop switching from AUTO to MANUAL and vice versa is strongly related to the concept of the Display Block. An additional feature allows to define other Control Blocks to be slaves of the Display Block with respect to the AUTO/MANUAL switch-over.

4.2.1. Input Blocks

Field-Name	Descriptor	AIN
MEAS	Measurement	x
SMFL	SIM. Reset flag	x
FLER	Force LERR	x
TAG	Tag	x
PUNI	Parent Unit	x

Field-Name	Descriptor	AIN
SCAN	Point On Scan	x
PHAS	Phase	x
POLL	Poll Time	x
ADDR	Card Addr	x
CHAN	Channel Number	x
ENSM	Enable Simul.	x
MHI	Mux Hi	x
MLO	Mux Lo	x
LINC	Linearization (BP-Tbl)	x
SPAN	Instr Span	x
SMOO	Smoothing	x
DES1	Descriptor 1	x
DES2	Descriptor 2	x
DES3	Descriptor 3	x
SAFE	Safe Input Value	x
EISV	Enable Safe Input Valu	x

4.2.2. Loop Header Blocks (scanned)

The following blocks are used as the first blocks of an analogue Control Loop. They typically read in the analogue signal (e.g. a SENSOR block) and perform a control scheme to control a part of the process.

Field-Name	Descriptor	SENSOR	SWSENSOR	PCI	CONTR
MEAS	Measurement	x	x	x	x
DTMS	Delta-Meas	x	x	x	
DINF	Aklarm Information	x	x	x	x
ALRM	Alarm Condition	x	x	x	x
DFLG	Display Flags	x	x	x	x
ASTR	Display Alarm	x	x	x	x
ALMS	Alarm Bits	x	x	x	x
ACOL	Alarm Color	x	x	x	x
AHIHI	HIHI Bit	x	x	x	
AHIGH	HIGH Bit	x	x	x	
ALOLO	LOLO Bit	x	x	x	
ALO	LOW Bit	x	x	x	
ADHI	DHI Bit	x	x	x	x
ADLO	DLO Bit	x	x	x	x
AROC	ROC Bit	x	x	x	
ALE	LERR Bit	x	x	x	x

Field-Name	Descriptor	SENSOR	SWSENSOR	PCI	CONTR
SMFL	SIM. Reset flag	x	x	x	x
FLER	Force LERR	x	x	x	x
TAG	Tag	x	x	x	x
PUNI	Parent Unit	x	x	x	x
SCAN	Point On Scan	x	x	x	x
PHAS	Phase	x	x	x	x
POLL	Poll Time	x	x	x	x
LINC	Linearization	x	x	x	
ADDR	Card Addr	x		x	
CHAN	Channel Number	x	x	x	
ENSM	Enable Simul.	x	x	x	x
SAFE	Safe Input Value	x	x	x	
EISV	Enable Safe Input Valu	x	x	x	
MHI	Mux Hi	x		x	
MLO	Mux Lo	x		x	
LINC	Linearization (BP-Tbl)	x	x	x	
SPAN	Instr Span	x	x	x	x
ZERO	Instr Zero	x	x	x	x
DSPN	Display Span	x	x	x	x
DZER	Display Zero	x	x	x	x
DPL	Decimal Pnt Loc	x	x	x	x
EGUT	EGU Tag	x	x	x	x
DES1	Descriptor 1	x	x	x	x
DES2	Descriptor 2	x	x	x	x
DES3	Descriptor 3	x	x	x	x
DHI	Deviation Hi	x	x	x	x
DLO	Deviation Lo	x	x	x	x
HIHI	HiHi Limit	x	x	x	
HI	High Limit	x	x	x	
LOLO	LoLo Limit	x	x	x	
LO	Low Limit	x	x	x	
ROC	Roc Limit	x	x	x	
DBND	Deadband	x	x	x	x
THLD	Threshold	x	x	x	x
SMOO	Smoothing	x	x	x	
UNIT	Unit	x	x	x	x
OPWK	Operation	x	x	x	x
SPWK	Step	x	x	x	x
EFNC	EXTOL Function	x	x	x	x
ENWA	EXTOL Act Enable	x	x	x	x

Field-Name	Descriptor	SENSOR	SWSENSOR	PCI	CONTR
CB1	Control Block 1	x	x	x	x
CB2	Control Block 2	x	x	x	x
CB3	Control Block 3	x	x	x	x
CB4	Control Block 4	x	x	x	x
CB5	Control Block 5	x	x	x	x
CB6	Control Block 6	x	x	x	x
CB7	Control Block 7	x	x	x	x
CB8	Control Block 8	x	x	x	x
MSBL	Meas Block	x	x	x	
DSPB	Display Block	x	x	x	x
DSP	Display SP	x	x	x	x
OIDR	Out Indictr D/R	x	x	x	x
OCB	Output Block	x	x	x	x
ODSP	Display Output	x	x	x	x
AD1	Alarm Dest 1	x	x	x	x
AD2	Alarm Dest 2	x	x	x	x
AD3	Alarm Dest 3	x	x	x	x
AD4	Alarm Dest 4	x	x	x	x
AD5	Alarm Dest 5	x	x	x	x
AD6	Alarm Dest 6	x	x	x	x
AD7	Alarm Dest 7	x	x	x	x
AD8	Alarm Dest 8	x	x	x	x
AD9	Alarm Dest 9	x	x	x	x
AD10	Alarm Dest 10	x	x	x	x
AD11	Alarm Dest 11	x	x	x	x
AD12	Alarm Dest 12	x	x	x	x
AD13	Alarm Dest 13	x	x	x	x
AD14	Alarm Dest 14	x	x	x	x
AD15	Alarm Dest 15	x	x	x	x
AD16	Alarm Dest 16	x	x	x	x
HHM	HIHI Alm Major	x	x	x	
HIM	HI Alm Major	x	x	x	
LLM	LOLO Alm Major	x	x	x	
LOM	LO Alm Major	x	x	x	
DHM	DHI Alm Major	x	x	x	x
DLM	DLO Alm Major	x	x	x	x
RCM	ROC Alm Major	x	x	x	
LEM	LERR Alm Major	x	x	x	x
WSFM	WSFL Alm Major				
WSDM	WSDA Alm Major				

Field-Name	Descriptor	SENSOR	SWSENSOR	PCI	CONTR
HHC	HiHi Critical	x	x	x	
HIC	Hi Critical	x	x	x	
LLC	LoLo Critical	x	x	x	
LOC	Lo Critical	x	x	x	
DHC	Dev Hi Critical	x	x	x	x
DLC	Dev Lo Critical	x	x	x	x
RCC	Roc Critical	x	x	x	
LEC	Lp Err Critical	x	x	x	x
LAHH	Latch HiHi	x	x	x	
LAHI	Latch Hi	x	x	x	
LALL	Latch LoLo	x	x	x	
LALO	Latch Lo	x	x	x	
LADH	Latch Dev Hi	x	x	x	x
LADL	Latch Dev Lo	x	x	x	x
LARC	Latch Roc	x	x	x	
LALE	Latch Lp Err	x	x	x	x
ANHH	Annun HiHi Alm	x	x	x	
ANHI	Annun Hi Alm	x	x	x	
ANLL	Annun LoLo Alm	x	x	x	
ANLO	Annun Lo Alm	x	x	x	
ANDH	Annun DHI Alm	x	x	x	x
ANDL	Annun DLO Alm	x	x	x	x
ANRC	Annun Roc Alm	x	x	x	
ANLE	Annun LERR Alm	x	x	x	x
ANFL	Annun WSFL				
ANDA	Annun WSDA				
IHHA	Inh HiHi Alm	x	x	x	
IHIA	Inh Hi Alm	x	x	x	
ILLA	Inh LoLo Alm	x	x	x	
ILOA	Inh Lo Alm	x	x	x	
IDHA	Inh Dev Hi Alm	x	x	x	x
IDLA	Inh Dev Lo Alm	x	x	x	x
IRCA	Inh Roc Alm	x	x	x	
ILEA	Inh Loop Error	x	x	x	x
AMIH	Alm Msg Inh	x	x	x	x
ATIM	Alm Delay Time	x	x	x	
ENHH	XTL HiHi Alm	x	x	x	
ENHI	XTL Hi Alm	x	x	x	
ENLL	XTL LoLo Alm	x	x	x	
ENLO	XTL Lo Alm	x	x	x	

Field-Name	Descriptor	SENSOR	SWSENSOR	PCI	CONTR
ENDH	XTL Dev Hi Alm	x	x	x	x
ENDL	XTL Dev Lo Alm	x	x	x	x
ENRC	XTL Roc Alm	x	x	x	
ENLE	XTL Loop Error	x	x	x	x
ENTU	XTL Thres Up	x	x	x	x
ENTD	XTL Thres Down	x	x	x	x
UFL1	User Float	x	x	x	x
USH1	User Short 1	x	x	x	x
USH2	User Short 2	x	x	x	x
UBT1	User Byte 1	x	x	x	x
UBT2	User Byte 2	x	x	x	x
UBT3	User Byte 3	x	x	x	x
UBT4	User Byte 4	x	x	x	x

4.2.3. Control Blocks (scanned, part of Control Loop)

The following blocks are used to form control strategies within the analogue Control Loops. They can be used as blocks 1 to 8 within a control loop structure.

Field-Name	Descriptor	CALC	MFCALC
RES	Result	x	x
CSTS	Cascade Status	x	x
EQN	Equation	x	x
AM	Auto/Manual	x	x
SLAVE	Slave Flag	x	x
STPT	Setpoint (A)	x	
ENA	Entry SP	x	
AHI	Setpoint Hi	x	
ALO	Setpoint Lo	x	
AROC	Setpoint Roc	x	
A	K * FLOW		x
B	Input B	x	
B	Gauge Press		x
C	Input C	x	
C	Temp Deg F		x
D	Input D	x	
D	Abs Press P		x
E	Input E	x	
E	Temp Scalar		x

Table 1: CALC Controller Blocks

Field-Name	Descriptor	CALC	MFCALC
F	Input F	x	
F	Temp Offset		x
OUT	Output	x	x
OHC	Out Hi Clamp	x	
OLC	Out Lo Clamp	x	
OROC	Out Roc	x	
NPOS	Neutral Pos.	x	
SNPN	Set Neutr Pos.	x	
MOUT	Manual Output	x	x
SMOUT	Set Manual Output	x	x
INIT	Initialize	x	
ITYP	Init Type	x	
CASC	Cascade	x	
OPNA	Open Action	x	
CLSA	Close Action	x	
AMLK	A/M Lock	x	
TMAN	To Manual	x	
STLK	Setpoint Lock	x	
VP	Vel/Pos Mode	x	

Table 1: CALC Controller Blocks

Field-Name	Descriptor
RES	Result
ENA	Entry SP
STPT	Input
OUT	Output
TLVL	Trigger Level
DEL	Delay in secs
AM	Auto/Manual
AMLK	A/M Lock
TMAN	To Manual
SLAVE	Slave Flag

Field-Name	Descriptor
RES	Result
CSTS	Cascade Status

Table 2: LL Control Block

Field-Name	Descriptor
AM	Auto/Manual
SLAVE	Slave Flag
STPT	Input/SP (A)
A	Input/SP (A)
ENA	Entry SP (A)
AHI	Input/SP Hi
ALO	Input/SP Lo
AROC	Input/SP Roc
STLK	Input/SP Lock
LEAD	Lead (Sec)
LAG	Lag (Sec)
K	Gain
OUT	Output
OHC	Out Hi Clamp
OLC	Out Lo Clamp
OROC	Out Roc
CASC	Cascade
OPNA	Open Action
CLSA	Close Action
AMLK	Auto/Man Lock
TMAN	To Manual
INIT	Initialize
ITYP	Init Type
VP	Out Vel/Pos

Table 2: LL Control Block

Field-Name	Descriptor	SPID	NLP	PID
RES	Result	x	x	x
CSTS	Cascade Status	x	x	x
STPT	Setpoint	x	x	x
ENA	Entry SP	x	x	x
AHI	Setpoint Hi	x	x	x
ALO	Setpoint Lo	x	x	x
AROC	Setpoint Roc	x	x	x
AM	Auto/Manual	x	x	x
SLAVE	Slave Flag	x	x	x
B	Input	x	x	x

Table 3: PID Control Blocks

Field-Name	Descriptor	SPID	NLP	PID
KP	Prop Gain	x	x	x
KI	Int Rpts/sec	x	x	x
KD	Deriv Sec	x	x	x
EB	Deriv Action			x
EI	Deriv Action	x	x	
ACT	Rev/Dir Action	x	x	x
OUT	Output	x	x	x
OHC	Out Hi Clamp	x	x	x
OLC	Out Lo Clamp	x	x	x
OROC	Out Roc	x	x	x
AMLK	Auto/Man Lock	x	x	x
TMAN	To Manual	x	x	x
STLK	Setpoint Lock	x	x	x
INIT	Initialize	x	x	x
ITYP	Init Type	x	x	x
CASC	Cascade	x	x	x
OPNA	Open Action	x	x	x
CLSA	Close Action	x	x	x
IF	Alg Inc/Full			x
VP	Out Vel/Pos			x
ILT	Int Lim Type	x	x	x
ILT1	Int Lim Inp 1	x	x	x
ILT2	Int Lim Inp 2			x
ILC	Int Lim Cont	x	x	x
FFW	Feed Forward	x	x	x
NPOS	Neutral Pos.	x	x	x
SNPN	Set Neutr Pos.	x	x	x
MOUT	Manual Output	x	x	
SMOUT	Set Manual Output	x	x	
VC	Velocity Comp.		x	
FILT	Filtering fact		x	
KNL	Nonlin Scalar		x	
NLPI	Nonlinear P/I		x	
ODHI	Outside DbHi		x	
ODLO	Outside DbLo		x	
IDHI	Inside DbHi		x	
IDLO	Inside DbLo		x	
NPAR	Norm Parameter	x		
PB	Prop Band	x		

Table 3: PID Control Blocks

Field-Name	Descriptor	SPID	NLP	PID
TR	Reset Time	x		
TD	Deriv Time	x		
TUNE	Tuning	x		
UFL1	User Float 1	x	x	
UFL2	User Float 2	x	x	
UFL3	User Float 3	x	x	
UFL4	User Float 4	x	x	

Table 3: PID Control Blocks

Field-Name	Descriptor
RES	Result
INP	Input
TYPE	Split Type
PX1	Param X1
PX2	Param X2
OUT1	Output1
OUT2	Output2
AM	Auto/Manual
AMLK	Auto/Man Lock
TMAN	To Manual
SLAVE	Slave Flag

Table 4: SPLIT Control Block

4.2.4. Output Blocks (not scanned)

The following block is not periodically processed. The output value must be calculated within the Contol Loop and then written to the output block from an Output field (OUT) of a control block.

Field-Name	Descriptor
CELT	Contr Tag
TAG	Tag
PUNI	Parent Unit
OVAL	Output Value
NMFL	Normal/Maint.
NMLK	N/M Lock
OUT	Output
CHAN	Channel Number

Table 5: AOT-Block

Field-Name	Descriptor
INV	Outp. Inversion
MHI	Mux Hi
MLO	Mux Lo
LINC	Linearization (Bkpt)
SPAN	Instr Span
ZERO	Instr Zero
EGUT	EGU Tag
DES1	Descriptor 1
DES2	Descriptor 2
DES3	Descriptor 3
SAFE	Loop Failsafe
FSLE	Loop FS Enable

Table 5: AOT-Block

4.3. Digital or Discrete Blocks

All scanned discrete or digital blocks are processed by one system task on the MFC: called **dcex.**

The discrete Control Loops unlike the analogue control loops **do not** use the concept of having:

- a Display Block
- a Measurement Block
- a Output Block

defined.

The AUTO/MANUAL switch over from the Operator Console or from EXTOL in these control loops does not influence the A/M state of the Control Blocks. There is no similar slave block concept implemented.

4.3.1. Input Blocks

The input blocks are scanned blocks, but cannot form a Control Loop. Their value read must be referenced (as input) by a discrete Loop-Header Block (or a Control Block within the Control Loop) for further processing.

Field-Name	Descriptor
TAG	Device Tag
PUNI	Parent Unit
SCAN	Point On Scan
PHAS	Phase
POLL	Poll

Table 6: DIN Input Block

Field-Name	Descriptor
NMFL	Normal/Maint.
NMLK	N/M Lock
DES1	Device Desc 1
DES2	Device Desc 2
DES3	Device Desc 3
IVAL	Input Value
INA	Input Address
IBIT	Input Bit #
INV	Inp Inversion
SDS1	State 1 Desc
SDS2	State 2 Desc

Table 6: DIN Input Block

4.3.2. Loop Header Blocks (scanned)

The following blocks are used as the first blocks of a discrete Control Loop. They typically read in digital signals (e.g. a DEV block) or signals from an Input Block (e.g. a DEVICE block) and perform a control scheme to manipulate discrete signals (e.g. interlocks) of the process.

Field-Name	Descriptor	DEVICE	DEV	DEVINP	DI	DEVOUT
ALRM	Alarm Condition	x	x	x	x	x
ASTR	Display Alarm	x	x	x	x	x
ALMS	Alarm Bits	x	x	x	x	x
ACOL	Alarm Color	x	x	x	x	x
MEAS	Measurement	x	x	x	x	
ADEV	DEV BIT	x	x	x	x	
AABS	ABS BIT	x	x	x	x	x
AUND	UNDEF BIT	x	x	x	x	x
ALOP	LOOP BIT	x	x	x	x	x
ACOS	COS BIT	x	x	x	x	x
SMFL	SIM. Reset flag	x	x	x	x	
FLER	Force LERR	x	x	x	x	x
TAG	Device Tag	x	x	x	x	x
PUNI	Parent Unit	x	x	x	x	x
SCAN	Point On Scan	x	x	x	x	x
PHAS	Phase	x	x	x	x	x
POLL	Poll	x	x	x	x	x
ALG	Device Alg	x	x			
OPTI	Stroke Time	x	x			

Table 7: DEVICE Loop Header Blocks

Field-Name	Descriptor	DEVICE	DEV	DEVINP	DI	DEVOUT
OTIM	Stroke Timer	x	x			
AM	Auto/Manual	x	x	x	x	x
AMLK	A/M Lock	x	x	x	x	x
ENSM	Enable Simul.	x	x	x	x	
ERES	Enable Reset	x	x	x	x	x
RESE	Reset	x	x	x	x	x
DES1	Device Desc 1	x	x	x	x	x
DES2	Device Desc 2	x	x	x	x	x
DES3	Device Desc 3	x	x	x	x	x
UNIT	Unit	x	x	x	x	x
OPWK	Operation	x	x	x	x	x
SPWK	Step	x	x	x	x	x
EFNC	EXTOL Function	x	x	x	x	x
ENWA	EXTOL Act Enable	x	x	x	x	x
DEST	Desired State	x	x			x
ACT	Actual State	x	x	x	x	
IVAL	Input Value	x	x	x	x	
OVR	Override Value	x	x			x
EOVR	Enable Override	x	x			x
INLK	Interlock value	x	x			x
INA	Input Addres		x	x	x	
IPNT	Input Point		x	x	x	
INM	Input Mask		x	x	x	
OUTA	Output Address		x			x
OPNT	Output Point		x			x
OUTM	Output Mask		x			x
INP1	Input 1	x				
INP2	Input 2	x				
INP3	Input 3	x				
INP4	Input 4	x				
INP5	Input 5	x				
INP6	Input 6	x				
INP7	Input 7	x				
INP8	Input 8	x				
OUT1	Output 1	x				
OUT2	Output 2	x				
OUT3	Output 3	x				
OUT4	Output 4	x				
OUT5	Output 5	x				

Table 7: DEVICE Loop Header Blocks

Field-Name	Descriptor	DEVICE	DEV	DEVINP	DI	DEVOUT
OUT6	Output 6	x				
OUT7	Output 7	x				
OUT8	Output 8	x				
DD1	Define State 1	x	x	x	x	x
SDS1	State 1 Desc	x	x	x	x	x
ENA1	State 1 Enable	x	x	x	x	x
S1IE	State 1 in Err	x	x	x	x	x
DD2	Define State 2	x	x	x	x	x
SDS2	State 2 Desc	x	x	x	x	x
ENA2	State 2 Enable	x	x	x	x	x
S2IE	State 2 in Err	x	x	x	x	x
DD3	Define State 3	x	x	x		x
SDS3	State 3 Desc	x	x	x		x
ENA3	State 3 Enable	x	x	x		x
S3IE	State 3 in Err	x	x	x		x
DD4	Define State 4	x	x	x		x
SDS4	State 4 Desc		x	x		x
ENA4	State 4 Enable	x	x	x		x
S4IE	State 4 in Err	x	x	x		x
DD5	Define State 5	x	x	x		x
SDS5	State 5 Desc	x	x	x		x
ENA5	State 5 Enable	x	x	x		x
S5IE	State 5 in Err	x	x	x		x
DD6	Define State 6	x	x	x		x
SDS6	State 6 Desc	x	x	x		x
ENA6	State 6 Enable	x	x	x		x
S6IE	State 6 in Err	x	x	x		x
DD7	Define State 7	x	x	x		x
SDS7	State 7 Desc	x	x	x		x
ENA7	State 7 Enable	x	x	x		x
S7IE	State 7 in Err	x	x	x		x
DD8	Define State 8	x	x	x		x
SDS8	State 8 Desc	x	x	x		x
ENA8	State 8 Enable	x	x	x		x
S8IE	State 8 in Err	x	x	x		x
S1I	State 1 Input	x	x	x	x	
S1O	State 1 Output	x	x			x
S2I	State 2 Input	x	x	x	x	
S2O	State 2 Output	x	x			x

Table 7: DEVICE Loop Header Blocks

Field-Name	Descriptor	DEVICE	DEV	DEVINP	DI	DEVOUT
S3I	State 3 Input	x	x	x		
S3O	State 3 Output	x	x			x
S4I	State 4 Input	x	x	x		
S4O	State 4 Output	x	x			x
S5I	State 5 Input	x	x	x		
S5O	State 5 Output	x	x			x
S6I	State 6 Input	x	x	x		
S6O	State 6 Output	x	x			x
S7I	State 7 Input	x	x	x		
S7O	State 7 Output	x	x			x
S8I	State 8 Input	x	x	x		
S8O	State 8 Output	x	x	x		x
LB1	Logic Block 1	x	x	x	x	x
LB2	Logic Block 2	x	x	x	x	x
LB3	Logic Block 3	x	x	x	x	x
LB4	Logic Block 4	x	x	x	x	x
LB5	Logic Block 5	x	x	x	x	x
LB6	Logic Block 6	x	x	x	x	x
LB7	Logic Block 7	x	x	x	x	x
LB8	Logic Block 8	x	x	x	x	x
AD1	Alarm Dest 1	x	x	x	x	x
AD2	Alarm Dest 2	x	x	x	x	x
AD3	Alarm Dest 3	x	x	x	x	x
AD4	Alarm Dest 4	x	x	x	x	x
AD5	Alarm Dest 5	x	x	x	x	x
AD6	Alarm Dest 6	x	x	x	x	x
AD7	Alarm Dest 7	x	x	x	x	x
AD8	Alarm Dest 8	x	x	x	x	x
AD9	Alarm Dest 9	x	x	x	x	x
AD10	Alarm Dest 10	x	x	x	x	x
AD11	Alarm Dest 11	x	x	x	x	x
AD12	Alarm Dest 12	x	x	x	x	x
AD13	Alarm Dest 13	x	x	x	x	x
AD14	Alarm Dest 14	x	x	x	x	x
AD15	Alarm Dest 15	x	x	x	x	x
AD16	Alarm Dest 16	x	x	x	x	x
MRST	State Alm Major	x	x	x	x	x
MRUN	Undef Alm Major	x	x	x	x	
MRLE	Lp Error Major	x	x	x	x	x

Table 7: DEVICE Loop Header Blocks

Field-Name	Descriptor	DEVICE	DEV	DEVINP	DI	DEVOUT
MRDV	Dev Alm Major	x	x			
MRCOS	COS Alm Major	x	x	x	x	
MRICS	ICOS Alm Major	x	x	x	x	
MRDCS	DCOS Alm Major	x	x	x	x	x
CRST	State Alm Crit	x	x	x	x	x
CRUN	Undef Alm Crit	x	x	x	x	
CRLE	Loop Error Crit	x	x	x	x	x
CRDV	Dev Alm Crit	x	x			
CRCOS	COS Alm Crit	x	x	x	x	
CRICS	ICOS Alm Crit	x	x	x	x	
CRDCS	DCOS Alm Crit	x	x	x	x	x
LAST	Latch State Alm	x	x	x	x	x
LAUN	Latch Undef Alm	x	x	x	x	
LALE	Latch Lp Error	x	x	x	x	x
LADV	Latch Dev Alm	x	x			
LACOS	Latch COS Alm	x	x	x	x	
LAICS	Latch ICOS Alm	x	x	x	x	
LADCS	Latch DCOS Alm	x	x	x	x	x
ANST	Annun State Alm	x	x	x	x	x
ANUN	Annun Undef	x	x	x	x	
ANLE	Annun LERR Alm	x	x	x	x	x
ANDV	Annun Dev Alm	x	x			
ANCOS	Annun COS Alm	x	x	x	x	
ANICS	Annun ICOS Alm	x	x	x	x	
ANDCS	Annun DCOS Alm	x	x	x	x	x
INST	Inh State Alm	x	x	x	x	x
INUN	Inh Undef Alm	x	x	x	x	
INLE	Inh Loop Error	x	x	x	x	x
INDV	Inh Dev Alm	x	x			
INCOS	Inh COS Alm	x	x	x	x	
INICS	Inh ICOS Alm	x	x	x	x	
INDCS	Inh DCOS Alm	x	x	x	x	x
AMIH	Alm Msg Inh	x	x	x	x	x
ATIM	Alm Delay Time	x	x	x	x	x
ENST	XTL State Alm	x	x	x	x	x
ENUN	XTL Undef Alm	x	x	x	x	x
ENLE	XTL Loop Error	x	x	x	x	x
ENDV	XTL Dev Alm	x	x			
ENCOS	XTL COS Alm	x	x	x	x	x

Table 7: DEVICE Loop Header Blocks

Field-Name	Descriptor	DEVICE	DEV	DEVINP	DI	DEVOUT
ENISC	XTL Inp St Chg	x	x	x	x	
ENDSC	XTL Des St Chg	x	x	x	x	x
UFL1	User Float	x	x	x	x	x
USH1	User Short 1	x	x	x	x	x
USH2	User Short 2	x	x	x	x	x
UBT1	User Byte 1	x	x	x	x	x
UBT2	User Byte 2	x	x	x	x	x
UBT3	User Byte 3	x	x	x	x	x
UBT4	User Byte 4	x	x	x	x	x

Table 7: DEVICE Loop Header Blocks

4.3.3. Control Blocks (scanned, part of Loop)

The following blocks are used to form interlocking strategies within the discrete Control Loops. They can be used as blocks 1 to 8 within a control loop structure.

Note: The **LB** block is the equivalent to the **CALC** block but slightly adapted for the discrete loop side. In particular it cannot be used as a controller with setpoint and in cascades.

Field-Name	Descriptor
RES	Result
STPT	Input
OUT	Output
TLVL	Trigger Level
DEL	Delay in secs
AM	Auto/Manual
AMLK	A/M Lock
TMAN	To Manual
SLAVE	Slave Flag

Table 8: DELAY Control Block

Field-Name	Descriptor
RES	Result
EQN	Equation
AM	Auto/Manual
A	Input A
B	Input B
C	Input C
D	Input D

Table 9: LB Controller Blocks

Field-Name	Descriptor
E	Input E
F	Input F
OUT	Output
OHC	Out Hi Clamp
OLC	Out Lo Clamp
AMLK	A/M Lock
TMAN	To Manual
SLAVE	Slave Flag

Table 9: LB Controller Blocks

4.3.4. Output Blocks (not scanned)

The following block is not periodically processed. The output value must be calculated within the Contol Loop and then written to the output block from an Output field (OUT) of a control block.

Field-Name	Descriptor
TAG	Device Tag
PUNI	Parent Unit
SCAN	Point On Scan
NMFL	Normal/Maint.
NMLK	N/M Lock
DES1	Device Desc 1
DES2	Device Desc 2
DES3	Device Desc 3
OVAL	Output Value
OUTA	Output Address
OPNT	Output Point
OBIT	Output Bit #
INV	Outp Inversion
SDS1	State 1 Desc
SDS2	State 2 Desc

Table 10: DOT Output Block

4.4. Miscellaneous Blocks

The following blocks are not processed but contain just data, used in appropriate configurations.

4.4.1. Cascade Terminal Block (not scanned)

Field-Name	Description
TAG	Tag
TBCC	Close Condition

Table 11: CTB Block

4.4.2. Rack Block (not scanned)

This block contains the link of the control package to the outside I/O Hardware.

Field-Name	Descriptor
TAG	Tag
MODEL	Rack Type
BUS	Bus Number
ADDR	Rack Address
C01	Card 01 Type
C02	Card 02 Type
C03	Card 03 Type
C04	Card 04 Type
C05	Card 05 Type
C06	Card 06 Type
C07	Card 07 Type
C08	Card 08 Type
C09	Card 09 Type
C10	Card 10 Type
C11	Card 11 Type
C12	Card 12 Type
C13	Card 13 Type
C14	Card 14 Type
C15	Card 15 Type
C16	Card 16 Type
C17	Card 17 Type
C18	Card 18 Type
C19	Card 19 Type
C20	Card 20 Type
C21	Card 21 Type
C22	Card 22 Type
C23	Card 23 Type

Table 12: RACK Block

Field-Name	Descriptor
C24	Card 24 Type
C25	Card 25 Type
C26	Card 26 Type
C27	Card 27 Type
C28	Card 28 Type
C29	Card 29 Type
C30	Card 30 Type
C31	Card 31 Type
C32	Card 32 Type

Table 12: RACK Block

4.4.3. Analogue Module / Digital Module (not scanned)

These blocks contain the link of the control package to the outside I/O Hardware.

Field-Name	Descriptor
TAG	Tag
MODEL	Module Type
BUS	Bus Number
ADDR	Module Address
CR1	Channel 01 Range
CR2	Channel 02 Range
CR3	Channel 03 Range
CR4	Channel 04 Range
CR5	Channel 05 Range
CR6	Channel 06 Range
CR7	Channel 07 Range
CR8	Channel 08 Range

Table 13: Analog Module Block (ANAMODULE)

Field-Name	Descriptor
TAG	Tag
MODEL	Module Type
ADDR	Module Address
BUS	Bus Number

Table 14: Digital Module Block (DIGMODULE)

4.4.4. Timer Block (scanned)

The TIMER blocks are processed by one system task on the MFC: called **timr.**

Field-Name	Descriptor
TAG	Tag
PUNI	Parent Unit
SCAN	On Scan
STAT	State
DIR	Direction
INC	Increment
TIME	Time
EGUT	EGU Tag
DES1	Descriptor 1
DES2	Descriptor 2
DES3	Descriptor 3
UNIT	Unit
OPWK	Operation
SPWK	Step
EFNC	EXTOL Function
ENWA	EXTOL Act Enable

Table 15: TIMER Block

4.4.5. Unit Definition Block (not scanned)

The UNIT block is required in connection with EXTOL applications. It contains data for the logical EXTOL unit. In cases where the Parent Unit (PUNI) field in individual Blocks are used, the Alarm destinations of the UNIT Block are also relevant.

Field-Name	Descriptor
BAFI	Batch File Name
TAG	Tag
DES1	Descriptor 1
DES2	Descriptor 2
DES3	Descriptor 3
SFOP	Safety Oper.
AD1	Alarm Dest 1
AD2	Alarm Dest 2
AD3	Alarm Dest 3
AD4	Alarm Dest 4

Table 16: UNIT block

Field-Name	Descriptor
AD5	Alarm Dest 5
AD6	Alarm Dest 6
AD7	Alarm Dest 7
AD8	Alarm Dest 8
AD9	Alarm Dest 9
AD10	Alarm Dest 10
AD11	Alarm Dest 11
AD12	Alarm Dest 12
AD13	Alarm Dest 13
AD14	Alarm Dest 14
AD15	Alarm Dest 15
AD16	Alarm Dest 16
UFL1	User Float
USH1	User Short 1
USH2	User Short 2
UBT1	User Byte 1
UBT2	User Byte 2
UBT3	User Byte 3
UBT4	User Byte 4

Table 16: UNIT block

5. TERMINOLOGY, FIELD TYPES AND ACCESS TO THE FIELDS

5.1. Field Types

HWLINK

This field is a reference to an element of the I/O hardware, either an input or an output. The hardware is specified either through a RACK block or through a DIGMODULE or a ANAMODULE block.

This field are actually two fields: one field for storing an address of the I/O channel on the I/O card and the other field (a float variable ADDR) for storing a numeric constant.

IFIELD

This is an input field, which allows to reference any other field in the database to be used as the input value to this field.

This field are actually two fields: one field for storing an address and the other field (a float variable) for storing a numeric constant.

If -using the DBB or the EXTOL instruction 'link'- a valid loop field name has been written in the IFIELD, then the address of this field is stored in the corresponding field. The name of the field is displayed by DBB by default or using option ';'. If there is a loop field address stored, then the package reads the actual value of this loop field into the corresponding float variable every time the block containing the IFIELD is processed and before the value of this field is "used" by the algorithm of this block. The value stored in the float variable of the IFIELD can be displayed using the option ',' of the DBB.

If -using the DBB- a constant has been written in the IFIELD, then this constant is stored in the corresponding float variable and the address field is marked to contain no address. The contents of the float variable is displayed by DBB in this case by default or using option ',' or ';'. If there is no loop field address stored, then the package "uses" the actual value of the float variable for the algorithm of this block.

The EXTOL instruction 'unlink' marks the address field of IFIELD to contain no address. The contents of the float variable is displayed by DBB in this case by default or using option ',' or ';'. If there is no loop field address stored, then the package "uses" the actual value of the float variable for the algorithm of this block.

OFIELDs of the database, EDB and EXTOL 'let' instruction applied on the IFIELD change the contents of the float variable only, **but not** the contents of the field for the address.

The EXTOL instructions 'link' and 'unlink' change the contents of the field for the address, **but not** the contents of the float variable.

Note:

The description given above means that there is a significant difference between cases, when in the IFIELD using DBB a loop field name has been written, or a numeric constant has been written using DBB in the IFIELD:

In case a loop field name has been specified, the attempts to write a constant by EXTOL 'let' instruction, OFIELDs, EDB are irrelevant for the work of the database: before the package uses the value from the field, the actual value of the loop field specified is read into the float variable (and overwrites the old contents, written there eventually by OFIELDs, EXTOL 'let' instruction or EDB).

In case a numeric constant has been specified by DBB, the actual value of this

constant is used by the packages when processing the database. It means the last value written by OFIELDs, EXTOL 'let' instruction or EDB and now stored in the float variable is used by the packages.

OFIELD

This is an output field, allowing to set the calculated output (result) of the block to any other suitable field in the database.

This field are actually two fields: one field for storing an address and the other a float variable for storing a numeric constant.

If -using the DBB- a valid loop field name has been written in the OFIELD, then the address of this field is stored in the corresponding field. The name of the field is displayed by DBB by default or using option ';'. If there is a loop field address stored, then the package writes the actual (last calculated) value of the corresponding float variable into (the float variable of) this loop field every time the block containing the OFIELD is processed after the value of this field has been calculated. The value stored in the float variable of the OFIELD can be displayed using the option ',' of the DBB.

