

IndraWorks 15VRS

Basis-Bibliotheken, IndraLogic 2G

**Bibliothek
R911398632**

Ausgabe 01



Titel	IndraWorks 15VRS Basis-Bibliotheken, IndraLogic 2G
Art der Dokumentation	Bibliothek
Dokumentations-Type	DOK-IL*2G*-BASLIB**V15-LI01-DE-P
Interner Ablagevermerk	RS-807b6ea0f5afb4e60a347e8632e3d462-1-de-DE-13
Änderungsverlauf	Ausgabe 01, 2019-07 Siehe Tab. 1-1 "Änderungsverlauf" auf Seite 1
Schutzvermerk	© Bosch Rexroth AG 2019 Alle Rechte vorbehalten, auch bzgl. jeder Verfügung, Verwertung, Reproduktion, Bearbeitung, Weitergabe sowie für den Fall von Schutzrechtsanmeldungen.
Verbindlichkeit	Die angegebenen Daten dienen allein der Produktbeschreibung und sind nicht als zugesicherte Eigenschaften im Rechtssinne zu verstehen. Änderungen im Inhalt der Dokumentation und Liefermöglichkeiten der Produkte sind vorbehalten.
Redaktion	Entwicklung Automationssysteme SPS-Engineering, KWa (MiNi)

Inhaltsverzeichnis

	Seite
1 Zu dieser Dokumentation.....	1
1.1 Gültigkeit der Dokumentation.....	1
1.2 Basis-Bibliotheken, IndraLogic 2G - Überblick.....	1
1.3 Gebrauch der Sicherheitshinweise.....	1
1.3.1 Aufbau der Sicherheitshinweise.....	1
1.3.2 Erläuterung der Signalwörter und der Signalgrafik.....	2
1.3.3 Verwendete Symbole.....	3
1.3.4 Erläuterung der Signalgrafik auf dem Gerät.....	3
1.4 Erforderliche und ergänzende Dokumentationen.....	3
2 Standardschnittstellen an Funktionsbausteinen.....	5
2.1 Motivation.....	5
2.2 Bausteintypen.....	5
2.3 Bausteintypen und ihre Schnittstellen.....	6
2.4 Ein- und Ausgänge zustandsgesteuerter Funktionsbausteine, abschließende Bearbeitung.....	7
2.5 Ein- und Ausgänge flankengesteuerter Funktionsbausteine, abschließende Bearbeitung.....	10
2.6 Ein- und Ausgänge zustandsgesteuerter Funktionsbausteine, permanente Bearbeitung.....	13
2.7 Ein- und Ausgänge flankengesteuerter Funktionsbausteine, permanente Bearbeitung.....	16
3 RIL_CommonTypes.library.....	19
3.1 Einführung und Übersicht.....	19
3.2 Globale_Variablen.....	20
3.2.1 IL_NULL.....	20
3.3 Datentypen zum Referenzieren von Objekten bei Motion-Funktionsbausteinen.....	20
3.3.1 Referenzierung von Objekten, Einführung und Übersicht.....	20
3.3.2 CONTROLS.....	21
3.3.3 AXIS_REF.....	21
3.3.4 OBJECTS.....	22
3.3.5 AXIS_GROUP_REF.....	23
3.3.6 KIN_OBJECTS.....	24
3.3.7 TOUCHPROBE_REF.....	25
3.3.8 TP_OBJECTS.....	26
3.4 Datentypen zur Unterstützung von Kurvenscheiben, Motion- und FlexProfilen.....	26
3.4.1 Datentyp zum Umschaltung zwischen Kurvenscheiben, allgemeines.....	26
3.4.2 MC_CAM_ID.....	26
3.4.3 MC_STEP_MODE, MotionProfile erweitert zu FlexProfile.....	27
3.5 Datentypen zum Aufsynchronisieren in den lagegeregelten Synchron-Betriebsarten.....	51
3.5.1 Synchronisation, Einführung und Übersicht.....	51
3.5.2 MC_START_MODE.....	51
3.5.3 MC_SYNC_DIRECTION.....	53
3.6 Datentypen zur Fehlerdiagnose bei Motion-Funktionsbausteinen.....	54
3.6.1 ERROR_CODE.....	54
3.6.2 ERROR_STRUCT.....	55

	Seite
3.6.3 ERROR_TABLE.....	56
3.7 Datentypen zur Wiedergabe des Datum und absoluter Zeitangaben bei Motion-Funktionsbausteinen.....	57
3.7.1 Wiedergabe des Datum und absoluter Zeitangaben, Einführung und Übersicht.....	57
3.7.2 SYS_TIME64.....	57
3.7.3 EXT_SYS_TIME_DATE.....	57
3.7.4 SYS_TIME_DATE.....	58
3.8 Datentypen zur Unterstützung von Messtaster-Funktionen.....	58
3.8.1 Datentypen zur Unterstützung von Messtaster-Funktionen, allgemeines.....	58
3.8.2 MB_DIFF_EVAL.....	59
3.8.3 MB_EXPECT_WINDOW_MODE.....	60
3.8.4 MB_INTERPOLATION_MODE.....	60
3.8.5 MB_POSITION_EVAL.....	61
3.8.6 MB_PROBE_MODULE.....	61
3.8.7 MB_PROBE_INPUT.....	61
3.8.8 MB_PROBE_DATA.....	62
3.8.9 MB_PROBE_CONFIG.....	63
3.8.10 MB_PROBE_CONFIG, Besonderheiten bei Sercos-basierten Achsen.....	63
4 RIL_Uutilities.library.....	65
4.1 Einführung und Übersicht.....	65
4.2 Vertippen von Variablen (Adjustment).....	69
4.2.1 Einführung und Übersicht.....	69
4.2.2 IL_ContinuousAdjustType01.....	72
4.2.3 IL_ContinuousAdjustType02.....	76
4.2.4 IL_IncrementalAdjustType01.....	79
4.2.5 Parametrierung der Funktionsbausteine zum Vertippen von Variablen.....	81
4.3 Basisklassen für Funktionsbausteinentwicklung (Base FB).....	82
4.3.1 Einführung und Übersicht.....	82
4.3.2 Basisfunktionsbaustein IL_EnableDoneBase.....	87
4.3.3 Basisfunktionsbaustein IL_EnableInOpBase.....	90
4.3.4 Basisfunktionsbaustein IL_EnableInOpShutdownBase.....	92
4.3.5 Basisfunktionsbaustein IL_ExecuteDoneBase.....	97
4.3.6 Basisfunktionsbaustein IL_ExecuteInOpBase.....	99
4.3.7 Basisfunktionsbaustein IL_ExecuteDoneAbortedBase.....	102
4.3.8 Basisfunktionsbaustein IL_ExecuteDoneAbortedShutdownBase.....	105
4.3.9 IL_FB_STATES.....	110
4.4 Konvertierung (Bit-Stream Connect).....	111
4.4.1 Einführung und Übersicht.....	111
4.4.2 IL_BoolToByte.....	111
4.4.3 IL_BoolToDWord.....	112
4.4.4 IL_BoolToWorld.....	115
4.4.5 IL_ByteToDWord.....	116
4.4.6 IL_ConcatByte.....	117
4.4.7 IL_ConcatWord.....	118
4.5 Konvertierung (Bit-Stream Split).....	119

	Seite
4.5.1 Einführung und Übersicht.....	119
4.5.2 IL_ByteToBool.....	119
4.5.3 IL_DWordToBool.....	120
4.5.4 IL_DWordToByte.....	123
4.5.5 IL_HighByte.....	123
4.5.6 IL_HighWord.....	124
4.5.7 IL_LowByte.....	124
4.5.8 IL_LowWord.....	125
4.5.9 IL_WordToBool.....	125
4.6 Konvertierung (ByteSwap).....	127
4.6.1 Einführung und Übersicht.....	127
4.6.2 IL_SwapDWord.....	127
4.6.3 IL_SwapWord.....	128
4.7 Prüfsummen.....	129
4.7.1 Einführung und Übersicht.....	129
4.7.2 IL_CRC16.....	129
4.8 Konvertierung (Datatype Change).....	130
4.8.1 Einführung und Übersicht.....	130
4.8.2 IL_AsciiToByte.....	130
4.8.3 IL_ByteToAscii.....	131
4.8.4 IL_RealToString.....	131
4.8.5 IL_LRealToString.....	132
4.9 Debuggen (Debugging).....	133
4.9.1 Einführung und Übersicht.....	133
4.9.2 IL_HighResTimeDiff.....	134
4.9.3 IL_HighResTimeTick.....	134
4.10 FlipFlops (FlipFlops).....	135
4.10.1 Einführung und Übersicht.....	135
4.10.2 IL_Toggle.....	136
4.11 Mathematische Funktionen (Mathematical Functions).....	137
4.11.1 Einführung und Übersicht.....	137
4.11.2 IL_InterpolationLinear.....	137
4.11.3 IL_MaxValue.....	139
4.11.4 IL_COORD_PAIR_TYPE01.....	140
4.12 Zeichenfolgen ohne Längenbegrenzung (StringFun_NoLimitStrLength).....	141
4.12.1 Einführung und Übersicht.....	141
4.12.2 IL_StrLower.....	144
4.12.3 IL_StrUpper.....	144
4.12.4 IL_StrCompare.....	145
4.12.5 IL_StrCompareNoCase.....	146
4.12.6 IL_StrNCompare.....	146
4.12.7 IL_StrNCompareNoCase.....	147
4.12.8 IL_AsciiStreamToBase64.....	148
4.12.9 IL_Base64ToAsciiStream.....	149
4.12.10 IL_BufferToString.....	151
4.12.11 IL_StringToBuffer.....	153

Inhaltsverzeichnis

	Seite
4.12.12 IL_StrCopy.....	154
4.12.13 IL_StrNCopy.....	155
4.12.14 IL_CONCAT.....	156
4.12.15 IL_DELETE.....	157
4.12.16 IL_FIND.....	158
4.12.17 IL_INSERT.....	158
4.12.18 IL_LEFT.....	159
4.12.19 IL_LEN.....	160
4.12.20 IL_MID.....	161
4.12.21 IL_REPLACE.....	162
4.12.22 IL_RIGHT.....	163
4.12.23 IL_Ascii_To_Char.....	164
4.12.24 IL_Char_To_Ascii.....	164
4.12.25 IL_Int_To_Str.....	164
4.12.26 ENUM_INT_OPTION.....	165
4.12.27 IL_Real_To_Str.....	165
4.12.28 ENUM_REAL_OPTION.....	166
4.12.29 IL_String_To_Unicode.....	166
4.12.30 ENUM_OPT_UTF.....	167
4.12.31 IL_Unicode_To_String.....	167
4.12.32 IL_BufferGetElementCount.....	168
4.12.33 IL_Code.....	169
4.12.34 IL_StrInverse.....	170
4.12.35 IL_StrSetEndOfString.....	171
4.12.36 IL_StrGetElement.....	171
4.12.37 IL_StrLChr.....	172
4.12.38 IL_StrRChr.....	173
4.12.39 IL_StrSetElement.....	174
4.12.40 IL_StrSplit.....	175
4.12.41 IL_StrTrimAll.....	176
4.12.42 IL_StrTrimSpace.....	177
4.12.43 ENUM_TRIM_OPT.....	178
4.13 Zeit- und Datumberechnung, Version 2 (Time).....	179
4.13.1 Einführung und Übersicht.....	179
4.13.2 IL_Date.....	179
4.13.3 IL_DateAndTime.....	180
4.13.4 IL_ExtSysTimeDate.....	180
4.13.5 IL_LocalTime.....	181
4.13.6 IL_SysTime64.....	183
4.13.7 IL_SysTime64ToSysTimeDate.....	184
4.13.8 IL_SysTimeDate.....	184
4.13.9 IL_SysTimeDateToSysTime64.....	185
4.13.10 IL_TimeOfDay.....	185
4.13.11 IL_TIMEZONE.....	186
4.14 Doppelt-verkettete Listen (Utilities/ Logic).....	189
4.14.1 Einführung und Übersicht.....	189

	Seite
4.14.2 IL_LinkedList.....	190
4.14.3 IL_LINKED_LIST_MAINTENANCE.....	194
4.14.4 IL_LINKED_LIST_MODE.....	194
4.15 Modulwert-Berechnung (Utilities/ Logic).....	195
4.15.1 Einführung und Übersicht.....	195
4.15.2 IL_GetModuloType01.....	196
4.15.3 IL_GetModuloType02.....	197
4.15.4 IL_GetModuloType03.....	198
4.15.5 IL_GetModuloType04.....	199
5 RIL_ModbusTCP.library.....	201
5.1 Einführung und Übersicht.....	201
5.2 ModbusTCP Server.....	202
5.2.1 ModbusTCP Server, Anwendungen.....	202
Allgemein.....	202
Projektierung mit Siemens WinCC-flexible 2007.....	204
Projektierung mit Pro-face GP-Pro EX 2.2.....	208
5.2.2 IL_ModbusTCPServer.....	216
5.2.3 IL_ModbusTCPServerType02.....	219
5.2.4 IL_MODBUS_STATS.....	225
5.3 ModbusTCP Client.....	226
5.3.1 Einführung und Übersicht.....	226
5.3.2 IL_ModbusTCPClient.....	226
5.3.3 IL_ModbusTCPReadCoils.....	229
5.3.4 IL_ModbusTCPReadDiscreteInputs.....	231
5.3.5 IL_ModbusTCPReadHoldingRegisters.....	234
5.3.6 IL_ModbusTCPReadInputRegisters.....	236
5.3.7 IL_ModBusTCPWriteSingleCoil.....	239
5.3.8 IL_ModBusTCPWriteSingleRegister.....	241
5.3.9 IL_ModBusTCPWriteMultipleCoils.....	243
5.3.10 IL_ModBusTCPWriteMultipleRegisters.....	245
5.3.11 IL_ModBusTCPReadWriteMultipleRegisters.....	248
5.3.12 IL_ModbusTCPReadDeviceIdentification.....	251
6 RIL_SocketComm.library.....	255
6.1 Einführung und Überblick.....	255
6.2 I/O Zeitverhalten der Funktionsbausteine.....	255
6.3 Funktionale Beschreibung.....	256
6.3.1 TCP.....	256
6.3.2 UDP.....	260
6.3.3 ICMP.....	267
6.3.4 FTP.....	267
6.3.5 TFTP.....	267
6.3.6 Zyklische Kommunikation.....	268
6.3.7 Byte-Reihenfolge in IP-basierter Kommunikation.....	268

Inhaltsverzeichnis

	Seite
6.3.8 Security - unsichere Protokolle abschalten.....	269
6.4 TCP Asynchrone Funktionsbausteine.....	269
6.4.1 Einführung und Überblick.....	269
6.4.2 IL_TCPConnectionAsync.....	269
6.4.3 IL_TCPInitialAsync.....	273
6.4.4 IL_TCPSendAsync.....	277
6.4.5 IL_TCPRecvAsync.....	279
6.4.6 IL_TCPRecvPacketAsync.....	281
6.4.7 IL_TCPCloseAsync.....	283
6.5 TCP Synchrone Funktionen.....	284
6.5.1 Einführung und Überblick.....	284
6.5.2 IL_TCPInitial.....	285
6.5.3 IL_TCPSend.....	286
6.5.4 IL_TCPRecv.....	287
6.5.5 IL_TCPRecvPacket.....	288
6.5.6 IL_TCPClose.....	290
6.6 UDP Asynchrone Funktionsbausteine.....	290
6.6.1 Einführung und Überblick.....	290
6.6.2 IL_UDPSocketAsync.....	291
6.6.3 IL_UDPInitialAsync.....	293
6.6.4 IL_UDPSendtoAsync.....	295
6.6.5 IL_UDPRecvfromAsync.....	297
6.6.6 IL_UDPCloseAsync.....	298
6.7 UDP Synchrone Funktionen.....	299
6.7.1 Einführung und Überblick.....	299
6.7.2 IL_UDPInitial.....	300
6.7.3 IL_UDPSendto.....	301
6.7.4 IL_UDPRecvfrom.....	302
6.7.5 IL_UDPClose.....	304
6.8 Asynchrone ICMP-Funktionsbausteine.....	304
6.8.1 IL_PingAsync.....	304
6.9 Synchrone ICMP-Funktionen.....	306
6.9.1 IL_Ping.....	306
6.10 Asynchrone FTP-Funktionsbausteine.....	307
6.10.1 Einführung und Überblick.....	307
6.10.2 IL_FTPGetAsync.....	307
6.10.3 IL_FTPPutAsync.....	310
6.10.4 IL_FTPGetFileAsync.....	313
6.10.5 IL_FTPPutFileAsync.....	315
6.11 Asynchrone TFTP-Funktionsbausteine.....	318
6.11.1 Einführung und Überblick.....	318
6.11.2 IL_TFTPGetFileAsync.....	318
6.11.3 IL_TFTPPutFileAsync.....	321
6.12 Asynchrone Kommunikationsbausteine zur zyklische Kommunikation.....	323
6.12.1 Einführung und Überblick.....	323
6.12.2 IL_TCPCyclic.....	323

	Seite
6.12.3 IL_UDPCyclic.....	328
6.12.4 IL_CONN_MODE.....	332
6.12.5 IL_CONN_STATUS.....	333
6.12.6 ETHERNET_TABLE, ERROR_TABLE 16#0150	334
ETHERNET_TABLE, Allgemeines.....	334
ETHERNET_TABLE, General, 16#00000001-16#00000212.....	335
ETHERNET_TABLE, IL_TcpCyclic, IL_UdpCyclic, 16#00001001-16#00001002.....	336
ETHERNET_TABLE, IL_FTPGetAsync, IL_FTPPutAsync, IL_FTPGetFileAsync, IL_FTP_PutFileAsync, 16#00002001-16#00002004.....	336
ETHERNET_TABLE, IL_TFTPGetFileAsync, IL_TFTPPutFileAsync, 16#00002101-16#00002108.....	337
7 RIL_LoopControl.library.....	339
7.1 Einführung und Übersicht.....	339
7.2 Zeitdiskrete Funktionsbausteine.....	341
7.3 Eventgesteuerte Funktionsbausteine.....	342
7.4 Regler (Controllers).....	343
7.4.1 Einleitung und Überblick.....	343
7.4.2 IL_PIDType01.....	343
7.4.3 IL_PIDType02.....	347
7.4.4 IL_PIDType03.....	351
7.4.5 IL_PIDType04.....	355
7.4.6 IL_PIType01.....	360
7.4.7 IL_PIType02.....	364
7.5 Grundelemente (Elements).....	368
7.5.1 Einführung und Übersicht.....	368
7.5.2 IL_DeadtimeType01.....	369
7.5.3 IL_DeadtimeType02.....	372
7.5.4 IL_DeviationResponseCurve.....	375
7.5.5 IL_DT1Type01.....	377
7.5.6 IL_DT1Type02.....	379
7.5.7 IL_PDT1Type01.....	381
7.5.8 IL_PDT1Type02.....	385
7.5.9 IL_PT1Type01.....	388
7.5.10 IL_PT1Type02.....	390
7.5.11 IL_TransportDelayType01.....	393
7.5.12 IL_TransportDelayType02.....	396
7.5.13 IL_ValueLimit.....	399
7.5.14 IL_ValueWindow.....	401
7.6 Filter (Filter).....	402
7.6.1 Einführung und Übersicht.....	402
7.6.2 IL_FIRType01.....	403
7.6.3 IL_FIRType02.....	405
7.6.4 IL_IIRType01.....	407
7.6.5 IL_IIRType02.....	409
7.6.6 IL_StateVariableType01.....	412