If -using the DBB- a constant has been written in the OFIELD, then this constant is stored in the corresponding float variable and the address field is marked to contain no address. The contents of the float variable is displayed by DBB in this case by default or using option ',' or ';'. If there is no loop field address stored, then the package "uses" the actual value of the float variable.

The EXTOL instruction 'unlink' marks the address field of OFIELD to contain no address. The contents of the float variable is displayed by DBB in this case by default or using option ',' or ';'. If there is no loop field address stored, then the package "uses" the actual value of the float variable for the algorithm of this block.

OFIELDs of the database, EDB and EXTOL 'let' instruction writing to the OFIELD change the contents of the float variable only, **but not** the contents of the field for the address.

The EXTOL instructions 'link' and 'unlink' change the contents of the field for the address, **but not** the contents of the float variable.

Note:

The description given above means that there is a significant difference between cases, when in the OFIELD using DBB a loop field name has been written, and cases when a numeric constant has been written using DBB in the OFIELD:

In case a loop field name has been specified, the attempts to write a constant by EXTOL 'let' instruction, OFIELDs, EDB are irrelevant for the work of the database: before the package uses the value from the field, the actual value of the loop field specified is read into the float variable (and overwrites the old contents, written there eventually by OFIELDs, EXTOL 'let' instruction or EDB).

In case a numeric constant has been specified by DBB, the actual value of this constant is used by the packages when processing the database. It means the last value written by OFIELDs, EXTOL 'let' instruction or EDB and now stored in the float variable is used by the packages.

FLOAT

This field contains a value of type float (6 decimal digits precision)

It is a float variable the contents of which can be modified

- by DBB (Data Base Builder)
- by EDB (EXTOL DeBugger)
- OFIELDs from the database
- by EXTOL 'let' instruction
- by EXTOL 'set' instruction in special cases
- from the Graphical User Interface

BYTE is a byte variable suitable for storing integer values between 0 and 255 the contents of which can be modified

- by DBB (Data Base Builder)
- by EDB (EXTOL DeBugger)
- OFIELDs from the database
- by EXTOL 'let' instruction
- from the Graphical User Interface

BIT is a bit variable suitable for storing integer values 0 or 1, the contents of which can be modified

- by DBB. DBB encrypts deliberately the values 0 and 1,
  - asking for input in the form of A,M,D,R etc.
  - and displaying then AUTO, MAN, DIR, REV etc.
- by EDB
- OFIELDs from the database
- by EXTOL 'let' instruction
- by EXTOL 'set' instruction in special cases
- from the faceplate in special cases

BINARY is a bit within a 8 or 16 bit entity, the other bits of which have different (independent) meaning. The bit can be set or cleared

- by DBB using the "loop field name" of the BINARY.

Changing the value of the BINARY bit

- by EDB
- OFIELDs from the database
- by EXTOL 'let' instruction

is possible, but it requires reading of the whole entity (using the loop field name of any of the BINARYs belonging to this word), bit manipulation of the contents of this entity using operators BNOT, BAND, BOR or using other means and writing the modified entity back to the database (again using the loop field name of any of the BINARYs belonging to this word).

The reference to an individual bit of the entity is as follows:

- Bits are numbered from 1 to 8 (or 1 to 16)
- Bit 1 (0x0001) being the least significant bit and Bit 8 (0x0080) or 16 (0x8000) being the most significant bit.

STRING is a character string of the specified length.

ELT Name is a string the contents of which is a valid ELT name.

WORD is a 16 bit unsigned word variable, the separate bits of which may have different (independent) meaning or it is handled as an unsigned short by the control package internally.

Range: 0...65535

Note: When reading or writing such fields as a whole from within EXTOL application programs, be aware, that EXTOL treats the variable type integer as a **signed short**. Therefore, if EXTOL intends to manipulate such database fields, it is recommended to use the variable type **long**.

Changing the value of separate bits in this word

- by DBB

- by EDB
- OFIELDs from the database
- by EXTOL 'let' instruction

is possible, but it requires reading of the whole word, bit manipulation of the contents of this word using operators BNOT, BAND, BOR or using other means and writing the modified word back to the database. Performing the instructions from EXTOL - use the variable type long.

DUE STATE is the number of the overwrite state OVR if OVR # 0 or the number of the interlock state INLK (if OVR=0 and INLK # 0) or the number of the desired state DEST (if OVR=0 and INLK=0).

Field Access For each field in “ALPHABETICAL LIST OF FIELDS” on page 46, there is a type given as:

- R,W - The field may be Read and Written to.
- R - The field may be read only. Note that this does not prevent the user from writing to the field. Fields marked as R (read only) should never be written to by the application engineer.

6. PROCESSING OF THE DATABASE

6.1. ANALOG SIGNAL PROCESSING

Analog data, such as temperature, pressure or chemical property, such as pH, is transmitted by the measurement transducer in the form of a standardized (ISA S50.1-1982) direct current (DC) analog signal which is varied between 4 and 20 mA or between 0 and 20 mA or between 0 and 5 V or between 1 and 5 V, depending on the transducer selected.

The electronic of the I/O card transforms the analog signal periodically in a digital signal (a bit pattern, which can be interpreted as a whole number). In case the electronic of the I/O card detects itself an error, it passes this information in the bit pattern, too. The rules, according to which an analog value is transformed in a bit pattern can (and do) differ from one type of the I/O electronic to the other type even if the whole electronics is provided by one supplier.

The bit pattern is stored in the block defining the hardware I/O modules for the MFC. This raw value is typically stored by the appropriate hardware scanner.

The further processing of the raw data is done by the control package "Continuous Control Executive" CCEX.

6.2. PROCESSING OF SENSOR BLOCK

Every 0.5 sec each input known in the SyCOS database to be a sensors are checked, whether it has to be processed. This is done using the fields SCAN, PHAS (and POLL) in the SENSOR block of the database.

If the given sensor has to be processed, its processing encompasses following steps:

- The scanner now provides a bit pattern with a numeric value between 0 and 65535
- Check whether the numeric value of the processed bit pattern is within the limits specified by the fields MLO and MHI. If not, the loop error on this sensor is raised
- In case either the hardware status reports an overrun or underrun of the input value (on 4-20mA signals) or the MLO/MHI values are hit there is automatically an error generated. The user can prevent the error by specifying a save input value and enabling the save input value function. In this case a fixed value will be used instead of the measured value.
- Transformation according to the breakpoint table, specified in the field LINC. The purpose of this processing should be to transform the integer value in the range of 0 to 65535 to a "normalized value" in the range between 0.0 and 1.0
- The value provided by the breakpoint table is linearly transformed using the contents of the fields ZERO and SPAN to the "value in engineering units".

- The transformed value in the engineering units is smoothed according to the contents of the field SMOO
  - In case DSPB >0 the smoothed value is lost; the output value of the DSPBth block is stored in the MEAS field instead. (**Only**) in case DSBL = 0, the smoothed value is stored in the field MEAS of the SENSOR block and **is available for further processing** by other blocks of the database. On the faceplate display is the MEAS value displayed within the limits DZER and (DSPN-DZER) and with DPL digits after decimal point.
  - The MEAS value is checked against the alarm limits stored in the fields DHI, DLO, HI, LO, HIHI, LOLO, ROC of the SENSOR block for those alarms, which have not been inhibited (see fields IDHA, IDLA, IHHA, ILLA, IHIA, ILOA, IRCA, ILEA). If any of the not inhibited alarms (or a not inhibited loop error) has been detected for a period of time longer than (ATIM * poll-time), the corresponding alarm bit is raised in the field ALMS.
- If an alarm bit has been raised and the MEAS value differs from its alarm value for more than DBND, the corresponding alarm bit in the ALMS field is cleared.
  - The most significant alarm of the alarms in the field ALMS is selected using the information from the fields DHM, DLM, HHM, LLM, HIM, LOM, RCM, LEM and DHC, DLC, HHC, LLC, HIC, LOC, RCC, LEC and the corresponding (single) bit is raised in the field ASTR. This alarm is displayed in the faceplate and in the graphics for the given measured value
  - New alarms are accompanied by messages on the alarm destinations, specified in the fields ADn, if the output of (all) messages has not been inhibited (AMIH)
  - New alarms can be accompanied by the Alarm Annunciation, if desired. This is ruled by the contents of the fields ANDH, ANDL, ANHH, ANLL, ANHI, ANLO, ANRC, ANLE.
  - New alarms on a SENSOR block or disappearing of these alarms causes a predefined action EFNC on predefined EXTOL operation OPWK and predefined EXTOL step SPWK.
- The alarms, which have to activate the action specified are defined in the fields ENDH, ENDL, ENHH, ENLL, ENHI, ENLO, ENRC, ENLE.
- Note that **any** of the alarms specified causes the **same** action EFNC on the EXTOL operation OPWK and its step SPWK
  - The same EXTOL action is activated if MEAS passes the threshold value THLD up and ENTU is set to 1 or if MEAS passes the threshold value THLD down and ENTD is set to 1.
  - The control blocks (as CALC, SPID, SPLIT) specified in the fields CB1 to CB2 in the SENSOR block are processed in the sequence 1, 2 ... 8.
  - If a control block is specified in the field CBn, the block is initialized if the flag in its INIT field is set. The initialisation is determined by the contents of the field ITYP.
- If the initialized block is the display block DSPL, the INIT flags of all slave blocks SBn are set as part of the initialization proce-

dure of the DSPL block.

Note: This means that slave blocks, which have a higher sequential number than the DSPB will be initialized and processed in the same scan in which the DSPB was initialized; slave blocks with sequential numbers lower than DSPB will be initialized and processed in the following scan.

6.3. PROCESSING OF SWSENSOR BLOCK

The processing of the software sensor blocks SWSENSOR is closely related to the processing of the SENSOR block.

The CCEX package decides every 0.5 seconds whether the given SWSENSOR block has to be processed or not, using the information in the fields SCAN, PHAS (and POLL).

Note, the SWSENSOR blocks are processed within the sequence of processing the SENSOR blocks, not after all SENSOR blocks have been processed! This means that the MEAS values of SENSOR blocks, which the given SWSENSOR uses for its calculations, need not to have been already updated even if they are going to be updated in the same scan.

The first steps of the processing, described for SENSOR block, are skipped in case of the processing of SWSENSOR blocks.

The processing of the SWSENSOR block depends on the way the data comes in its INP field(!):

- If the data are **read in** the INP field, they are transformed in engineering units using the contents of the fields ZERO and SPAN. Because (virtually all) readable data in the SyCOS database (e.g. MEAS) are already in engineering units, the meaningful values of ZERO = 0 and of SPAN = 1 (which means multiplying the value by 1 and adding 0, i.e. not changing the data at all).
  After having been "transformed" to engineering units the read-in value from the INP field is smoothed according to the contents of the SMOO field.
- If the data are **written to** the INP field, the values in ZERO and in SPAN are irrelevant, because no transformation takes place. To compensate this disadvantage, the written-to data are not smoothed either; the contents of the field SMOO is irrelevant.

After this first step of processing of the SWSENSOR block, the following processing steps are equal to the steps, described in connection with the processing of SENSOR blocks, which have been provided with the thick bullet

6.4. PROCESSING OF CONTR BLOCK

After all SENSOR and SWSENSOR blocks have been processed, the CONTR blocks are processed.

The processing of the CONTR block itself is very similar to the processing of the SWSENSOR block. The main differences are

- The CONTR block does not do any conversion from raw input data to an engineering value. The engineering value is read in from another block (typically a SENSOR).
- The CONTR block takes over the SPAN from the SENSOR

block it reads its measurement from.

7. ALPHABETICAL LIST OF FIELDS

The following chapter contains an alphabethical list of all fields listed in the tables of the previous sections.

7.1. A - Uncompensated flow in MFCALC block

Field type: IFIELD

Field Access: R,W

Defined in Blocks: MFCALC

Properties: A is the value of the **volumetric** flow of the gas, measured with an orifice at temperature T and pressure P.

The **volumetric** flow of the gas is transformed to the **mass** flow at reference temperature (e.g. 0 Deg C) and reference pressure (e.g. 1 atm) using the equation

A*S(E*(B+D)/(C+F))

where:

- A is the uncompensated volumetric flow rate,
- S is the square root operator,
- B is the gauge pressure reading,
- C is the temperature in degrees F or C,
- D is the offset of the gauge pressure to absolute pressure,
- E is the reference absolute temperature divided by the reference absolute pressure: Tr/Pr,
- F is offset of temperature in degrees F to degrees Rankine (460), or offset of temperature in degrees C to degrees Kelvin (273).

For further details: See “MFCALC - COMPENSATED GAS MASS FLOW BLOCK” on page 139

7.2. A - Block input in LB block

Field type: IFIELD

Field Access: R,W

Defined in Blocks: LB

Properties: This input may be a variable read from another block, a variable written by an other block, or a constant. Input B is read after input A has been read, but before input C is read.

7.3. A - Setpoint STPT in LL block

Field type: IFIELD

Field Access: R,W

Defined in Blocks: LL

Properties: Current setpoint A of the display block is displayed on the faceplate as C. Note that changing the "setpoint" from the faceplate does not change directly the

value of A of the display block: it is the field ENA which is accessed and changed from the faceplate.

The behaviour of the loop body block is different depending on whether the value A is **read in A** or **written to A** or **a constant**.

If the value A is **read in** A, then at each poll the value read is written to the field ENA. If AROC is set to zero (0.0) then the field A (STPT) is set equal to ENA (=value read). If AROC is not zero (0.0) then the field A (STPT) is ramped from it's previous value in increments (or decrements) of AROC each poll cycle until the target value in ENA is reached.

If a value is **written to** A (STPT) from another block of the database, then no ramping willtake place at all, irrespective of the value in AROC.

If a value is **written to** ENA from another block of the database, then ramping of A takes place, if AROC is not zero (0.0). The mechanism is the same as when the setpoint is read in or the setpoint is changed from a faceplate.

If the setpoint of the block is changed either from a faceplate or through the EXTOL instruction `set <elt> setpoint to <value>` then the new value is placed into field ENA. If AROC is set to zero (0.0) then the field A (STPT) is set equal to ENA. If AROC is not zero (0.0) then the field A (STPT) is ramped from it's previous value in increments (or decrements) of AROC each poll cycle until the target value in ENA is reached.

Depending on the value of ITYP the value of A and simultaneously the value of ENA can be adjusted during the initialization:

7.4. AABS - State Alarm bit

Field type: Bit

Field Access: R

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: The AABS bit is set when the loop is in state alarm.

7.5. ACOL - Alarm Colour, Alarm Rank

Field type: Word

Field Access: R

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT, SENSOR, SWSENSOR, PCI

Properties: The ACOL field provides information about the colour of the currently highest ranking alarm. The field is used to animate graphical elements with the colour of alarm.

7.6. ACOS - Change of State Alarm bit

Field type: Bit

Field Access: R

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: The ACOS bit is set when the loop's Change of State alarm is set.

7.7. ACT - Actual state in digital devices

Field type: BYTE

Field Access: R

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: The ACT field contains the number of the state to be displayed as the actual, i.e. green coloured state on the faceplate. The contents of the ACT field is written by DCEX package in the digital device block (DEVINP, DEV, DEVICE, DEVOUT, DI).

The ACT field becomes the number of the first of the defined states, whose pattern of the input (defined in fields S1I to S8I) matches the pattern obtained either from the hardware read (for DEV, DI, DEVINP blocks) or from the input combination obtained from the input fields (INP1 to INP8) of a DEVICE block.

The input bit pattern, when read from the hardware, is a straight forward pattern, read from card INA, from the bank IPNT whose pattern is binary anded (BAND) with the input mask INM.

In case of a DEVICE block, the input values obtained from inputs INP1 to INP8 are used to form a bit pattern of 8 bits, where the value of INP1-1 represents bit 0 (the least significant bit) and the value of INP8 -1 represents bit 7 (the most significant bit).

For DEVOUT blocks (blocks involving no digital inputs) the actual state ACT is set by DCEX equal the due state (i.e. OVR if OVR # 0 or INLK (if OVR=0 and INLK # 0) or DEST (if OVR=0 and INLK=0)).

7.8. ACT - Direct or reverse action control

Field type: BINARY

Field Access: R,W

ACT is the 13th bit in the word ACT (decimal value 4096, hex value 0x1000).

This bit cleared (=0) means DIRect action,

This bit set (=1) means REVerse action of the SPID block.

Defined in Blocks: SPID, NLP

Properties: If ACT = D (i.e. DIRect action, i.e. the 13th bit of the word IF is cleared) then the error e(k) is computed by subtracting the value of the B input (measurement) from the setpoint:

e(k) = STPT - B

This means that B values higher than the setpoint cause the error to become negative and will result in outputs that tend to decrease. This is the desired case for typical flow, heating/cooling or supply side level controls.

If ACT=R (i.e. REVerse action, i.e. the 13th bit of the word is set) then the error e(k) is computed by subtracting the setpoint from the B value:

e(k) = B - STPT

This means that B values higher than the setpoint cause the error to become positive and will result in outputs that tend to increase. This is the desired case for discharge side level controls.

7.9. ADDR - Bus Address

Field type: WORD

Field Access: R,W

Defined in Blocks: RACK

Properties: The bus address specifies the address assigned to a rack on a process bus (or a similar identification). This is typically the address defined for the bus coupler of this hardware unit.

This number must be unique to avoid identity confits. The value may be between 0x0 & 0xFF but depends on the hardware in use.

Like BUS & MODEL, this field is used to access the IO interface, it depends on the choice by the Automation Engineer & on the system supported IO's. See the seperate manual describing the special I/O hardware and how to define it on the SyCOS system.

7.10. ADDR - Module Address

Field type: WORD

Field Access: R,W

Defined in Blocks: ANAMODULE, DIGMODULE, ANACARD, DIGCARD

Properties: The module address specifies the address assigned to a module to provide the possibility to the system to identify it. If the hardware in use consists of independent modules on a process bus, then the field ADDR contains the bus address. If the hardware is rather a "module" or "card" in a RACK, the field ADDR identifies the address of the "card" in the RACK.

This number must be unique to avoid identity confits. The value lies between 0x0 & 0xFF but depends on the hardware in use.

Like BUS & MODEL, this field is used to access the IO interface, it depends on the choice done by the Automation Engineer & on the system supported IO's. Feel free to contact us for further information.

7.11. ADDR - Card address

Field type: HWLINK

Field Access: R,W

Defined in Blocks: SENSOR

Properties: The input is assigned to a physical location which must be made known to the scanning software. The physical location is identified by the following data:

- **rack**:
  name (or number) of the rack chassis on this node. The rack is normally identified by it's tag name. The method using a number should be avoided and is only there for compatibility reasons with older version of the system. Note, that when using the number, it actually refers to the internally defined rack blocks, they being numbered sequentially from 1 to n (0 to n-1 are the ivn numbers shown by the database builder utility "DBB").
- **card**:
  slot number within the rack. This is equivalent to the card number in the

rack-block. This number is related to the card addressing and depends on the type of the I/O hardware used. Please refer to the appropriate I/O hardware description (or scanner type).

- **channel**:
  channel on that card.

The card address is stored in the ADDR field, entered in the following format:

- #rack-card
- #module

The channel number is entered separately to the CHAN field (see CHAN).

Examples: The following examples demonstrate the syntax:

Read analog inputs from card 1 in rack named MODB_RACK1:

```
#MODB_RACK1-1
```

Write to analog output on card 5 in rack named MODB_RACK2:

```
#MODB_RACK2-5
```

Read digital inputs from a module named CAN_DIN1:

```
#CAN_DIN1
```

Write digital outputs to a module named CAN_DOT3:

```
#CAN_DOT3
```

Notes: It is important to differentiate between addressing a card in a RACK or a module (ANAMODULE, DIGMODULE) directly. In case of a RACK, there are no modules with tag names defined, therefore the reference is trhtough the RACK and the card number. In case of a named DIGMODULE or ANAMODULE the reference is through the module's tag name.

7.12. ADEV - Deviation Alarm bit

Field type: Bit

Field Access: R

Defined in Blocks: DEV, DEVICE

Properties: The ADEV bit is set when the loop's Deviation alarm is set.

7.13. ADHI - DHI alarm bit

Field type: Bit

Field Access: R

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: ADHI bit is set when the deviation alarm high (difference between MEAS value and the A (STPT) value of the display block higher than the contents of the DHI field) is present. See also ALMS.

7.14. ADLO - DLO alarm bit

Field type: Bit

Field Access: R

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: ADLO bit is set when the deviation alarm low (difference between the A (STPT) value of the display block and MEAS value higher than the contents of the DLO field) is present. See also ALMS.

7.15. ADn - n-th alarm destination (AD1 to AD16)

Field type: Byte

Field Access: R

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR, DEV, DEVICE, DEVINP, DI, DEVOUT, TIMER, UNIT

Properties: ADn contains the destinations (internal numbers, associated with the alarm devices), where (all and any) alarms have to be reported.

However, if the loop header block has a valid parent unit specified in the PUNI field, the alarm destinations specified **in the parent unit block** will be used instead and alarm destinations specified in the ADn fields of the loop header block **will be neglected**.

The possible destinations are the alarm typers, alarm summary displays, batch history (BHF) and log files configured anywhere on the network. Up to sixteen (16) separate devices may simultaneously be designated to receive the alarm data. The designation is by name to the disk file, display or printer at each operator console as assigned when the system was installed. The valid names of alarm destination names can be found in the configuration text file of the system.

BHF, OCX01, AT01 can be valid examples of the contents of ADn.

7.16. AHI - Setpoint high limit value

Field type: Float

Field Access: R,W

Defined in Blocks: CALC, LL, MFCALC, NLP, SPID,PID

Properties: AHI contains the maximum value the setpoint can be given by the operator from faceplate or from EXTOL application programs.

Restrictions: The attempt to set setpoint higher than AHI from EXTOL results in EXTOL runtime error.

7.17. AHIGH - HI alarm bit

Field type: Bit

Field Access: R

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: AHIGH bit is set when the alarm high (MEAS value higher than the contents of the HI field) is present. (See “ALMS - All present alarm bits” on page 53)

7.18. AHIHI - HIHI alarm bit

Field type: Bit

Field Access: R

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: AHIHI bit is set when the alarm high-high (MEAS value higher than the contents of the HIHI field) is present. (See “ALMS - All present alarm bits” on page 53)

7.19. ALE - Loop error alarm flag

Field type: Bit

Field Access: R

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: The ALE bit is set when the loop is in loop error.

The term loop error (sometimes referred to as LERR) is an error, which may be caused by the following errors:

- Input from hardware is faulty (either an analog channel outside the defined hardware signal limits or a severe error in communication with the hardware).
- An internal error has been detected (e.g. an error performing a calculation caused an exception such as a division by 0).
- A system internal error has occurred.

7.20. ALG - Device algorithm

Field type: Byte

Field Access: R,W

Defined in Blocks: DEV, DEVICE

Properties: Device algorithm specifies the method of processing digital signals.

The device algorithm type can be any of the numbers in the following chart. There is no prompt for this number when a DBB prototype like DEVINP, DEVOUT or DI is being used because the algorithm value is implicit in those cases (i.e., DEVINP will automatically put a value of 1 into the algorithm field). The DEV and DEVICE prototype will prompt for the algorithm type in case the user wishes to change the default value of 2. This is useful if the user wishes to create a block that has no corresponding prototype like, for example, a latching input. The prototypes mentioned in the following chart are special prototypes of the DEV block.

Algorithms for different digital devices

Type	Prototype	Explanation
1	DEVINP	Discrete input only Allows up to 8 states
2	DEV DEVICE	Full discrete I/O With feedback Allows up to 8 states. Feedback of the inputs may be supervised if field OPTI is set to something >0. A descrepancy after the specified time causes a discrepancy alarm.
3	DEVOUT	Discrete output only Allows up to 8 states
4	DI	Single bit input only Allows 2 states 'Contact input'
5	---	Software device for simulation of 8 states Can simulate actions of real descrete output with feddback input
7	---	Single bit latching discrete input

Table 17: Algorthms for differnt device behaviour

7.21. ALMS - All present alarm bits

Field type: Word

Field Access: R

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR, DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: This is a 16 bit non-prompted field, the individual bits of which are set by system software according to all actual alarms.

If two or more alarms exist simultaneously on a block, the ALMS field has two or more bits set (1 bit set for each alarm) (note the difference to ASTR field).

The alarms and the positions of the bits raised are listed in the following table:

Alarm	Bit mask (HEX)	Condition of Alarm	Alarm Rank
Limit Alarms			
HI	0x0001	MEAS value > HI value	5
HIHI	0x0002	MEAS value > HIHI value	4
LO	0x0004	MEAS value < LO value	5
LOLO	0x0008	MEAS value < LOLO value	4
Deviation Alarms			
Deviation Hi	0x0010	difference between MEAS value and A (STPT) value of display block greater than DHI value	6
Deviation Lo	0x0020	difference between A (STPT) value of display block and MEAS value greater than DLO value	6
Rate Alarm			
MV rate of change	0x0040	difference between MEAS values in two succeeding scans greater than ROC value.	7
Loop Error			
LERR	0x0080	error related to the I/O rack, e.g. hardware overscale, or any internal error, such as floating point exceptions, or database mismatches	1

Table 18: Types and Classes of Alarms in SENSOR block

Alarm	Bit mask (HEX)	Condition of Alarm	Alarm Rank
Deviation	0x0001	Desired state and actual state are different after OPTI scans	8
State	0x0002	The actual state is a state with the SnIE field set	7
Undefined	0x0004	Input bit mask does not match any of the enabled defined states	9
Loop Error	0x0008	Loop Error, e.g. an attempt to read or write from/to hardware which has failed or is not properly defined, or any internal error, such as floating point exceptions, or database mismatches	1
COS	0x0010	The actual state (ACT) has changed	6
ICOS	0x0020	The input value (IVAL) has changed	5
DCOS	0x0040	The desired state (DEST, INLK or OVR) has changed	10

Table 19: Types and Classes of Alarms in digital blocks

Note: There are alarm conditions, which by definition do only exist for 1 scan cycle. The following alarm types are of this nature:

- COS
- ICOS
- DCOS

7.22. ALO - Setpoint low limit value

Field type: Float

Field Access: R,W

Defined in Blocks: CALC, LL, MFCALC, NLP, SPID, PID

Properties: ALO contains the minimum value the setpoint can be given by the operator from faceplate or from EXTOL application programs.

The attempt to set setpoint lower than ALO from EXTOL results in EXTOL runtime error.

7.23. ALO - LO alarm bit

Field type: Bit

Field Access: R

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: The ALO bit is set when LO alarm (MEAS lower than the value of LO field) is present.

7.24. ALOLO - LOLO alarm bit

Field type: Bit

Field Access: R

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: The ALOLO bit is set when LoLo alarm (MEAS lower than the value of LOLO field) is present.

7.25. ALOP - Loop error alarm bit

Field type: Bit

Field Access: R

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: The ALOP bit is set when the loop is in loop error.

7.26. ALRM - Alarm condition (color) code

Field type: Word

Field Access: R

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR, DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: The value of the ALRM field is zero if no alarm is present, it contains the colour code of the most relevant (most critical) alarm present.

- 1 = alarm no more present, not yet acknowledged
- 2 = not major alarm
- 3 = major alarm
- 4 = critical alarm
- 16 = annunciator on (or'ed together with other information)

7.27. AM - Auto / Manual status flag for loop body blocks

Field type: Bit

Field Access: R,(W)

Defined in Blocks: CALC, DELAY, LB, LL, MFCALC, NLP, SPID, PID, SPLIT

Properties: AM flag is checked by the control package to decide whether the particular loop body block has to be processed or not.

If AM is set (1 or Y), then the block is processed, if AM is cleared (0 or N), then the block is not processed.

The AM flag is set during the initialization process (when INIT has been 1).

The AM flag is cleared when the block is set to MANUAL after flag TMAN has been set.

When the flag is set (=1) or cleared (=0), it has the same effect as if the block is set to AUTO or set to MANUAL, i.e. the complete initialization is performed or

the complete setting to MNUAL processing is performed.

Type of processing if AM = 1 depends on the type of the block involved, , initialization is performed as defined through ITYP flags:

SPID block calculates the output according to the PID algorithmus;

CALC block calculates the output according to the equation EQN;

DELAY block checks whether the input value STPT has to be passed to the output;

SPLIT block calculates the outputs according to the algorithmus selected;

When AM = 0 the output of the block is set according to normal behaviour when block is switched to MNUAL (i.e. the output may stay unchanged or is driven to a predefined position).

The usage of TMAN, INIT and AM fields have the same effects.

See “SLAVE - Block to be a slave block” on page 165

Example:

The usage of TMAN, INIT and AM fields can be demonstrated on following example:

Assume the 'slave' loop LOOPS should behave according to the 'master' loop LOOPM:

if	LOOPM has just been switched from MANUAL to AUTO,
then	LOOPS should be initialized (set to AUTO via flag INIT)
endif	

if	LOOPM has just been switched from AUTO to MAN,
then	LOOPS should be de-initialized (set to MAN via flag TMAN)
endif	

Assume, the display block of LOOPM is block 3, the display block of LOOPS is block 5.

The task can be solved including two CALC block to LOOPS, e.g.

CB6:

EQN:	A=B&C=D?E:F	If
A:	LOOPS:6.C	mode of LOOPM during the last poll
B:	0	was MANUAL and
C:	LOOPM:3.AM	mode of LOOPM now
D:	1	is AUTO
E:	1	then set 1
F	LOOPS:5.INIT	else leave unchanged
OUT:	LOOPS:5.INIT	the INIT flag of the display block of LOOPS
OLC:	0	
OHC:	1	

CB7:

EQN:	A=B&C=D?E:F	If
A:	LOOPS:7.C	mode of LOOPM during the last poll

B:	1	was AUTO and
C:	LOOPM:3.AM	mode of LOOPM now
D:	0	is MANUAL
E:	1	then set 1
F	LOOPS:5.TMAN	else leave unchanged
OUT:	LOOPS:5.TMAN	the TMAN flag of the display block of LOOPS
OLC:	0	
OHC:	1	

This technique works reliably even if the two loops involved have completely different POLL times and PHASes.

7.28. AM - Enable manual interventions flag for digital devices

Field type: Bit

Field Access: R,W

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: AM = A (1) means that the device is in non-manual mode, the operator has no access to the device (using the SETPOINT function).

To allow the operator manipulations the device must be in MANUAL mode (AM = M (0)).

EXTOL can operate the device in either mode.

Restrictions: The AM field change (i.e. switching the device from MANUAL to AUTO or vice versa) through DBB or from the operator console can be prohibited with the field ALMK (See “AMLK - Auto manual lock” on page 58)

7.29. AMIH - Alarm message inhibit

Field type: Bit

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR, DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: AMIH = Y causes (all types of) alarm messages from a particular loop to be inhibitted.

Note:
- With AMIH = Y alarm states are still displayed on the group display.
- With AMIH = Y alarm bits in fields ALMS and ASTR are still set according to same rules as with AMIH = N.

7.30. AMLK - Auto manual lock

Field type: Bit

Field Access: R,W

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT, CALC, DELAY, LB, LL, MFCALC, NLP, SPID, SPLIT

Properties: If AMLK field is set (Y or 1) then the operator on the OC is prevented to switch the loop / the device over from AUTO into MANUAL mode and vice versa.

Clearing the AMLK field allows the Operator again to manipulate the loop / the device.

7.31. ANCOS - Annunciate change of actual state alarm

Field type: Binary

Field Access: R,W

ANCOS is the 5th bit in the word ANCOS (decimal value 16, hex value 0x0010).

Defined in Blocks: DEV, DEVICE, DEVINP, DI

Properties: An audible annunciator is located at each operator console for the purpose of calling attention to the fact that an alarm has occurred.

ANCOS = Y causes the change of actual state alarm to be annunciated on those consoles which are specified in the ADn fields of the loop header block or if PUNI field contains a valid unit name- in the ADn fields of the parent unit block. (For change of actual state alarm: See “ALMS - All present alarm bits” on page 53.)

7.32. ANDCS - Annunciate change of desired state alarm

Field type: Binary

Field Access: R,W

ANDCS is the 7th bit in the word ANDCS (decimal value 64, hex value 0x0040).

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: An audible annunciator is located at each operator console for the purpose of calling attention to the fact that an alarm has occurred.

ANDCS = Y causes the change of desired state alarm to be annunciated on those consoles which are specified in the ADn fields of the loop header block or if PUNI field contains a valid unit name- in the ADn fields of the parent unit block. (For change of desired state alarm: See “ALMS - All present alarm bits” on page 53)

7.33. ANDH - Annunciate deviation high alarm

Field type: Binary

Field Access: R,W

ANDH is the 5th bit in the word ANDH (decimal value 16, hex value 0x0010).

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: An audible annunciator is located at each operator console for the purpose of calling attention to the fact that an alarm has occurred.

ANDH = Y causes the deviation high alarm to be annunciated on those consoles which are specified in the ADn fields of the loop header block or if PUNI field contains a valid unit name- in the ADn fields of the parent unit block. (For deviation high alarm: See “ALMS - All present alarm bits” on page 53)

7.34. ANDL - Annunciate deviation low alarm

Field type: Binary

Field Access: R,W

ANDL is the 3th bit in the word ANDL (decimal value 4, hex value 0x0004).

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: An audible annunciator is located at each operator console for the purpose of calling attention to the fact that an alarm has occurred.

ANDL = Y causes the deviation low alarm to be annunciated on those consoles which are specified in the ADn fields of the loop header block or if PUNI field contains a valid unit name- in the ADn fields of the parent unit block. (For ldeviation low alarm: See “ALMS - All present alarm bits” on page 53)

7.35. ANDV - Annunciate difference between actual and desired state alarm

Field type: Binary

Field Access: R,W

ANDV is the 1st bit in the word ANDV (decimal value 1, hex value 0x0001).

Defined in Blocks: DEV, DEVICE

Properties: An audible annunciator is located at each operator console for the purpose of calling attention to the fact that an alarm has occurred.

ANDV = Y causes the difference between the actual and the desired states alarm to be annunciated on those consoles which are specified in the ADn fields of the loop header block or if PUNI field contains a valid unit name- in the ADn fields of the parent unit block. (For difference alarm: See “ALMS - All present alarm bits” on page 53)

7.36. ANHH - Annunciate high-high alarm

Field type: Binary

Field Access: R,W

ANHH is the 2nd bit in the word ANHH (decimal value 2, hex value 0x0002).

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: An audible annunciator is located at each operator console for the purpose of calling attention to the fact that an alarm has occurred.

ANHH = Y causes the high-high alarm to be annunciated on those consoles which are specified in the ADn fields of the loop header block or if PUNI field contains a valid unit name- in the ADn fields of the parent unit block. (For high-high alarm see: See “ALMS - All present alarm bits” on page 53)

7.37. ANHI - Annunciate high alarm

Field type: Binary

Field Access: R,W

ANHI is the 1th bit in the word ANHI (decimal value 1, hex value 0x0001).

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: An audible annunciator is located at each operator console for the purpose of calling attention to the fact that an alarm has occurred.

ANHI = Y causes the high alarm to be annunciated on those consoles which are specified in the ADn fields of the loop header block or if PUNI field contains a valid unit name- in the ADn fields of the parent unit block. (For Hi alarm: See “ALMS - All present alarm bits” on page 53)

7.38. ANICS - Annunciate change of input value alarm

Field type: Binary

Field Access: R,W

ANICS is the 6th bit in the word ANICS (decimal value 32, hex value 0x0020).

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: An audible annunciator is located at each operator console for the purpose of calling attention to the fact that an alarm has occurred.