Inhaltsverzeichnis

	Seite
7.6.7 IL_StateVariableType02.....	414
7.6.8 IL_FILTER_TYPE.....	417
7.6.9 IL_FIR_COEFF.....	417
7.6.10 IL_IIR_COEFF.....	417
7.7 Signalgeneratoren (Generators).....	418
7.7.1 Einführung und Übersicht.....	418
7.7.2 IL_NoiseGeneratorType01.....	419
7.7.3 IL_NoiseGeneratorType02.....	420
7.7.4 IL_PRBSGeneratorType01.....	422
7.7.5 IL_PRBSGeneratorType02.....	424
7.7.6 IL_AngleGeneratorType01.....	426
7.7.7 IL_AngleGeneratorType02.....	428
7.7.8 IL_PhaseController.....	429
7.7.9 IL_SignalGeneratorType01.....	431
7.7.10 IL_SignalGeneratorType02.....	435
7.7.11 IL_AmplitudeControllerType01.....	439
7.7.12 IL_AmplitudeControllerType02.....	442
7.7.13 IL_AMP_CONTROL.....	445
7.7.14 IL_CENTERLINE_CONTROL.....	445
7.7.15 IL_CURVETYPE.....	446
7.8 Statistikbausteine (Statistics).....	446
7.8.1 Einführung und Überblick.....	446
7.8.2 IL_HistogramType01.....	447
7.8.3 IL_HistogramType02.....	449
7.8.4 IL_StatisticType01.....	451
7.8.5 IL_StatisticType02.....	453
7.8.6 IL_SuppressionType01.....	455
7.8.7 IL_SuppressionType02.....	460
7.8.8 IL_SuppressionType03.....	465
7.8.9 IL_SuppressionType04.....	470
7.9 Temperaturregler (TemperatureController).....	477
7.9.1 Einführung und Überblick.....	477
7.9.2 IL_TempControlType01.....	477
7.9.3 IL_TempControlType02.....	495
7.9.4 IL_TEMPCONTROL_CONTR_PARAM.....	506
7.9.5 IL_TEMPCONTROL_CONTROL_MODE.....	506
7.9.6 IL_TEMPCONTROL_CONTROL_TYPE.....	507
7.9.7 IL_TEMPCONTROL_EXTENDED_SETTINGS.....	507
7.9.8 IL_TEMPCONTROL_LIMITS.....	508
7.9.9 IL_TEMPCONTROL_MODEL_PARAM.....	509
7.10 Zweipunktregler.....	509
7.10.1 IL_TwoPosCtrlType01.....	509
7.11 Utilities.....	511
7.11.1 IL_COORD_TYPE01.....	511
7.11.2 IL_SUPPRESSION_LIMITS.....	511
7.11.3 IL_SUPPRESSION_MODE.....	512

	Seite
8 RIL_HMI_Uilities.library.....	513
8.1 Einführung und Übersicht.....	513
8.2 IL_MKeys.....	513
8.3 IL_UdpKeys.....	514
9 IL_VCP_DP.library.....	517
9.1 Einführung und Übersicht.....	517
9.2 VCP_PBS16_A4096	517
9.3 VCP_PBS32_A4096	518
9.4 VCP_PBS32_A65536	519
10 Service und Support.....	521
Index.....	523

1 Zu dieser Dokumentation

Ausgaben dieser Dokumentation

Ausgabe	Stand	Bemerkung
01	2019-07	Erstausgabe zur Version 15VRS

Tab. 1-1: Änderungsverlauf

1.1 Gültigkeit der Dokumentation

Zielgruppe Diese Dokumentation richtet sich an Anwender, die sich über die grundlegenden Bibliotheken der Systeme MLC und MTX informieren möchten.

Erforderliche Qualifikationen: Person, die aufgrund ihrer fachlichen Ausbildung, Kenntnisse und Erfahrungen sowie Kenntnis der einschlägigen Normen und Bestimmungen die ihr übertragenen Arbeiten beurteilen und mögliche Gefahren erkennen kann.

1.2 Basis-Bibliotheken, IndraLogic 2G - Überblick

Nachstehend die Beschreibung der **IndraLogic 2G-Basis-Bibliotheken**, geordnet nach den Systemen **MLC**, **MTX** und **MLD**:

Name	MTX	MLC	MLD
RIL_CommonTypes.library , Seite 19,	x	x	
RIL_Uilities.library , Seite 65, allgemeine IL-Funktionen und IL-Funktionsbausteine		x	
RIL_ModbusTCP.library , Seite 201, Diagnose- und Kommunikationsdienste für ModbusTCP	x	x	x
RIL_SocketComm.library , Seite 255, Funktionen zur Kommunikation über TCP/IP und UDP	x	x	
RIL_LoopControl.library , Seite 339, Regler-Bausteine	x	x	x
RIL_NetXLoad, nur für interne Verwendung		x	
RIL_HMI_Uilities.library , Seite 513, Funktionen zum Einlesen von HMI-Mkeys	x	x	
IL_VCP_DP.library , Seite 517, Funktionen zur Ansprache von Kleinbedienfeldern (VCP) über Profibus-DP			

MLC Bibliothek setzt das Basis-Funktionspaket PLC voraus
 Tab. 1-2: Basis-Bibliotheken und durch sie unterstützte Systeme

1.3 Gebrauch der Sicherheitshinweise

1.3.1 Aufbau der Sicherheitshinweise

Die Sicherheitshinweise sind wie folgt aufgebaut:

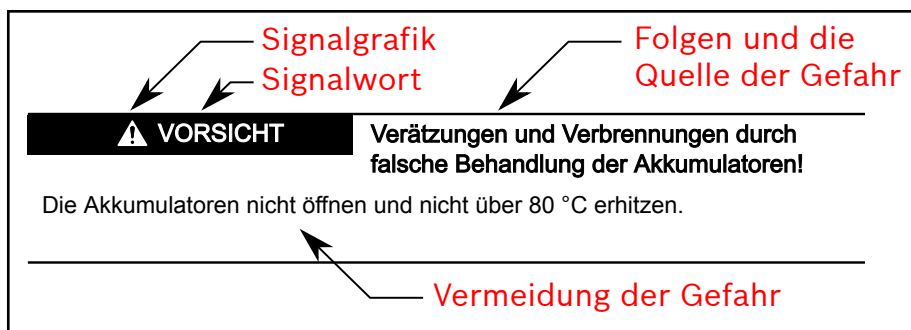


Abb. 1-1: Aufbau der Sicherheitshinweise

1.3.2 Erläuterung der Signalwörter und der Signalgrafik

Die Sicherheitshinweise in der vorliegenden Dokumentation beinhalten bestimmte Signalwörter (Gefahr, Warnung, Vorsicht, Hinweis) und gegebenenfalls eine Signalgrafik (nach ANSI Z535.6-2006).

Das Signalwort soll die Aufmerksamkeit auf den Sicherheitshinweis lenken und bezeichnet die Schwere der Gefährdung.

Die Signalgrafik (Warndreieck mit Ausrufezeichen), welche den Signalwörtern Gefahr, Warnung und Vorsicht vorangestellt wird, weist auf Gefährdungen für Personen hin.

⚠ GEFAHR

Bei Nichtbeachtung dieses Sicherheitshinweises **werden** Tod oder schwere Körperverletzung eintreten.

⚠ WARNUNG

Bei Nichtbeachtung dieses Sicherheitshinweises **können** Tod oder schwere Körperverletzung eintreten.

⚠ VORSICHT

Bei Nichtbeachtung dieses Sicherheitshinweises können mittelschwere oder leichte Körperverletzung eintreten.

HINWEIS

Bei Nichtbeachtung dieses Sicherheitshinweises können Sachschäden eintreten.

1.3.3 Verwendete Symbole

Fingerzeige werden wie folgt dargestellt:



Dies ist ein Hinweis.

Tipps werden wie folgt dargestellt:



Dies ist ein Tipp.

1.3.4 Erläuterung der Signalgrafik auf dem Gerät



Wenn dieses Symbol am Gerät angebracht ist, beachten Sie unbedingt die Dokumentation zu dem Gerät. In der jeweiligen Dokumentation finden Sie die Art der Gefährdung sowie die notwendigen Schritte zur Vermeidung der Gefährdung.

1.4 Erforderliche und ergänzende Dokumentationen

SPS	MTX	MLC
IndraWorks 15VRS IndraLogic 2G SPS-Programmiersystem DOK-IWORKS-IL2GPRO*V15-APRS-DE-P, R911396134 Diese Dokumentation beschreibt das SPS-Programmierzwerkzeug IndraLogic 2G und seine Verwendung. Beschrieben werden die Basisverwendung, Erste Schritte, die Visualisierung, Menüpunkte und die Editoren.	x	x
Feldbusse	MTX	MLC
MLC 15VRS PLCopen-Bibliotheken DOK-XLCMLC-FUNLIB**V15-LIRS-DE-P, R911393306 Diese Dokumentation beschreibt die Funktionsbausteine, Funktionen und Datentypen der Bibliotheken RIL_CommonTypes, ML_Base und ML_PLCopen für das System MLC. Die Dokumentation enthält ferner die Fehlerreaktion der Bausteine.		x
IndraWorks 15VRS Feldbusse DOK-IWORKS-FB*****V15-APRS-DE-P, R911393282 Die Dokumentation dient der Beschreibung der Feldbus- und lokalen Peripherieanschlüssen, die von den Systemen MLC und MTX unterstützt werden. Der Schwerpunkt dieser Dokumentation liegt in der Konfiguration, Parametrierung, Inbetriebnahme und Diagnose der verschiedenen Peripherieanschlüssen. Sie ist die Grundlage für die Online-Hilfe.	x	x
IndraWorks 15VRS Feldbus-Bibliotheken DOK-IWORKS-FB*LIB**V15-LIRS-DE-P, R911393274 Das vorliegende Handbuch beschreibt die Bibliotheken der Feldbusse: RIL_ProfinetDP_02, RIL_ProfinetDPSlave, RIL_ProfinetIO, RIL_ProfinetIODevice, RIL_EtherNetIPAdapter, RIL_MappingList, RIL_SERCOSIII, RIL_Inline, einschließlich deren Diagnosen, sowie die Fehlerreaktion der Bausteine.	x	x

2 Standardschnittstellen an Funktionsbausteinen

2.1 Motivation

Die Mehrzahl der Funktionsbausteine hat einen Eingang zur Aktivierung und einen Ausgang, der die fehlerfreie Bearbeitung anzeigt.

Zusätzlich wird oft ein Ausgang benötigt, der den Bearbeitungszeitraum kennzeichnet.

Zum Anzeigen von Fehlern werden weitere Ausgänge definiert.

Eine einheitliche Namensgebung sowie gleiches Verhalten dieser Standardschnittstellen erhöht das Verständnis, verkürzt die Einarbeitung und entlastet den Support.

2.2 Bausteintypen

Mit Funktionsbausteinen ist es möglich, komplexe Aufgaben wiederverwendbar zu kapseln und diese Aufgaben über definierte Schnittstellen anzusprechen. Dabei kann die Bearbeitung entweder zustands- oder flankengesteuert erfolgen. Zudem unterscheidet man zwischen Aufträgen, die abschließend behandelt werden können und solchen die einmal angestoßen einen permanenten Eingriff erfordern.

Zustandsgesteuert Wiederholt ein Funktionsbaustein nach dem Einschalten seinen Auftrag immer dann, wenn der Funktionsbaustein einen definierten Zustand erreicht hat, arbeitet der Funktionsbaustein "zustandsgesteuert".

Flankengesteuert Erfüllt ein Funktionsbaustein nach dem Einschalten seine Aufgabe genau einmal, so arbeitet der Funktionsbaustein "flankengesteuert".

Abschließende Bearbeitung Kann ein Baustein seinen Auftrag endgültig bearbeiten und seine Arbeit anschließend einstellen, so ist dies eine "abschließende Bearbeitung".

Permanente Bearbeitung Kann ein Baustein seinen Auftrag nie endgültig abschließen, sondern ist dauerhaft im Eingriff, wird dies als "permanente Bearbeitung" definiert.

In Kombination ergeben sich somit vier mögliche Bausteintypen:

Steuerung	Bearbeitung	Beispiel
Zustandsgesteuert (Enable)	Abschließend (Done)	Dauerhaftes Lesen eines Parameters Ist das Lesen abgeschlossen, hat der Baustein seinen Auftrag beendet. Der Baustein wiederholt diesen Auftrag solange, wie der Steuereingang (Enable) gesetzt bleibt, z. B. MB_ReadParameter
Zustandsgesteuert (Enable)	Permanent (In....)	Beendbare Regelung Ist die Regelung aktiv, arbeitet die Regelung solange, bis der Steuereingang (Enable) zurückgesetzt wird und die Regelung damit deaktiviert
Flankengesteuert (Execute)	Abschließend (Done)	Einmaliges Schreiben eines Parameters Ist das Schreiben abgeschlossen, hat der Baustein seinen Auftrag beendet. Es ist eine erneute Flanke am Steuereingang (Execute) notwendig, um ein erneutes Schreiben zu starten, z. B. MB_WriteParameter
Flankengesteuert (Execute)	Permanent (In....)	Nicht beendbare Regelung Ist die Regelung einmal über den Steuereingang (Execute) gestartet, kann die Regelung nicht mehr über die Schnittstellen dieses Bausteines deaktiviert werden, z. B. MC_MoveVelocity

Tab. 2-1: Übersicht Bausteintypen

2.3 Bausteintypen und ihre Schnittstellen

Eingänge Um nach außen kenntlich zu machen, ob ein Funktionsbaustein zustands- oder flankengesteuert arbeitet, werden zwei verschiedene Variablennamen für die Eingänge zur Aktivierung des Funktionsbaustein verwendet.

- Enable = zustandsgesteuert
- Execute = flankengesteuert

Ausgänge Um nach außen kenntlich zu machen, ob ein Funktionsbaustein abschließend oder permanent arbeitet, werden verschiedene Variablennamen für die Ausgänge verwendet, welche den Bearbeitungszustand kennzeichnen.



- Done = Abschließende Bearbeitung
- In..... = Permanente Bearbeitung (z.B. InOperation, InSync, InVelocity, InGear, InTorque...)

Der optionale Ausgang Active kann bei abschließend und permanent arbeitenden Bausteinen verwendet werden, hat aber jeweils ein anderes Verhalten!

Alle Ausgänge zur Fehlerkennzeichnung (Error, ErrorID, ErrorIdent) finden sich bei abschließend und permanent arbeitenden Bausteinen und haben bei beiden das gleiche Verhalten.

Standardisierte Bezeichner

Bezeichner	Beschreibung
Eingänge	
Enable	Aktivierungseingang zustandsgesteuerter FBs
Execute	Aktivierungseingang flankengesteuerter FBs. Andere FBs können diesen FB unterbrechen
ExecuteLock	Aktivierungseingang flankengesteuerter FBs. Solange dieser Eingang TRUE ist, kann der FB nicht durch andere unterbrochen werden
Ausgänge	
Active	Ausgang, der den Bearbeitungszeitraum kennzeichnet
Done	Bearbeitung erfolgreich beendet und die Datenausgänge sind gültig
In.....	Ausgang signalisiert, dass der Baustein an seiner Aufgabe arbeitet und die Datenausgänge gültig sind
Shutdown (optional)	Ausgang signalisiert, dass sich der Baustein gerade in einen definierten Endzustand bringt (z. B. Achsen stoppt, Ressourcen freigibt, ...). Der Baustein muss solange aufgerufen werden, bis "Shutdown" FALSE wird
CommandAborted	Ausgang zeigt an, dass der FB unterbrochen wurde (z. B. von einem anderen FB)
Error	Bearbeitung mit Fehler beendet
ErrorID	Ausgang für grobe Fehlerklassifikation
ErrorIdent	Ausgang für detaillierte Fehlerklassifikation

Tab. 2-2: Übersicht standardisierter Bezeichner

2.4 Ein- und Ausgänge zustandsgesteuerter Funktionsbausteine, abschließende Bearbeitung

Steuerung / Bearbeitung	I/O	Variablenname	Beschreibung
Zustandsge- steuert / ab- schließend	I	Enable	Mit einer positiven Flanke von "Enable" werden die Eingangsvariablen erfasst. Neue Eingangswerte werden erst über eine erneute positive Flanke von "Enable" wirksam. "Enable" muss während der gesamten Bausteinbearbeitung TRUE sein! Das Deaktivieren von "Enable" unterbricht die Bearbeitung und setzt die Ausgänge "Done", "Active", "CommandAborted" und "Error" auf FALSE. Falls notwendig wird über "Shutdown" signalisiert, dass weitere Aufrufe notwendig sind, um den Baustein in einen definierten Endzustand zu bringen
	O	Done	Ist "Done" TRUE, hat der Funktionsbaustein seinen Auftrag erfolgreich beendet und ist in einem Endzustand. Datenausgänge sind jetzt gültig. "Active", "Error" und "CommandAborted" sind FALSE! "Done" bleibt für genau einen SPS-Zyklus TRUE. Ist anschließend "Enable" noch TRUE, startet der Baustein seine Bearbeitung erneut
	O	Active	Ist "Active" TRUE, arbeitet der Funktionsbaustein an seiner eigentlichen Aufgabe und ist in einem Zwischenzustand. Eventuelle Vorverarbeitungen werden nicht mit diesem Ausgang gekennzeichnet! "Done", "Error" und "CommandAborted" sind FALSE!
	O	Shutdown (optional)	Ist ein "Shutdown"-Ausgang vorhanden, benötigt der Baustein beim Deaktivieren von "Enable" noch mehrere Zyklen an Rechenzeit, um sich in einen definierten Endzustand zu bringen und verwendete Ressourcen freizugeben. Der Baustein muss solange aufgerufen werden, bis "Shutdown" FALSE wird!
	O	CommandAborted	Ist "CommandAborted" TRUE, wurde der Funktionsbaustein unterbrochen und befindet sich in einem Endzustand. "Done", "Active" und "Error" sind FALSE! "CommandAborted" bleibt solange TRUE, bis der Steuereingang "Enable" deaktiviert wird. Ein zusätzlicher Reset-Eingang ist daher nicht notwendig. Ist ein "Shutdown"-Ausgang vorhanden, wird dieser Ausgang gleichzeitig mit "CommandAborted" TRUE. Der Baustein muss dann solange weiterhin aufgerufen werden, bis "Shutdown" FALSE wird
	O	Error	Ist "Error" TRUE, wurde der Funktionsbaustein wegen eines Fehlers abgebrochen und befindet sich in einem Endzustand. "Error" bleibt solange TRUE, bis der Steuereingang "Enable" deaktiviert wird. Ein zusätzlicher Reset-Eingang ist daher nicht notwendig. Ist ein "Shutdown"-Ausgang vorhanden, wird dieser Ausgang gleichzeitig mit "Error" TRUE. Der Baustein muss dann solange weiterhin aufgerufen werden, bis "Shutdown" FALSE wird

Tab. 2-3: I/O zustandsgesteuerter Funktionsbausteine, abschließende Bearbeitung



Ist es notwendig, dass bestimmte Eingänge nicht nur mit der Flanke an "Enable" übernommen werden, sondern zyklisch während der Bausteinbearbeitung, so ist dies explizit zu dokumentieren und nicht über Variablenamen kenntlich zu machen!