ANICS = Y causes the change of input value alarm to be annunciated on those consoles which are specified in the ADn fields of the loop header block or if PUNI field contains a valid unit name- in the ADn fields of the parent unit block. (For change of input value alarm: See “ALMS - All present alarm bits” on page 53)

7.39. ANLE - Annunciate loop error alarm for sensor blocks

Field type: Binary

Field Access: R,W

ANLE is the 8th bit in the word ANLE (decimal value 128, hex value 0x0080).

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: An audible annunciator is located at each operator console for the purpose of calling attention to the fact that an alarm has occurred.

ANLE = Y causes the loop error alarm to be annunciated on those consoles which are specified in the ADn fields of the loop header block or if PUNI field contains a valid unit name- in the ADn fields of the parent unit block. (For loop error alarm: See “ALMS - All present alarm bits” on page 53)

Note: There is a different bit used for the same error type on device blocks: See “ANLE - Annunciate loop error alarm for device blocks” on page 61

7.40. ANLE - Annunciate loop error alarm for device blocks

Field type: Binary

Field Access: R,W

ANLE is the 4th bit in the word ANANLE (decimal value 8, hex value 0x0008).

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: An audible annunciator is located at each operator console for the purpose of calling attention to the fact that an alarm has occurred.

ANLE = Y causes the loop error alarm to be annunciated on those consoles

which are specified in the ADn fields of the loop header block or if PUNI field contains a valid unit name- in the ADn fields of the parent unit block. (For loop error alarm: See “ALMS - All present alarm bits” on page 53)

Note: There is a different bit used for the same error type on device blocks: See “ANLE - Annunciate loop error alarm for sensor blocks” on page 61

7.41. ANLL - Annunciate low-low alarm

Field type: Binary

Field Access: R,W

ANLL is the 4th bit in the word ANLL(decimal value 8, hex value 0x0008).

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: An audible annunciator is located at each operator console for the purpose of calling attention to the fact that an alarm has occurred.

ANLL = Y causes the low-low alarm to be annunciated on those consoles which are specified in the ADn fields of the loop header block or if PUNI field contains a valid unit name- in the ADn fields of the parent unit block. (For LoLo alarm: See “ALMS - All present alarm bits” on page 53)

7.42. ANLO - Annunciate low alarm

Field type: Binary

Field Access: R,W

ANLO is the 2nd bit in the word ANLO (decimal value 2, hex value 0x0002).

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: An audible annunciator is located at each operator console for the purpose of calling attention to the fact that an alarm has occurred.

ANLO = Y causes the low alarm to be annunciated on those consoles which are specified in the ADn fields of the loop header block or if PUNI field contains a valid unit name- in the ADn fields of the parent unit block. (For Lo alarm: See “ALMS - All present alarm bits” on page 53)

7.43. ANRC - Annunciate measured value rate of change alarm

Field type: Binary

Field Access: R,W

ANRC is the 7th bit in the word ANRC (decimal value 64, hex value 0x0040).

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: An audible annunciator is located at each operator console for the purpose of calling attention to the fact that an alarm has occurred.

ANRC = Y causes the measured value rate of change alarm to be annunciated on those consoles which are specified in the ADn fields of the loop header block or if PUNI field contains a valid unit name- in the ADn fields of the parent unit block. (For definition of measured value rate of change alarm: See “ALMS - All present alarm bits” on page 53)

7.44. ANST - Annunciate error status alarm

Field type: Binary

Field Access: R,W

ANST is the 2nd bit in the word ANST (decimal value 2, hex value 0x0002).

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: An audible annunciator is located at each operator console for the purpose of calling attention to the fact that an alarm has occurred.

ANST = Y causes the error status alarm to be annunciated on those consoles which are specified in the ADn fields of the loop header block or if PUNI field contains a valid unit name- in the ADn fields of the parent unit block. (For definition of annunciate error status alarm: See “ALMS - All present alarm bits” on page 53)

7.45. ANUN - Annunciate undefined state alarm

Field type: Binary

Field Access: R,W

ANUN is the 3th bit in the word ANUN (decimal value 4, hex value 0x0004).

Defined in Blocks: DEV, DEVICE, DEVINP, DI

Properties: An audible annunciator is located at each operator console for the purpose of calling attention to the fact that an alarm has occurred.

ANUN = Y causes the undefined state alarm to be annunciated on those consoles which are specified in the ADn fields of the loop header block or if PUNI field contains a valid unit name- in the ADn fields of the parent unit block. (For definition of annunciate undefined state alarm: See “ALMS - All present alarm bits” on page 53)

7.46. AOT - Analog output block

Block-Type: AOT

Record-Type: AOT

Properties: This is the analogue output block. No periodical processing is performed on this block type. It is a "placeholder" for all parameters required to define the behaviour of an analogue output. The analogue out is immediately performed, when a link in any OUT field exists to the AOT block, and the control block with the OUT field is processed.

The value to be written out by the AOT blocks has to be written to the field OUT.

Example:

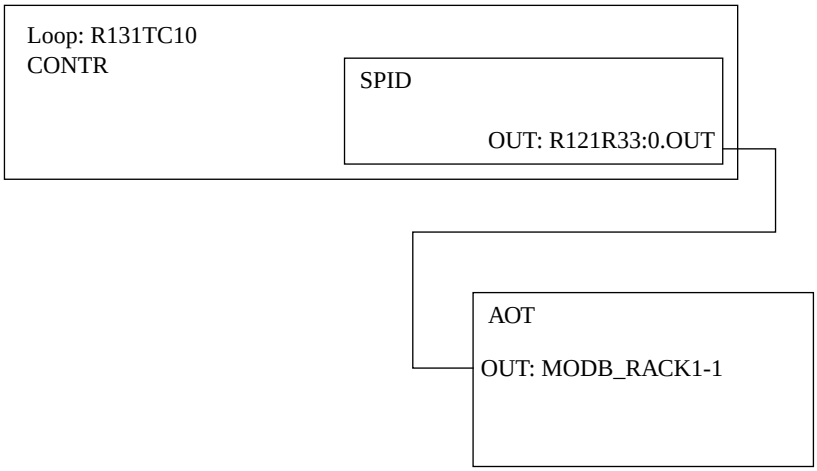


Figure 2: Ooutput from a controller to AOT block

7.47. AROC - Setpoint rate of change limit

Field type: Float

Field Access: R,W

Defined in Blocks: CALC, LL, MFCALC, NLP, SPID, PID

Properties: AROC contains the maximum allowed change of setpoint per POLL time.

The target setpoint is found in field ENA and the incremented setpoint is found in field STPT. If the absolute value of the difference between ENA and STPT is greater than AROC, then STPT + AROC (if ENA > STPT) or STPT - AROC (for ENA < STPT) is stored in STPT.

AROC = 0 means no limitation on rate of change (AROC = 0 and AROC = 99999 are equivalent)

This feature can be used to ramp a setpoint.

Note: The user should be aware of the following fact:

when the setpoint of a control block has been set, while the block is still in MANUAL mode and AROC is not equal to 0.0, and the control block is then switched to AUTO mode, the setpoint STPT is already incremented by the the amount specified in AROC, i.e. before the first normal scan of the block has taken place. I.e. the value in STPT (or A) on the first normal scan has already a value incremented twice by the amount of AROC.

7.48. AROC - Measured value rate of change alarm bit

Field type: Bit

Field Access: R

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: The AROC bit is set when the rate of change alarm (difference between MEAS values in two following scans greater than the value in the ROC field) is present.

7.49. ASTR - Most critical (most relevant) alarm bit

Field type: Word

Field Access: R

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR, DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: This field is a 16 bit non-prompted field which contains a bit-pattern indicating the most critical alarm of the set of alarms existing on the block at this moment of time. It is the alarm which is currently displayed on the faceplate display of the operator console.

If two (2) or more alarms exist simultaneously on the sensor block, the ASTR field has never more than 1 bit set (note the difference to ALMS field).

The alarms and the positions of the bits raised are to be found under ALMS (See “ALMS - All present alarm bits” on page 53).

The currently most important alarm indicated by this field is determined as follows:

- Determine, which alarms are of the most important type:
  - 1 critical alarm
  - 2 major alarm
  - 3 minor alarm
- Determine, which alarm has the highest "Alarm Rank" (as defined in “ALMS - All present alarm bits” on page 53), where the most important one is rank 1.

7.50. ATIM - Alarm delay time (in seconds)

Field type: WORD

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR, DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: ATIM contains the number of seconds the alarm detection (of any and all alarms) has to be delayed by.

This is useful to suppress intermittent error conditions. The normal alarm actions only take place after the alarms has permanently been present for the given period of time. The time starts to count again, when the alarm has cleared. Note that this applies to all alarm types.

Note: The user should be aware of the fact, that there are alarms which are by definition only present for 1 scan cycle (e.g. DCOS and ICOS). Therefore if ATIM is set to a value of nonzero, these alarms will never show up anymore.

7.51. AUND - Undefined state alarm bit

Field type: Bit

Field Access: R

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: The AUND bit is set when the undefined alarm (actual state input bit pattern does not fit any enabled defined state) is present.

7.52. B - Gauge pressure in MFCALC block

Field type: IFIELD

Field Access: R,W

Defined in Blocks: MFCALC

Properties: B is the gauge pressure reading.

The **volumetric** flow of the gas is transformed to the **mass** flow at reference temperature (e.g. 0 Deg C) and reference pressure (e.g. 1 atm) using the equation

A*S(E*(B+D)/(C+F))

where:

- A is the uncompensated volumetric flow rate,
- S is the square root operator,
- B is the gauge pressure reading,
- C is the temperature in degrees F or C,
- D is the offset of the gauge pressure to absolute pressure,
- E is the reference absolute temperature divided by the reference absolute pressure: Tr/Pr,
- F is offset of temperature in degrees F to degrees Rankine (460), or offset of temperature in degrees C to degrees Kelvin (273).

See “MFCALC - COMPENSATED GAS MASS FLOW BLOCK” on page 139

7.53. B - Block input of CALC blocks, LB blocks and PID blocks

Field type: IFIELD

Field Access: R,W

Defined in Blocks: CALC, LB, NLP, SPID, PID

Properties: This input may be a variable read from another block, a variable written by another block, or a constant. Input B is read after input A has been read, but before input C is read. Depending on the value of the ITYP field the contents of the B field can determine the contents of the ENA and STPT (A) fields during the initialization of the block. See “ITYP - Initialization type for control blocks” on page 119.

The input term B to a PID-block is usually the MEAS field of a sensor or a controller block.

During initialization of a CALC block the B input is read before its value is used for initializing A value (see ITYP = 0, ITYP = 2).

7.54. BAFI - Batch file number

Field type: String

Field Access: R

The field is implicitly set from EXTOL, using the instruction

```
batchlog into ...
```

Defined in Blocks: UNIT

Properties: Actual name of the Batch history files for this unit.

Warning: It is strictly prohibited to change this field from EXTOL, because this may obstruct the normal function of the MFC!

7.55. BUS - Bus Number

Field type: Byte

Field Access: R,W

Defined in Blocks: ANAMODULE, DIGMODULE, RACK

Properties: The bus number specifies which bus is used to access to the I/O modules.

The bus number depends on the I/O hardware used as well as on the used MFC. See the specific manual for the I/O Hardware used.

Examples: Example for the CAN modules (ANAMODULE, DIGMODULE) with a 486DX2 MFC:

Table 20: Bus Number Signification (example)

Bus Number	Signification
0	The module is connected using a external bus cable
1	The module is connected using the internal bus of the MFC

Like ADDR & MODEL, this field is used to access the IO interface, it depends on the choice done by the Automation Engineer & on the system supported IO's. Feel free to contact us for further information.

7.56. C - Temp deg F in MFCALC block

Field type: IFIELD

Field Access: R,W

Defined in Blocks: MFCALC

Properties: C is the temperature in degrees F or C

The **volumetric** flow of the gas is transformed to the **mass** flow at reference temperature (e.g. 0 Deg C) and reference pressure (e.g. 1 atm) using the equation

A*S(E*(B+D)/(C+F))

where:

- A is the uncompensated volumetric flow rate,
- S is the square root operator,
- B is the gauge pressure reading,
- C is the temperature in degrees F or C,
- D is the offset of the gauge pressure to absolute pressure,
- E is the reference absolute temperature divided by the reference absolute pressure: Tr/Pr,
- F is offset of temperature in degrees F to degrees Rankine (460), or offset of temperature in degrees C to degrees Kelvin (273).

For further details: See "MFCALC - COMPENSATED GAS MASS FLOW

BLOCK” on page 139

7.57. C - Block input in CALC block and LB block

Field type: IFIELD

Field Access: R,W

Defined in Blocks: CALC, LB

Properties: This input may be a variable read from another block, a variable written by an other block, or a constant. Input C is read after input B has been read, but before input D is read.

7.58. Cn - Card C01 to C32

Field type: Internal field type

Field Access: This field can only be entered and displayed using DBB.

Defined in Blocks: RACK

Properties: The fields can be configured with the actual names of the I/O cards in the appropriate rack or (or chassis). The rules to be applied depend on the type of hardware in use.

See the specific manual for the I/O Hardware used, where a list of the card types will be found.

7.59. CALC - CALCULATION BLOCK and LB - LOGIC BLOCK

Block-Type: CALC or LB

Record-Type: CALC

Properties: The CALC or LB block is a loop body block which provides a general calculation capability for up to six (6) inputs and two special variables M and R.

The A input (STPT) of the CALC block can be initialized (see A).

The inputs STPT(A), B, C, D, E, F are read in in this sequence.

The value of the output is calculated according to the equation EQN.

The output value, limited by the values OHC and OLC, is written to the OUT field. If there is an address of a database field in the OUT field, the same value is written also in this field.

Note: The CALC block is used within analogue loops and may be used to form a controller, i.e. a block with setpoint and output.

The LB block is used in the context of discrete loops to handle logical equations, (as interlocks). Therefore no setpoint handling is available. All calculation capabilities are identical.

7.60. CASC - Cascade status definition field

Field type: Bit

Field Access: R

Defined in Blocks: CALC, LL, NLP, SPID

Properties: CASC = Y causes a loop body block to be displayed with CASCade status dis-

played.

This will inform the process operator that the setpoint may not be changed for that block while the CASCade indication is on.

All blocks, that have cascades opened or closed (be it from the group display or from EXTOL, with the instruction "open/close .... cascade") must have CASC = Y.

7.61. CBn - Type of the n-th control block (CB1 to CB8)

Field type: Internal

Field Access: R,W but only from DBB! No access from EXTOL!

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: CBn indicates the type of the n-th block. This can be one of the blocks:

Frequently used block-types:

- CALC
- SPID
- DELAY
- SPLIT
- LL

Exceptionally used block types:

- NLP
- PID
- MFCALC

The combination of these blocks gives the control engineer the possibility to construct the control loops which perform the desired function.

Restrictions: These fields can only be accessed by DBB and not by EXTOL!

7.62. CELT - Controlling ELT

Field type: String (Size: 10 characters)

Field Access: R

Defined in Blocks: AOT

Properties: This field contains the ELT name of the loop who is currently driving the analogue output block. This field is for information purposes only.

7.63. CHAN - Channel number

Field type: Byte

Field Access: R

Defined in Blocks: AOT, PCI, SENSOR

Properties: Each channel of analog input/output is assigned to a card or module on which there may be several analog inputs/outputs. CHAN is the channel number on that card. The number of channels depends on the type of card. For specific information see the manual for the hardware used.

7.64. CLSA - Close cascade action

Field type: Byte

Field Access: R,W

Defined in Blocks: CALC, LL, NLP, SPID

Properties: The CLSA parameter specifies the action of the control block when the output cascade (the downstream loop of the cascade) changes from open to closed. CLSA is in action **only** if the output of this block (link in the OUT field) goes to a Cascade Termination Block CTB.

The block action will be one of the following: If the block had been previously set to MANUAL:

CLSA	Action
0	no action
1	return to AUTO
2	return to AUTO and close upstream cascade

7.65. CNTR - Card address

Field type: HWLINK

Field Access: R,W

Defined in Blocks: PCI

Properties: The input is assigned to a physical location which must be made known to the scanning software. The physical location is identified by the following data:

- **rack**:
  name (or number) of the rack chassis on this node. The rack is normally identified by it's tag name. The method using a number should be avoided and is only there for compatibility reasons with older version of the system. Note, that when using the number, it actually refers to the internally defined rack blocks, they being numbered sequentially from 1 to n (0 to n-1 are the ivn numbers shown by the database builder utility "DBB").
- **card**:
  slot number within the rack. This is equivalent to the card number in the rack-block. This number is related to the card addressing and depends on the type of the I/O hardware used. Please refer to the appropriate I/O hardware description (or scanner type).
- **channel**:
  channel on that card.

The card address is stored in the ADDR field, entered in the following format:

- #rack-card
- #module

The channel number is entered separately to the CHAN field (see CHAN).

Examples: The following examples demonstrate the syntax:

Read analog inputs from card 1 in rack named MODB_RACK1:

```
#MODB_RACK1-1
```

Write to analog output on card 5 in rack named MODB_RACK2:

```
#MODB_RACK2-5
```

Read digital inputs from a module named CAN_DIN1:

```
#CAN_DIN1
```

Write digital outputs to a module named CAN_DOT3:

```
#CAN_DOT3
```

Notes: It is important to differentiate between addressing a card in a RACK or a module (ANAMODULE, DIGMODULE) directly. In case of a RACK, there are no modules with tag names defined, therefore the reference is trhtough the RACK and the card number. In case of a named DIGMODULE or ANAMODULE the reference is through the module's tag name.

7.66. CRn - Channel Range 1 to 8 (CR1 to CR8)

Field type: Word

Field Access: This field can only be entered and displayed using DBB.

Defined in Blocks: ANAMODULE, ANACARD (through RACK)

Properties: The Channel Range is specific to each module model used.

Example: For the analogue modules the definition is:

Channel Range Value (in hex)	I/O configuration
0x0	Not defined (default value)
0x1	0 V / 10 V
0x2	-10 V / +10 V
0x4	0 mA / 20 mA
0x8	4 mA / 20 mA
0x10	0 mV / 500 mV
0x20	-10 mV / +50 mV
0x40	-10 mV / +100 mV
0x100	0 V / 5 V
0x200	-5 V / +5 V

Like ADDR, BUS & MODEL, this field is used to access the IO interface, it depends on the choice done by the Automation Engineer & on the system supported IO's. Feel free to contact us for further information.

For details consult the sepcific manual for the hardawre used.

7.67. CONTR - CONTROL-LOOP HEADER BLOCK

Block-Type: CONTR

Record-Type: CONTR

Properties: CONTR block is the loop header block designed for building process control loops. It does not include any input signal processing (as the SENSOR block). It obtains its input from an other block, typically from a SENSOR block (MEAS field) or from a SWSENSOR block (MEAS field).

The figure below shows a simple control loop configuration using a CONTR block.

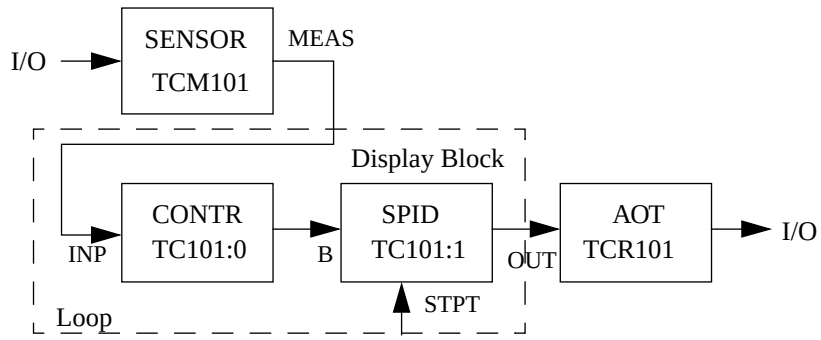


Figure 3: Simple Control Loop using CONTR block

7.68. CRCOS - Change of state alarm to be critical

Field type: Binary

Field Access: R,W

CRCOS is the 5th bit in the word CRCOS (decimal value 16, hex value 0x0010).

Defined in Blocks: DEV, DEVICE, DEVINP, DI

Properties: CRCOS = Y specifies that change of state alarm will be displayed as critical.

Critical alarms will appear to the operator in RED and will take precedence over reporting of simultaneous major or minor alarms.

If two alarms of equivalent severity are present simultaneously, the displayed alarm will be the one closer to the top of this list:

- LERR
- ICOS
- COS
- STATE
- DEV
- UNDEFINED
- DCOS

7.69. CRDCS - Change of desired state alarm to be critical

Field type: Binary

Field Access: R,W

CRDCS is the 7th bit in the word CRDCS (decimal value 64, hex value 0x0040).

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: CRDCS = Y specifies that change of desired state alarm will be displayed as critical.

Critical alarms will appear to the operator in RED and will take precedence over reporting of simultaneous major or minor alarms.

If two alarms of equivalent severity are present simultaneously, the displayed alarm will be the one closer to the top of this list:

- LERR
- ICOS
- COS
- STATE
- DEV
- UNDEFINED
- DCOS

7.70. CRDV - Deviation between actual and desired state alarm to be critical

Field type: Binary

Field Access: R,W

CRDV is the 1st bit in the word CRDV (decimal value 1, hex value 0x0001)

Defined in Blocks: DEV, DEVICE

Properties: CRDV = Y specifies that change of desired state alarm will be displayed as critical.

Critical alarms will appear to the operator in RED and will take precedence over reporting of simultaneous major or minor alarms.

If two alarms of equivalent severity are present simultaneously, the displayed alarm will be the one closer to the top of this list:

- LERR
- ICOS
- COS
- STATE
- DEV
- UNDEFINED
- DCOS

7.71. CRICS - Change of input state alarm to be critical

Field type: Binary

Field Access: R,W

CRICS is the 6th bit in the word CRICS (decimal value 32, hex value 0x0020)

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: CRICS = Y specifies that change of input state alarm will be displayed as critical.

Critical alarms will appear to the operator in RED and will take precedence over reporting of simultaneous major or minor alarms.

If two alarms of equivalent severity are present simultaneously, the displayed alarm will be the one closer to the top of this list:

- LERR
- ICOS
- COS
- STATE
- DEV
- UNDEFINED
- DCOS

7.72. CRLE - Loop error alarm to be critical

Field type: Binary

Field Access: R,W

CRLE is the 4th bit in the word CRLE (decimal value 8, hex value 0x0008))

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: CRLE = Y specifies that loop error alarm will be displayed as critical.

Critical alarms will appear to the operator in RED and will take precedence over reporting of simultaneous major or minor alarms.

If two alarms of equivalent severity are present simultaneously, the displayed alarm will be the one closer to the top of this list:

- LERR
- ICOS
- COS
- STATE
- DEV
- UNDEFINED
- DCOS

7.73. CRST - Error State alarm to be critical

Field type: Binary

Field Access: R,W

CRST is the 2nd bit in the word CRST (decimal value 2, hex value 0x0002)

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: CRST = Y specifies that undefined state alarm will be displayed as critical.

Critical alarms will appear to the operator in RED and will take precedence over reporting of simultaneous major or minor alarms.

If two alarms of equivalent severity are present simultaneously, the displayed alarm will be the one closer to the top of this list:

- LERR
- ICOS
- COS
- STATE
- DEV
- UNDEFINED
- DCOS

7.74. CRUN - Undefined state alarm to be critical

Field type: Binary

Field Access: R,W

CRUN is the 3rd bit in the word CRUN (decimal value 4, hex value 0x0004)

Defined in Blocks: DEV, DEVICE, DEVINP, DI

Properties: CRUN = Y specifies that undefined state alarm will be displayed as critical.

Critical alarms will appear to the operator in RED and will take precedence over reporting of simultaneous major or minor alarms.

If two alarms of equivalent severity are present simultaneously, the displayed alarm will be the one closer to the top of this list:

- LERR
- ICOS
- COS
- STATE
- DEV
- UNDEFINED
- DCOS

7.75. CSTS - Cascade status

Field type: Bit

Field Access: R

Defined in Blocks: CALC, LL, NLP, SPID

Properties: This flag indicates in a analogue control block, if the cascade status is open (flag = 0) or closed (flag = 1). The cascade switching is always down through the downstream loop of the cascade, therefore CSTS indicates, whether a cascade formed by the block containing the field CSTS and the upstream loop is open or closed.

7.76. CTB - CASCADE TERMINAL BLOCK

Block-Type: CTB

Record-Type: CTB

Properties: The Cascade Terminal Block (CTB) is the SyCOS block that is used to provide the linkage necessary for doing cascade control functions between analogue control blocks. CTBs must be used in order to provide the option for a cascade loop to switch between manually entered setpoint values and setpoints from another control block. By definition an 'Open' cascade controller receives its setpoint manually from the operator by entering a setpoint on the Operator Console (or from an EXTOL program). A 'Closed' cascade controller receives its setpoint from the CTB block which in turn receives its setpoint from upstream control blocks. Condition checks (upstream and downstream) are available when switching between Open and Close cascade status.

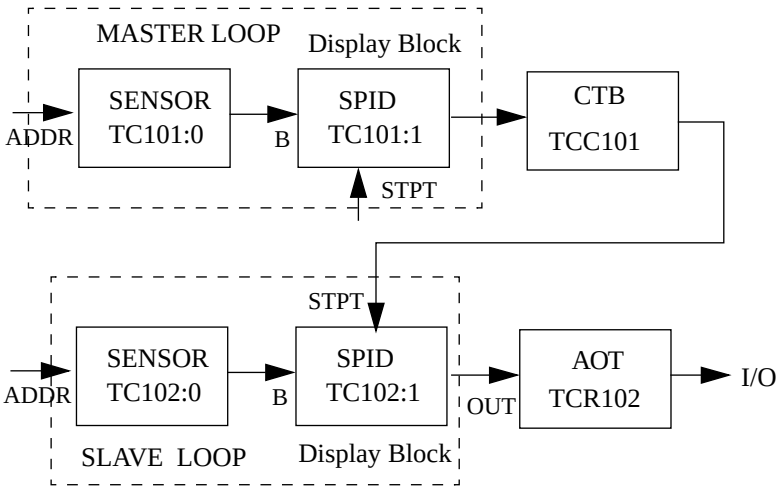


Figure 4: Simple Cascaded Loops

A cascade consists of a Master Loop (the upstream loop), a cascade terminal block (a CTB) and a Slave Loop (the downstream loop. The following characteristics apply to a cascade:

- The Slave Loop is the one, which is defined as the cascaded loop, i.e. it's field CASC is set to Y.
- Cascade openening and closing is performed on the Slave Loop.
- The current cascade status can be obtained by reading the field CSTS of the Slave Loop, which indicates, if the cascade is open or closed.
- Cascade Closing is only allowed, when the Slave Loop is in AUTO.
- The behaviour of the Master Loop (the upstream loop) on cascade open and close switching is defined through the fields OPNA (open action) and CLSA (close action) in the Master Loop.
- The Master Loop's AUTO/MANUAL switching can be made dependent on the opening and closing of the cascade (on the Slave Loop).
- The integral limitation behaviour of theMaster Loop can be made dependent on the integral limitation of the Slave Loop by choosing the appropriate integral limitation types (See "ILT - Integral limiting action type" on page 107)

on the Master Loop control block.

Cascade Terminal Block Usage:

The usage of the CTB block must be restricted to positional mode operations. This limits the number of inputs to the CTB to 1. For historical reasons the parameters can be defined as "input" numbers 1 to 4 and as "output" numbers 1 to 8. The user should restrict himself to use input 1, output 1 and set the field TBCC to 1. This allows to use the CTB block as shown in Figure 4: "Simple Cascaded Loops". This functionality is clearly defined. It connects one input to the CTB block, which is defined in the OUT field of the driving Control Block (written as "TCC101 1").

The block which receives its setpoint from a CTB should use an I/O field specification of CTBNAME Lk, where Lk is a number in the range of 1 through 8 (use the number 1 for normal cases). This name of the I/O field should be the STPT field; the control block should be specified as cascade; and in order for Open and Close actions to work correctly, this control block should be specified as the display control block. For blocks which fetch an input (STPT) from a CTB, they simply get the value which was placed there last (from any one of the four possible inputs).

When an upstream control block, which outputs to a CTB, is placed in AUTO, the Open action is tested and if it is 1 (go to MANUAL) the operation is denied as long as the cascade is open in one or more downstream links. Alternatively if the cascade controller is later opened, and Open action is specified as a 1 on an upstream control block which outputs to the CTB the (upstream) control block is switched to MANUAL. The Cascade Terminal Block close condition (TBCC) field specifies further what conditions must be satisfied before the cascade controller is allowed to go to the Close status.

Note:

Database links to CTB blocks are an exception. The syntax is:

<elt-name> <number>

where number is in the range 1 to 4 or 1 to 8 (but use alwasys 1).

Note:

The CTB blocks are designed to establish links between analogue control blocks only used for direct digital control. These are the following blocks:

- SPID, NLP
- CALC
- LL

There is no cascade support on any other block.

7.77. D - Block input in CALC block and LB block

Field type: IFIELD

Field Access: R,W

Defined in Blocks: CALC, LB

Properties: This input may be a variable read from another block, a variable written by an other block, or a constant. Input D is read after input C has been read, but before input E is read.

7.78. D - Absolute pressure in MFCALC block

Field type: IFIELD

Field Access: R,W

Defined in Blocks: MFCALC

Properties: D is the offset of the gauge pressure to absolute pressure.

The **volumetric** flow of the gas is transformed to the **mass** flow at reference temperature (e.g. 0 Deg C) and reference pressure (e.g. 1 atm) using the equation

A*S(E*(B+D)/(C+F))

where:

- A is the uncompensated volumetric flow rate,
- S is the square root operator,
- B is the gauge pressure reading,
- C is the temperature in degrees F or C,
- D is the offset of the gauge pressure to absolute pressure,
- E is the reference absolute temperature divided by the reference absolute pressure: Tr/Pr,
- F is offset of temperature in degrees F to degrees Rankine (460), or offset of temperature in degrees C to degrees Kelvin (273).

For further details: See “MFCALC - COMPENSATED GAS MASS FLOW BLOCK” on page 139

7.79. DBND - Alarm Deadband

Field type: Float

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: DBND contains the value of an adjustable (but common, i.e. single for all mentioned alarms) deadband for HiHi, Hi, LoLo, Lo, DHI, DLO (deviation) and ROC alarms. The deadband is applied in such a way that the process value must return DBND amount into the acceptable range before the alarm indication is cancelled.

Restrictions: The value of DBND should never be greater then the values of DHI or DLO.

7.80. DDn - Define n-th State (DD1 to DD8)

Field type: Binary

Field Access: R

The n-th state corresponds the n-th bit in the field DDn

Defined in Blocks: DEV, DEVICE, DEVINP, DEVOUT

Properties: DDn = Y declares the n-th state as an existing (defined) state.

7.81. DEL - Delay time

Field type: IFIELD

Field Access: R,W

Defined in Blocks: DELAY

Properties: DEL contains the delay time in seconds. The internal timer starts counting when the input (STPT) became the value of trigger level (TLVL). After DEL seconds the output is set to TLVL (STPT).

All other than TLVL values of input are passed to output immediately in the moment of their occurence.

The purpose is to delay the state transition from any state to the one specified in TLVL.

7.82. DELAY - Delay Block

Block-Type: DELAY

Record-Type: DELAY

Properties: The DELAY block is a specialized control block, which provides a delay timing function on a signal. A signal is comapred on its input with a trigger signal level. If the input signal is equal to the input trigger level, then the output is only changed to this level after a specified time has elapsed. Otherwise the input is copied immediately to the output. In case the input signal does not stay at the trigger level for at least the specified delay time, the running delay is cancelled.

7.83. DESn - n-th descriptor tag field (DES1, DES2 and DES3)

Field type: String (Size: 12 characters)

Field Access: R,W

The Descriptor field is 11 usable characters wide

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR, DEV, DEVICE, DEVINP, DI, DEVOUT, AOT, DIN, DOT, UNIT

Properties: DESn is one of three twelve-character fields used to specify a description of this TAG to the operator on the faceplates.

Any alphabetic or numeric characters may be used. The descriptors are normally printed sequentially on a log. Blank spaces left in these fields will print as spaces on the log.

7.84. DEST - Desired state

Field type: Byte

Field Access: R,W

Defined in Blocks: DEV, DEVICE, DEVOUT

Properties: DEST is the number of the state of the device to which the system tries to drive this device, if OVR = 0 and INLK = 0.

In devices realized by DEV or DEVOUT loop header blocks, the system reads 16 bits from the address OUTA from the bank selector OPNT. The system sets the bits, specified by the output mask OUTM, to their states defined in the field OUTn, where n = DEST. Other bits stay as they have been. The 16 bit pattern is then written to the address OUTA on bank selector OPNT.

The desired state DEST (also called setpoint) can be set either from operator console or from another block or by DBB in remote mode.

If the actual state is different from the desired state after OPTI * POLL time seconds, a deviation alarm will be generated. The maximum delay time may be (OPTI + 1) * POLL, this is because the event may occurr immediately after the processing of the block.

Note: An EXTOL program, which would like to drive a device to a new desired state should use the ***drive*** instruction and not write to the field DEST.

7.85. DEV - DIGITAL BLOCK FOR HARDWARE I/O

Block-Type: DEV

Record-Type: DEV

Properties: DEV is the loop header block designed for building digital devices having up to 8 states, using directly hardware inputs (up to 16 on one selector bank of the same input card) and directly hardware outputs (up to 16 on one selector bank of the same output card).

Note: Separate hardware inputs and outputs, used in DEV, **can not** be visualized on the faceplate. I.e. hardware elements (I/O), known from P&I Ds, **have not** their counterparts in the SyCOS database, if we use DEV. That is the reason for recommending the usage of DEVICE instead.

7.86. DEVICE - DIGITAL BLOCK FOR DIN/DOT I/O

Block-Type: DEVICE

Record-Type: DEV

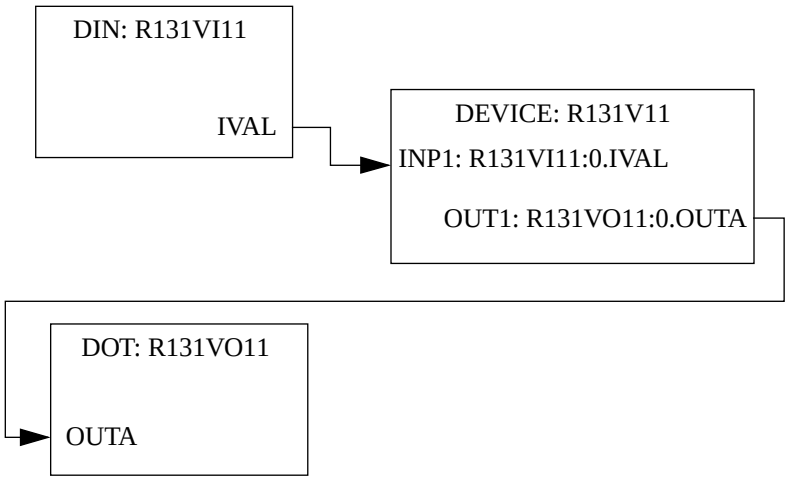
Properties: DEVICE is the loop header block designed for building digital devices with up to 8 states, using database links (addresses) as inputs (up to 8; the linked field must be in the same MFC) and database adresses as outputs (up to 8; the linked field must be in the same MFC).

Note: Separate hardware inputs and outputs can not be directly used in DEVICE; they must be realized using preferrably DIN loop header blocks (for inputs) and DOT loop header blocks (for outputs). It is recommended to provide DIN blocks with the same names as the inputs have in P&I Ds; to provide DOT blocks with the same names as the outputs have in P&I Ds. This allows the visualization of hardware inputs and outputs on the faceplate under the names they have in P&I Ds.

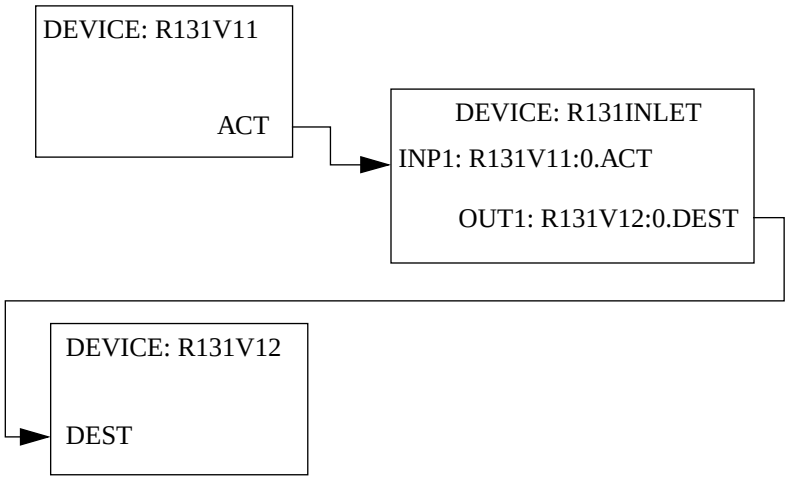
That is the reason for recommending the usage of DEVICE, DIN and DOT blocks instead of DEV blocks to handle complex I/O devices.

The IVAL fields of DINs are used as software links in DEVICE block for inputs; OUTA fields of DOTs are used as software links for outputs.

Example:



Cascading several DEVICE-blocks is also possible, as given in the followin example:



7.87. DEVINP - DIGITAL BLOCK FOR HARDWARE INPUTS

Block-Type: DEVINPUT

Record-Type: DEV

Properties: DEVINP is the loop header block designed for building digital devices having up to 8 states, using directly hardware inputs (up to 16 on one selector bank of the same input card).

Note: Separate hardware inputs, used in DEVINP, **can not** be visualized on the face-plate. I.e. hardware elements (inputs), known from P&I Ds, **have not** their coun-

terparts in the SyCOS database, if we use DEVINP for building devices **with more than one input**. That is the reason for recommending the usage of DEVICE in these cases instead.

7.88. DEVOUT - DIGITAL BLOCK FOR HARDWARE OUTPUTS

Block-Type: DEVOUT

Record-Type: DEV

Properties: DEVOUT is the loop header block designed for building digital devices having up to 8 states, using directly hardware outputs (up to 16 on one selector bank of the same output card).

Note: Separate hardware outputs, used in DEVOUT, **can not** be visualized on the face-plate. I.e. hardware elements (outputs), known from P&I Ds, **have not** their counterparts in the SyCOS database, if we use DEVOUT for building devices **with more than one output**. That is the reason for recommending the usage of DEVICE in these cases instead.

7.89. DHC - Deviation high alarm to be critical

Field type: Binary

Field Access: R,W

DHC is the 5th bit in the word DHC (decimal value 16, hex value 0x0010)

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: DHC = Y specifies that deviation high alarm will be displayed as critical. Critical alarm will appear to the operator in RED and will take precedence over reporting of simultaneous major or minor alarms.

If two alarms of equivalent severity are present simultaneously, the displayed alarm will be the one closer to the top of this list:

- LERR
- HIHI and LOLO
- HI and LO
- DHI and DLO
- ROC

7.90. DHI - Deviation high alarm limit value

Field type: Float

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: DHI is the value with which the difference between MEAS value of the loop and the A (STPT) value of the display block is compared. If the difference exceeds DHI (and deviation high alarm is not inhibited, i.e. IDHA = N), the deviation high alarm is raised.

Setting the deviation high alarm limit to zero deactivates this alarm.

Restriction: The value of DHI should never be smaller or equal to the value of DBND.

7.91. DHM - Deviation high alarm to be major

Field type: Binary

Field Access: R,W

DHM is the 5th bit in the word DHM (decimal value 16, hex value 0x0010).

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: DHM = Y specifies that deviation high alarm will be displayed as major alarm.

Major alarm will appear to the operator in YELLOW and will take precedence over reporting of simultaneous minor alarms. However, reporting of simultaneously present critical alarms takes precedence over reporting major alarms.

If two alarms of equivalent severity are present simultaneously, the displayed alarm will be the one closer to the top of this list:

- LERR
- HIHI and LOLO
- HI and LO
- DHI and DLO
- ROC

7.92. DI - SINGLE BIT TWO STATE BLOCK

Block-Type: DI

Record-Type: DEV

Properties: DI is a loop header block designed for building devices for contacts with two states. It is a simple subset of the general DEV block type and more precise of the DEVINP block type.

7.93. DIN - DISCRETE INPUT BLOCK

Block-Type: DIN

Record-Type: DIN

Properties: DIN is a loop header block designed for building devices for single digital inputs.

It is recommended to provide the DIN with the name of the input from the P&ID.

The status of this input can be then vizualized on the faceplate.

The IVAL field of the DIN can be used as a link to digital inputs in the DEVICE blocks. This is the recommended usage of DIN blocks.

See “DEVICE - DIGITAL BLOCK FOR DIN/DOT I/O” on page 81

7.94. DIR - Direction

Field type: Bit

Field Access: R,W

Defined in Blocks: TIMER

Properties: Bit DIR set causes the timer to count down,

Bit DIR cleared causes the timer to count up.

7.95. DLC - Deviation low alarm to be critical

Field type: Binary

Field Access: R,W

DLC is the 6th bit in the word DLC (decimal value 32, hex value 0x0020)

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: DLC = Y specifies that deviation low alarm will be displayed as critical.

Critical alarm will appear to the operator in RED and will take precedence over reporting of simultaneous major or minor alarms.

If two alarms of equivalent severity are present simultaneously, the displayed alarm will be the one closer to the top of this list:

- LERR
- HIHI and LOLO
- HI and LO
- DHI and DLO
- ROC

7.96. DLM - Deviation low alarm to be major

Field type: Binary

Field Access: R,W

DLM is the 6th bit in the word DLM (decimal value 32, hex value 0x0020)

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: DLM = Y specifies that deviation low alarm will be displayed as major alarm.

Major alarm will appear to the operator in YELLOW and will take precedence over reporting of simultaneous minor alarms. However, reporting of simultaneously present critical alarms takes precedence over reporting major alarms.

If two alarms of equivalent severity are present simultaneously, the displayed alarm will be the one closer to the top of this list:

- LERR
- HIHI and LOLO
- HI and LO
- DHI and DLO
- ROC

7.97. DLO - Deviation low alarm limit value

Field type: Float

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: DLO contains the value with which the difference between A (STPT) value of

the display block and the MEAS value of the loop is compared. If the difference exceeds DLO (and deviation low alarm is not inhibited, i.e. IDLA = N), the deviation low alarm is raised.

Setting the deviation low alarm limit to zero deactivates this alarm.

Restrictions: The value of DLO should never be smaller than the DBND value defined.

7.98. DOT - DISCRETE OUTPUT BLOCK

Block-Type: DOT

Record-Type: DOT

Properties: DOT is a loop header block designed for building devices for single digital output.

It is recommended to use the DOT with the name of the output from the P&ID. The status of this output can then be vizualized on the faceplate.

The OUTA field of the DOT block can be used as a link to digital outputs in the DEVICE blocks. This is the recommended usage of DOT blocks.

See “DEVICE - DIGITAL BLOCK FOR DIN/DOT I/O” on page 81

7.99. DPL - Display decimal point location

Field type: Byte

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: DPL is the number of decimal places for numerical values to be presented on the display may be set. The decimal point location parameter may be from 0 to 5, and refers to the number of decimal places appearing to the right of the decimal. An entry of "1" means to show one decimal place as in "17.2". An entry of "0" means do not show any decimal places. These are the most common.

An entry of 5 means to show 5 decimal places as in ".12345".

Note: Only a limited amount of digits can be displayed on the group display (5, if decimal point required, 6 without decimal point).

7.100. DSP - Display the setpoint numerical value

Field type: Bit

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: DSP = Y will display both the slider for indicating setpoint (of the display block) on the faceplate and the button SETP with the values T: (target setpoint, i.e. ENA of the display block) and C: (i.e. current setpoint STPT of the display block).

7.101. DSPB - Display block in continuous loops

Field type: Byte

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: DSPB contains the number of the loop block which is designed to be display block, i.e. the block which delivers information for display purposes.

DSPB = 0 indicates the loop header block is to be displayed: no status AM button, no setpoint slider, no output bar, no setpoint button will be displayed. The measured value displayed is the MEAS value of the loop (see MEAS).

DSPB > 0 contains the number of the loop body block which is designed to be display block, i.e. to provide the setpoint, output and status for display purposes, as well as the block which is switched from MANUAL to AUTO and vice versa.

The display block has other outstanding roles, too:

- The setpoint of this block is used for determining deviation high and deviation low limit
- The output of this block (if displayed as a bar on the faceplate - see ODSP), can be manipulated by the operator from the faceplate. If in the OUT field of the display block there is the name (address) of the OUT field of an AOT block, than the input value will be immediately passed to the hardware of the AOT (manual positioning of AOT in loops, set to MANUAL - if the display block is on AUTO (AM = A), then the output value is overwritten (and if needed, passed to the hardware of the AOT every poll time of the display block)).
- The status of the display block determines the stati of slave blocks,

Restrictions: This field is in an internal format. It may only be changed by DBB! Do never use EXTOL to access this field! Remember, access of the display block of a loop is always achieved by specifying block number 9 in the expression designating the database element (e.g. R101T2011:9.K).

If the display block number is changed using DBB and the elt is currently displayed on a group display, the information displayed may not reflect the new situation. The point must remain undisplayed for a while and redisplayed until the new definition is properly reflected.

7.102. DSPB - Display Block in discrete control blocks

Field type: Byte

Field Access: R,W

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: This field is always 0 and has no meaning at the moment.

7.103. DSPN - Span of displayed values

Field type: Float

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: DSPN determines the range (span) for both trend and meter bar and pointer presentation on the operator station display (group display).

The lowest point of the displayed range is determined by DZERO.

The highest point of the displayed range is determined by (DZERO + DSPN)

Note: The displayed range, specified by DZERO and DSPN **does not need** to be a subset of the instrument range, specified by ZERO and SPAN.

DSPN must be a positive number.

7.104. DTMS - Delta Measurement

Field type: Float

Field Access: R

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: The field DTMS contains the difference of the current measurement and the measurement of the previous scan:

DTMS = MEAS(k) - MEAS(k-1)

where:

MEAS(k) = Measurement at current scan cycle
MEAS(k-1) = Measurement of previous scan cycle

7.105. DZER - Lowest displayed value

Field type: Float

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: DZERO specifies the lowest point of the displayed range for both trend and meter bar and pointer presentation on the operator station display (group display).

The range (span) of the displayed range is determined by DSPN.

The highest point of the displayed range is determined by (DZERO + DSPN)

Note: The displayed range, specified by DZERO and DSPN **does not need** to be a subset of the instrument range, specified by ZERO and SPAN.

7.106. E - Block input in CALC block and LB block

Field type: IFIELD

Field Access: R,W

Defined in Blocks: CALC, LB

Properties: This input may be a variable read from another block, a variable written by an other block, or a constant. Input E is read after input D has been read, but before input F is read.

7.107. E - Ref. abs. temperature divided by the ref. abs. pressure in MFCALC block

Field type: IFIELD

Field Access: R,W

Defined in Blocks: MFCALC

Properties: E is the reference absolute temperature divided by the reference absolute pressure: Tr/Pr,.

The **volumetric** flow of the gas is transformed to the **mass** flow at reference temperature (e.g. 0 Deg C) and reference pressure (e.g. 1 atm) using the equation

A*S(E*(B+D)/(C+F))

where:

- A is the uncompensated volumetric flow rate,
- S is the square root operator,
- B is the gauge pressure reading,
- C is the temperature in degrees F or C,
- D is the offset of the gauge pressure to absolute pressure,
- E is the reference absolute temperature divided by the reference absolute pressure: Tr/Pr,
- F is offset of temperature in degrees F to degrees Rankine (460), or offset of temperature in degrees C to degrees Kelvin (273).

For further details: See “MFCALC - COMPENSATED GAS MASS FLOW BLOCK” on page 139

7.108. EFNC - EXTOL function to be triggered

Field type: Byte

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR, DEV, DEVICE, DEVINP, DI, DEVOUT, TIMER

Properties: This field specifies the EXTOL action triggered on unit UNIT on EXTOL operation OPWK on EXTOL main step SPWK at selected (enabled) events (see ENST, ENUN, ENLE, ENDV, ENCOS, ENICS, ENDSC, ENDH, ENDL, ENTU, ENTD).

The following (integer) values for EFNC are valid:

Value	Function
1	Run the specified EXTOL operation [step] on unit
2	Kill the specified EXTOL operation on unit
3	Hold the specified EXTOL operation [step] on unit
4	Wakeup the specified EXTOL operation (default value) on unit

Table 21: EXTOL Function Codes

Triggering event see fields:

ENCOS, ENDH, ENDL, ENDSC, ENDV, ENHH, ENHI, ENISC
ENLE, ENLL, ENLO, ENRC, ENST, ENTD, ENTU, ENUN

7.109. EGUT - Engineering units

Field type: String

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR, AOT

Properties: EGUT - the engineering units descriptor may be any string of 1 to 5 characters.

When the measurement value for the point is displayed at the operator's console or is logged, the engineering units field usually follows so that the reader may know the dimensions of the data. Typical Engineering Units are:

Kg, Min., DEGC, Bar, mBar et cetera.

7.110. EI - Derivative Action

Field type: Binary

Field Access: R,W

EI is the 14th bit in the field EI (hex value 0x2000)

Defined in Blocks: SPID, NLP, PID

Properties: This specifies, whether the derivative action is calculated using the full error term (option ‘E’) or the input value (measurement) only (option ‘I’).

The following algorithm for the derivative term is applied:

derivative = KP * KD / POLL * (d(k) - d(k-1))

where:

B(k)	= B input of the SPID block ("measurement")
e(k)	= STPT(k) - B(k) in the case of direct action
e(k)	= B(k) - STPT(k)in the case of reverse action
d(k)	= either e(k) if field EI is set to ’E’, or B(k) if field EI is set to ’I’
KD	= KD value of the SPID block (derivative gain)
KP	= KP value of SPID block (proportional gain)

POLL = the poll-time of the loop

EI value is relevant only when KD not equal 0.

In case a pid algorithm is used with the derivative term and a setpoint change should not have an effect on the derivative term, then set EI=I (i.e. use Input only). In case a setpoint change should effect the derivative term, then set EI=E.

7.111. ENA - Target setpoint

Field type: Float

Field Access: R,W

Defined in Blocks: CALC, LL, NLP, SPID

Properties: ENA - target setpoint is a variable written by another block or a constant. ENA influences the behaviour of input A (setpoint STPT) of the loop body block and is also influenced by A.

The behaviour of ENA is different depending on whether the **value A** is **read in A** or **written to A** or **a constant**.

If the value A is **read in** A, then at each poll simultaneously with A (STPT) also the ENA value is set equal A.

If the value A is **written to** A or is **a constant**, then the ENA value stays as it is (set from faceplate or by initialization).

If AROC is not equal zero (0) then at each poll time the value just written in A is increased (or decreased) by the value of AROC as long as it does not exceed ENA.

If AROC equals 0, the A value is immediately set equal to ENA without respect to the value written in (or the constant value previously present)

Depending on the value of ITYP the value of ENA and simultaneously the value of A can be adjusted during initialization. If ITYP = 1 or 3, then the value ENA (and A) stays unchanged during the initialization; if ITYP = 0 or 2 then ENA (and simultaneously A) is set equal the value of B. See ITYP.

The contents of the ENA field of the display block can be changed from the faceplate using the button SETP or the setpoint-slider. The current value of ENA is displayed on the faceplate as T (target setpoint).

Restrictions: This field should not be altered directly from EXTOL, but the command `set ... setpoint to ...` must be used instaed to change a setpoint.

7.112. ENAn - Enable state n (ENA1 to ENA8)

Field type: Bit

Field Access: R,W

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: ENAn = Y enables the state n of a device.

The device cannot be driven to a not enabled state (from operator faceplate or by manipulating DEST field). An attempt to do so will result in an error message or in a runtime error in EXTOL.

If the input bit pattern matches the state's bit pattern, the actual state of the device ACT will become that state without respect to the contents of the ENAn flag.

7.113. ENCOS - Enable EXTOL action on events

Field type: Bit

Field Access: R,W

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: ENCOS = Y causes the change of state alarm to trigger the action specified in the field EFNC on the step specified in the field SPWK of the operation specified in the field OPWK on the unit specified in the field UNIT.

7.114. ENDH - Enable EXTOL action on deviation high alarm

Field type: Bit

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: ENDH = Y causes the deviation high alarm to trigger the action specified in the field EFNC on the step specified in the field SPWK of the operation specified in the field OPWK on the unit specified in the field UNIT.

7.115. ENDL - Enable EXTOL action on deviation low alarm

Field type: Bit

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: ENDL = Y causes the deviation low alarm to trigger the action specified in the field EFNC on the step specified in the field SPWK of the operation specified in the field OPWK on the unit specified in the field UNIT.

7.116. ENDSC - Enable EXTOL action on change of desired state alarm

Field type: Bit

Field Access: R,W

Defined in Blocks: DEV, DEVICE, DEVOUT

Properties: ENDSC = Y causes the change of desired state alarm to trigger the action specified in the field EFNC on the step specified in the field SPWK of the operation specified in the field OPWK on the unit specified in the field UNIT.

7.117. ENDV - Enable EXTOL action on diff. between desired and actual state alarm

Field type: Bit

Field Access: R,W

Defined in Blocks: DEV, DEVICE, DEVOUT

Properties: ENDV = Y causes the difference between desired and actual state alarm to trigger the action specified in the field EFNC on the step specified in the field SPWK of the operation specified in the field OPWK on the unit specified in the field

UNIT.

7.118. ENHH - Enable EXTOL action on high-high alarm

Field type: Bit

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: ENHH = Y causes the high-high alarm to trigger the action specified in the field EFNC on the step specified in the field SPWK of the operation specified in the field OPWK on the unit specified in the field UNIT.

7.119. ENHI - Enable EXTOL action on high alarm

Field type: Bit

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: ENHI = Y causes the high alarm to trigger the action specified in the field EFNC on the step specified in the field SPWK of the operation specified in the field OPWK on the unit specified in the field UNIT.

7.120. ENISC - Enable EXTOL action on change of input state alarm

Field type: Bit

Field Access: R,W

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: ENISC = Y causes the change of input state alarm to trigger the action specified in the field EFNC on the step specified in the field SPWK of the operation specified in the field OPWK on the unit specified in the field UNIT.

7.121. ENLE - Enable EXTOL action on loop error alarm

Field type: Bit

Field Access: R,W

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT, SENSOR, SWSENSOR, PCI, CONTR

Properties: ENLE = Y causes the loop error alarm to trigger the action specified in the field EFNC on the step specified in the field SPWK of the operation specified in the field OPWK on the unit specified in the field UNIT.

7.122. ENLL - Enable EXTOL action on low-low alarm

Field type: Bit

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: ENLL = Y causes the low-low alarm to trigger the action specified in the field EFNC on the step specified in the field SPWK of the operation specified in the field OPWK on the unit specified in the field UNIT.

7.123. ENLO - Enable EXTOL action on low alarm

Field type: Bit

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: ENLO = Y causes the low alarm to trigger the action specified in the field EFNC on the step specified in the field SPWK of the operation specified in the field OPWK on the unit specified in the field UNIT.

7.124. ENRC - Enable EXTOL action on measured value rate of change alarm

Field type: Bit

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: ENRC = Y causes the rate of change alarm to trigger the action specified in the field EFNC on the step specified in the field SPWK of the operation specified in the field OPWK on the unit specified in the field UNIT.

7.125. ENSM - Enable simulation flag

Field type: Bit

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR, DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: ENSM = Y allows this loop to be set in simulation mode by the operator, i.e. to enable him to simulate the measured value MEAS of the loop after having set the simulation falg SMFL to Y from the faceplate.

For a device type of loop this allows the operator to simulate the feedback input of the device, after havoing set the simulation falg SMFL to Y from the faceplate.

Note: When trying to simulate from EXTOL, the SMFL field has to be set to 1.

If ENSM = N, there is no switching to simultation from the faceplate possible.

DANGER: Be carful with simulating values, they may lead to problems in a real process.

7.126. ENST - Enable EXTOL action on error state alarm

Field type: Bit

Field Access: R,W

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: ENST = Y causes the error state alarm to trigger the action specified in the field EFNC on the step specified in the field SPWK of the operation specified in the field OPWK on the unit specified in the field UNIT.

7.127. ENTD - Enable EXTOL action on threshold down

Field type: Bit

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: ENTD = Y causes that MEAS passing through threshold value THLD from higher values to lower values will trigger the action specified in the field EFNC on the step specified in the field SPWK of the operation specified in the field OPWK on the unit specified in the field UNIT.

7.128. ENTU - Enable EXTOL action on threshold up

Field type: Bit

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: ENTU = Y causes that MEAS passing through threshold value THLD from lower values to higher values will trigger the action specified in the field EFNC on the step specified in the field SPWK of the operation specified in the field OPWK on the unit specified in the field UNIT.

7.129. ENUN - Enable EXTOL action on undefined state alarm

Field type: Bit

Field Access: R,W

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: ENUN = Y causes the change of undefined state alarm to trigger the action specified in the field EFNC on the step specified in the field SPWK of the operation specified in the field OPWK on the unit specified in the field UNIT.

7.130. ENWA - Enable EXTOL Action on Loop

Field type: Bit

Field Access: R,W

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT, SENSOR, SWSENSOR, PCI

Properties: ENWA = Y (=1) enables EXTOL Actions to take place, when the appropriate conditions are met. When ENWA is set to N (=0), all EXTOL Actions on this Block are disabled. When the ENWA is set to Y and there is at least one alarm condition present for which the individual EXTOL Action Enable flag is set, the EXTOL Action will immediately be triggered.

7.131. EOVR - Enable override

Field type: Bit

Field Access: R,W

Defined in Blocks: DEV, DEVICE, DEVOUT

Properties: EOVR = Y enables manipulation of the override field OVR by the operator from

the faceplate. EOVR = N disables the manipulation of the override field OVR by the operator from the console and from EXTOL and forces the override field OVR to 0.

Note: Non-zero contents of the override field (OVR) has the highest priority compared to interlock field (INLK) and to desired state (DEST): **all software safety "interlocks" realized in the database on the level of interlock (INLK) and desired state (DEST) are overwritten (=gone) by the OVR field manipulation by the operator.** That is the reason EOVR should be normally set to N.

7.132. EQN - General control equation

Field type: Binary (String max size 40 characters)

Field Access: None, except from DBB

Defined in Blocks: CALC, LB

Properties: The EQN field (40 characters long) specifies the logical and arithmetical operations which have to be performed in the CALC or LB block. The equation can use constant values and all 8 input fields of the block (i.e. A (or STPT), B,C,D,E,F and the special identifiers M (Measurement), R (Resulting output).

Variables of Equation: The following variables can be used:

- IFIELDs A (STPT), B, C, D, E
- Constant values as float numbers (no decimal point required)
- Special values identified with M -the measured value (equals value of field MEAS) and R - the last resulting output value.

Operators: The EQN can contain different operators found in the following table:

OPERATOR	DESCRIPTION	UNARY BINARY	PRIORITY	EXAMPLE
X	exponentiation base e	unary	7	XA
W	exponentiation base 10	unary	7	WA
|	absolute value	unary	7	|E
()	order of execution			(A-B)
~	negate	unary	7	~F
S	square root	unary	7	SD
L	log base 10	unary	7	L E *)
N	log base e	unary	7	NF
^	exponent	binary	7	A^B
!	NOT, logical not	unary	7	!(A>B)
*	multiplication	binary	5	A*B
/	division	binary	5	A/B
%	modulo (remainder)	binary	5	A%B
+	addition	binary	4	A+B
-	minus	binary	4	A-B
> GT	if greater than	binary	3	A>B
< LT	if less than	binary	3	A<B
# NE	if not equal	binary	3	A#B
= EQ	if equal	binary	3	A=B
} GE	if greater or equal	binary	3	A}B
{ LE	if less or equal	binary	3	A LE B
& AND	logical and	binary	2	A&B
U BAND	bit and	binary	2	AUB
V OR	logical inclusive or	binary	1	A OR B
. XOR	logical exclusive or	binary	1	A XOR B
O BOR	bit inclusive or	binary	1	AOB
\ BXOR	bit exclusive or	binary	1	A\B
_	one's complement	unary	7	_A
?:	then - else condition		8	A>B?C:D

Table 22: Operators for Equations (EQN) in CALC and LB blocks

***)** In case Log10 of field E should be calculated either have a blank between the L and the field (**L E**) or use brackets (**L(E)**). This is to distinguish it from the operator LE (less or equal).

Unary operators transform the following variable.

Binary operators can be arithmetic i.e.perform the arithmetical operation on the variables on either side.

Binary operators can be logical i.e.perform the logical operation on the variables on either side.

Binary operators can be relational, i.e to compare the two variables on either side and generate a true or false result for further use.

There are conditional operators ? (representing then) and : (representing else) in conditional expressions. They use the value of a previous (in the same line) relational expression to set the value of an output for either the true or false condition.

There can be as many parenthesis pairs, unary, binary and relational expressions as the 40 character limit permits.

Priority 1 is the lowest, priority 8 the highest priority.

Conditional expressions may be nested, but they must always be of the form if (...), then (...) else (...). It is not allowed to have only the if (...) then (...) part.

Inputs A,B,C,D,E and F may be variables fetched from another block or constants adjusted by the operator or EXTOL.

All integer operations (such as ones's complement, bit AND, etc.) are carried out after having converted the floating point number into a 32-bit wide unsigned integer. The integer is converted back after the operation into a float (internally a double) value.

Output high (OHC) and low (OLC) clamps are applied to the output of the equation to guarantee that the CALC block's output will remain within the allowable range from OLC to OHC.

The equation EQN is user defined with up to eight operands (A,B,C,D,E,F,M,R). A through F are the six CALC block inputs mentioned above. M is the measurement of the control loop (in continuous control loops) or the actual state (ACT) from the DEV block (in discrete control loops).

R, another special operand, is the previous result (See "RES - Result" on page 162) of this CALC block. These operands may be used repeatedly within the equation.

Although " GE ", " LE " take up more of the allowed 24 characters of equation space than their functionally equivalent "}", "{", etc., they are preferred to "{" and "}" because the operator console will not display "}" or "{". The meaning of the multi-letter mnemonic is also clearer. Please note that the multi-letter mnemonics must be surrounded with spaces.

After a flow control operator (?:) a further expression may be used. The following expression is valid:

A>B ? C<M ? R+D: E-D : F * B

Restrictions: It should never be attempted to change this field from EXTOL. Any attempt to do so will not result in a change of the executed equation, but the equation text displayed will have changed (the field consists of two parts internally, a string and a encoded equation.

7.133. ERES - Enable reset

Field type: Bit

Field Access: R,W

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: ERES = Y causes the reset functionality (the reset button on the Group Display) to be enabled.

7.134. F - Temperature offset in MFCALC block

Field type: IFIELD

Field Access: R,W

Defined in Blocks: MFCALC

Properties: F is offset of temperature in degrees F to degrees Rankine (460), or offset of temperature in degrees C to degrees Kelvin (273).

The **volumetric** flow of the gas is transformed to the **mass** flow at reference temperature (e.g. 0 Deg C) and reference pressure (e.g. 1 atm) using the equation

A*S(E*(B+D)/(C+F))

where:

- A is the uncompensated volumetric flow rate,
- S is the square root operator,
- B is the gauge pressure reading,
- C is the temperature in degrees F or C,
- D is the offset of the gauge pressure to absolute pressure,
- E is the reference absolute temperature divided by the reference absolute pressure: Tr/Pr,
- F is offset of temperature in degrees F to degrees Rankine (460), or offset of temperature in degrees C to degrees Kelvin (273).

For further details: See “MFCALC - COMPENSATED GAS MASS FLOW BLOCK” on page 139

7.135. F - Block input in CALC and LB block

Field type: IFIELD

Field Access: R,W

Defined in Blocks: CALC, LB

Properties: This input may be a variable read from another block, a variable written by an other block, or a constant. Input F is read after input E has been read, but before the equation is being evaluated.

7.136. FFW - Feed forward term

Field type: IFIELD

Field Access: R,W

Defined in Blocks: SPID, NLP, PID

Properties: FFW value (and NPOS value, too!) is just added onto the result of the SPID control algorithm before it is written to OUT. FFW may be a variable read from another block, a variable written by an other block, or a constant.

This is normally used to increase the stability of the controller for a process, where the output to the process is an approximately known function of the required setpoint. This can be implemented through a calculation (CALC) block prior to the SPID block. The CALC block reads the STPT of the SPID block and prepares the output value according to the approximately known function between setpoint and output. The calculated value can be written by the CALC

block into the FFW field of the SPID block, or better, read in the FFW field of the SPID block.

Example: Flow controller FLCONT.

We know that 0% position of the control valve corresponds to zero flow, by 100% position the flow is 20 m3/h.

Assume the dependence of the flow on the valve position to be linear.

The FLCONT controller can consist of a CALC block as CB1 and a SPID block as CB2.

CB1:	CALC
EQN:	A*B
A:	FLCONT:2.STPT
B:	5
OLC:	0
OHC:	100
OUT:	0
CB2:	SPID
FFW:	FLCONT:1.OUT

7.137. FILT - Filtering Factor

Field type: Float

Field Access: R,W

Defined in Blocks: NLP

Properties: The FILT field is used in conjunction with the VC (Velocity Compensation) in the NLP control block (See “NLP - NONLINEAR PID BLOCK” on page 145 and See “VC - Velocity compensation factor” on page 179). In order to be able to calculate the derivative of the input measurement, even when the signal is noisy, needs some filtering. The following method is used:

dmdt(k) = (mv(k) - mv(k-1)) / poll
dmdt(k) = FILT * (dmdt(k) - dmdt(k-1)) + dmdt(k-1)

where:

mv(k) = measurement at poll cycle k
poll = poll time in seconds
FILT = filtering value (range 0.0 to 1.0)
dmdt(k) = derivative of measurement at poll cycle k

Note that the value dmdt(k) is generated as a running average over 3 poll cycles.

7.138. FLER - Force to loop error

Field type: Bit

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR, DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: FLER = Y forces the loop in loop error - i.e. the loop is switched to MANUAL (only the analogue types of loop, not the digital devices) with "de-initialization"

and loop error is displayed on the faceplate and loop error alarm is raised.

Note: FLER is **not** reset automatically! It is the users's responsibility to clear the bit.

7.139. FSLE - Loop FS enable

Field type: Bit

Field Access: R,W

Defined in Blocks: AOT

Properties: FSLE = Y enables the loop failsafe output feature. I.e. the output is driven to the value defined in the field SAFE of the AOT Block in case of an error in the driving loop.

See "SAFE - Loop failsafe output" on page 163

7.140. HHC - High-high alarm to be critical

Field type: Binary

Field Access: R,W

HHC is the 2nd bit in the word HHC (decimal value 2, hex value 0x0002).

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: HHC = Y specifies that high-high alarm will be displayed as critical. Critical alarm will appear to the operator in RED and will take precedence over reporting of simultaneous major or minor alarms.

If two alarms of equivalent severity are present simultaneously, the displayed alarm will be the one closer to the top of this list:

- LERR
- HIHI and LOLO
- HI and LO
- DHI and DLO
- ROC

7.141. HHM - High-high alarm to be major

Field type: Binary

Field Access: R,W

HHM is the 2nd bit in the word HHM (decimal value 2, hex value 0x0002).

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: HHM = Y specifies that high-high alarm will be displayed as major alarm.

Major alarm will appear to the operator in YELLOW and will take precedence over reporting of simultaneous minor alarms. However, reporting of simultaneously present critical alarms takes precedence over reporting major alarms.

If two alarms of equivalent severity are present simultaneously, the displayed alarm will be the one closer to the top of this list:

- LERR
- HIHI and LOLO

- HI and LO
- DHI and DLO

7.142. HI - High alarm limit value

Field type: Float

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: The high alarm limit value HI is compared with the measured value MEAS. If MEAS is greater than HI, then high alarm is raised.

Restrictions: The HI alarm limit cannot be greater than the HIHI alarm value. Any attempt to set the limit above the HIHI will result in an error.

7.143. HIC - High alarm to be critical

Field type: Binary

Field Access: R,W

HIC is the 1st bit in the word HIC (decimal value 1, hex value 0x0001).

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: HIC = Y specifies that high alarm will be displayed as critical.

Critical alarm will appear to the operator in RED and will take precedence over reporting of simultaneous major or minor alarms.

If two alarms of equivalent severity are present simultaneously, the displayed alarm will be the one closer to the top of this list:

- LERR
- HIHI and LOLO
- HI and LO
- DHI and DLO
- ROC

7.144. HIHI - High-high alarm limit value

Field type: Float

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: The high-high alarm limit value HIHI is compared with the measured value MEAS. If MEAS is greater than HIHI, then high-high alarm is raised.

Restrictions: The HIHI alarm limit cannot be less than the HI alarm value. Any attempt to set the limit below the HI will result in an error.

7.145. HIM - High limit alarm to be major

Field type: Binary

Field Access: R,W

HIM is the 1st bit in the word HIM (decimal value 1, hex value 0x0001).

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: HIM = Y specifies that deviation high alarm will be displayed as major alarm.

Major alarm will appear to the operator in YELLOW and will take precedence over reporting of simultaneous minor alarms. However, reporting of simultaneously present critical alarms takes precedence over reporting major alarms.

If two alarms of equivalent severity are present simultaneously, the displayed alarm will be the one closer to the top of this list:

- LERR
- HIHI and LOLO
- HI and LO
- DHI and DLO
- ROC

7.146. IBIT - Input bit number

Field type: Byte

Field Access: R,W

Defined in Blocks: DIN

Properties: This value represents the bit number in the current 16 bit channel bank (point) on a DIN card.

Example: The DIN signal is on the 22nd physical channel on the Digital card. The point (IPNT) will be 2 and IBIT will be 6.

7.147. IDHA - Inhibit deviation high alarm

Field type: Binary

Field Access: R,W

IDHA is the 5th bit in the word IDHA (decimal value 16, hex value 0x0010).

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: IDHA = Y inhibits the deviation high alarm (raised when measured value MEAS exceeds the setpoint STPT of the display block for more than DHI).

An alarm inhibiting effect can also be achieved by widening the deviation high limit DHI to full scale or by setting deviation high limit DHI to zero (0). The use of the inhibit parameter is preferred.

7.148. IDHI - Inside Deadband High (Gap action control)

Field type: Float

Field Access: R,W

Defined in Blocks: NLP

Properties: Some processes are naturally very noisy and cause excessive control actions which are not needed. With some types of process actuators, the constant movement may shorten the life of the equipment. In other cases, excessive control action increases the instability of the control system or forces the control system

to be detuned so as to react too slowly for serious problems. The gap action capability of the NLP algorithm is designed to overcome these problems.