Bei Bausteinen mit "Shutdown" muss genau definiert werden, wie sich der Baustein verhält, wenn "Enable" wieder auf TRUE gesetzt wird, solange "Shutdown" noch gesetzt ist (Retriggern).

Standardschnittstellen an Funktionsbausteinen

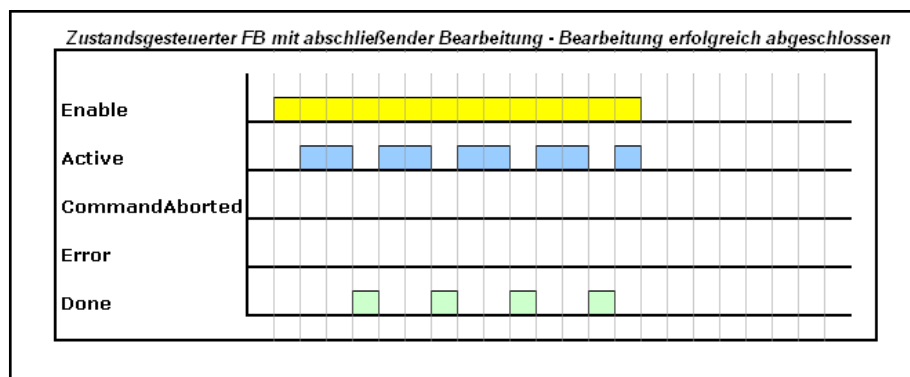


Abb. 2-1: I/O zustandsgesteuerter Funktionsbausteine, abschließende Bearbeitung ohne "Shutdown". Bearbeitung erfolgreich abgeschlossen

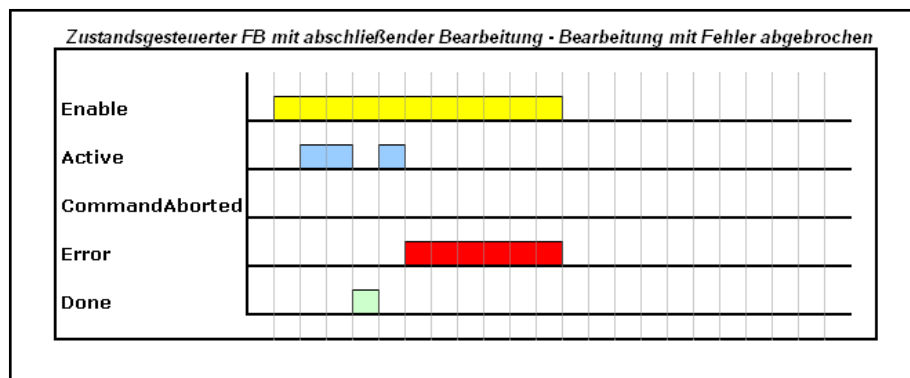


Abb. 2-2: I/O zustandsgesteuerter Funktionsbausteine, abschließende Bearbeitung ohne "Shutdown". Bearbeitung mit Fehler abgeschlossen

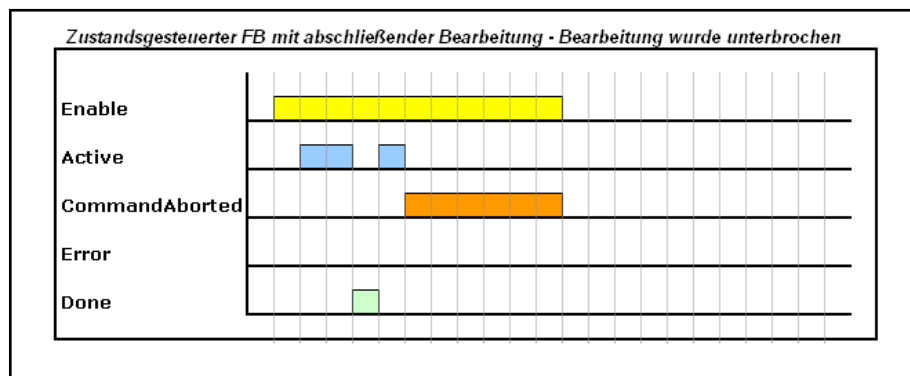


Abb. 2-3: I/O zustandsgesteuerter Funktionsbausteine, abschließende Bearbeitung ohne "Shutdown". Bearbeitung unterbrochen

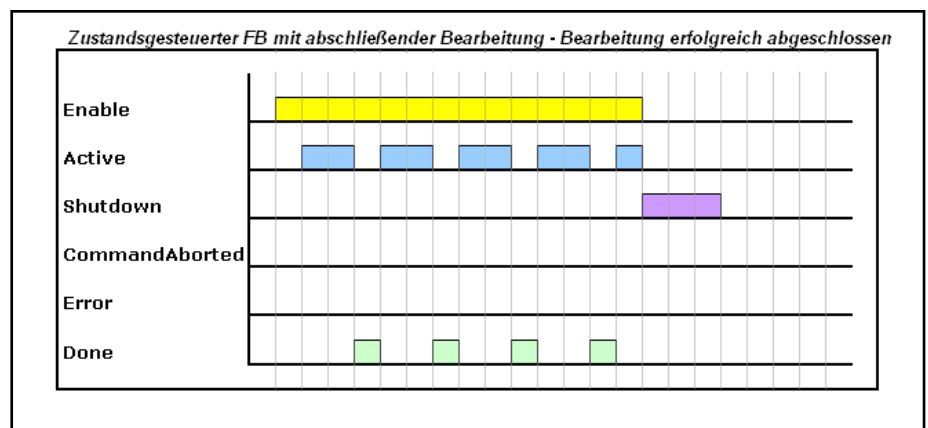


Abb. 2-4: I/O zustandsgesteuerter Funktionsbausteine, abschließende Bearbeitung mit "Shutdown". Bearbeitung erfolgreich abgeschlossen

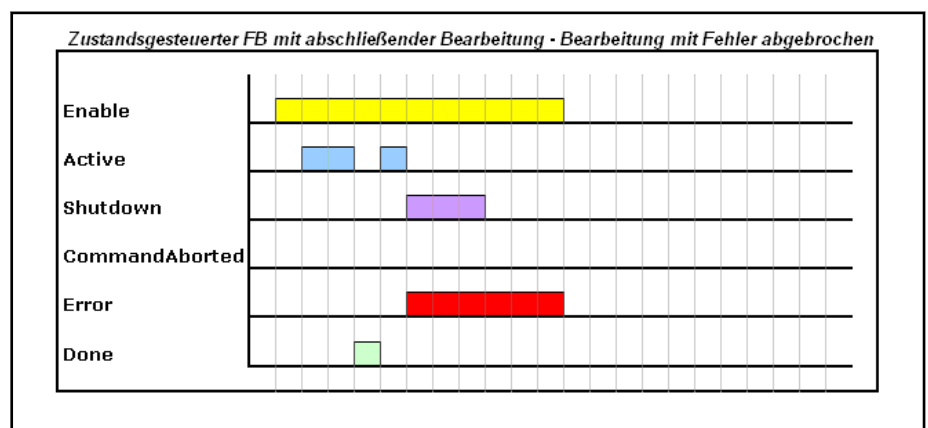


Abb. 2-5: I/O zustandsgesteuerter Funktionsbausteine, abschließende Bearbeitung mit "Shutdown". Bearbeitung mit Fehler abgeschlossen

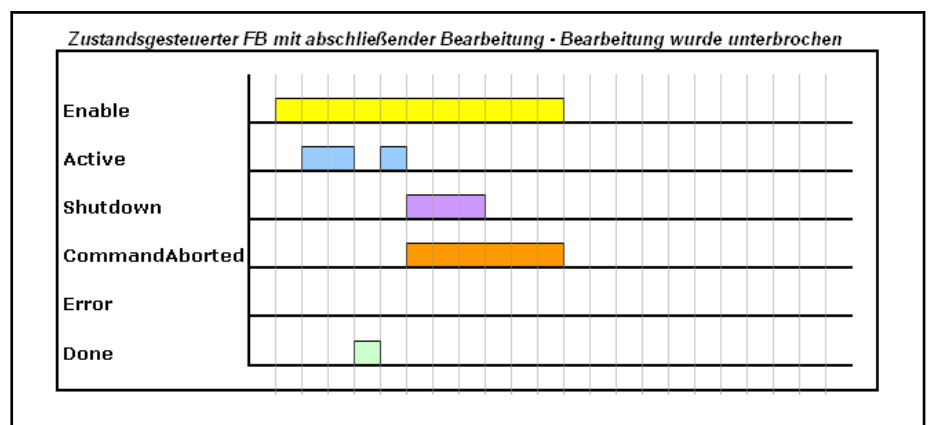


Abb. 2-6: I/O zustandsgesteuerter Funktionsbausteine, abschließende Bearbeitung mit "Shutdown". Bearbeitung wurde unterbrochen