Gap action may be applied relative to the current setpoint by specifying a gap, or deadband around the setpoint. Control calculations are suppressed for any error that is within the gap. Normal control occurs outside the gap based on the actual error. Reset accumulation (wind-up) does not occur while the error is within the gap. The gap is specified so that measurements within the inner limits (between IDHI and IDLO) are within the gap, while measurements above and below the outer limits (above ODHI and below ODLO) are outside the gap. The space between the outer and inner limits is a deadband applied so that gap action control will not "chatter". Figure 5: "Gap Action Control" illustrates the functions of Gap Action Control and the relative positions of the limits and the setpoint. No control action will occur in zones 3 and 4. Full control action will always occur in zones 1 and 6. As a measurement signal moves from zone 3 or 4, it must enter zones 1 or 6 before control action will occur. Control action will then continue while the measurement moves through zones 2 and 5, and stop when zone 3 or 4 is entered. Zones 2 and 5 are deadbands to the Gap Action Control.

Note that ODHI, ODLO, IDHI, IDLO are values related to the setpoint, i.e. IDLO and ODLO are negative values.

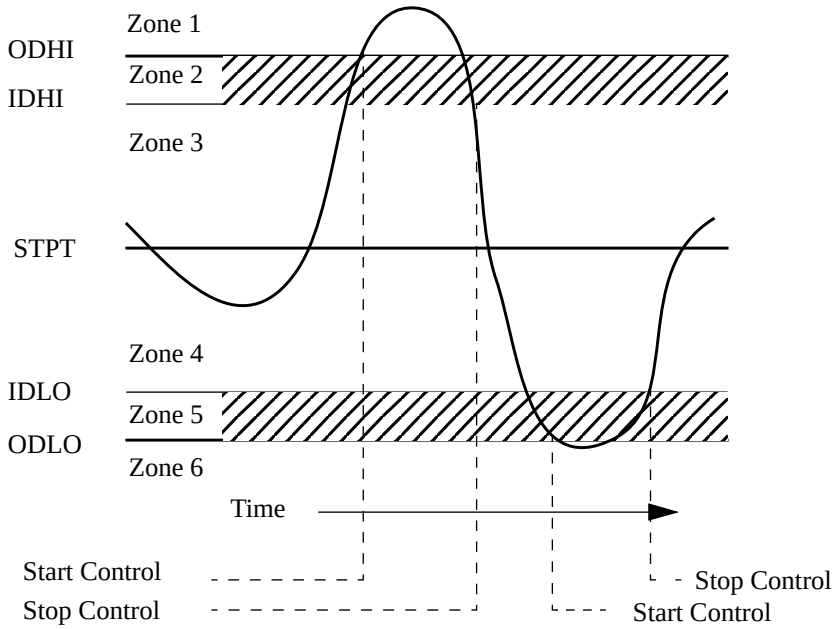


Figure 5: Gap Action Control

7.149. IDLA - Inhibit deviation low alarm

Field type:	Binary
Field Access:	R,W
	IDLA is the 6th bit in the word IDLA (decimal value 32, hex value 0x0020).
Defined in Blocks:	SENSOR, SWSENSOR, PCI, CONTR
Properties:	IDLA = Y inhibits the deviation low alarm (raised when the setpoint STPT of the display block exceeds measured value MEAS for more than DLO).

An alarm inhibiting effect can also be achieved by widening the deviation low limit DLO to full scale or by setting deviation low limit DLO to zero (0). The use of the inhibit parameter is preferred.

7.150. IDLO - Inside Deadband Low (Gap action control)

Field type: Float

Field Access: R,W

Defined in Blocks: NLP

Properties: Some processes are naturally very noisy and cause excessive control actions which are not needed. With some types of process actuators, the constant movement may shorten the life of the equipment. In other cases, excessive control action increases the instability of the control system or forces the control system to be detuned so as to react too slowly for serious problems. The gap action capability of the NLP algorithm is designed to overcome these problems.

Gap action may be applied relative to the current setpoint by specifying a gap, or deadband around the setpoint. Control calculations are suppressed for any error that is within the gap. Normal control occurs outside the gap based on the actual error. Reset accumulation (wind-up) does not occur while the error is within the gap. The gap is specified so that measurements within the inner limits (between IDHI and IDLO) are within the gap, while measurements above and below the outer limits (above ODHI and below ODLO) are outside the gap. The space between the outer and inner limits is a deadband applied so that gap action control will not "chatter". Figure 5: "Gap Action Control" illustrates the functions of Gap Action Control and the relative positions of the limits and the setpoint. No control action will occur in zones 3 and 4. Full control action will always occur in zones 1 and 6. As a measurement signal moves from zone 3 or 4, it must enter zones 1 or 6 before control action will occur. Control action will then continue while the measurement moves through zones 2 and 5, and stop when zone 3 or 4 is entered. Zones 2 and 5 are deadbands to the Gap Action Control.

Note that ODHI, ODLO, IDHI, IDLO are values related to the setpoint, i.e. IDLO and ODLO are negative values.

See “IDHI - Inside Deadband High (Gap action control)” on page 103 See “Gap Action Control” on page 104

7.151. IHHA - Inhibit high-high alarm

Field type: Binary

Field Access: R,W

IHHA is the 2nd bit in the word IHHA (decimal value 2, hex value 0x0002).

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: IHHA = Y inhibits the high-high alarm.

A temporary alarm inhibiting effect can also be achieved by setting the alarm limit values to the end of the scale (99999).

7.152. IHIA - Inhibit high alarm

Field type: Binary

Field Access: R,W

IHIA is the 1st bit in the word IHIA (decimal value 1, hex value 0x0001).

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: IHIA = Y inhibits the high alarm.

A temporary alarm inhibiting effect can also be achieved by setting the alarm limit values to the end of the scale (99999).

7.153. ILC - Integral limiting continuous

Field type: Binary

Field Access: R,W

ILC is the 15th bit in the field ILC (decimal value 16384, hex value 0x4000)

Defined in Blocks: SPID, NLP

Properties: ILC = Y causes the integral calculation and limitation in the SPID algorithm to be performed continuously.

ILC = N causes the integral calculation and limitation in the SPID algorithm to be applied only when the measurement is within a band of +/- ILT1 of the set-point.

This function can be used to switch a controller from single P-control to PI-control automatically, when the measurement gets close to the setpoint. This avoids the problem of integral wind-up.

7.154. ILEA - Inhibit loop error alarm

Field type: Binary

Field Access: R,W

ILEA is the 8th bit in the word ILEA (decimal value 128, hex value 0x0080).

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: ILEA = Y inhibits the loop error alarm.

7.155. ILLA - Inhibit low-low alarm

Field type: Binary

Field Access: R,W

ILLA is the 4th bit in the word ILLA (decimal value 64, hex value 0x0008).

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: ILLA = Y inhibits the low-low alarm.

A temporary alarm inhibiting effect can also be achieved by setting the alarm limit value to the end of the scale (ZERO).

7.156. ILOA - Inhibit low alarm

Field type: Binary

Field Access: R,W

ILOA is the 3th bit in the word ILOA (decimal value 4, hex value 0x0004).

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: ILOA = Y inhibits the low limit alarm.

A temporary alarm inhibiting effect can also be achieved by setting the alarm limit value to the end of the scale (ZERO).

7.157. ILT - Integral limiting action type

Field type: Byte

Field Access: R,W

Defined in Blocks: SPID, NLP

Properties: ILT value determines the way integral term of the SPID algorithm will be handled. This allows to specify the desired form of wind-up-protection.

ILT=0

Integral term is determined according to its specification

iterm(k) = iterm(k-1) + KP * KI * POLL * e(k)

where:

B = B input of the SPID block ("measurement")

e(k) = STPT(k) - B(k) in the case of direct action

e(k) = B(k) - STPT(k) in the case of reverse action

fKP = fKP = KP für KP # 0, fKP = 1 für KP = 0

KI = KI value of the SPID block (integral gain)

KP = KP value of SPID block (proportional gain)

POLL = poll time of the loop

STPT = STPT value of the SPID block ("setpoint")

For the initialization of the algorithm See “SPID - SPID BLOCK” on page 169 or See “NLP - NONLINEAR PID BLOCK” on page 145

Option ILT = 0 does not provide any protection against reset wind-up.

ILT=1

ILT = 1 limits the integral term so that output never exceeds the limits of the OHC or OLC. When OHC or OLC are exceeded with the proportional part of the SPID algorithm alone, then the integral term is set to 0.
This behaviour of integral limitation can be disadvantageous when the output is mainly formed by the integral part and short large disturbances occurr. In such cases, the behaviour of the controller becomes unacceptable. In such cases consider using ILT-type 3.
ILT = 1 is usually used in normal non-cascade loops or in the slave loop of a cascade.

ILT=2

ILT = 2 stops integration (i.e. iterm(k) = iterm(k-1)) whenever the downstream loop of the cascade is saturated (i.e. the control algorithm is currently limiting the integration). Otherwise it behaves as for ILT = 1 limitation.
ILT = 2 is usually used in a master loop of a cascade (cascade formed using a CTB block).

ILT=3

ILT = 3 stops the integration (i.e. iterm(k) = iterm(k-1)) whenever the controller output reaches OHC or OLC, but the integral term is preserved.

ILT = 3 is usually used for non-cascade loops or for slave loops of a cascade.

ILT=4

ILT = 4 stops integration (i.e. iterm(k) = iterm(k-1)) whenever the downstream loop of the cascade is saturated (i.e. the control algorithm is currently limiting the integration). Otherwise it behaves as for ILT = 3 limitation. ILT = 4 is usually used a master loop with a cascade structure

7.158. ILT - Integral limiting action type (PID block)

Field type: Byte

Field Access: R,W

Defined in Blocks: PID

Properties: ILT value determines the way integral term of the PID algorithm (TCS compatible) will be handled. This allows to specify the desired form of wind-up-protection.

ILT=0

Integral term is determined according to its specification

iterm(k) = iterm(k-1) + KP * KI * POLL * e(k)

where:

B	= B input of the PID block ("measurement")	
e(k)	= STPT(k) - B(k)	in the case of direct action
e(k)	= B(k) - STPT(k)	in the case of reverse action
fKP	= fKP = KP für KP # 0, fKP = 1 für KP = 0	
KI	= KI value of the PID block (integral gain)	
KP	= KP value of PID block (proportional gain)	
POLL	= poll time of the loop	
STPT	= STPT value of the PID block ("setpoint")	

The algorithm is initialised

e(k-1) = 0 and

iterm (k-1) = 0 for ITYP = 0 and 1

iterm (k-1) = OUTfor ITYP = 2 and 3

all the time.

Option ILT = 0 does not provide any protection against reset wind-up.

ILT=1

Type 1 Integral Limiting Action uses any downstream output to measure the performance of the control loop downstream of the block. As that output begins to deviate from the loops actual internally calculated result (RES), the term becomes larger and decreases the effect of the reset accumulation. This correc-

tion may be applied continuously (ILC = "Y") or it may be applied only when the block output limits or output rate of change limit is detected (ILC = "N"). Continuous application to the block output simulates an analog controller with reset wind-up protection.

ILT = 1 is usually used in normal non-cascade loops or in the slave loop of a cascade.

For type 1 action, the equation used to calculate the integral limiting term is:

intg = KI * POLL
if intg > 1.0, then intg = 1.0

In case of Incremental Algorithm:

ILTerm = intg * ILT1

In case of Positional Algorithm:

ILTerm = intg * (ILT1 - LR)

where:

KI = Integral gain (repeats /sec)
POLL = Scan Time or Poll time
ILT1 = Integral Limiting Input one
LR = Last Result, i.e. this is y(k) of the previous scan

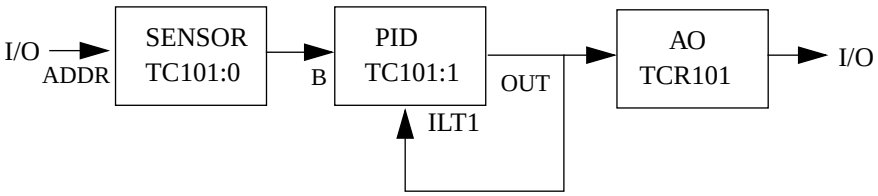


Figure 6: Type 1 Integral Limiting Action in PID block

ILT=2

Type 2 Integral Limiting Action uses either of two different points to detect and compensate for excessive reset wind-up. The first feedback input (ILI1) provides data for normal operations only. When an output limiting condition is detected, the second feedback input (ILI2) is used. The option for action only on limit condition has no effect. This option would only be used to decouple the effects of the interaction that occurs when complex cascades are tuned.

For type 2 action, the equation used to calculate the integral limiting term is:

We have to differentiate two situations, when the downstream loop is already limited:

intg = KI * POLL
if intg > 1.0, then intg = 1.0

In case of Incremental Algorithm:

ILTerm = intg * ILT2

In case of Positional Algorithm:

ILTerm = intg * (ILT2 - LR)

where:

KI = Integral gain (repeats /sec)
POLL = Scan Time or Poll time
ILT2 = Integral Limiting Input two
LR = Last Result, i.e. this is y(k) of the previous scan

When the downstream loop is not limited:

Type 1 limiting is applied, in this case the value of ILC is again taken into account.

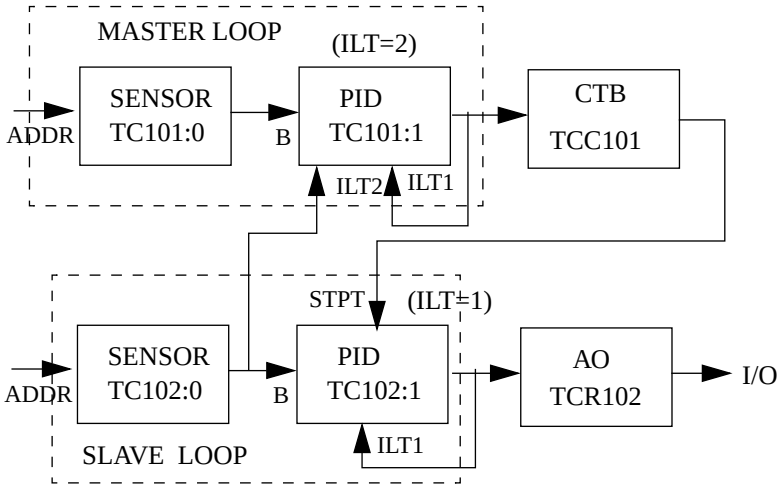


Figure 7: Type 2 Integral Limiting Action in PID block

ILT=3

Type 3 Integral Limiting Action is specifically designed for the elimination of the open loop wind-up originating from feedforward control schemes when the primary control loop is out of control. In this mechanism, two different feedback points may be used to compensate for excessive wind-up in the primary loop. One of these (ILI1) is typically the output of the primary or any downstream loop. The other (ILI2) is typically the output of a feedforward calculation. Since feedforward calculations are somewhat sensitive to process variations, the application of type 3 integral limiting action is almost always set to occur when there is an output limiting condition (ILC = "Y").

For type 3 action, the equation used to calculate the integral limiting term is:

intg = KI * POLL
if intg > 1.0, then intg = 1.0

In case of Incremental Algorithm:

ILTerm = intg * (ILT1 - ILT2)

In case of Positional Algorithm:

ILTerm = intg * (ILT1 - ILT2 - LR)

where:

KI = Integral gain (repeats /sec)
POLL = Scan Time or Poll time
ILT1 = Integral Limiting Input one
ILT2 = Integral Limiting Input two
LR = Last Result, i.e. this is y(k) of the previous scan

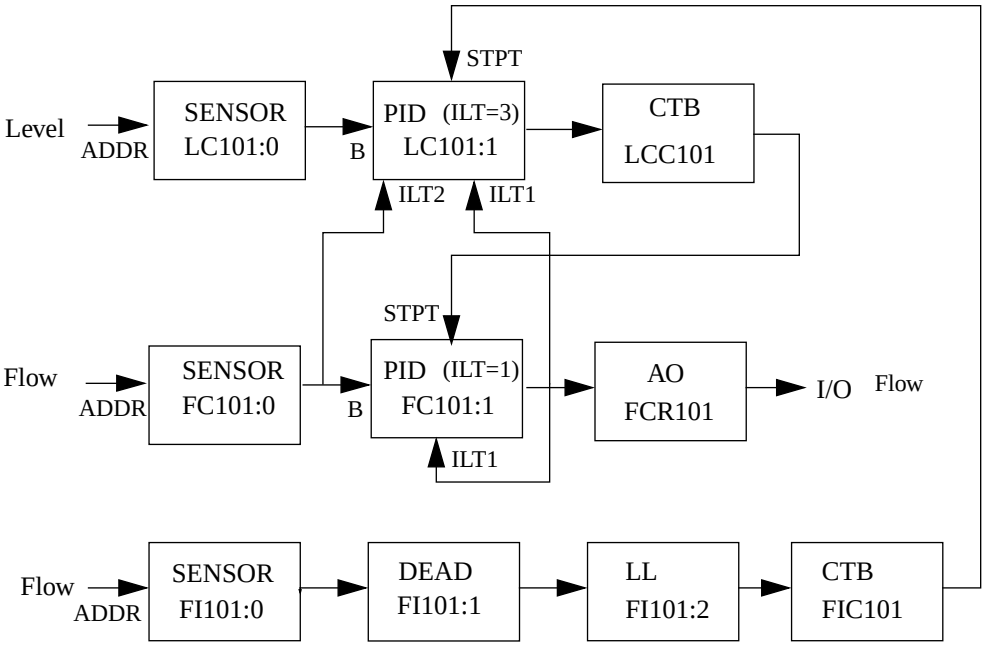


Figure 8: Type 3 Integral Limiting Action

ILT=4

ILT = 4 limits the integral term so that output never exceeds the limits of the OHC or OLC. When OHC or OLC are exceeded with the proportional part of the PID algorithm alone, then the integral term is set to 0.

This behaviour of integral limitation can be disadvantageous when the output is mainly formed by the integral part and short large disturbances occurr. In such cases, the behaviour of the controller becomes unacceptable. In such cases consider using one of the lower types.

It is essential that this type of limitation is used when feedforward is not equal to zero i.e. feed forward control is in effect.

ILT=5

ILT = 5 stops integration (i.e. iterm(k) = iterm(k-1)) whenever the downstream loop of the cascade is saturated (i.e. the control algorithm is currently limiting the integration). Otherwise it behaves as for ILT = 4 limitation.

ILT = 5 is usually used in a master loop of a cascade (cascade formed using a CTB block).

7.159. ILT1 - Halfband for integration

Field type: IFIELD

Field Access: R,W

Defined in Blocks: SPID, NLP

Properties: ILT1 specifies the band width (+/- ILT1 of the setpoint) within which the integral calculation and limitation of the SPID algorithm takes place. ILT1 is used (i.e. its value matters) only if ILC = N.

This function can be used to switch a controller from single P-control to PI-control automatically, when the measurement gets close to the setpoint. This avoids the problem of integral wind-up.

7.160. ILT1 - Halfband for integration in PID algorithm

Field type: IFIELD

Field Access: R,W

Defined in Blocks: PID

Properties: Input field for Integral Limitation purposes. The Integral Limitation functionality is described further in the field ILT (See “ILT - Integral limiting action type (PID block)” on page 108). This field is used in connection with Integral Limiting Types 1,2,3.

7.161. ILT2 - Halfband for integration in PID algorithm

Field type: IFIELD

Field Access: R,W

Defined in Blocks: PID

Properties: Input field for Integral Limitation purposes. The Integral Limitation functionality is described further in the field ILT (See “ILT - Integral limiting action type (PID block)” on page 108). This field is used in connection with Integral Limiting Types 2,3.

7.162. INA - Digital input address

Field type: HWLINK

Field Access: Only accessable via DBB

Defined in Blocks: DEV, DEVINP, DI, DIN

Properties: Logical address of the digital input card. The format is:

- #rack-card
- #module

7.163. INC - Increment

Field type:	Float
Field Access:	R,W
Defined in Blocks:	TIMER
Properties:	The increment value will be added or subtracted from the value in the TIME field, after each execution of the TIMER block.

7.164. INCOS - Inhibit change of state alarm

Field type:	Binary
Field Access:	R,W
	INCOS is the 5th bit in the word INCOS (decimal value 16, hex value 0x0010).
Defined in Blocks:	DEV, DEVICE, DEVINP, DI
Properties:	INCOS = Y inhibits the change of state alarm.

7.165. INDCS - Inhibit desired state change alarm

Field type:	Binary
Field Access:	R,W
	INDCS is the 7th bit in the word INDCS (decimal value 64, hex value 0x0040).
Defined in Blocks:	DEV, DEVICE, DEVOUT
Properties:	INDCS = Y inhibits the change of desired state (DEST or INLK or OVR) alarm.

7.166. INDV - Inhibit difference between actual and desired state alarm

Field type:	Binary
Field Access:	R,W
	INDV is the 1st bit in the word INDV (decimal value 1, hex value 0x0001).
Defined in Blocks:	DEV, DEVICE
Properties:	INDV = Y inhibits the difference between actual and desired state alarm.
	INDV = N is not sufficient to achieve processing of difference alarm: there must be also a non-zero value in the field OPTI (stroke time), indicating the number of scan cycles to count down before comparing the actual state with the due state (DEST or INLK or OVR)

7.167. INICS - Inhibit input change of state alarm

Field type:	Binary
Field Access:	R,W
	INICS is the 6th bit in the word INICS (decimal value 32, hex value 0x0020).
Defined in Blocks:	DEV, DEVICE, DEVINP, DI
Properties:	INICS = Y inhibits the input change of state alarm.

7.168. INIT - Block initialization flag

Field type: Bit

Field Access: R,W

Defined in Blocks: CALC, DELAY, LL, MFCALC, NLP, SPID, PID, SPLIT

Properties: INIT = Y specifies that the block will be initialized when processed next. After the initialization has been performed, the INIT flag is cleared to No (N or 0) and the block is set to AUTO (i.e. its AM flag is set to Y) by the system.

"Initialization" is performed even if the block with INIT = Y has been already in AUTO mode (AM = A).

The effect of setting INIT of the display block to Y or 1 is very similar to switching the loop to AUTO from the faceplate or from EXTOL by the instruction

set loop ... to auto".

Note: The INIT field can not be changed from the operator console. The INIT field is always reset (set to 0, i.e. to N) by the system itself.

The usage of INIT and AM fields can be demonstrated on the following example

Example: Assume loop LOOPS should behave according to LOOPM:

if	LOOPM has just been switched from MANUAL to AUTO,
then	LOOPS should be initialized
endif	

if	LOOPM has just been switched from AUTO to MAN,
then	LOOPS should be de-initialized (set to MAN via flag TMAN)
endif	

Assume, the display block of LOOPM is block 3, the display block of LOOPS is block 5.

The task can be solved including two CALC blocks to LOOPS, e.g.

CB6:

EQN:	A=B&C=D?E:F	If
A:	LOOPS:6.C	mode of LOOPM **during the last poll**
B:	0	was MANUAL and
C:	LOOPM:3.AM	mode of LOOPM **now**
D:	1	is AUTO
E:	1	then set 1
F	**LOOPS:5.INIT**	else leave unchanged
OUT: LOOPS	**LOOPS:5.INIT**	the INIT flag of the display block of
OLC:	0	
OHC:	1	

CB7:

EQN:	A=B&C=D?E:F	If
A:	LOOPS:7.C	mode of LOOPM **during the last poll**

B:	1	was AUTO and
C:	LOOPM:3.AM	mode of LOOPM **now**
D:	0	is MANUAL
E:	1	then set 1
F	LOOPS:5.TMANelse leave unchanged	
OUT:	LOOPS:5.TMANthe TMAN flag of the display block of LOOPS	
OLC:	0	
OHC:	1	

This technique works reliably even if the two loops involved have completely different POLL times and PHASes.

7.169. INLE - Inhibit loop error alarm

Field type: Binary

Field Access: R,W

INLE is the 4th bit in the word INLE (decimal value 8, hex value 0x0008).

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: INLE = Y inhibits the loop error alarm.

7.170. INLK - Interlock state

Field type: Byte

Field Access: R,W

Defined in Blocks: DEV, DEVICE, DEVOUT

Properties: The non-zero state number specified in the field INLK (interlock state) takes precedence over any state specified in the field DEST (desired state) - i.e.the system tries to drive the device to the interlock state. When the interlock state (the field INLK) is set to zero, the original desired state specified in the field DEST will become the due state again. State number are in the range 1 to 8.

The non-zero state specified in the field OVR (override state) has precedence over any interlock state and desired state.

7.171. INM - Input mask

Field type: Word

Field Access: R,W

Defined in Blocks: DEV, DEVINP, DI

Properties: INM contains the hexadecimal mask applied to the value read from the input DIN card to isolate the bits used in this block prior to determining the state by comparing the masked input to the state bit patterns in the fields S1I to S8I until a match is detected.

Example: In a DEV block the inputs are physically connected to the card specified in the field INA to the 16 bit bank specified in the field IPNT.

Only bits 6,7,8,9,10 and 11 "belong" to the DEV block considered, other bits

"belong" to other devices. This is reflected in the values of SnI, which contain the bit combinations corresponding to the states of our device, and which encompass only bits 6 to 10. Assume, our device has three defined states:

OFF	- S1I = 0060 h	i.e. '0000 0**000 011**0 0000'
STATE1	- S2I = 0180 h	i.e. '0000 0**001 100**0 0000'
STATE2	- S3I = 0600 h	i.e. '0000 0**110 000**0 0000'

The field INM should contain bits 6,7,8,9,10 and 11 set,

i.e. INM = 003C h i.e. '0000 0**111 111**0 0000'

Suppose the 16 bit hardware point contains the value

A61A h i.e. '1010 0**110 000**1 1010'

(not bold bits on the bank considered belong to other devices).

The processing goes as follows: From the 16 bit pattern of the hardware point the bits "belonging" to our device are masked out by BAND operation with the INM. The result is

0600 h i.e. '0000 0**110 000**0 0000'

The comparison with the fields SnI of this device shows match with STATE2. Number 3 will be stored in the field ACT by the DCEX package. Simple, isn't it

7.172. INP - Input field of the CONTR block

Field type: IFIELD

Field Access: R,W

Defined in Blocks: CONTR

Properties: This field serves as the input for analog values from other fields of the process database. The value can be either read in the INP field (recommended technique), or written into the INP field by the OUT field of an other block (not recommended technique). This value is then used as the measurement value (stored in field MEAS). Therefore further references to this value should be made through the field MEAS.

7.173. INP - Input field of the SPLIT block

Field type: IFIELD

Field Access: R,W

Defined in Blocks: SPLIT

Properties: The value of INP is used to calculate the two outputs OUT1 and OUT2 of the SPLIT block according to the value of split type TYPE chosen. The INP value is usually the output value (OUT field) of an SPID block. The value is expected to be in the range 0 to 100.

7.174. INP - Input field of the SWSENSOR block

Field type: IFIELD

Field Access: R,W

Defined in Blocks: SWSENSOR

Properties: This field serves as the input for analog values from other fields of the process database. The value can be either read in the INP field or written into the INP field by the OUT field of an other block.

The value is not processed , but just used as an ordinary value in engineering units. SPAN and ZERO are not used to process the value written to INP or read in intp INP.

7.175. INPn - n-th Input of the DEVICE block (INP1 to INP8)

Field type: IFIELD

Field Access: R,W (can only be changed by DBB)

EXTOL may change the links using the instruction LINKIO.

Defined in Blocks: DEVICE

Properties: The input field INPn of the DEVICE block contains the read-address of a data-base field. INPn becomes so linked to other process database field and reads the value from there. The value read must be in the range 1 to 2. An input value is typically obtained from the field IVAL of a DIN block or from the field ACT of another DEV or DEVICE block.

The value read in will be transformed so that 2 -> 1 and 1 -> 0.

The transformed (logical) values is then interpreted as the value of the n-th bit in the 8 bit imput field, which is compared with the bit patterns of the fields S1I to S8I to find out the ACT state of the DEVICE.

Example: Assume, our device has three defined states:

OFF	- S1I = 0003 h	i.e. '0000 0000 0000 0011'
STATE1	- S2I = 000C h	i.e. '0000 0000 0000 1100'
STATE2	- S3I = 0030 h	i.e. '0000 0000 0011 0000'

Assume the values read in INP1 to INP6 are

INP1 -	1
INP2 -	1
INP3 -	1
INP4 -	1
INP5 -	2
INP6 -	2

These inputs will be transformed to the bit pattern

0030 h i.e. "0000 0000 0011 0000"

and the ACT field gets the value 3, corresponding to STATE2 of the device.

7.176. INST - Inhibit error status alarm

Field type: Binary

Field Access: R,W

INST is the 2nd bit in the word INST (decimal value 2, hex value 0x0002).

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: INST = Y inhibits the error status alarm.

7.177. INUN - Inhibit undefined state alarm

Field type: Binary

Field Access: R,W

INUN is the 3rd bit in the word INUN (decimal value 4, hex value 0x0004).

Defined in Blocks: DEV, DEVICE, DEVINP, DI

Properties: INUN = Y inhibits the undefined state alarm.

7.178. INV - Inversion flag in digital inputs (DIN)

Field type: Bit

Field Access: R,W

Defined in Blocks: DIN

Properties: If INV = N, then the 1st state corresponds to value IVAL = 0, the 2nd state to value IVAL = 1.

If INV = Y the inversion takes place: the 1st state corresponds to IVAL = 1 and the 2nd state to IVAL = 0.

7.179. INV - Inversion flag in digital outputs (DOT)

Field type: Bit

Field Access: R,W

Defined in Blocks: DOT

Properties: If INV = N, then the 1st state corresponds to value 0 output, the 2nd state to value 1 output.

If INV = Y the inversion takes place: the 1st state corresponds to output = 1 and the 2nd state to output = 0.

7.180. INV - Output inversion in AOT blocks

Field type: Bit

Field Access: R,W

Defined in Blocks: AOT

Properties: INV = Y inverts the output, i.e. zero value in OUT field of AOT will cause the highest value to be sent to the analog output channel.

7.181. IPNT - Input point

Field type: Byte

Field Access: R,W

Defined in Blocks: DEV, DEVINP, DI, DIN

Properties: Bank selector on the digital input card. Each digital input card may have up to four (4) channel banks of 16 bit channels. The channel banks are consecutively numbered from 1 to 4. Note, however, that all references within a device block can only be made to one 16 bit channel bank.

Example: The DIN signal is on the 22nd physical channel on the Digital card. The point (IPNT) will be 2 and IBIT will be 6.

7.182. IRCA - Inhibit measured value rate of change alarm

Field type: Binary

Field Access: R,W

IRCA is the 7th bit in the word IRCA (decimal value 64, hex value 0x0040).

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: IRCA = Y inhibits the measured value rate of change alarm.

A temporary alarm inhibiting effect can also be achieved by setting the alarm limit value to full scale. The use of the inhibit parameter is preferred.

7.183. ITYP - Initialization type for control blocks

Field type: Byte

Field Access: R,W

Defined in Blocks: SPID, NLP, PID

Properties: This parameter defines the different types of initialization performed when the block is switched from MANUAL to AUTO

- from the operator console (display block only)
- as a slave block when the master (display) block of the loop has been switched to AUTO from the faceplate or by setting INIT = 1
- by setting the flag INIT of this block to Y.

The following list defines the initialization types:

ITYP	integral term	STPT	OUT
0	0	input B (measurement)	from output link
1	0	unchanged	from output link
2	OUT	input B (measurement)	from output link
3	OUT	unchanged	from output link
4	OUT-FFW-NPOS	input B	from output link
5	OUT-FFW-NPOS	unchanged	from output link

Table 23: Initialization Types for SPID, NLP

If the SPID, NLP block is the display block and is set to MANUAL, then the output of the loop may be positioned from the group display (faceplate). With the initialization types 2 and 3, when switched to AUTO, the controller output will control starting from the current position of the output.

Figure 9: Initialization Types for PID

ITYP	integral term	STPT	OUT
0	0	input B (measurement)	from output link
1	0	unchanged	from output link

7.184. ITYP - Initialization type control blocks

Field type: Byte

Field Access: R,W

Defined in Blocks: CALC, LL

Properties: This parameter defines the different types of initialization performed when the block is switched from MANUAL to AUTO

- from the operator console (display block only)
- as a slave block when the master (display) block of the loop has been switched to AUTO from the faceplate or by setting INIT = 1
- by setting the flag INIT of this block to Y.

The following lists defines the initialization types:

ITYP	STPT	OUT	Result
0	input B (measurement)	from output link	from output link
1	unchanged	0.0	0.0
2	from output link	0.0	0.0
3	unchanged	from output link	from output link
4	unchanged	from B	from B

Table 24: Initialization Types for CALC

ITYP	STPT	Last Input (An-1)	Result
0	0.0	0.0	0.0
1	unchanged	from STPT	from output link

Table 25: Initialization Types for LL

ITYP	STPT	OUT	Result
0	unchanged	unchanged	unchanged
1	unchanged	0.0	0.0

Table 26: Initialization Types for LB

7.185. IVAL - Input value in DIN

Field type:	Byte
Field Access:	R,W
Defined in Blocks:	DIN
Properties:	IVAL contains the actual value of the block (actual status of the DIN). This is in the range 1 to 2. IVAL value can be read by other database blocks.

7.186. IVAL - Simulated input value in digital devices

Field type:	Byte
Field Access:	R,W
Defined in Blocks:	DEV, DEVICE, DEVINP, DI
Properties:	IVAL is copied in the field ACT as actual state if the flag SMFL is set. It represents the state value (1->8) we want to simulate.

7.187. K - Gain of Lead Lag Block

Field type: IFIELD

Field Access: R,W

Defined in Blocks: LL

Properties: The operation of the lead/lag block uses the equation:

R = [K*(LEAD/Ts)*(A-La) + K*A +(LAG/Ts)*Lr] / ((LAG/TS)+1)

where:

R = Result
Lr = Last Result , R(n-1)
K = Gain
A = Input (or setpoint if used as a controller)
La = Last input, A(n-1)
LEAD = Lead time
LAG = Lag time
Ts = Time scan

7.188. KD - Derivative (rate) tuning (constant)

Field type: IFIELD

Field Access: R,W

Defined in Blocks: SPID, NLP, PID

Properties: KD is the derivative tuning constant value. It is a constant, a value read in or a value written to this field.

The derivative term is provided in the classical PID control equation to compensate for sudden deviations of the process. This is a form of anticipatory control; take action before the process actually deviates too much. In actual practice, derivative control is often too sensitive to random noise and causes too much instability. This is corrected by tuning the derivative constant, KD, to a smaller value than unity, typically 0.2 or less. Unfortunately, the actual derivative gain is the product of KP and KD which causes the control to become even less stable when KP is increased.

For control engineers which are more familiar with the term 'derivative time TD' has to be said that KD is exactly the same.

KD = TD

7.189. KI - Integral (reset) tuning constant

Field type: IFIELD

Field Access: R,W

Defined in Blocks: SPID, NLP, PID

Properties: KI is the integral tuning constant value. It is a constant, a value read in or a value written to this field.

The integral term is provided to eliminate the offset error which would occur if proportional gain would only be applied. The integral term also provides an output to correct errors that persist for longer periods of time. The longer an error persists, the larger the output will become. This requires that the integral term

have memory between control cycles called the "reset remainder". This is the term that will "wind up", or grow larger if the output of the control loop is decoupled from the process in any way. This decoupling may occur from output linkages (cascades) that are opened or by a control valve that is fully opened or closed rendering it ineffective.

The dimensions of the SyCOS integral term are in "1 / second", which means that as the number for KI grows larger, the more effective the integral action will be. The effect of a larger value for KI will be to damp or slow down the action of the control block after the first effects of an error are compensated. To some degree, the interaction with KP provides additional stability since the effect of increasing KP is to also increase the actual integral gain which is in effect KP * KI. (See “ILT - Integral limiting action type” on page 107 and “ILC - Integral limiting continuous” on page 106)

For control engineers which are more familiar with the term 'reset time TR' here is the relationship between KI and reset time.

KI = 1 / TR

KNL - Error squared limit parameter

KNL is the maximum absolute error to be used for the error square calculation. It is a constant, a value read in or a value written to this field.

The NLP algorithm allows an external limitation of the error squared gain applied to the proportional or integral terms to be specified. If the absolute value of the error exceeds KNL, then the error squared function is defeated. This helps to stabilize the error squared algorithm option. If KNL is set to zero (the default), the effect of the error square option is defeated. If KNL is set to the span, the error squared option applies over the entire control span.

7.190. KNL - Nonlinear Scalar

Field type: IFIELD

Field Access: R,W

Defined in Blocks: NLP

Properties: The KNL field is used to specify a scalar value for calculation of the absolute error term (|e| / KNL), which is used as a multiplication factor on either the KP or the KI gain. If KLN is equal to 0, or if the absolute error (|e|) is greater than KNL, the absolute error term is set to 1.

This results in:

if |e(k)| <= KNL

e(k) = e(k) * abs (e(k)) /KNL

if |e(k)| > KNL

e(k) = e(k)

See “NLP - NONLINEAR PID BLOCK” on page 145

7.191. KP - Proportional gain (constant)

Field type: IFIELD

Field Access: R,W

Defined in Blocks: SPID, NLP, PID

Properties: KP is the proportional gain tuning constant value. It is a constant, a value read in or a value written to this field.

The Proportional Gain is the overall sensitivity adjustment for the control block. The Proportional term of the SPID provides an output correction which is proportional to the size of the error. This is a gain term. Increases in KP will cause more rapid response to error conditions, and if set too high, may cause control instability. The proportional gain in SyCOS is interactive with the integral (reset) and derivative (rate) terms, which means that a change in the value of KP may then also require adjustments to the reset and rate constants.

For control engineers which are more familiar with the term 'proportional band PB' here is the relationship between KP and proportional band.

KP = (100 / PB) * (100 / SPAN)

7.192. LACOS - Change of state alarm to be latched

Field type: Binary

Field Access: R,W

LACOS is the 5th bit in the word LACOS (decimal value 16, hex value 0x0010).

Defined in Blocks: DEV, DEVICE, DEVINP, DI

Properties: LACOS = Y specifies that change of state alarm will be latched.

A latched alarm stays on the screen even after the alarm has cleared. A latched defunct alarm blinks in dark blue and must be acknowledged to clear on the displays. Any active alarm will take display precedence over a latched defunct alarm.

Unlatched alarms will disappear from the display screen if the underlying condition which caused the alarm goes away even if the alarm was not acknowledged.

Alarm latching ensures, that the operator will see alarms even if they are momentary in nature.

7.193. LADCS - Change of desired state alarm to be latched

Field type: Binary

Field Access: R,W

LADCS is the 7th bit in the word LADCS (decimal value 64, hex value 0x0040).

Defined in Blocks: DEV, DEVICE, DEVOUT

Properties: LADCS = Y specifies that change of desired state alarm will be latched.

A latched alarm stays on the screen even after the alarm has cleared. A latched defunct alarm blinks in dark blue and must be acknowledged to clear on the displays. Any active alarm will take display precedence over a latched defunct alarm.

Unlatched alarms will disappear from the display screen if the underlying condition which caused the alarm goes away even if the alarm was not acknowledged.

Alarm latching ensures, that the operator will see alarms even if they are momentary in nature.

7.194. LADH - Deviation High alarm to be latched

Field type: Binary

Field Access: R,W

LADH is the 4th bit in the word LADH (decimal value 16 hex value 0x0010).

Defined in Blocks: SENSOR, SWSENSOR, PCI CONTR

Properties: LADH = Y specifies that deviation high alarm will be latched.

A latched alarm stays on the screen even after the alarm has cleared. A latched defunct alarm blinks in dark blue and must be acknowledged to clear on the displays. Any active alarm will take display precedence over a latched defunct alarm.

Unlatched alarms will disappear from the display screen if the underlying condition which caused the alarm goes away even if the alarm was not acknowledged.

Alarm latching ensures, that the operator will see alarms even if they are momentary in nature.

7.195. LADL - Deviation Low alarm to be latched

Field type: Binary

Field Access: R,W

LADL is the 5th bit in the word LADL (decimal value 32 hex value 0x0020).

Defined in Blocks: SENSOR, SWSENSOR, PCI CONTR

Properties: LADL = Y specifies that deviation low alarm will be latched.

A latched alarm stays on the screen even after the alarm has cleared. A latched defunct alarm blinks in dark blue and must be acknowledged to clear on the displays. Any active alarm will take display precedence over a latched defunct alarm.

Unlatched alarms will disappear from the display screen if the underlying condition which caused the alarm goes away even if the alarm was not acknowledged.

Alarm latching ensures, that the operator will see alarms even if they are momentary in nature.

7.196. LADV - Difference between actual and desired state alarm to be latched

Field type: Binary

Field Access: R,W

LADV is the 1st bit in the word LADV (decimal value 1, hex value 0x0001).

Defined in Blocks: DEV, DEVICE

Properties: LADV = Y specifies that difference between actual and desired state alarm will be latched.

A latched alarm stays on the screen even after the alarm has cleared. A latched defunct alarm blinks in dark blue and must be acknowledged to clear on the displays. Any active alarm will take display precedence over a latched defunct alarm.

Unlatched alarms will disappear from the display screen if the underlying condi-

tion which caused the alarm goes away even if the alarm was not acknowledged.

Alarm latching ensures, that the operator will see alarms even if they are momentary in nature.

7.197. LAG - Lag

Field type: IFIELD

Field Access: R,W

Defined in Blocks: LL

Properties: Lag time in seconds.

7.198. LAHH - High High alarm to be latched

Field type: Binary

Field Access: R,W

LAHH is the 2nd bit in the word LAHH (decimal value 2, hex value 0x0002).

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: LAHH = Y specifies that high high alarm will be latched.

A latched alarm stays on the screen even after the alarm has cleared. A latched defunct alarm blinks in dark blue and must be acknowledged to clear on the displays. Any active alarm will take display precedence over a latched defunct alarm.

Unlatched alarms will disappear from the display screen if the underlying condition which caused the alarm goes away even if the alarm was not acknowledged.

Alarm latching ensures, that the operator will see alarms even if they are momentary in nature.

7.199. LAHI - High alarm to be latched

Field type: Binary

Field Access: R,W

LAHI is the 1st bit in the word LAHI (decimal value 1, hex value 0x0001).

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: LAHI = Y specifies that high alarm will be latched.

A latched alarm stays on the screen even after the alarm has cleared. A latched defunct alarm blinks in dark blue and must be acknowledged to clear on the displays. Any active alarm will take display precedence over a latched defunct alarm.

Unlatched alarms will disappear from the display screen if the underlying condition which caused the alarm goes away even if the alarm was not acknowledged.

Alarm latching ensures, that the operator will see alarms even if they are momentary in nature.

7.200. LAICS - Change of input state alarm to be latched

Field type: Binary

Field Access: R,W

LAICS is the 6th bit in the word LAICS (decimal value 32, hex value 0x0020).

Defined in Blocks: DEV, DEVICE, DEVINP, DI

Properties: LAICS = Y specifies that change of input state alarm will be latched.

A latched alarm stays on the screen even after the alarm has cleared. A latched defunct alarm blinks in dark blue and must be acknowledged to clear on the displays. Any active alarm will take display precedence over a latched defunct alarm.

Unlatched alarms will disappear from the display screen if the underlying condition which caused the alarm goes away even if the alarm was not acknowledged.

Alarm latching ensures, that the operator will see alarms even if they are momentary in nature.

7.201. LALE - Loop error alarm to be latched

Field type: Binary

Field Access: R,W

LALE is the 4th bit in the word LALE (decimal value 08, hex value 0x0008).

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT, SENSOR, SWSENSOR, CONTR, PCI

Properties: LALE = Y specifies that loop error alarm will be latched.

A latched alarm stays on the screen even after the alarm has cleared. A latched defunct alarm blinks in dark blue and must be acknowledged to clear on the displays. Any active alarm will take display precedence over a latched defunct alarm.

Unlatched alarms will disappear from the display screen if the underlying condition which caused the alarm goes away even if the alarm was not acknowledged.

Alarm latching ensures, that the operator will see alarms even if they are momentary in nature.

7.202. LALL - Low Low alarm to be latched

Field type: Binary

Field Access: R,W

LALL is the 4th bit in the word LAHI (decimal value 8, hex value 0x0008).

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: LALL = Y specifies that lowlow alarm will be latched.

A latched alarm stays on the screen even after the alarm has cleared. A latched defunct alarm blinks in dark blue and must be acknowledged to clear on the displays. Any active alarm will take display precedence over a latched defunct alarm.

Unlatched alarms will disappear from the display screen if the underlying condition which caused the alarm goes away even if the alarm was not acknowledged.

Alarm latching ensures, that the operator will see alarms even if they are momentary in nature.

7.203. LALO - Low alarm to be latched

Field type: Binary

Field Access: R,W

LALO is the 3rd bit in the word LAHI (decimal value 4, hex value 0x0004).

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: LALO = Y specifies that high alarm will be latched.

A latched alarm stays on the screen even after the alarm has cleared. A latched defunct alarm blinks in dark blue and must be acknowledged to clear on the displays. Any active alarm will take display precedence over a latched defunct alarm.

Unlatched alarms will disappear from the display screen if the underlying condition which caused the alarm goes away even if the alarm was not acknowledged.

Alarm latching ensures, that the operator will see alarms even if they are momentary in nature.

7.204. LARC - Rate of Change alarm to be latched

Field type: Binary

Field Access: R,W

LARC is the 7th bit in the word LARC (decimal value 64, hex value 0x0040).

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: LARC = Y specifies that high alarm will be latched.

A latched alarm stays on the screen even after the alarm has cleared. A latched defunct alarm blinks in dark blue and must be acknowledged to clear on the displays. Any active alarm will take display precedence over a latched defunct alarm.

Unlatched alarms will disappear from the display screen if the underlying condition which caused the alarm goes away even if the alarm was not acknowledged.

Alarm latching ensures, that the operator will see alarms even if they are momentary in nature.

7.205. LAST - Error status alarm to be latched

Field type: Binary

Field Access: R,W

LAST is the 2nd bit in the word LAST (decimal value 2, hex value 0x0002).

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: LAST = Y specifies that error status alarm will be latched.

A latched alarm stays on the screen even after the alarm has cleared. A latched defunct alarm blinks in dark blue and must be acknowledged to clear on the displays. Any active alarm will take display precedence over a latched defunct alarm.

Unlatched alarms will disappear from the display screen if the underlying condition which caused the alarm goes away even if the alarm was not acknowledged.

Alarm latching ensures, that the operator will see alarms even if they are

momentary in nature.

7.206. LAUN - Undefined state alarm to be latched

Field type: Binary

Field Access: R,W

LAUN is the 3rd bit in the word LAUN (decimal value 4, hex value 0x0004).

Defined in Blocks: DEV, DEVICE, DEVINP, DI

Properties: LAUN = Y specifies that undefined state alarm will be latched.

A latched alarm stays on the screen even after the alarm has cleared. A latched defunct alarm blinks in dark blue and must be acknowledged to clear on the displays. Any active alarm will take display precedence over a latched defunct alarm.

Unlatched alarms will disappear from the display screen if the underlying condition which caused the alarm goes away even if the alarm was not acknowledged.

Alarm latching ensures, that the operator will see alarms even if they are momentary in nature.

7.207. LBn - Type of the n-th loop body block (LB1-LB8)

Field type: Internal Type

Field Access: This field can only be changed by DBB

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: Loop body blocks specified in the fields LBn are either calculation blocks (LB or CALC,) or delay blocks (DELAY).

The calculation blocks (LB or CALC) use six inputs in an expression with mathematical, logical, relational and conditional operators to determine the output of the logic block.

Delay blocks (DELAY) are used to delay a control signal.

Typical Blocks: The following blocks are typically used:

- LB (equivalent to the CALC block)
- DELAY

7.208. LEAD - Lead (sec)

Field type: Float

Field Access: R,W

Defined in Blocks: LL

Properties: Lead time in seconds.

7.209. LEC - Loop error alarm to be critical

Field type: Binary

Field Access: R,W

LEC is the 8th bit in the word LEC (decimal value 128, hex value 0x0080).

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: LEC = Y specifies that loop error alarm will be displayed as critical.

Critical alarm will appear to the operator in RED and will take precedence over reporting of simultaneous major or minor alarms.

If two alarms of equivalent severity are present simultaneously, the displayed alarm will be the one closer to the top of this list:

- LERR
- HIHI and LOLO
- HI and LO
- DHI and DLO
- ROC

7.210. LEM - Loop error alarm to be major

Field type: Binary

Field Access: R,W

LEM is the 8th bit in the word LEM (decimal value 128, hex value 0x0080).

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: LEM = Y specifies that loop error alarm will be displayed as major alarm.

Major alarm will appear to the operator in YELLOW and will take precedence over reporting of simultaneous minor alarms. However, reporting of simultaneously present critical alarms takes precedence over reporting major alarms.

If two alarms of equivalent severity are present simultaneously, the displayed alarm will be the one closer to the top of this list:

- LERR
- HIHI and LOLO
- HI and LO
- DHI and DLO
- ROC

7.211. LINC - Linearization

Field type: String (Size: 4 characters)

Field Access: R

Defined in Blocks: SENSOR, PCI, AIN, AOT

Notes: Remote ON modification has no effect. A change will only become effective after a reboot of the MFC!

Properties: The LINC field specifies the conversion curve in the Breakpoint-table to be used for the conversion of raw data of this sensor.

In many cases the raw data delivered by the hardware are depending on the type of the hardware and are not directly the measure of the desired physical property.

The breakpoint table allows the conversion of raw input integer values into a floating point number, representing either the final engineering value or a normalized engineering value. The following figure illustrates how the breakpoint table approximates a curve by a piecewise linear approximation.

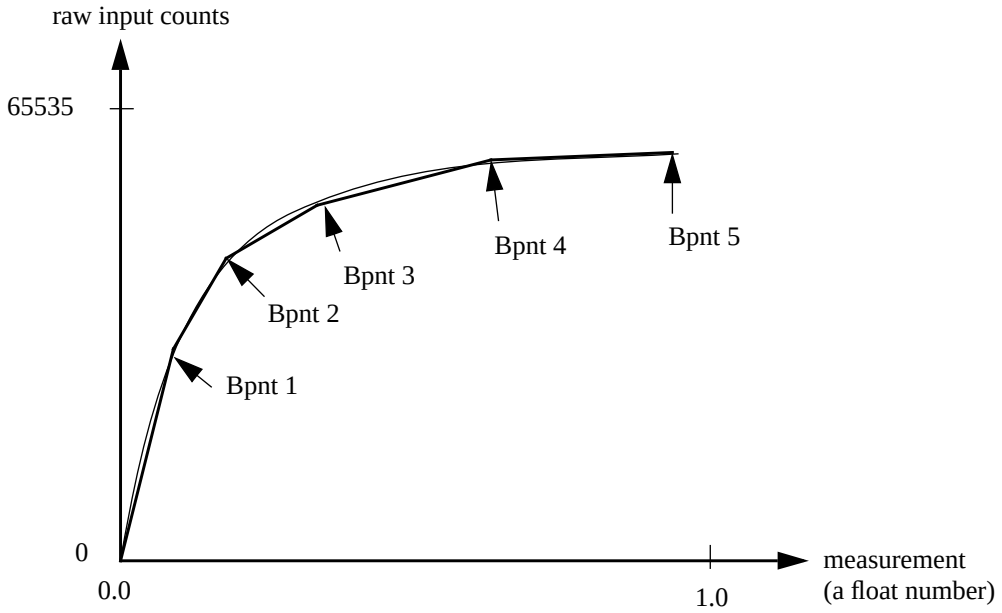


Figure 10: Breakpoint Table Illustr

Example: To determine the weight of the liquid in a horizontal cylindrical tank we measure the hydrostatic pressure of the liquid in the tank. The volume of the liquid (the weight of the liquid) is a non-linear function of the measured hydrostatic pressure.

Example: To determine the flow of a liquid we measure the pressure drop over an orifice. The flow is proportional to the square root of the pressure drop.

Example: Siemens AI card provides numeric data in the interval between 0 and 4095, these data are multiplied by CCEX by 16 and so the raw data lie in the interval 0 to 65520.

For compensating such non-linearities user alterable conversion curves, specified in a breakpoint table, are used. The structure of the breakpoint table is visible in the following example:

```
# comment
BREAK_POINT_TABLE:
#
CURVE_START: name_of_the_1st_curve,_10_characters
LOWCNT: count_1                HIGHCNT: count_2
LOWVAL: value_at_count_1    HIGHVAL: value_at_count_2
LOWCNT: count_2                HIGHCNT: count_3
LOWVAL: value_at_count_2    HIGHVAL: value_at_count_3
LOWCNT: count_3                HIGHCNT: count_4
LOWVAL: value_at_count_3    HIGHVAL: value_at_count_4
CURVE_END:
#
CURVE_START: name_of_the_2nd_curve,_10_characters
LOWCNT: count_1                HIGHCNT: count_2
```

LOWVAL: value_at_count_1 HIGHVAL: value_at_count_2

LOWCNT: count_2 HIGHCNT: count_3

LOWVAL: value_at_count_2 HIGHVAL: value_at_count_3

LOWCNT: count_3 HIGHCNT: count_4

LOWVAL: value_at_count_3 HIGHVAL: value_at_count_4

CURVE_END:

#

#next curves#

TABLE_END:

The raw data (after their transformation according to the scanner used and test, whether they are within MLO and MHI) are transformed using the corresponding conversion curve (field LINC) by linear interpolation between HIGHVAL and LOWVAL, which correspond to the LOWCNT <= raw data < HIGHCNT.

It is a good practice to use the conversion curve to convert the raw data in such a way that the useable interval of the data is transformed in the interval between 0.0 and 1.0 (Normalized value).

The process of linearization can be demonstrated on the following example of the Siemens analog input card 6ES5 498-1AA71, used for inputs between 4 to 20 mA:

mA	Counts ex Siemens Card	Rawdata = Counts * 16	Normalized
0	0	0	-0.25000
3	384	6144	-0.06250
4	512	8192	0.00000
5	640	10240	0.06250
6	768	12288	0.12500
7	896	14336	0.18750
8	1024	16384	0.25000
9	1152	18432	0.31250
10	1280	20480	0.37500
11	1408	22528	0.43750
12	1536	24576	0.50000
13	1664	26624	0.56250
14	1792	28672	0.62500
15	1920	30720	0.68750
16	2048	32768	0.75000
17	2176	34816	0.81250
18	2304	36864	0.87500
19	2432	38912	0.93750
20	2560	40960	1.00000

Table 27: Relationship between Interface Counts and Break Table

mA	Counts ex Siemens Card	Rawdata = Counts * 16	Normalized
21	2688	43008	1.06250
22	2816	45056	1.12500
23	2944	47104	1.18750
24	3072	49152	1.25000
25	3200	51200	1.31250
26	3328	53248	1.37500
27	3456	55296	1.43750
28	3584	57344	1.50000
29	3712	59392	1.56250
30	3840	61440	1.62500
31	3968	63488	1.68750
32	4096	65536	1.75000

Table 27: Relationship between Interface Counts and Break Table

The table above has been assembled using the following data:

- For Raw-Value = 8192 the normalized value must be 0.0
- For Raw-Value = 40960 the normalized value must be 1.0
- Normalized Value = (1.0 - 0.0)/(40960 - 8192) * (Raw-Value - 8192) + 0
- Normalized Value = Raw-Value / 32768 - 0.25

This assumes that the norm,al measurement range lies within the equivalent of 4 to 20 mA.

The breakpoint table values will therefore be:

- LOWCNT: 0HIGHCNT: 61440
- LOWVAL: -0.25HIGHVAL: 1.625

The MUX Low and High values will be set as follows:

- MLO: 6144 (the equivalent of 3 mA)
- MHI: 51200 (the equivalent of 25 mA)

The resulting Breakpoint Table is as follows:

```
#
#               B R E A K P O I N T - T A B L E
#
BREAK_POINT_TABLE:
#
# Curve for analog input for S5 interface,   Module: 6ES5-498-1AA71
# Measuring interval  4 to 20 mA
#
CURVE_START: S5A1
LOWCNT: 0            HIGHCNT: 61440
LOWVAL: -0.25        HIGHVAL: 1.625
CURVE_END:
```

#

TABLE_END:

The recommended values for MLO is 6144 (corresponding 3 mA) and for MHI is 51200 (corresponding to 25 mA) for S5 interfaces

Example: CAN Analog Input which are linear, uses the following entry from the Breakpoint Table:

#

Curve for CAN AI

#

CURVE_START: CAN_AI

LOWCNT: 0 HIGHCNT: 65535

LOWVAL: 0 HIGHVAL: 1.0

CURVE_END:

#

7.212. LL - LEAD/LAG BLOCK

The Lead/Lag algorithm is often configured into feed forward and model reference control schemes. In these schemes, the output of the feedforward loop often supplies a computed measurement term to a PID controller configured for integral-only action. When any part of the feedback control scheme becomes open loop, the feedforward elements will often wind up. When the feedback control loops are closed again, the entire feedforward mechanism must be re-initialized.

The Lead/Lag algorithm provides the capability to simulate a simple first order dynamic process (Lag term) combined with a second order dynamic process (Lead term). The result is a reasonable dynamic approximation to a wide variety of real processes. By tuning the lead (LEAD) and lag (LAG) coefficients, it is possible to closely match the dynamic performance of the process sufficient for control modelling purposes. The pure delay functions provided by the Dead Time algorithm are almost always required to complete these models.

When configured into a feedforward or model reference control loop, the CLSA term should be set to 2 to detect that a downstream cascade has gone from open to closed and to close the upstream cascade. The ITYP term should be set to 1 which will cause the output to be set to the input, and the calculation delayed until the next scan cycle. The figure Type 3 Integral Limiting Action, which can be found earlier in this document, illustrates the configuration of both Lead/Lag and Dead Time blocks into a feedforward control scheme.

The operation of the lead/lag block uses the equation:

R = [K*(LEAD/Ts)*(A-La) + K*A +(LAG/Ts)*Lr] / ((LAG/TS)+1)

where:

R = Result
Lr = Last Result , R(n-1)
K = Gain
A = Input (or setpoint if used as a controller)
La = Last input, A(n-1)
LEAD = Lead time
LAG = Lag time
Ts = Time scan

7.213. LLC - Low-low alarm to be critical

Field type: Binary

Field Access: R,W

LLC is the 4th bit in the word LLC (decimal value 8, hex value 0x0008).

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: LLC = Y specifies that low-low alarm will be displayed as critical.

Critical alarm will appear to the operator in RED and will take precedence over reporting of simultaneous major or minor alarms.

If two alarms of equivalent severity are present simultaneously, the displayed alarm will be the one closer to the top of this list:

- LERR
- HIHI and LOLO
- HI and LO
- DHI and DLO
- ROC

7.214. LLM - Low-low alarm to be major

Field type: Binary

Field Access: R,W

LLM is the 4th bit in the word LLM (decimal value 8, hex value 0x0008).

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: LLM = Y specifies that low-low alarm will be displayed as major alarm.

Major alarm will appear to the operator in YELLOW and will take precedence over reporting of simultaneous minor alarms. However, reporting of simultaneously present critical alarms takes precedence over reporting major alarms.

If two alarms of equivalent severity are present simultaneously, the displayed alarm will be the one closer to the top of this list:

- LERR
- HIHI and LOLO
- HI and LO
- DHI and DLO
- ROC

7.215. LO - Low alarm limit value

Field type: Float

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: The low alarm limit value LO is compared with the measured value MEAS. If MEAS is less than LO, then low alarm is raised.

Restrictions: The LO alarm limit cannot be less than the LOLO alarm value. Any attempt to set the limit below the LOLO will result in an error.

7.216. LOC - Low alarm to be critical

Field type: Binary

Field Access: R,W

LOC is the 3rd bit in the word LOC (decimal value 4, hex value 0x0004).

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: LOC = Y specifies that low-low alarm will be displayed as critical.

Critical alarm will appear to the operator in RED and will take precedence over reporting of simultaneous major or minor alarms.

If two alarms of equivalent severity are present simultaneously, the displayed alarm will be the one closer to the top of this list:

- LERR
- HIHI and LOLO
- HI and LO
- DHI and DLO
- ROC

7.217. LOLO - Low-low alarm limit value

Field type: Float

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: The low-low alarm limit value LOLO is compared with the measured value MEAS. If MEAS is less than LOLO, then low-low alarm is raised.

Restrictions: The LOLO alarm limit cannot be higher than the LO alarm value. Any attempt to set the limit above the LO will result in an error.

7.218. LOM - Low alarm to be major

Field type: Binary

Field Access: R,W

LOM is the 3rd bit in the word LOM (decimal value 4, hex value 0x0004).

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: LOM = Y specifies that low alarm will be displayed as major alarm.

Major alarm will appear to the operator in YELLOW and will take precedence over reporting of simultaneous minor alarms. However, reporting of simultaneously present critical alarms takes precedence over reporting major alarms.

If two alarms of equivalent severity are present simultaneously, the displayed alarm will be the one closer to the top of this list:

- LERR
- HIHI and LOLO
- HI and LO
- DHI and DLO
- ROC

7.219. MEAS - Measurement in analogue Loops

Field type: Float

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: The Measurement (MEAS) field is a non-prompted field which can be accessed from other blocks in the database, or from EXTOL programs.

This value depends on the value of the MSBL (Meas Block) Field.

If MSBL = 0, then MEAS field contains the result of the processing of the input (linearization, smoothing, transformation in engineering units in case of SENSOR loop header block, the input INP in case of CONTR block and SWSENSOR loop header block (exception - if INP field of SWSENSOR reads the value from elsewhere, MEAS is calculated using the SPAN and ZERO of the block; see INP field description).

If MSBL > 0 (1 to 8) then MEAS contains the value of the OUT field of the MSBL-th loop body block. I.e. MSBL (MeaSurement BLock) designates the block number from which the contents of the MEAS field is derived.

Example: Assume a Software sensor has to be built to provide the average T_AVER of two temperature measurement TEMP1 and TEMP2, realized by two SENSORs.

Additional requirement is to raise a deviation (high or low) alarm if the readings of TEMP1 and TEMP2 differ by more than 3 DegC.

Solution:

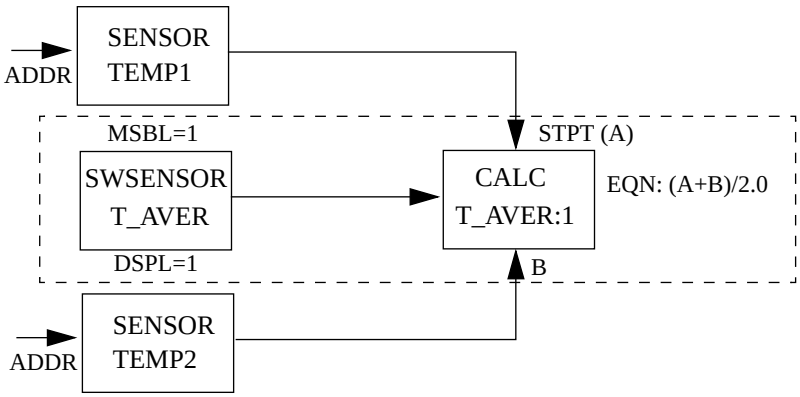


Figure 11: Block structure of example

SWSENSOR T_AVER:

DHI = 1.5
DLO = 1.5
MSBL = 1
DSPL = 1
CB1 = CALC

CALC T_AVER:

EQN = (A+B)/2.0
STPT = TEMP1:0.MEAS
B = TEMP2:0.MEAS
AMLK = Y

The average of the two temperature sensors is calculated in the CALC block. Because MSBL (measurement Block) is defined to be 1, the calculated average is taken as the new MEAS value for the SENSOR loop. Defining DSPL (display block) as 1, causes the control package to use MEAS and STPT of the CALC block (the display block) to check for deviation. Because deviation is defined not to exceed 3, the two parameters DHI and DLO are both set to 1.5.

7.220. MEAS - Measurement in discrete or digital Loops

Field type: Word

Field Access: R

Defined in Blocks: DEV, DEVICE, DI, DEVINP, DEVICE

Properties: The Measurement (MEAS) field is a non-prompted field which can be accessed from other blocks in the database, or from EXTOL programs.

The value of this field represents the bit-mask read from the hardware as input for the device.

7.221. MFCALC - COMPENSATED GAS MASS FLOW BLOCK

Block-Type: This is a specially preconfigured CALC-block

Record-Type: CALC

Properties: The flow of compressible gases follows natural laws which allow the calculation of mass flow if the temperature and pressure are known as well as the volume of flow at those conditions. The MFCALC is a specific configuration of the CALC block designed to calculate mass flow from a volumetric flow rate, a temperature and a pressure.

The **volumetric** flow of the gas is transformed to the **mass** flow at reference temperature (e.g. 0 DegC C) and reference pressure (e.g. 1 atm) using the equation

A*S(E*(B+D)/(C+F))

where:

- A is the uncompensated volumetric flow rate,
- S is the square root operator,
- B is the gauge pressure reading,
- C is the temperature in degrees F or C,
- D is the offset of the gauge pressure to absolute pressure,
- E is the reference absolute temperature divided by the reference absolute pressure: Tr/Pr,
- F is offset of temperature in degrees F to degrees Rankine (460), or offset of temperature in degrees C to degrees Kelvin (273).

For temperatures in degrees F or C, there are a variety of coefficient values for various units of pressure. The following table provides a convenient selection table for values of D, E and F to be used for each combination of pressure and temperature units:

Coefficients for MFCALC

Pressure Units	Temp. Units	D	F	E
psig	F	14.7	460	33.4694
psig	C	4.7	273	18.5714
atm	F	1	460	492
atm	C	1	273	273
in H2O	F	406.67	460	1.2098
in H2O	C	406.67	273	0.6713
g/sqcm	F	1033.3	460	0.4761
g/sqcm	C	1033.3	273	0.2642
mmHg	F	760	460	0.6474
mmHg	C	760	273	0.3592

Table 28: MFCALC Coefficients

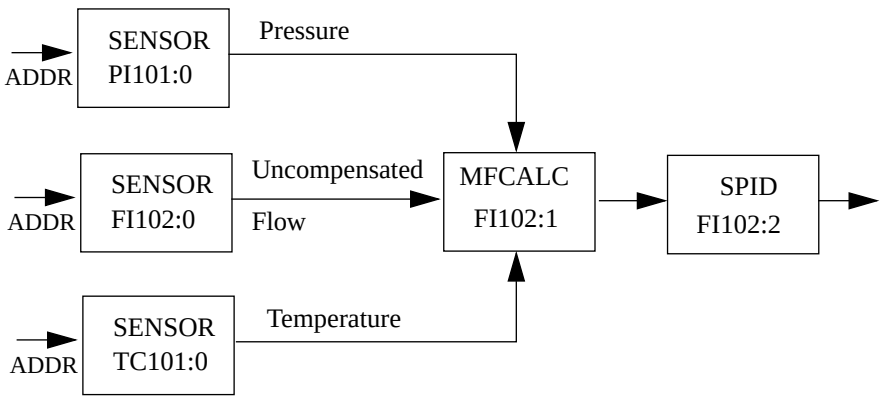


Figure 12: Mass Flow Loops using MFCALC block

7.222. MHI - Mux high clamp

Field type: Word

Field Access: R,W

Defined in Blocks: SENSOR, PCI, AOT

Properties: The value defined in MHI is called the Multiplexer High Limit. In both cases, i.e. for analogue inputs and outputs it represents the physical limitations of the measurement equipment (including the A/D and D/A converters). It means, that values above this limitations are invalid or meaningless. The value is in raw counts, in the range 0 to 65535. The exact values to be used depend on the hardware in use.

Two fields are provided in the data base to allow clamping (and loop error generation) or out-of-range inputs. In most cases, MHI will be 65535 and MLO will be zero. This effectively turns off the MUX HI and MUX LO checking. A similar but more readily understandable mechanism for checking out-of-range sig-

nals is built into the breakpoint table mechanism; "falling off the end" of the table generates a loop error.

In case of the normal 4 to 20 mA signals, MHI and MLO are to be set to those values which represent a point below 4 mA resp. above 20 mA. These values are normally the raw values representing the appropriate limit (for MLO: 0 to 12000).

In case of AOT blocks, the output value is clamped to MHI, if the value converted to raw counts is greater than MLO. For further details on AOT block usage, See “SPAN - Instrument span of AOT” on page 168

7.223. MLO - Mux low input clamp

Field type: Word

Field Access: R,W

Defined in Blocks: SENSOR, PCI, AOT

Properties: The value defined in MLO is called the Multiplexer Low Limit. In both cases, i.e. for analogue inputs and outputs it represents the physical limitations of the measurement equipment (including the A/D and D/A converters). It means, that values below this limitations are invalid or meaningless. The value is in raw counts, in the range 0 to 65535. The exact values to be used depend on the hardware in use.

MLO specifies the minimum output raw count below which the output may not fall. Typically this is the counts produced by 4mA current normalised to 16 bits. This value depends on the card type used.

Two fields are provided in the data base to allow clamping (and loop error generation) or out-of-range inputs. In most cases, MHI will be 65535 and MLO will be zero. This effectively turns off the MUX HI and MUX LO checking. A similar but more readily understandable mechanism for checking out-of-range signals is built into the breakpoint table mechanism; "falling off the end" of the table generates a loop error.

In case of the normal 4 to 20 mA signals, MHI and MLO are to be set to those values which represent a point below 4 mA resp. above 20 mA. These values are normally the raw values representing the appropriate limit (for MLO: 0 to 12000).

In case of AOT blocks, the output value is clamped to MLO, if the value converted to raw counts is less than MLO.

7.224. MODEL- Rack-, Module- or Card-Type

Field type: String

Field Access: This field can only be entered and displayed using DBB.

Defined in Blocks: ANAMODULE, DIGMODULE, ANACARD, DIGCARD, RACK

Properties: The Model is specific to each module or rack. The names of the models will be found in the I/O Hardware specific manual.

Example: For the analogue module from WEIDMUELLER, the models are:

Model Name	Designation	Reference
WIN 8DIP	Digital Input 8 Channels	830716
WIN 16DIP	Digital Input 16 Channels	830717
WIN 16DIP 2x8	Digital Input 16 (2x8) Channels	830718
WIN 32DIP 2x16	Digital Input 32 (2x16) Channels	830719
WIN 4AIUI	Analog Input 4 Channels	830727
WIN 8DOPK	Digital Output 8 Channels	830721
WIN 16DOPK	Digital Output 16 Channels	830722
WIN 16DOP 2x8	Digital Output 16 (2x8) Channels	831497
WIN 32DOP 2x16	Digital Output 32 (2x16) Channels	830723
WIN 4AOUI	Analog Output 4 Channels	830728

Table 29: WEIDMUELLER CAN Module

For the cards from Beckhoff, the models are:

Model Name	Designation	Reference
KL1104	Digital Input 4 Channels	
KL2134	Digital Output 4 Channels	
KL3054	Analog Input 4 Channels	
KL4022	Analog Output 2 Channels	

Table 30: BECKHOFF Modbus RTU Module

Like ADDR, BUS & CRn, this field is used to access the IO interface, it depends on the choice done by the Automation Engineer & on the system supported IO's. Feel free to contact us for further information.