2.5 Ein- und Ausgänge flankengesteuerter Funktionsbausteine, abschließende Bearbeitung

Steuerung / Bearbeitung	I/O	Name	Beschreibung
Flankenge- steuert / ab- schließend	I	Execute	Mit einer positiven Flanke von "Execute" werden die Eingangsvariablen erfasst. Neue Eingangswerte werden erst über eine erneute positive Flanke von "Execute" wirksam. Der Flankenwechsel an "Execute" reicht aus, um den Baustein zu starten. Zur weiteren Bearbeitung ist der Zustand von "Execute" nicht relevant. Durch einen erneuten Flankenwechsel noch während der Bearbeitung wird der alte Auftrag verworfen, die Eingänge erneut übernommen und mit den neuen Werten der Auftrag fortgesetzt (Retriggerung, Nachtriggern). Ist ein "Shutdown"-Ausgang vorhanden, wird Retriggering möglicherweise nicht unterstützt. Die Bausteindokumentation schafft hierüber Klarheit
	O	Done	Ist "Done" TRUE, hat der Funktionsbaustein seinen Auftrag erfolgreich beendet und ist in einem Endzustand. Datenausgänge sind jetzt gültig. "Active", "Error" und "CommandAborted" sind FALSE! Ist bei Auftragsabschluss der Steuerungseingang "Execute" FALSE, bleibt "Done" für genau einen SPS-Zyklus TRUE. Ist "Execute" TRUE, bleibt "Done" solange TRUE bis "Execute" deaktiviert wird. Ist ein "Shutdown"-Ausgang vorhanden wird dieser Ausgang gleichzeitig mit "Done" TRUE. Der Baustein muss dann solange weiterhin aufgerufen werden, bis "Shutdown" FALSE wird
	O	Active	Ist "Active" TRUE, arbeitet der Funktionsbaustein an seiner eigentlichen Aufgabe und ist in einem Zwischenzustand. Eventuelle Vorverarbeitungen werden nicht mit diesem Ausgang gekennzeichnet! "Done", "Error", "Shutdown" und "CommandAborted" sind FALSE!
	O	Shutdown (optional)	Ist ein "Shutdown"-Ausgang vorhanden, benötigt der Baustein nach Beendigung der Aufgabe (signalisiert durch "Done") noch mehrere Zyklen an Rechenzeit, um sich in einen definierten Endzustand zu bringen und verwendete Ressourcen freizugeben. Der Baustein muss solange aufgerufen werden, bis "Shutdown" FALSE wird!
	O	CommandAborted	Ist "CommandAborted" TRUE, wurde der Funktionsbaustein unterbrochen und befindet sich in einem Endzustand. "Done", "Active" und "Error" sind FALSE! Ist zum Unterbrechungszeitpunkt der Steuerungseingang "Execute" FALSE, bleibt "CommandAborted" für genau einen SPS-Zyklus TRUE. Ist "Execute" TRUE, bleibt "CommandAborted" solange TRUE bis "Execute" deaktiviert wird. Ist ein "Shutdown"-Ausgang vorhanden, wird dieser Ausgang gleichzeitig mit "CommandAborted" TRUE. Der Baustein muss dann solange weiterhin aufgerufen werden, bis "Shutdown" FALSE wird
	O	Error	Ist "Error" TRUE, wurde der Funktionsbaustein wegen eines Fehlers abgebrochen und befindet sich in einem Endzustand. Ist bei Auftreten des Fehlers der Steuerungseingang "Execute" FALSE, bleibt "Error" für genau einen SPS-Zyklus TRUE. Ist "Execute" TRUE, bleibt "Error" solange TRUE bis "Execute" deaktiviert wird. Ist ein "Shutdown"-Ausgang vorhanden, wird dieser Ausgang gleichzeitig mit "Error" TRUE. Der Baustein muss dann solange weiterhin aufgerufen werden, bis "Shutdown" FALSE wird

Tab. 2-4: I/O zustandsgesteuerter Funktionsbausteine, abschließende Bearbeitung



Werden bestimmte Eingänge nicht nur mit der 0/1-Flanke an "Execute" übernommen, sondern zyklisch während der Bausteinbearbeitung, so ist dies explizit zu dokumentieren und nicht über Variablennamen kenntlich zu machen!



Bei Bausteinen mit "Shutdown" muss genau definiert werden, wie sich der Baustein verhält, wenn an "Execute" eine erneute Flanke erkannt wird, solange "Shutdown" noch gesetzt ist (Retriggern).

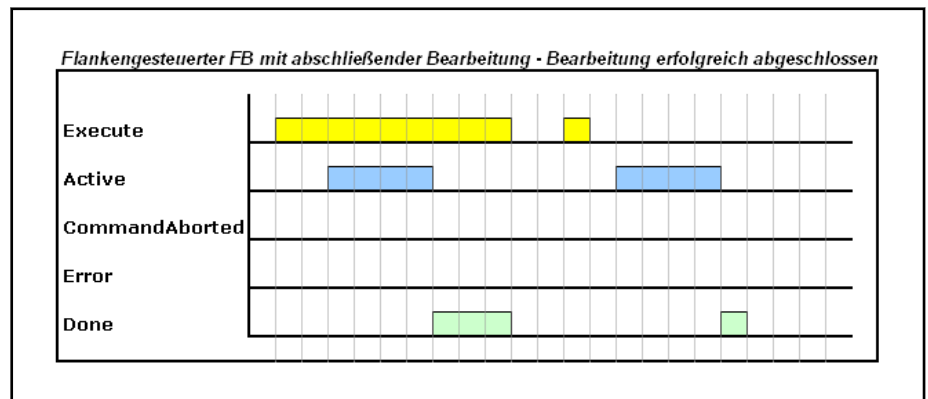


Abb. 2-7: I/O flankengesteuerter Funktionsbausteine, abschließende Bearbeitung ohne "Shutdown". Bearbeitung erfolgreich abgeschlossen

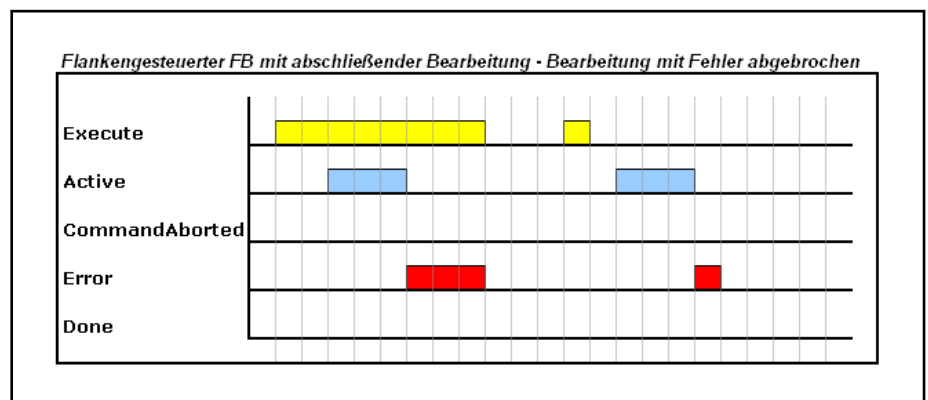


Abb. 2-8: I/O flankengesteuerter Funktionsbausteine, abschließende Bearbeitung ohne "Shutdown". Bearbeitung mit Fehler abgeschlossen

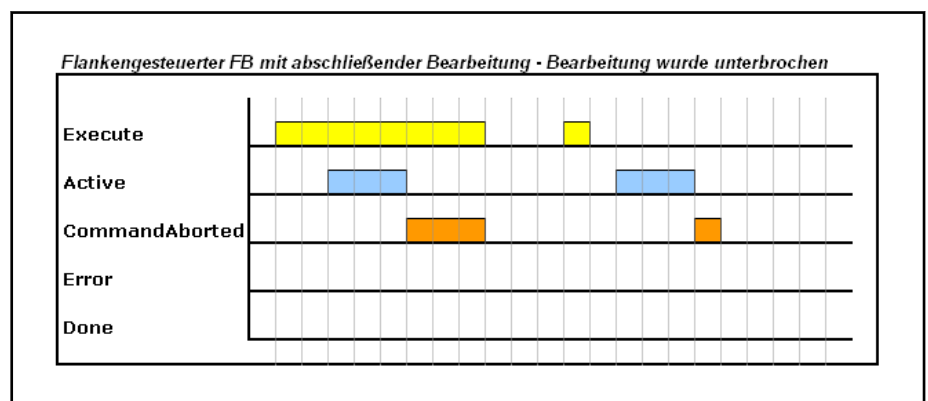


Abb. 2-9: I/O flankengesteuerter Funktionsbausteine, abschließende Bearbeitung ohne "Shutdown". Bearbeitung unterbrochen