7.225. MOUT - Manual output position

Field type: Float

Field Access: R,W

Defined in Blocks: SPID, NLP, PID, CALC

Properties: MOUT specifies the output value which appears in the OUT when the loop is switched from AUTO to MAN from the console, or switched to MANUAL by the EXTOL instruction

set loop ... to manual

or switched to MANUAL as slave of the display block.

In this sense output of MOUT is part of "de-initializing" when switched to MANUAL.

Note: MOUT value is normally 0 (= output control valve closed).

For applications using 2 output values in a split range mode for a heating/cooling system, this value is typically 50. (**Note**: The SPLIT block **must not** be a slave of the SPID block to get output valves really opened / closed!)

This field replaces the former NPOS field.

MOUT value is simply added to the result of the SPID algorithm. (See SPID)

7.226. MRCOS- Change of state alarm to be major

Field type: Binary

Field Access: R,W

MRCOS is the 5th bit in the word MRCOS (decimal value 16, hex value 0x0010).

Defined in Blocks: DEV, DEVICE, DEVINP, DI

Properties: MRCOS = Y specifies that change of state alarm will be displayed as major alarm.

Major alarm will appear to the operator in YELLOW.

Critical alarms will take precedence over simultaneous major alarms.

Major alarm will take precedence over reporting of simultaneous minor alarms.

7.227. MRDCS - Change of desired state alarm to be major

Field type: Binary

Field Access: R,W

MRDCS is the 7th bit in the word MRDCS (decimal value 64, hex value 0x0040).

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: MRDCS = Y specifies that change of desired state alarm will be displayed as major alarm.

Major alarm will appear to the operator in YELLOW.

Critical alarms will take precedence over simultaneous major alarms.

Major alarm will take precedence over reporting of simultaneous minor alarms.

7.228. MRDV - Difference between actual and desired state alarm to be major

Field type: Binary

Field Access: R,W

MRDV is the 1st bit in the word MRDV (decimal value 1, hex value 0x0001).

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: MRDV = Y specifies that difference between actual and desired state alarm will be displayed as major alarm.

Major alarm will appear to the operator in YELLOW.

Critical alarms will take precedence over simultaneous major alarms.

Major alarm will take precedence over reporting of simultaneous minor alarms.

7.229. MRICS - Change of input state alarm to be major

Field type:	Binary
Field Access:	R,W
	MRICS is the 6th bit in the word MRICS (decimal value 32, hex value 0x0020).
Defined in Blocks:	DEV, DEVICE, DEVINP, DI, DEVOUT
Properties:	MRICS = Y specifies that change of input state alarm will be displayed as major alarm.
	Major alarm will appear to the operator in YELLOW.
	Critical alarms will take precedence over simultaneous major alarms.
	Major alarm will take precedence over reporting of simultaneous minor alarms.

7.230. MRLE - Loop error alarm to be major

Field type:	Binary
Field Access:	R,W
	MRLE is the 4th bit in the word MRLE (decimal value 8, hex value 0x0008).
Defined in Blocks:	DEV, DEVICE, DEVINP, DI, DEVOUT
Properties:	MRLE = Y specifies that loop error alarm will be displayed as major alarm.
	Major alarm will appear to the operator in YELLOW.
	Critical alarms will take precedence over simultaneous major alarms.
	Major alarm will take precedence over reporting of simultaneous minor alarms.

7.231. MRST - Error status alarm to be major

Field type:	Binary
Field Access:	R,W
	MRST is the 2nd bit in the word MRST (decimal value 2, hex value 0x0002).
Defined in Blocks:	DEV, DEVICE, DEVINP, DI, DEVOUT
Properties:	MRST = Y specifies that error status alarm will be displayed as major alarm.
	Major alarm will appear to the operator in YELLOW.
	Critical alarms will take precedence over simultaneous major alarms.
	Major alarm will take precedence over reporting of simultaneous minor alarms.

7.232. MRUN - Undefined state alarm to be major

Field type:	Binary
Field Access:	R,W
	MRUN is the 3rd bit in the word MRUN (decimal value 4, hex value 0x0004).
Defined in Blocks:	DEV, DEVICE, DEVINP, DI
Properties:	MRUN = Y specifies that undefined state alarm will be displayed as major

alarm.

Major alarm will appear to the operator in YELLOW.

Critical alarms will take precedence over simultaneous major alarms.

Major alarm will take precedence over reporting of simultaneous minor alarms.

7.233. MSBL - Measurement display block

Field type: Byte

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: MSBL = 0 indicates that the MEAS value is the processed input value from the loop header block, i.e. the value obtained from the input of the loop header block.

MSBL > 0 (1 to 8) specifies that the output of the block in the loop with that sequence number is to be used as the loop measurement (i.e. when read as MEAS or as M in EQN, and displayed on the faceplate).

7.234. NLP - NONLINEAR PID BLOCK

Block-Type: NLP

Record-Type: PID

Properties: The Nonlinear PID is an algorithmic variant of the standard SPID sharing the majority of the parameters. The differences are primarily in the additional options which are provided: error squared, limits for error squared, gap action and velocity compensation.

The algorithm implemented is:

$$y(k) = KP \{e(k) + KD / POLL * [d(k) - d(k-1)]\} + iterm(k) + FFW$$

where:

FFW = feed forward term

e(k) = without velocity compensation:

$$e(k) = STPT(k) - B(k)$$

with velocity compensation

$$e(k) = STPT(k) - (B + vc * dB/dt))$$

in case of reverse action, e(k) = - e (k)

d(k) = either e(k) if field EI is set to 'E', or B(k) if field EI is set to 'I'

fKP = fKP = KP for KP # 0

fKP = 1 for KP = 0

$$iterm(k) = iterm(k-1) + fKP * KI * POLL * e(k)$$

dB/dt is calculated as a running average of the last 3 samples

$$(B(k) - B(k-1)) / POLL$$

NPOS = neutral position

The algorithm is initialised

e(k-1)	= STPT - B in the case of direct action	
e(k-1)	= B - STPT in the case of reverse action	
d(k-1)	= e(k-1) if EI is set to E, else =B(k)	
iterm(k-1)	= iterm (k-1) = 0	for ITYP = 0 and 1
iterm (k-1) = OUT		for ITYP = 2 and 3

With the gap action defined, the controller freezes within a deadband around the setpoint.

With error squared, the above error calculation is e(k) = e(k) * abs (e(k)) /KNL

7.235. NLPI - Error squared option

Field type: Binary

Field Access: R,W

NLPI is the 12th bit in the word NLPI (decimal value 2048, hex value 0x0800).

Defined in Blocks: NLP

Properties: NLPI = P (=0) defines error squared to be applied to the proportional term only (default),

NLPI = I (=1) defines error squared to be applied to the integral term only.

The ability to respond to slow acting and integrating processes is improved by using the square of the error in the control calculation. SyCOS supports this fully by allowing the option to specify that the error squared be applied either to the proportional term, or (sometimes more effectively) to the integral term only. The term "error squared" actually means that the numerical value of the error is squared, but the sign of the error is preserved. This magnifies the effect of large errors.

7.236. NMFL - Normal / Maintenance flag

Field type: Bit

Field Access: R,W

Defined in Blocks: DIN, DOT, AOT

Properties: NMFL = N (0) determines that the block is in normal mode (on faceplate appears NORM)

NMFL = M (1) determines that the block is in maintenance mode (on faceplate appears MAIN).

If a block is switched to maintenance mode, it may be driven manually from the operator console. On the DOT block the output may be driven directly, on a AOT block the output may be positioned directly and on a DIN block the input may be simulated.

Switching the flag NMFL from the faceplate can be enabled or disabled with the field NMLK.

Note: When an other block of the database tries to drive an output block, which is in maintenance mode, a loop error will occur.

7.237. NMLK - Normal / Maintenance lock flag

Field type: Bit

Field Access: R,W

Defined in Blocks: DIN, DOT, AOT

Properties: NMLK = Y (1) disables the changing of the state of the Normal / Maintenance flag (NMFL) by the operator from the faceplate; NMLK = N (0) enables it. This does not prevent from changing the flag NMFL neither from EXTOL nor from the database itself.

7.238. NPAR - Convert from normalized parameters (PB, TR, TD)

Field type: Binary

Field Access: R,W

NPAR is the 10th bit in the field NPAR (decimal value 512, hex value 0x0200

Defined in Blocks: SPID

Properties: Some process engineers are more familiar with the PID terms ‘proportional band PB’, ‘reset time RT’ and ‘derivative time DT’.

As long as NPAR is set to NO, then the values KP, KI, KD are converted at initialisation time into three normalized values for display purposes.

The formulae used are:

PB = 100 / KP * 100 / SPAN

TR = 1/ KI

TD = KD

If NPAR is switched to YES, then the values of KP, KI and KD are calculated from PB, TR and TD at loop initialization.

The formulae used are:

KP = 100 / PB * 100 / SPAN

KI = 1 / TR

KD = TD

All subsequent calculations are performed using KP, KI and KD.

Note: Be aware, that the calculation is only performed at initialization, i.e. when switching from MANUAL to AUTO.

7.239. NPOS - Neutral output position

Field type: Float

Field Access: R,W

Defined in Blocks: SPID, NLP, PID, CALC

Properties: NPOS specifies the output value which appears in the OUT when the loop is switched from AUTO to MAN from the console, or switched to MANUAL by the EXTOL instruction

set loop ... to manual

or switched to MANUAL as slave of the display block.

In this sense output of NPOS is part of "de-initializing" when switched to MANUAL.

Note:

NPOS value is normally 0 (= output control valve closed).

For applications using 2 output values in a split range mode for a heating/cooling system, this value is typically 50. (**Note**: The SPLIT block **must not** be a slave of the SPID block to get output valves really opened / closed!)

NPOS value is simply added to the result of the SPID algorithm. (See SPID)

7.240. OBIT - Output bit

Field type: Word

Field Access: R,W

Defined in Blocks: DOT

Properties: This value represents the bit number in the current 16 bit channel bank (point) on a DOT card.

Example: The DOT signal is on the 14th physical channel on the Digital card.

The point (OPNT) will be 1 and OBIT will be 14.

7.241. OCB - Output control block

Field type: Byte

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: OCB specifies the number of the loop body block, the number of which and the address of the output of which will be shown in the field GET LINK INFOS on the faceplate.

OCB would normally contain the number of the SPID block for a simple control loop outputting to one AOT, or the number of the CALC blocks outputting to a digital device. In case of a splitrange output it would contain the number of the SPLIT block.

The fact that OCB is used for display purposes only allows unconventional usage of OCB, as demonstrated in Example.

Example: Assume we have to control pressure in a vessel using two on/off valves: If the pressure is below a given value, valve N2IN should be opened and kept open, until the pressure comes close to the desired pressure; if the pressure is above a given value, the device VENT has to be opened until the pressure comes to the desired value again.

We solve the problem building a control loop, consisting of the loop header block of type CONTR, with the loop body block CB1 of type CALC, which controls the valve N2IN and with the loop body block CB2 of type CALC which controls the device VENT.

We would like to indicate the operator that the control of the pressure is performed through the two devices N2IN and VENT.

The (only) possibility to achieve this is to add a third loop body block (e.g.CB8) of type SPLIT with the two outputs

OUT1 = N2IN:0.DEST
and
OUT2 = VENT:0.DEST

This SPLIT block in this example must have AM flag cleared (be constantly in MANUAL), must be not slave of the display block and should have AMLK flag = Y to indicate that switching to AUTO is not desired. Its task in this case is only to force the faceplate to display the names of the two devices as GET LINK INFOS.

7.242. ODHI - Outside Deadband High (Gap action control)

Field type: Float

Field Access: R,W

Defined in Blocks: NLP

Properties: Some processes are naturally very noisy and cause excessive control actions which are not needed. With some types of process actuators, the constant movement may shorten the life of the equipment. In other cases, excessive control action increases the instability of the control system or forces the control system to be detuned so as to react too slowly for serious problems. The gap action capability of the NLP algorithm is designed to overcome these problems.

Gap action may be applied relative to the current setpoint by specifying a gap, or deadband around the setpoint. Control calculations are suppressed for any error that is within the gap. Normal control occurs outside the gap based on the actual error. Reset accumulation (wind-up) does not occur while the error is within the gap. The gap is specified so that measurements within the inner limits (between IDHI and IDLO) are within the gap, while measurements above and below the outer limits (above ODHI and below ODLO) are outside the gap. The space between the outer and inner limits is a deadband applied so that gap action control will not "chatter". Figure 5: "Gap Action Control" illustrates the functions of Gap Action Control and the relative positions of the limits and the setpoint. No control action will occur in zones 3 and 4. Full control action will always occur in zones 1 and 6. As a measurement signal moves from zone 3 or 4, it must enter zones 1 or 6 before control action will occur. Control action will then continue while the measurement moves through zones 2 and 5, and stop when zone 3 or 4 is entered. Zones 2 and 5 are deadbands to the Gap Action Control.

Note that ODHI, ODLO, IDHI, IDLO are values related to the setpoint, i.e. IDLO and ODLO are negative values.

7.243. ODLO - Outside Deadband Low (Gap action control)

Field type: Float

Field Access: R,W

Defined in Blocks: NLP

Properties: Some processes are naturally very noisy and cause excessive control actions which are not needed. With some types of process actuators, the constant movement may shorten the life of the equipment. In other cases, excessive control action increases the instability of the control system or forces the control system to be detuned so as to react too slowly for serious problems. The gap action capability of the NLP algorithm is designed to overcome these problems.

Gap action may be applied relative to the current setpoint by specifying a gap, or deadband around the setpoint. Control calculations are suppressed for any error that is within the gap. Normal control occurs outside the gap based on the actual error. Reset accumulation (wind-up) does not occur while the error is within the gap. The gap is specified so that measurements within the inner limits (between IDHI and IDLO) are within the gap, while measurements above and below the outer limits (above ODHI and below ODLO) are outside the gap. The space between the outer and inner limits is a deadband applied so that gap action control will not "chatter". Figure 5: "Gap Action Control" illustrates the functions of Gap Action Control and the relative positions of the limits and the setpoint. No

control action will occur in zones 3 and 4. Full control action will always occur in zones 1 and 6. As a measurement signal moves from zone 3 or 4, it must enter zones 1 or 6 before control action will occur. Control action will then continue while the measurement moves through zones 2 and 5, and stop when zone 3 or 4 is entered. Zones 2 and 5 are deadbands to the Gap Action Control.

Note that ODHI, ODLO, IDHI, IDLO are values related to the setpoint, i.e. IDLO and ODLO are negative values.

See “Figure 5: Gap Action Control” on page 104.

7.244. ODSP - Display output of the display block

Field type: Bit

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: ODSP = Y displays the output of the display block in the form of a horizontal bar on the faceplate.

There is a slider on the bar, which indicates the position of the output within the OLC and OHC limits. The numbers displayed over the slider are calculated according to the formula

displayed number = (OUT - OLC) / (OHC - OLC) * 100

I.e. it is a value in percent of the range between OLC and OHC.

The leftmost position means the output is equal OLC, the rightmost position means the output is equal OHC. There is no information about the actual values of OLC and OHC.

Positions of the slider within the bar are between OLC and OHC.

Unfortunately if OHC = OLC then the last position of the slider and the last number displayed stay on the faceplate.

7.245. OHC - Output Hi clamp

Field type: Float

Field Access: R,W

Defined in Blocks: CALC, NLP, SPID, SPLIT, AOT, LL, MFCALC, LB, CTIM

Properties: OHC value is the maximum value the output of the loop body block can achieve. If the result of the processing of the loop body block would exceed OHC value, OHC value is output. Nevertheless, the internal calculation result is not clamped.

OHC also corresponds to the console's 100% output range for output display bar. (See “ODSP - Display output of the display block” on page 151).

Note, that attempting to set the output above the OHC limit from EXTOL causes an EXTOL runtime error.

7.246. OIDR - Direction for displaying the display block output

Field type: Bit

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: OIDR = D (DIRect) causes the display the position of the slider on the output of the display block (assumption: ODB = Y) to increase from left to right.

OIDR = R (REVerse) causes the display the position of the slider on the output of the display block (assumption: ODB = Y) to increase from right to left.

7.247. OLC - Output Low clamp

Field type: Float

Field Access: R,W

Defined in Blocks: CALC, NLP, SPID, SPLIT, AOT, LL, MFCALC, LB, CTIM

Properties: OLC value is the minimum value the output of the loop body block can achieve. If the result of the processing of the loop body block would be less than OLC value, OLC value is output. Nevertheless, the internal calculation result is not clamped.

OLC also corresponds the console's 0% output range for output display bar. (See “ODSP - Display output of the display block” on page 151).

Note, that attempting to set the output below the OLC limits from EXTOL causes an EXTOL runtime error.

7.248. OPNA - Open cascade action

Field type: Byte

Field Access: R,W

Defined in Blocks: CALC, CTIM, LL, MFCALC, NLP, SPID

Properties: This parameter specifies the AUTO/MANUAL status of the block whenever the downstream cascade linkage is broken (opened). The block may be reset to MANUAL, if OPNA = 1, or allowed to remain in AUTO, if OPNA = 0. If the block is in MANUAL, it will remain in MANUAL in either case.

7.249. OPNT - Output point

Field type: Byte

Field Access: R,W

Defined in Blocks: DEV, DEVOUT, DOT

Properties: Bank selector on the digital output card. On each digital output card there may be one to several16-bit banks of outputs numbered from 1 to 4. Note, however, that all references within a device block can only be made to one 16 bit channel bank.

7.250. OPTI - Stroke time (in seconds)

Field type: Byte

Field Access: R,W

Defined in Blocks: DEV DEVICE

Properties: The interval between setting the desired state and having the actual state match it. If the actual state is different from the desired state after this period of time

has elapsed, a deviation alarm will be generated. The delay between the change of the desired state and an alarm being generated is:

- Min.: OPTI seconds
- Max.: OPTI + 1 seconds

The stroke time is a number between 0 and 255 inclusive.

Zero means that the checking of the deviation is deactivated and the deviation alarm will never be raised. OPTI = 0 has so the same effect as INDV = Y.

Any number different from zero (0) is the number of seconds of the loop to count down before comparing the actual state to the desired state.

Note: The stroke timer is only activated when the desired state of the device has changed. The alarm is not delayed when the actual state of the device changes!

7.251. OPWK - Name of the EXTOL operation to be triggered

Field type: String (14 characters)

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR, DEV, DEVICE, DEVINP, DI, DEVOUT, TIMER

Properties: OPWK = character string specifies the name of an EXTOL operation in the unit UNIT. This field is mandatory for all actions to be triggered and may not be left blank. No check is made to see if the operation name or the step name is valid while running DBB.

Triggering event see

ENCOS, ENDH, ENDL, ENDSC, ENDV, ENHH, ENHI, ENISC
ENLE, ENLL, ENLO, ENRC, ENST, ENTD, ENTU, ENUN

7.252. OROC - Output rate of change limit

Field type: Float

Field Access: R,W

Defined in Blocks: AOT, CALC, DELAY, LL, MFCALC, NLP, SPID

Properties: OROC contains the maximum value the output can change in one control cycle.

The output of a control block may be damped by setting a maximum rate of change to something less than 100% per control cycle. The maximum rate at which the output will be changed can be set by this parameter. Any attempt to adjust the process by more than the allowable limit forces the output to the maximum output ramp rate. On each control cycle, a new output will be computed for positionel algorithms, which allows the remainder from the previous output to be discarded. For velocity output algorithms, the output remainder (if any) is retained and added to the output for the next cycle.

OROC = 0 means no limitation on output rate of change (OROC = 0 and OROC = 99999 are equivalent).

7.253. OUT - Card address in the AOT block

Field type: Ofield

Field Access: R,W

Defined in Blocks: AOT

Properties: OUT is the field where the output block has to write the value in engineering units, which has to be output to the hardware after being transformed to raw counts using the equation:

For INV = NO (0)

raw count = ((output value - ZERO) / (SPAN - ZERO)) * 65536

For INV = YES (1)

raw count = [1 - ((output value - ZERO) / (SPAN - ZERO))] * 65536

Note: By convention, the output value must always be in the range:

ZERO <= output value <= (SPAN - ZERO)

Analog Outputs are serviced by cards that plug into I/O racks. The card location is noted by the rack number and the slot in the rack where the analog output card is located.

The output is assigned to a physical location which must be made known to the scanning software. The physical location is identified by the following data:

- **rack**:
  number of the rack chassis on this node. Note, that this number actually refers to the internally defined Rack blocks, they being numbered sequentially from 1 to n (0 to n-1 are the ivn numbers shown by the database builder utility “DBB”).
- **card**:
  slot number within the rack. This is equivalent to the card number in the rack-block.
- **channel**:
  channel on that card

The card address is stored in the OUT field, entered in the following format:

- #rack-card
- #module

The channelnumber is entered separately to the CHAN field (see CHAN).

7.254. OUT - Output from the control blocks

Field type: Ofield

Field Access: R,W

A new output link can only be defined using either DBB or the special LINKIO instruction in EXTOL.

Defined in Blocks: CALC, LB, LL, MFCALC, NLP, SPID

Properties: The OUT field can contain either a constant or an address of an other database field. The result of the algorithm performed in the control block is stored (after the OHC and OLC clamp and the OROC has been applied) in the OUT field.

If there is a constant in the OUT field , the result of the new calculation is stored as a new constant.

If there is an address of an other database field in the output field , the result of the calculation is stored in the OUT field and also written into the field, specified by the address. If the addressed field is the OUT field of the AOT block, then in addition to the actions described until now the result is written to the hardware, too.

Note: Although the fields OHC, OLC and OROC are not visible on the MFCALC block, they still exist and are defined as default values (OHC=99’999, OLC=-99’999 and OROC=0). They are applied as on a normal CALC block.

7.255. OUT - Output in DELAY block

Field type: Ofield

Field Access: R,W

A new output link can only be defined using either DBB or the special LINKIO instruction in EXTOL.

Defined in Blocks: DELAY

Properties: The input value (state) STPT is read and copied into the OUT field. If the input value changes to the value defined in field TLVL, the output retains the previous value until the delay time has elapsed, then it changes to the current input value (=value in TLVL). Note, that the delay time is restarted on each transition of the input to the value specified in TLVL.

The same rules as defined in "OUT - Output from the control blocks" do also apply.

7.256. OUTA - Output address

Field type: HWLINK

Field Access: R,W

Defined in Blocks: DEV, DEVOUT, DOT

Properties: Logical address of the output card or module. Links the block’s output to the I/O hardware. The physical location is identified by the following data:

- **rack**:
  number of the rack chassis on this node. Note, that this number actually refers to the internally defined Rack blocks, they being numbered sequentially from 1 to n (0 to n-1 are the ivn numbers shown by the database builder utility “DBB”).
- **card**:
  slot number within the rack. This is equivalent to the card number in the rack-block. This number is related to the card addressing and depends on the type of the I/O hardware used. Please refer to the appropriate I/O hardware description (or scanner type).
- **channel**:
  channel on that card.

The card address is stored in the OUT field, entered in the following format:

- #rack-card
- #module

The channel number is entered separately to the CHAN field (see CHAN).

7.257. OUTM - Output mask

Field type: Word

Field Access: R,W

Defined in Blocks: DEV, DEVOUT, DOT

Properties: OUTM contains the hexadecimal mask applied to output bits prior to driving the specified output address and point.

Example: In a DEV block the outputs are physically connected to the card specified in the field OUTA to the 16 bit bank specified in the field OPNT.

Only bits 3,4,5 and 6 "belong" to the DEV block considered, other bits "belong" to other devices. This is reflected in the values of SnO, which contain the bit combinations corresponding to the states of our device, and which encompass only bits 3 to 6. Assume, our device has three defined states:

OFF - S1O = 0000 h

STATE1 - S2O = 0018 h i.e. '0000 0000 00**01 10**00'

STATE2 - S3O = 0024 h i.e. '0000 0000 00**10 01**00'

The field OUTM should contain bits 3,4,5 and 6 set,

i.e. OUTM = 003C h i.e. '0000 0000 00**11 11**00'

Suppose the 16 bit hardware point was previously set to

E1A5 h i.e. '1110 0001 10**10 01**01'

it is obvious that the previous due state of our device was STATE2 (not bold bits on the bank considered belong to other devices).

Binary complement to the OUTM value is

FFC3 h i.e. '1111 1111 11**00 00**11'

Assume the new due state is the STATE2.

The processing goes as follows: From the 16 bit pattern of the hardware point the bits "belonging" to our device are masked out by BAND operation with the complement of OUTM. The result is

E181 h i.e. '1110 0001 10**00 00**01'

The second BAND operation between this result and the bit mask stored in S2O for STATE2 provides

E199 h i.e. '1110 0001 10**01 10**01'

To this value the 16 hardware bits will be driven now. Simple.

7.258. OUTn - n-th output (OUT1 and OUT2) for SPLIT block

Field type: Ofield

Field Access: R,W

A new output link can only be defined using either DBB or the special LINKIO instruction in EXTOL.

Defined in Blocks: SPLIT

Properties: Output fields to specify the output to be driven by the SPLIT block. These fields usually contain the address of the OUT field of an analog output block (A0).

7.259. OUTn - n-th output (OUT1 to OUT8) for digital devices

Field type: Ofield

Field Access: R,W

A new output link can only be defined using either DBB or the special LINKIO

instruction in EXTOL.

Defined in Blocks: DEVICE

Properties: Output fields are linked to other process database fields. The new output state (state1 or state2) is written out to the destination field. Typically links are made to field OUTA of a DOT block (this causes an output to a physical point to be made) or field DEST of another DEV or DEVICE block.

7.260. OVAL - Output value

Field type: Byte

Field Access: R

Defined in Blocks: DOT

Properties: This field represents the actual value of the DOT block.

7.261. OVR - Override state of device

Field type: Byte

Field Access: R,W

Defined in Blocks: DEV, DEVICE, DEVOUT

Properties: The override state takes precedence over the Interlock State (INLK) and over the desired state (DEST). Whenever the Override State is cleared (the field OVR set to zero) then the Interlock State if it is defined, or if undefined the Desired State will be in effect again. It means, there is a three level hierarchy, where the highest level is always in effect and as soon as it is cleared the device will fall back to the next lower level state.

Please note, that this field is also influenced by the EOVR field, See “EOVR - Enable override” on page 95

7.262. PB - Proportional Band

Field type: Float

Field Access: R,W

Defined in Blocks: SPID

Properties: This is the alternative parameter of KP. It may be used instead of KP in the SPID block to specify the gain of the controller.

See “NPAR - Convert from normalized parameters (PB, TR, TD)” on page 147

7.263. PCI - Pulse Counter Input block

Block-Type: PCI

Record-Type: SENSOR

Properties: The pulse counter input block is basically the same as a sensor block except that the hardware parameters are set for input from a pulse counter input module. They are scanned at a regular poll cycle and the current contents of the counter is the raw input reading. Pulse input is used where the external device supplies pulse data. The following describe only the differences from the sensor block definitions.

A special feature allows the user to reset pulse counter values to zero (by setting the field CNTR to 0) when PCI blocks are used. This allows the user to reset the Pulse Counter at the beginning of a dosing operation. This feature is active only, when the field LINC is set to the value 'TPCI' (Totalizing Pulse Counter Input).

7.264. PHAS - Poll phase

Field type: Byte

Field Access: R

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR, DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: The PHAS contains at any time the actual number of 0.25s intervals to wait before executing the loop header block the next time.

For any given POLL time "t", there are only 4t unique quarter second phases.

The PHAS field as specified on the database on the disc allows the user to level the load placed upon the processor. For example, two blocks can both have poll times of one second yet never run simultaneously if the first has a phase of 0 and the second a phase of 2.

If two loops, both with a poll time of 5 second, must always be processed in a specific order, i.e. loop1 must always run before loop2, then this can be achieved by setting the initial value of PHAS accordingly. By setting PHAS of loop1 to 0 and of loop2 to 2 the two loops will be processed with a difference of 0.5 seconds. First loop1 and 0.5 seconds later loop2.

When loop header block is scanned, every 0.5 seconds the contents of the POLL field is decremented by 2. If the contents after decrementation is equal or less zero, the contents of the PHAS field is set equal to the contents of the POLL field and the block is processed.

If the contents of the PHAS field after decrementation is greater than zero, the block is not processed in this scanning cycle.

Note: If the value of the field PHAS (or the value of the field POLL) is changed at runtime, the relative processing order of blocks may be changed. A specific order can only be kept (if relevenat at all) by not altering the fields PHAS and POLL at runtime.

7.265. PID - TCS COMPATIBLE PID BLOCK

Block-Type: PID

Record-Type: PID

Properties: PID block compatible with the TCS system.

The PID block: **should not be used** in new applications. Do use SPID or NLP instead.

The PID algorithm used allows to specify either a positionel or a velocity output mode. Velocity mode is only possible with the PID algorithm. The SPID and NLP algorithms work in positionel mode only.

The integral limitation algorithm used is different from the one used on the SPID and NLP algorithm.

The algorithm implemented for the PID controller is in principal:

In case of Incremental Output:

$$y(k) = KP * [(e(k) - e(k-1)) + KD / POLL * (d(k) - 2*d(k-1) + d(k-2))] + fKP * KI * POLL*e(k) + ILTerm$$

In case of Positional Output:

$$y(k) = KP * [(e(k) + KD / POLL * (d(k) - d(k-1))] + iterm(k) + FFW + ILTerm$$

where:

B(k)	= B input of the PID block ("measurement")
e(k)	= STPT(k) - B(k) in the case of direct action
e(k)	= B(k) - STPT(k)in the case of reverse action
d(k)	= either e(k) if field EB is set to ’E’, or B(k) if field EB is set to ’B’
FFW	= feed forward term
fKP	= KP for KP # 0
	= 1 for KP = 0
iterm(k) = iterm(k-1) + fKP * KI * POLL * e(k)	
KD	= KD value of the PID block (derivative gain)
KI	= KI value of the PID block (integral gain)
KP	= KP value of PID block (proportional gain)
POLL	= the poll-time of the loop
STPT	= STPT value of the PID block ("setpoint")
ILTerm	= Integral Limiting Term. This term is calculated as described in “ILT - Integral limiting action type” on page 107

The algorithm is initialised

e(k-1)	= STPT - B in the case of direct action	
e(k-1)	= B - STPT in the case of reverse action	
d(k-1)	= e(k-1) if EB is set to E, else =B(k)	
d(k-2)	= e(k-1) if EB is set to E, else =B(k)	
iterm(k-1)	= OUT	if FFW = 0.0
itemr(k-1)	= 0.0	if FFW is linked, or nonzero

Note: The internal calculation is not exactly as described in the above equation. The equation explains the principal way of the algorithm. In particular when the PID is used in the full output mode (IF=F) the internal algorithm still works incrementally, it just integrates all the individual increments. Therefore the initialisation influences the result. E.g. the P-term is calculated by summing the following expression:

(e(k) - e(k-1)) * KP

Because of the initialization of e(k-1) equal to e(k), the P-part of the first scan in mode AUTO will always be 0 even though the error is nonzero at this time.

7.266. POLL - Poll time

Field type: Byte

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR, DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: POLL field contains the poll time in seconds (displayed by DBB and input by DBB), but in reality it contains the poll time * 4 (read by database blocks, read by EXTOL from the field POLL).

Note: The POLL time is stored internally in 0.25 second units. I.e if EXTOL reads a value of 8, this is the equivalent of a poll time of 2 seconds.

See “PHAS - Poll phase” on page 158.

Note: If the value of the field POLL (or the value of the field PHAS) is changed at runtime, the relative processing order of blocks may be changed. A specific order can only be kept (if relevenat at all) by not altering the fields PHAS and POLL at runtime.

7.267. PUNI - Parent unit

Field type: String (Size 10 characters)

Field Access: R

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR, DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: PUNI, the parent unit name is a 10 character string. The Parent Unit determines to which unit (which is also a SyCOS database block) this device 'belongs to'. It's purpose is to be able to direct alarm messages to the appropriate batch history file, if BHF has been defined as one of the alarm destinations. If this field contains the name of a valid unit, alarm messages are sent to the alarm destinations defined in the parent Unit’s block and not the alarm destinations defined in this block.

Restrictions: The parent unit must be a unit in the local database. It is absolutely forbidden to modify or delete the name of the parent unit from an EXTOL application program! This may lead to internal system problems.

7.268. PXn - Split range parameters (PX1 and PX2)

Field type: Ifield)

Field Access: R,W

Defined in Blocks: SPLIT

Properties: PX1 and PX2 define the detailed characteristics of the split-algorithm applied:

Split type TYPE = 1

- For input values INP between 0 and PX1 the output OUT1 decreases linearly from 100% (at 0 input) to 0% (at PX1 input) and stays 0% for all higher values if input INP.
- For input values INP between 0 and PX2 the output OUT2 stays 0%; it

increases then linearly from 0% (at PX2 input) to 100% (at 100% input).

Split type TYPE = 2

- For input values INP between 0 and PX1 the output OUT1 increases linearly from 0% (at 0 input) to 100% (at PX1 input) and stays 100% for all higher values if input INP.
- For input values INP between 0 and PX2 the output OUT2 stays 0%; it increases then linearly from 0% (at PX2 input) to 100% (at 100% input).

7.269. RCC - Measured value rate of change alarm to be critical

Field type: Binary

Field Access: R,W

RCC is the 7th bit in the field RCC (decimal value 64, hex value 0x0040)

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: RCC = Y specifies that measured value rate of change alarm will be displayed as critical.

Critical alarm will appear to the operator in RED and will take precedence over reporting of simultaneous major or minor alarms.

If two alarms of equivalent severity are present simultaneously, the displayed alarm will be the one closer to the top of this list:

- LERR
- HIHI and LOLO
- HI and LO
- DHI and DLO
- ROC

7.270. RCM - Measured value rate of change alarm to be major

Field type: Binary

Field Access: R,W

RCM is the 7th bit in the field RCM (decimal value 64, hex value 0x0040)

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: RCM = Y specifies that rate of change alarm will be displayed as major alarm.

Major alarm will appear to the operator in YELLOW and will take precedence over reporting of simultaneous minor alarms. However, reporting of simultaneously present critical alarms takes precedence over reporting major alarms.

If two alarms of equivalent severity are present simultaneously, the displayed alarm will be the one closer to the top of this list:

- LERR
- HIHI and LOLO
- HI and LO
- DHI and DLO
- ROC

7.271. RES - Result

Field type: Float

Field Access: R

Defined in Blocks: CALC, DELAY, LB, LL, MFCALC, NLP, SPID, SPLIT

Properties: The field RES contains the result of the latest calculation. This value reflects the resulting value of the algorithm applied by the block (this value may alraedy be limited by the algorithm itself, such as the integral limitation of a pid algorithm).

Clamping with OHC and OLC is then applied to this resulting value before it is placed into the output field OUT.

Exception: On the SPLIT Block the contents of the RES field is set to the value read from the input filed INP.

7.272. RESE - Reset Flag

Field type: Bit

Field Access: R

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: This field has been added for the convenience of the application programmer. In case the field ERES (Enable Reset) is set to the value Y (=1), then there is a button displayed on the group display's faceplate of this device. When the button is pressed, the field RESE is set to 1 for exactly one processing cycle. It is then cleared again. This allows to implement an interlock reset logic, through a LB block. The field RESE will be set to 1 for exactly one processing cycle.

The field can be used in the application to perform any resetting action required by applying the appropriate logic to a loop.

7.273. ROC - Measured value rate of change limit

Field type: Float

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: If the absolute value of the difference between the actual value of the measured value MEAS and the measured value MEAS in the previous scan exceeds the value in the field ROC, the rate of change alarm is raised.

7.274. SAFE - Loop failsafe output

Field type: Float

Field Access: R,W

Defined in Blocks: AOT

Properties: SAFE field contains the value to which the analog output has to be driven to when a Loop error is detected on the input of the loop and if FSLE flag is set (Y). This output value is typically 100 (percent output) or 0 (percent output). A similar functionality can be achieved, by the definition of the Neutral Position in a SPID, NLP or CALC block and enabling setting the Neutral Position (SNPN = Y) when the loop is switched to MANUAL.

Restrictions: This function is only in effect, when the output was not driven manually by the operator (or from EXTOL), but was driven by an output of a control block in automatic mode.

7.275. SAFE - Input Safe Value

Field type: Float

Field Access: R,W

Defined in Blocks: AIN, SENSOR

Properties: SAFE field contains the value to which the measurement value should take in case the input from the hardware is unavailable or indicates a fault. If the Input Safe Value handling is enabled, no LERR is caused by an invalid input from the hardware.

The typical usage may be to simulate a value, which is outside the alarm limits to cause appropriate alarm handling actions.

Restrictions:

This function is only in effect, when the apporpriate Enable field: EISV is set .

7.276. SCAN - On / Off scan

Field type: Bit

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR, DEV, DEVICE, DEVINP, DI, DEVOUT, DIN

Properties: SCAN = Y (1) means that the loop will be processed. (i.e. have its input scanned, logic blocks activated, outputs driven and alarms serviced).

SCAN = N (0) means the loop is OFF SCAN: the loop will not be processed at all (including all loop body blocks). When the SCAN bit is cleared, there can be no alarm checking or other update of point values at all. OFFSCAN is displayed on the faceplate.

The DIN block with SCAN = Y will be updated corresponding to its input value. The DIN block with SCAN OFF will stay in a frozen state (keeping the last active state when the scan flag changed).

Note: All blocks set to scan off will still get their PHAS updated regularly, in order to keep the loops in the correct order of processing.

7.277. SDSn - Name of the n-th state (SDS1 - SDS2)

Field type: String (Size: 6 characters)

Field Access: R

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: SDSn is the six-character-name of the n-th state of a digital device.

7.278. SENSOR - Analogue Input Block

Block-Type: SENSOR

Record-Type: SENSOR

Properties: Loop header block used to build measuring loops processing hardware based analog signals (i.e.analog signals coming from the hardware). The SENSOR block can also be used as the loop header block to form a control loop. A simple control loop example is given in the figure below.

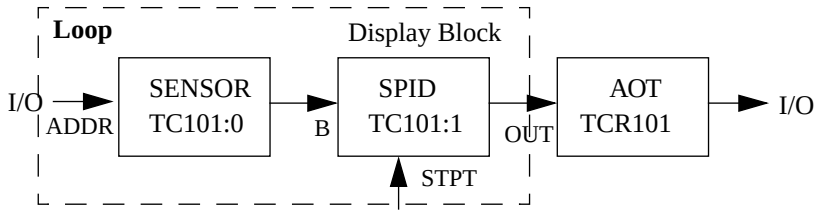


Figure 13: Simple Control Loop using SENSOR Block

7.279. SFOP - Safety Operation

Field type: String (14 characters)

Field Access: R,W

Defined in Blocks: UNIT

Properties: SFOP contains the name of the EXTOL operation which should be started, in case of the detection of a runtime error in EXTOL. If SFOP contains an invalid or currently not loaded operation name, the system watchdog will drop instead.

7.280. SLAVE - Block to be a slave block

Field type: Binary)

Field Access: R,W

Defined in Blocks: CALC, DELAY, LB, LL, MFCALC, NLP, PID, RAMP, SPID, SPLIT

Properties: When the loop is set to MANUAL from faceplate or from EXTOL via the instruction

set loop ... to manual,

the display block and also all the blocks which are defined as “slaves” of this display block will be set to MANUAL, too.

When the loop is set to AUTO from faceplate or from EXTOL via the instruction

set loop ... to auto,

the display block and also all the blocks which are defined as “slaves” of this display block will be set to AUTO, too.

In short, the switch over from MANUAL to AUTO and from AUTO to MANUAL is simultaneously performed on the display block and all control blocks defined as “slaves”. The system periodically checks for the synchronity of all blocks marked as slave blocks. In case of a discrepancy detected, the the loop is marked with a LERR and the loop will automatically be switched to MANUAL.

7.281. SMFL - Simulation Flag

Field type: Bit

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR, DEV, DEVICE, DI, DEVINP

Properties: The flag SMFL when set to 1, allows the simulation of either the analogue input

signal or of the digital input signal.

This feature is normally used to write process simulation programs, using EXTOL operations. Simulation is typically done in the following way:

- On a SENSOR block:

  let SENS100:0.SMFL = 1
  let SENS100:0.ADDR = SENS100:0.ADDR + 1.5

- On a device block:

  let DEV100:0.SMFL = 1
  let DEV100:0.INP = 2

7.282. SMOO - Smoothing

Field type: Float

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI

Properties: Because SyCOS I/O often provides smoothing prior to conversion to digital form, SMOO may usually be turned off. In other cases, in order to eliminate the natural noise of the input data signal, a simple digital filter may be applied. The formula used is called "exponential smoothing" and is the digital equivalent of a resistive/capacitive filter. The "smoothing constant" is the equivalent of the filter time constant referring to the sampling interval. The smoothing equation is as follows:

nsv = (1 - SMOO) * adcv + SMOO * lsv

where:

adcv is the current input value
nsv is the new smoothed value
lsv is the last smoothed value
SMOO is the smoothing constant, between 0 and 1.0

The value of SMOO should be set to approximately p/tc, where tc is the same as the process time constant for the measurement expressed in seconds and p is the poll rate in seconds, such that it takes for a step change in the process to cause 63% of its effect on the variable. Values for SMOO between 0.8 and 0.25 are normal.

A value of 0 automatically results in no smoothing.

7.283. SMOUT - Set Manual Output

Field type: Bit

Field Access: R,W

Defined in Blocks: CALC, NLP, SPID

Properties: SMOUT = Y (1) causes the output OUT to be set to the Manual Output MOUT on switchover from AUTO to MANUAL.

SMOUT = N (0) causes the output OUT to stay on its last value on switchover from AUTO to MANUAL.

7.284. SnI - Input corresponding to n-th state (S1I to S8I)

Field type: Word

Field Access: R,W

Defined in Blocks: DEV, DEVICE, DI, DEVINP

Properties: SnI is the value to which the device driver compares the mask, formed by the input states in INP1 to INP8, to determine the actual state.

Example: If the input mask is equal to 0 x 70, this means that INP 7, 6, 5, 4, 2 are logical 1 (in state 2) and INP 8, 3, 1 are logical 0 (in state 1).

This is the value to which the device driver compares the masked input to determine the state. If the hex value for state 1 is 2000, state 1 name is "OFF", input mask is 2000, and input is 6F80. The point would be in state 1 and display the state OFF.

7.285. SnIE - State-n is an error state (S1IE to S8IE)

Field type: Binary

Field Access: R,W

SnIE is the n-th bit in the field SnIE. The n-th bit corresponds to the n-th state.

Defined in Blocks: DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: SnIE = Y causes an alarm to be generated if the device is in this state. This alarm is called the error state alarm.

7.286. SnO - State-n output (S1O to S8O)

Field type: Word

Field Access: R,W

Defined in Blocks: DEV, DEVICE, DEVOUT

Properties: SnO is the value the device driver outputs to drive the desired state. If the output for state 1 is 0xF000 and the output mask is 0x3000, 0x3000 will be output when the device is driven to state 1.

This is the value the device driver outputs to drive the desired state. Each bit of the 8-bit mask determines the output sent to OUT1 to OUT8. I.e. an output mask of 30h will drive OUT5 and OUT6 to state 2, whereas OUT1 to OUT4 and OUT7 to OUT8 are driven to state 1.

7.287. SNPN - Set neutral position

Field type: Bit

Field Access: R,W

Defined in Blocks: CALC, NLP, SPID. PID

Properties: SNPN = Y (1) causes the output OUT to be set to the neutral position NPOS on switchover from AUTO to MANUAL.

SNPN = N (0) causes the output OUT to stay on its last value on switchover from AUTO to MANUAL.

7.288. SPAN - Engineering units conversion constant

Field type: Float

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: After linearization, the input value may be rescaled to reflect the actual engineering units desired. This is a simple slope and offset calculation using the following formula:

MEAS = csl * SPAN + ZERO

where:

MEAS is the value in engineering units
csl is the current smoothed and linearized value
SPAN is the transformation constant
ZERO is the transformation constant

In case that csl is normalized to the interval between 0 and 1 (as it usually is), then

SPAN is the engineering units upper value minus the lower value and
ZERO is the engineering units lower value

7.289. SPAN - Instrument span of AOT

Field type: Float

Field Access: R,W

Defined in Blocks: AOT

Properties: This field specifies the SPAN of the output in engineering units. This value is used when converting between percentage output and raw counts. The raw output counts are computed as follows:

For INV = NO (0)

raw count = ((OUT - ZERO) / SPAN) * 65536

For INV = YES (1)

raw count = ((OUTmax + OUTmin - OUT) - ZERO) / SPAN * 65536

where:

OUT = Current output value in engineering units
OUTmin = smallest engineering value, which should drive the AOT to the value of MLO.
OUTmax = highest engineering value, which should drive the AOT to the value of MHI.

The parameters for SPAN and ZERO have to be calculated with the knowledge of the interface cards to be driven. The important values, which must be known are the MHI and MLO values. They represent the MUX Low and MUX High raw counts (e.g. MLO=12800 and MHI=64000 for an EXICON AOT card). The actual values are calculated as follows:

ZERO = (MHI * OUTmin - MLO * OUTmax) / (MHI - MLO)
SPAN = ((OUTmin -ZERO) * 65535) / MLO

where:

OUTmin = smallest engineering value, which should drive the AOT to the

value of MLO.
OUTmax = highest engineering value, which should drive the AOT to the value of MHI.

The values of OUTmax and OUTmin can be calculated as follows:

OUTmax = (MHI * SPAN) / 65535 + ZERO
OUTmin = (MLO *SPAN) / 65535 + ZERO

Note: It is important, that the values for MLO and MHI are defined to reflect exactly the minimum and maximum value for the AOT card in use. These values differ significantly for different I/O types.

See the special manual for the I/O hardware in use for further configuration information.

7.290. SPID - SPID BLOCK

Block-Type: SPID

Record-Type: PID

Properties: The algorithm implemented for the SPID controller is:

$$y(k) = KP * [e(k) + KD / POLL * (d(k) - d(k-1))] + iterm(k) + FFW$$

where:

B	= B input of the SPID block ("measurement")
e(k)	= STPT(k) - B(k) in the case of direct action
e(k)	= B(k) - STPT(k)in the case of reverse action
d(k)	= either e(k) if field EI is set to ’E’, or B(k) if field EI is set to ’I’
FFW	= feed forward term
fKP	= KP for KP # 0
	= 1 for KP = 0

iterm(k) = iterm(k-1) + fKP * KI * POLL * e(k)

KD	= KD value of the SPID block (derivative gain)
KI	= KI value of the SPID block (integral gain)
KP	= KP value of SPID block (proportional gain)
POLL	= the poll-time of the loop
STPT	= STPT value of the SPID block ("setpoint")

The algorithm is initialised

e(k-1)	= STPT - B in the case of direct action	
e(k-1)	= B - STPT in the case of reverse action	
d(k-1)	= e(k-1) if EI is set to E, else =B(k)	
iterm(k-1)	= iterm (k-1) = 0	for ITYP = 0 and 1
iterm (k-1) = OUT		for ITYP = 2 and 3

Compatibility Note: The algorithm used to add to the total output the NPOS as well. This is no longer the case. If the output must have a constant offset, then use the FFW field

instead. The field MOUT does define the output position when the loop is switched to MANUAL.

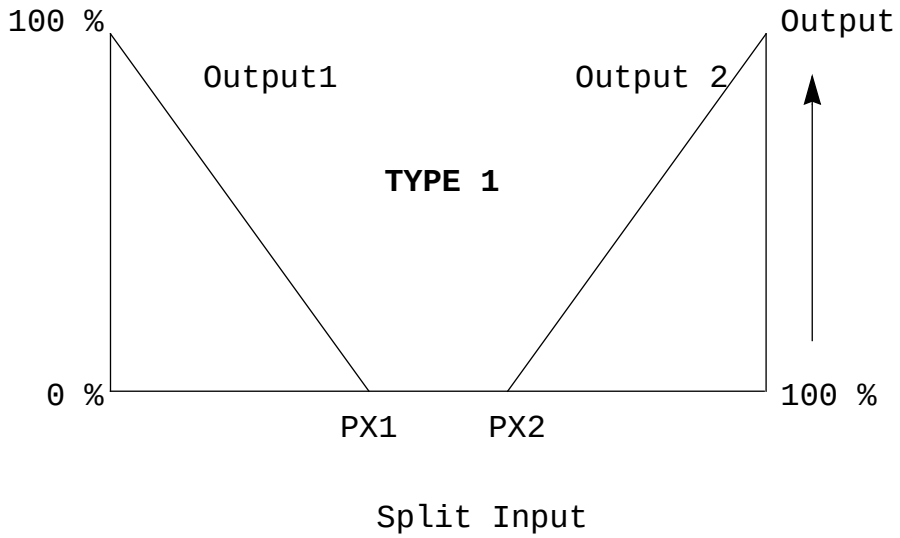
7.291. SPLIT - SPLIT RANGE BLOCK

Block-Type: SPLIT

Record-Type: SPLIT

Properties: The split range block provides an easy way to drive two analog outputs from one control loop in a predefined way. In applications where two outputs have to be driven from one PID controller this is the solution to choose.

Two different algorithms are provided.



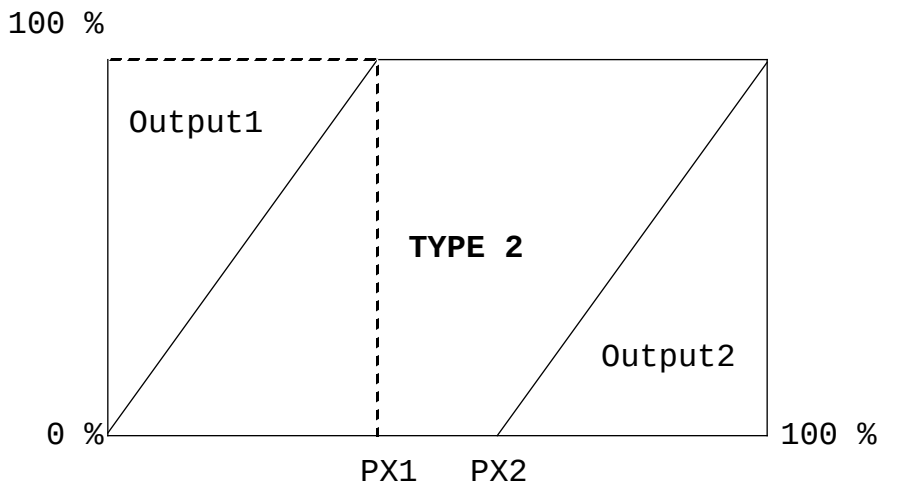


Figure 14: Split Range Types

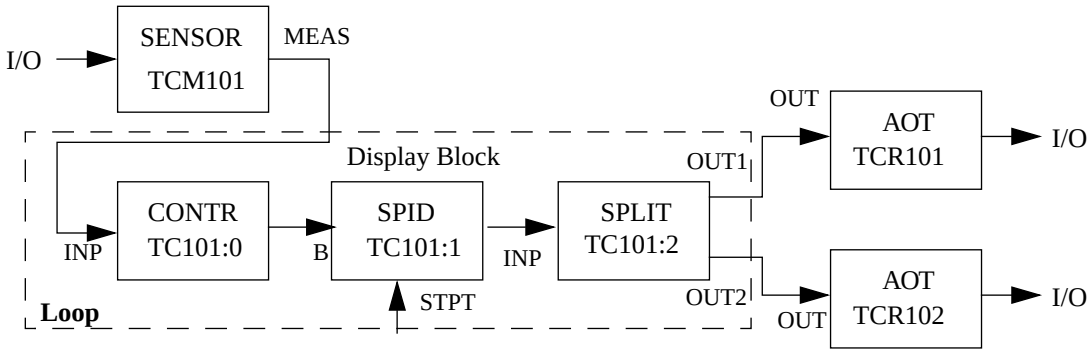


Figure 15: Simple Control Loop Structure using SPLIT Block

7.292. SPWK - EXTOL step the action on which is to be triggered

Field type: String (Size: 14 characters)

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR, DEV, DEVICE, DEVINP, DI, DEVOUT

Properties: SPWK field specifies the name of the main EXTOL step of the operation specified in the field OPWK to be triggered to action specified in EFNC (only used in conjunction with the functions RUN and WAKEUP) on a database event. This parameter is an optional parameter for RUN and WAKEUP operations. No check is made to see if the step name or the operation name is valid during running DBB.

For triggering events event see also

ENCOS, ENDH, ENDL, ENDSC, ENDV, ENHH, ENHI, ENISC
ENLE, ENLL, ENLO, ENRC, ENST, ENTD, ENTU, ENUN

See “OPWK - Name of the EXTOL operation to be triggered” on page 153.

7.293. STAT - State

Field type: Bit

Field Access: R,W

Defined in Blocks: TIMER

Properties: The STAT field contains information on the operational status of the TIMER block. A TIMER block can be in the Run or Stop state. The Run value is 1, and the Stop value is 0. This bit can either be used as a reference value, but also to run/stop a timer.

7.294. STLK - Setpoint lock

Field type: Bit

Field Access: R,W

Defined in Blocks: CALC, LL, NLP, SPID, PID

Properties: STLK = Y locks the access to the setpoint for the operator.

When an upstream block is configured to cascade its output through a cascade control block, and the upstream block is on control, then the downstream block

setpoint is automatically locked.

From EXTOL setpoint can be changed without respect to the contents of STLK field.

7.295. STPT - Input for DELAY block

Field type: Ifield

Field Access: R,W

Defined in Blocks: DELAY

Properties: STPT is the input value which is passed to the OUT field of the DELAY block either immediately (if it is different from the value of TLVL) or after the delay of DEL seconds (if it is equal to TLVL value).

7.296. STPT - Setpoint in CALC Block (input A)

Field type: Ifield

Field Access: R,W

Defined in Blocks: CALC

Properties: The behaviour of the loop body block is different depending on whether the value of STPT is **read in STPT** or **written to STPT** or is **a constant**.

If the value STPT is **read in** STPT, then at each poll the value read is written to the field ENA. If AROC is set to zero (0.0) then the field STPT is set equal to ENA (=value read). If AROC is not zero (0.0) then the field STPT is ramped from it's previous value in increments (or decrements) of AROC each poll cycle until the target value in ENA is reached.

If a value is **written to** STPT from another block of the database, then no ramping willtake place at all, irrespective of the value in AROC.

If a value is **written to** ENA from another block of the database, then ramping of STPT takes place, if AROC is not zero (0.0). The mechanism is the same as when the setpoint is read in or the setpoint is changed from a faceplate.

If the setpoint of the block is changed either from a faceplate or through the EXTOL instruction `set <elt> setpoint to <value>` then the new value is placed into field ENA. If AROC is set to zero (0.0) then the field STPT is set equal to ENA. If AROC is not zero (0.0) then the field STPT is ramped from it's previous value in increments (or decrements) of AROC each poll cycle until the target value in ENA is reached.

Depending on the value of ITYP the value of STPT and simultaneously the value of ENA can be adjusted during the initialization: If ITYP = 1, then the value STPT (and ENA) stays unchanged; if ITYP = 0 then first the **value of B is read in B and only then STPT** (and simultaneously ENA) **is set equal the value of B**. See ITYP. If ITYP = 2 then STPT is set equals to the value of the output field.

Note: STPT (Input A) is read prior to Inputs B, C, D, E, F.

7.297. STPT - Setpoint in control blocks

Field type: Ifield

Field Access: R,W

Defined in Blocks: SPID, NLP. PID

Properties: Current setpoint STPT of the display block is displayed on the faceplate as C. Note that changing the "setpoint" from the faceplate does not change directly the value of STPT of the display block: it is the field ENA which is accessed and changed from the faceplate.

The behaviour of the loop body block is different depending on whether the value STPT is **read in STPT** or **written to STPT** or **a constant**.

If the value STPT is **read in** STPT, then at each poll the value read is written to the field ENA. If AROC is set to zero (0.0) then the field STPT is set equal to ENA (=value read). If AROC is not zero (0.0) then the field STPT is ramped from it's previous value in increments (or decrements) of AROC each poll cycle until the target value in ENA is reached.

If a value is **written to** STPT from another block of the database, then no ramping willtake place at all, irrespective of the value in AROC.

If a value is **written to** ENA from another block of the database, then ramping of STPT takes place, if AROC is not zero (0.0). The mechanism is the same as when the setpoint is read in or the setpoint is changed from a faceplate.

If the setpoint of the block is changed either from a faceplate or through the EXTOL instruction `set <elt> setpoint to <value>` then the new value is placed into field ENA. If AROC is set to zero (0.0) then the field STPT is set equal to ENA. If AROC is not zero (0.0) then the field STPT is ramped from it's previous value in increments (or decrements) of AROC each poll cycle until the target value in ENA is reached.

Depending on the value of ITYP the value of STPT and simultaneously the value of ENA can be adjusted during the initialization:

if ITYP = 1 or 3, then the value of STPT (and ENA) stays unchanged;
if ITYP = 0 or 2, then **new value is read in B and only then STPT** (and simultaneously ENA) **is set equal the value of B during initialization**. See ITYP.

7.298. SWSENSOR - SOFTWARE SENSOR BLOCK

Block-Type: SWSENSOR

Record-Type: SENSOR

Properties: Loop header block used for loops, processing analog signals coming from SENSOR or other SWSENSOR blocks. This block does not fetch its input value from the process I/O hardware.

Note, that there is a different behaviour whether the input value for the SWSENSOR is fetched from the other field , or if the value is written from an other control block to the SWSENSOR.

7.299. TAG - Tag name

Field type: String (Size: 10 Characters

Field Access: This field can only be accessed by DBB.

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR, DEV, DEVICE, DEVINP, DI, DEVOUT, UNIT, DIN, DOT, CTB, ANAMODULE, DIGMODULE

Properties: TAG contains a ten character tag-name, which is the name given to the database loop header block. This name allows the loop to be referenced by other database blocks, by EXTOL programs, and by the operator console.

The tag-name is ten (10) characters in length where the first character must be alphabetic. The remaining characters may be alphabetic letters A - Z or a - z, the numerals 0 through 9 or the underline character (_) which provides internal space.

7.300. TBCC - CTB close condition

Field type: Internal format

Field Access: This field can only be accessed by DBB.

Defined in Blocks: CTB

Properties: The TBCC field is used to define the necessary conditions for a cascade to be closed, and the conditions which will cause it to open.

A number one through eight. This number represents the number of downstream blocks which must be closed before the upstream blocks are connected. Alternatively, it is the minimum number of closed downstream blocks to maintain the cascade in a close condition. When the number of downstream closed blocks becomes less than this, the upstream outputs to the CTB are opened and appropriate action taken as defined in those control blocks.

Normally this is always set to 1, because cascading with several downstream blocks does not make sense when using positionel algorithms.

7.301. TD - Derivative Time

Field type: Ifield

Field Access: R,W

Defined in Blocks: SPID

Properties: This is the alternative parameter of KD. It may be used instead of KD in the SPID block to specify the derivative part of the controller.

See “NPAR - Convert from normalized parameters (PB, TR, TD)” on page 147

7.302. THLD - Threshold value

Field type: Float

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: The contents of these fields is used only if explicitly referred (read or written) from an other database field.

These field can be used by the user to store float values according to his needs and assignment.

Threshold value THLD is the value, the passing of which by MEAS triggers the specified EXTOL function, when ‘threshold up’ and/or ‘threshold down’ is enabled.

7.303. TIME - Timer count

Field type: Float

Field Access: R,W

Defined in Blocks: TIMER

Properties: This field contains the value of the timer. The user may enter an initial value prior to activating, i.e., setting the state to run, in the TIMER block.When the TIMER block is in the run state, the value of TIME is increased (if direction is up) or decreased, (if direction is down) by an amount equal to INC each time the TIMER block is executed (every 0.5 seconds).

7.304. TIMER - Timer Block

Block-Type: TIMER

Record-Type: TIMER

Properties: The Timer block is used to provide calculated time delays. EXTOL programs, in particular can use TIMER blocks to determine whena prescribed time delay has expired and to initiate an EXTOL action, using the action triggering mechanism (i.e. specifying a UNIT, OPWK, STWK and EFNC). An EXTOL action is initiated, when the expired flag is set. On down timers this is on reaching zero.

7.305. TLVL - Delayed state

Field type: Byte

Field Access: R,W

Defined in Blocks: DELAY

Properties: TLVL contains the number of the state, which (and only which) would be delayed by the DELAY block.

If STPT (the input of DELAY block) becomes TLVL, the delay time is started with the value DEL. If during the whole delay time the input STPT has stayed equal TLVL, then after DEL seconds the STPT (=TLVL) value is written to OUT field.

If STPT becomes **not** TLVL, the STPT value ic written to OUT **immediately**.

7.306. TMAN - To manual flag

Field type: Bit

Field Access: R,W

Defined in Blocks: CALC, DELAY, LB, LL, NLP, SPID, SPLIT

Properties: TMAN = Y (=1) forces the loop body block to MANUAL state (AM = M (=0)) with "de-initialization" done (e.g. output of NPOS from SPID block if SNPN =

Y (1), switching of slave loop body blocks to MANUAL if the display block has been sent to MANUAL by TMAN). When this has happened, TMAN is automatically reset to 0. **Note** that the "de-initialisation" is not identical for all of the control blocks, e.g. the LB, LL and SPLIT blocks do not support the neutral position output setting.

"De-initialization" is performed even if the block with TMAN set was already in MANUAL mode (AM = M (=0)).

TMAN is performed immediately after the (eventually by INIT = Y required) initialization before the processing of the block. I.e. if TMAN = Y then the block is not processed. The details can be demonstrated on the following (fictive) case:

Question: What value will the fields have in the following CALC block after the next poll?

CB3:

EQN:	A	
A:	2	
B:	3	
INIT:	1	
ITYP :	0	
TMAN:	1	
OUT:	0	
AM:	irrelevant	

Answer:

EQN:	A	
A:	3	set equal B during initialization
B:	3	
INIT:	0	
ITYP:	0	set to 0 after initialization
TMAN	0	set to 0 after setting to MANUAL
OUT	0	block **has not been** processed, i.e. old value left here
AM:	M	set as part of "de-initialization"

The usage of TMAN and AM fields can be demonstrated on following

Example: Assume loop LOOPS should behave according to LOOPM:

if LOOPM has just been switched from MAN to AUTO,
then LOOPS should be initialized
endif

if LOOPM has just been switched from AUTO to MAN,
then LOOPS should be de-initialized (set to MAN via flag TMAN)
endif

Assume, the display block of LOOPM is block 3, the display block of LOOPS is block 5.

The task can be solved including two CALC block to LOOPS, e.g.

CB6:

EQN:	A=B&C=D?E:F	If

A:	LOOPS:6.C	mode of LOOPM **during the last poll**
B:	0	was MANUAL and
C:	LOOPM:3.AM	mode of LOOPM **now**
D:	1	is AUTO
E:	1	then set 1
F	LOOPS:5.INIT	else leave unchanged
OUT:	LOOPS:5.INITt	he INIT flag of the display block of LOOPS
OLC:	0	
OHC:	1	

CB7:

EQN:	A=B&C=D?E:F	If
A:	LOOPS:7.C	mode of LOOPM **during the last poll**
B:	1	was AUTO and
C:	LOOPM:3.AM	mode of LOOPM **now**
D:	0	is MANUAL
E:	1	then set 1
F	**LOOPS:5.TMAN**	else leave unchanged
OUT:	**LOOPS:5.TMAN**	the TMAN flag of the display block of LOOPS
OLC:	0	
OHC:	1	

This technique works reliably even if the two loops involved have completely different POLL times and PHASes.

7.307. TR - Reset Time

Field type: Ifield

Field Access: R,W

Defined in Blocks: SPID

Properties: This is the alternative parameter of KI. It may be used instead of KI in the SPID block to specify the derivative part of the controller.

See “NPAR - Convert from normalized parameters (PB, TR, TD)” on page 147

7.308. TUNE - Tuning

Field type: Bit

Field Access: R,W

Defined in Blocks: SPID

Properties: When this field is set to 1, then a special data collection is started, which collects data of the control loop, suitable to be used later as input to a utility program "*tuning*". When the flag TUNE is reset to 0, the data collection is stopped and

a file is written to $COSAPPL/database called <elt-name>.TUN.

See also the separate document "SyCOS Loop Tuning and Control Examples".

7.309. TYPE - Split type

Field type: Word

Field Access: R,W

Defined in Blocks: SPLIT

Properties: TYPE specifies the type of split range functions to be applied. Two types of functions are provided:

Split type TYPE = 1

- For input values INP between 0 and PX1 the output OUT1 decreases linearly from 100% (at 0 input) to 0% (at PX1 input) and stays 0% for all higher values if input INP.
- For input values INP between 0 and PX2 the output OUT2 stays 0%; it increases then linearly from 0% (at PX2 input) to 100% (at 100% input).

Split type TYPE = 2

- For input values INP between 0 and PX1 the output OUT1 increases linearly from 0% (at 0 input) to 100% (at PX1 input) and stays 100% for all higher values if input INP.
- For input values INP between 0 and PX2 the output OUT2 stays 0%; it increases then linearly from 0% (at PX2 input) to 100% (at 100% input).

7.310. UBTn - User bytes (USBT1 to USBT4)

Field type: Byte

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR, DEV, DEVICE, DEVINP, DI, DEVOUT, UNIT

Properties: The contents of these fields is used only if explicitly referred (read or written) from an other database field.

These fields can be used by the user to store byte values according to his needs and assignment.

7.311. UFLn - User float (UFL1 to UFL4)

Field type: Float

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR, DEV, DEVICE, DEVINP, DI, DEVOUT, UNIT

Properties: The contents of these fields is used only if explicitly referred (read or written) from an other database field.

These fields can be used by the user to store float values according to his needs and assignment.

7.312. UNIT - Unit on which EXTOL action will be triggered

Field type: String (Size: 10 characters)

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR, DEV, DEVICE, DEVINP, DI, DEVOUT, TIMER

Properties: UNIT specifies the name of the unit in which the EXTOL operation OPWK on the step SPWK should be triggered to the action EFNC on specified triggering events. This field is mandatory for proper action triggering.

No check is made to see if the operation name in OPWK or the step name in SPWK is valid while running DBB.

7.313. UNIT - EXTOL Unit Block

Block-Type: UNIT

Record-Type: UNIT

Properties: The unit block is a block, which is not processed by a periodical scan program, like most other database blocks. It is a "place-holder" for data, most of it related to EXTOL and the concept of using units. Where a unit (a logical entity) typically represents a physical unit of the installation.

The unit block contains a list of alarm destinations (AD1 ... AD16). Whenever the field PUNI in any of the database blocks is defined (i.e. contains the elt name of a UNIT block within the same database node) the system uses automatically the list of alarm destinations defined in the parent unit's UNIT block, instead of the ones in its own block.

The functionality of EXTOL applications depends on the existence of the appropriate UNIT block in the database. I.e. all programs (operations and steps) of a unit can only be loaded and run, when the UNIT block exists in the database.

7.314. USHn - User short (USH1 to USH4)

Field type: Word

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR, DEV, DEVICE, DEVINP, DI, DEVOUT, UNIT

Properties: The contents of these fields is used only if explicitly referred (read or written) from an other database field.

These fields can be used by the user to store integer values according to his needs and assignment.

7.315. VC - Velocity compensation factor

Field type: Ifield

Field Access: R,W

Defined in Blocks: NLP

Properties: The velocity compensation can be used to improve the stability of a control loop. The purpose of it can be determined from the figure below. Simply spoken, it

acts in a way, such that the controller error (en) reaches zero before the measurement has really reached the setpoint. If the VC field is set to 0.0, then no compensation will be carried out.

The equation applied is as follows:

```
if VC = 0.0
    en = stpt -inp

if VC != 0.0
    en = stpt - inp - (vc * (d(inp)/dt))
```

The expression dinp/dt is calculated using a running average and applying a filter, which can be influenced by the field FILT (range 0.0 to 1.0, where 1.0 represents no filtering)

.

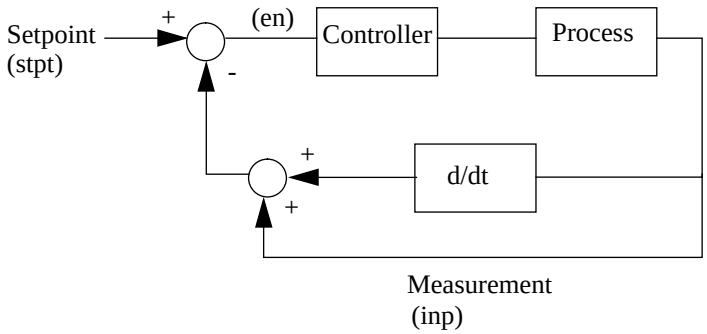


Figure 16: Velocity Compensation

7.316. VP - Velocity / Position mode

Field type: Bit

Field Access: R,W

Defined in Blocks: CALC, LL, MFCALC

Properties: VP = P causes the algorithm to generate a positionel output. This mode is the default one.

In the positional mode, the output value is the result obtained from executing the algorithm, where finally the Hi/Low Clamping (OHC, OLC) is applied:

Output = Current Result (clamped, if necessary)

VP = V causes the algorithm to generate a velocity output value. It is recommended not to use this mode for normal control loops.

In the velocity mode, the output is incremented by a value calculated from the current and previous processing:

Increment = Current Result - Previous Result

The increment value is limited by OROC if OROC is not equal to zero.

Output = Output from Output Link + Increment

7.317. ZERO - Engineering units conversion constant

Field type: Float

Field Access: R,W

Defined in Blocks: SENSOR, SWSENSOR, PCI, CONTR

Properties: After smoothing and linearization, the input value may be rescaled to reflect the actual engineering units desired. This is a simple slope and offset calculation using the following formula:

MEAS = cslv * SPAN + ZERO

where:

MEAS: is the value in engineering units
cslv: is the current smoothed and linearized value
SPAN: is the Engineering units conversion constant
ZERO: is the Engineering units conversion constant

In case that cslv is normalized to the interval between 0 and 1 (as it usually is), than

SPAN is the engineering units upper value minus the lower value and
ZERO is the engineering units lower value

7.318. ZERO - Engineering units conversion constant for AOT

Field type: Float

Field Access: R,W

Defined in Blocks: AOT

Properties: This value is used to calculate the raw output value from the engineering value fed into the AOT-block.

See “SPAN - Instrument span of AOT” on page 168See “SPAN - Instrument span of AOT” on page 168

8. Index

S

Support 13