Standardschnittstellen an Funktionsbausteinen

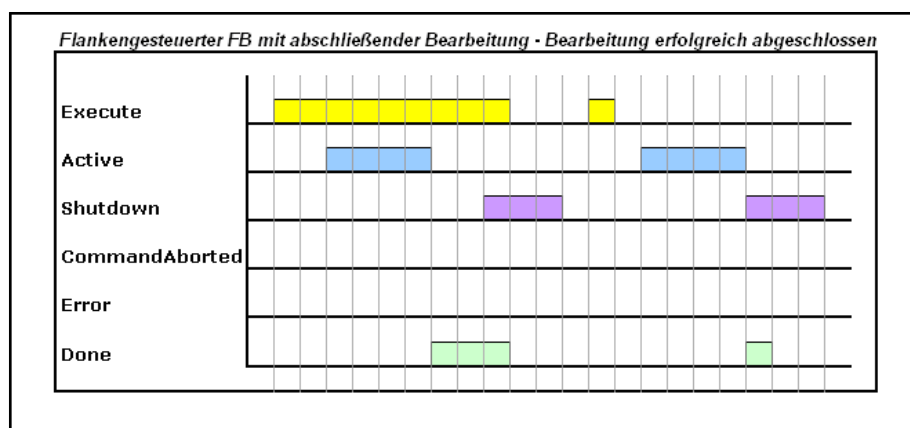


Abb. 2-10: I/O flankengesteuerter Funktionsbausteine, abschließende Bearbeitung mit "Shutdown". Bearbeitung erfolgreich abgeschlossen

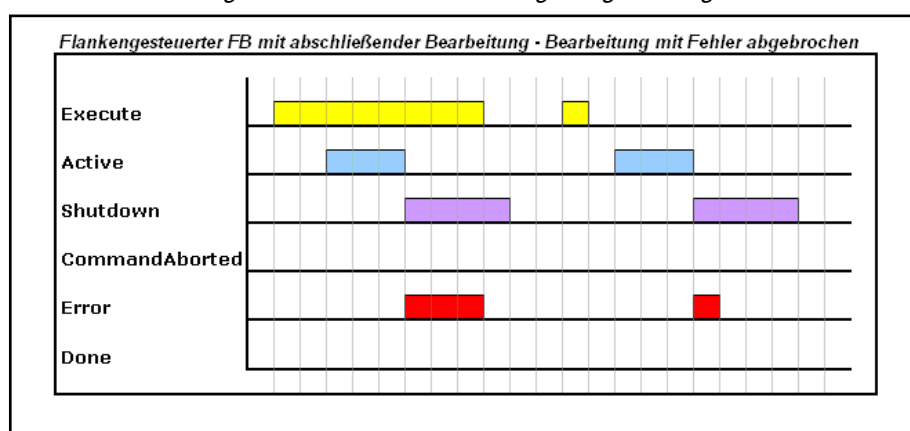


Abb. 2-11: I/O flankengesteuerter Funktionsbausteine, abschließende Bearbeitung mit "Shutdown". Bearbeitung mit Fehler abgeschlossen

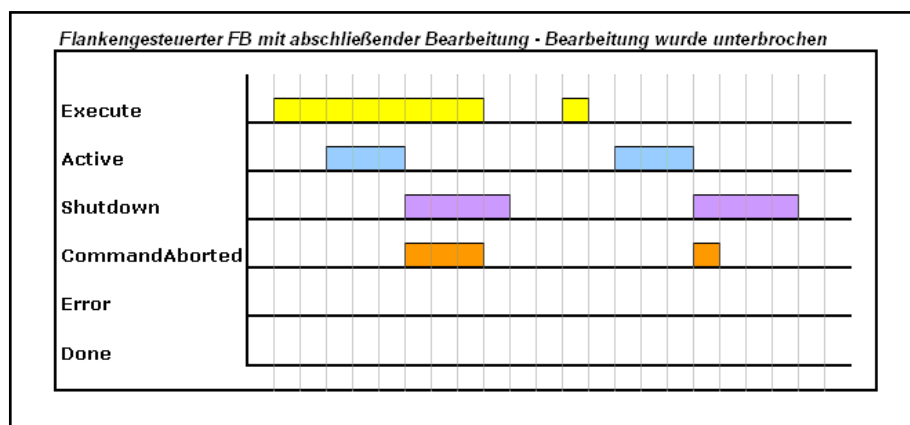


Abb. 2-12: I/O flankengesteuerter Funktionsbausteine, abschließende Bearbeitung mit "Shutdown". Bearbeitung unterbrochen

2.6 Ein- und Ausgänge zustandsgesteuerter Funktionsbausteine, permanente Bearbeitung

Steuerung / Bearbeitung	I/O	Variablenname	Beschreibung
Zustandsge- steuert / per- manent	I	Enable	Mit einer positiven Flanke von "Enable" werden die Eingangsvariablen erfasst. Neue Eingangswerte werden erst über eine erneute positive Flanke von "Enable" wirksam. "Enable" muss während der gesamten Bausteinbearbeitung TRUE sein! Das Deaktivieren von "Enable" unterbricht die Bearbeitung und setzt die Ausgänge "In....", "Active", "CommandAborted" und "Error" auf FALSE. Falls notwendig wird über "Shutdown" signalisiert, dass weitere Aufrufe notwendig sind, um den Baustein in einen definierten Endzustand zu bringen
	O	In....	Ist "In...." TRUE, hat der Funktionsbaustein sein Ziel erreicht, befindet sich aber weiter im Eingriff, um das Erreichte zu "bewahren" und befindet sich damit in einem dauerhaften Endzustand. Datenausgänge sind jetzt gültig. Da der FB weiter im Eingriff bleibt, muss auch "Active" TRUE bleiben. "Error" und "CommandAborted" sind FALSE! Solange "Enable" TRUE ist, bleibt "In...." ebenfalls TRUE
	O	Active	Ist Active TRUE, arbeitet der Funktionsbaustein an seiner eigentlichen Aufgabe. Eventuelle Vorverarbeitungen werden nicht mit diesem Ausgang gekennzeichnet! Da der FB permanent im Eingriff bleibt, bleibt "Active" TRUE bis der FB über den Steuereingang "Enable" abgeschaltet wird oder durch "Error" oder "CommandAborted" beendet wird. Solange "Active" TRUE ist, müssen "Error" oder "CommandAborted" FALSE sein!
	O	Shutdown (optional)	Ist ein "Shutdown"-Ausgang vorhanden, benötigt der Baustein beim Deaktivieren von "Enable" noch mehrere Zyklen an Rechenzeit, um sich in einen definierten Endzustand zu bringen und verwendete Ressourcen freizugeben. Der Baustein muss solange aufgerufen werden, bis "Shutdown" FALSE wird!
	O	CommandAborted	Ist "CommandAborted" TRUE, wurde der Funktionsbaustein unterbrochen und befindet sich in einem Endzustand. "In....", "Active" und "Error" sind FALSE! "CommandAborted" bleibt solange TRUE, bis der Steuereingang "Enable" deaktiviert wird. Ein zusätzlicher Reset-Eingang ist daher nicht notwendig. Ist ein "Shutdown"-Ausgang vorhanden, wird dieser Ausgang gleichzeitig mit "CommandAborted" TRUE. Der Baustein muss dann solange weiterhin aufgerufen werden, bis "Shutdown" FALSE wird
	O	Error	Ist "Error" TRUE, wurde der Funktionsbausteins wegen eines Fehlers abgebrochen und befindet sich in einem Endzustand. "Error" bleibt solange TRUE bis der Steuereingang "Enable" deaktiviert wird. Ein zusätzlicher Reset-Eingang ist daher nicht notwendig. Ist ein "Shutdown"-Ausgang vorhanden, wird dieser Ausgang gleichzeitig mit "Error" TRUE. Der Baustein muss dann solange weiterhin aufgerufen werden, bis "Shutdown" FALSE wird

Tab. 2-5: I/O zustandsgesteuerter Funktionsbausteine, abschließende Bearbeitung



Ist es notwendig, dass bestimmte Eingänge nicht nur mit der Flanke an "Enable" übernommen werden, sondern zyklisch während der Bausteinbearbeitung, so ist dies explizit zu dokumentieren und nicht über Variablennamen kenntlich zu machen!



Bei Bausteinen mit "Shutdown" muss genau definiert werden, wie sich der Baustein verhält, wenn "Enable" wieder TRUE gesetzt wird, solange "Shutdown" noch gesetzt ist (Retriggern).

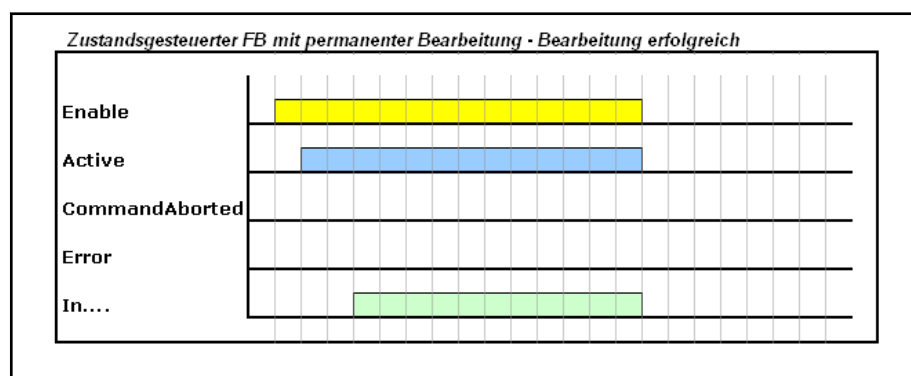


Abb. 2-13: I/O zustandsgesteuerter Funktionsbausteine, permanente Bearbeitung ohne "Shutdown". Bearbeitung erfolgreich

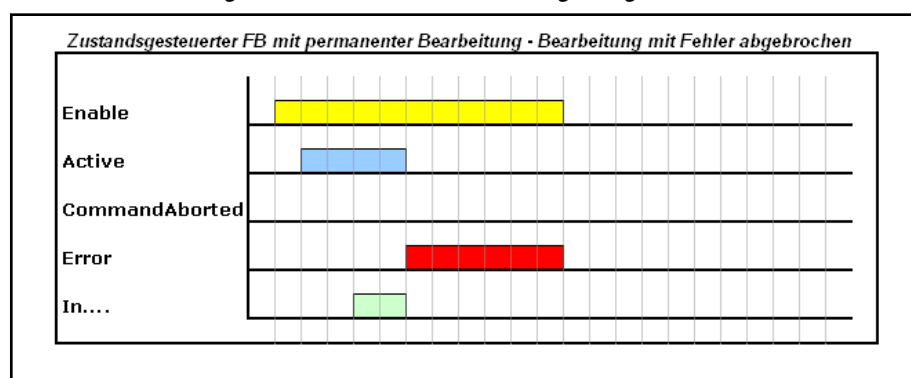


Abb. 2-14: I/O zustandsgesteuerter Funktionsbausteine, permanente Bearbeitung ohne "Shutdown". Bearbeitung mit Fehler abgebrochen

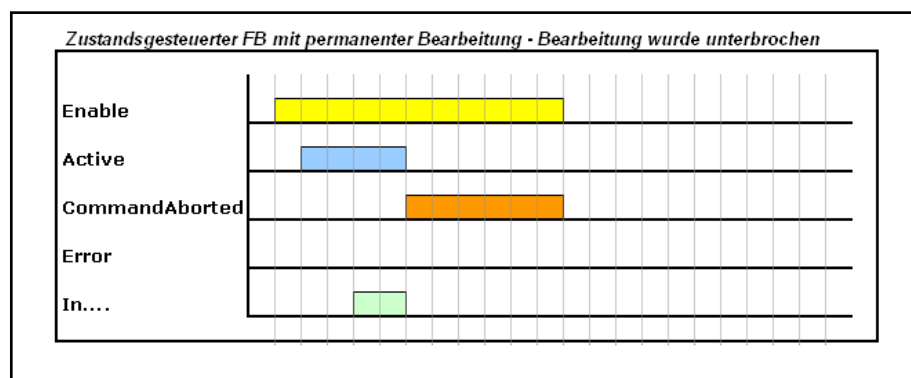


Abb. 2-15: I/O zustandsgesteuerter Funktionsbausteine, permanente Bearbeitung ohne "Shutdown". Bearbeitung unterbrochen

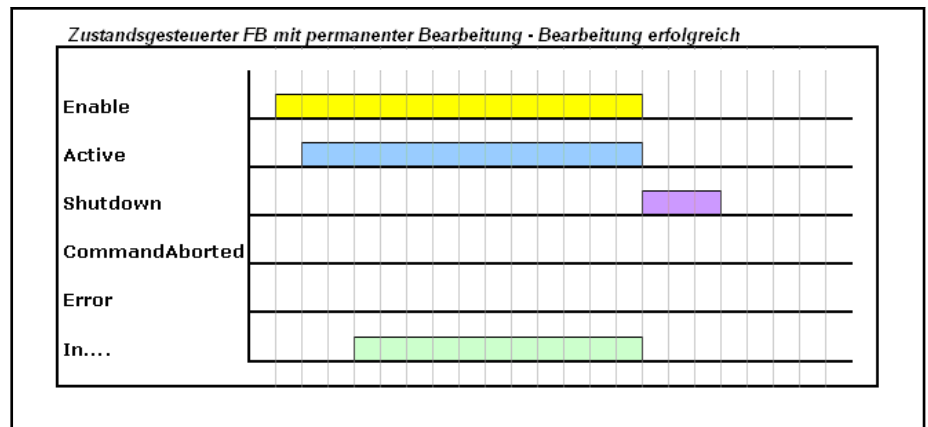


Abb. 2-16: I/O zustandsgesteuerter Funktionsbausteine, permanente Bearbeitung mit "Shutdown". Bearbeitung erfolgreich

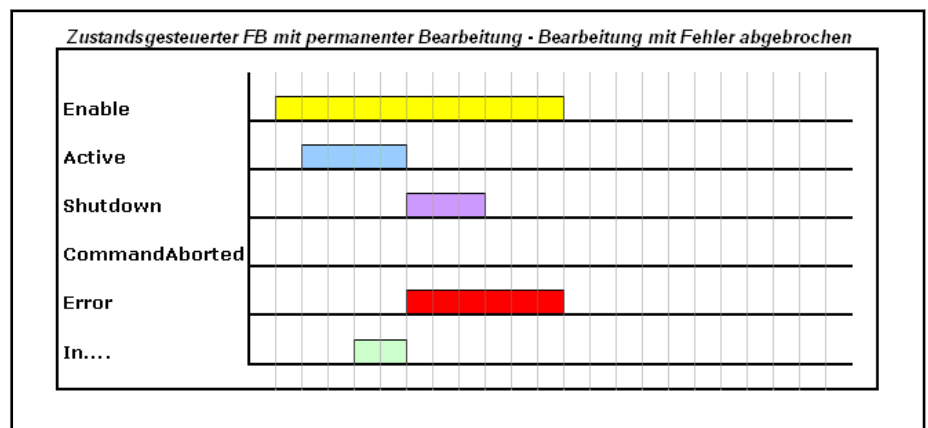


Abb. 2-17: I/O zustandsgesteuerter Funktionsbausteine, permanente Bearbeitung mit "Shutdown". Bearbeitung mit Fehler abgebrochen

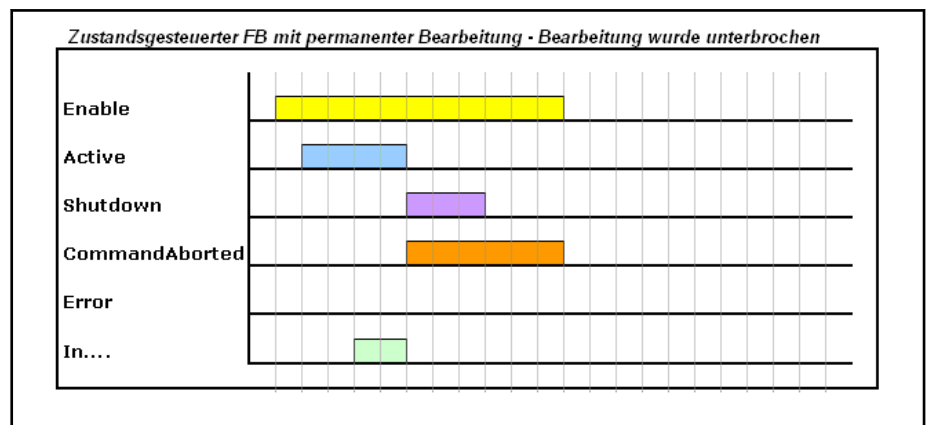


Abb. 2-18: I/O zustandsgesteuerter Funktionsbausteine, permanente Bearbeitung mit "Shutdown". Bearbeitung unterbrochen

2.7 Ein- und Ausgänge flankengesteuerter Funktionsbausteine, permanente Bearbeitung

Steuerung / Bearbeitung	I/O	Variablenname	Beschreibung
Flankengesteuert / permanent	i	Execute	Mit einer positiven Flanke von "Execute" werden die Eingangsvariablen erfasst. Neue Eingangswerte werden erst über eine erneute positive Flanke von "Execute" wirksam. Der Flankenwechsel an "Execute" reicht aus, um den Baustein zu starten. Zur weiteren Bearbeitung ist der Zustand von "Execute" nicht relevant. Durch einen erneuten Flankenwechsel noch während der Bearbeitung werden der alte Auftrag verworfen, die Eingänge erneut übernommen und mit den neuen Werten der Auftrag fortgesetzt (Retriggerung, Nachtriggern). Hat der Baustein einen "Shutdown"-Ausgang, ist dieses Retriggern eventuell nicht erlaubt
	O	In....	Ist "In...." TRUE, hat der Funktionsbaustein sein Ziel erreicht, befindet sich aber weiter im Eingriff, um das Erreichte zu "bewahren" und ist damit in einem dauerhaften Endzustand. Datenausgänge sind jetzt gültig. Da der FB weiter im Eingriff bleibt, muss auch "Active" TRUE bleiben. Das gilt auch dann, wenn "In...." wieder abfällt. "Error" und "CommandAborted" sind FALSE! Ist bei Auftragsabschluss der Steuerungseingang "Execute" FALSE, bleibt "In...." für genau einen SPS-Zyklus TRUE. Ist "Execute" TRUE, bleibt "In...." solange TRUE, bis "Execute" deaktiviert wird
	O	Active	Ist "Active" TRUE, arbeitet der Funktionsbaustein an seiner eigentlichen Aufgabe. Eventuelle Vorverarbeitungen werden nicht mit diesem Ausgang gekennzeichnet! Da der FB permanent im Eingriff bleibt, bleibt "Active" TRUE bis der Baustein durch "Error" oder "CommandAborted" beendet wird. Solange "Active" TRUE ist, müssen "Error" oder "CommandAborted" sowie „Shutdown“ FALSE sein!
	O	Shutdown (optional)	Ist ein "Shutdown"-Ausgang vorhanden, benötigt der Baustein, sobald der Baustein seine Aufgabe abbricht ("CommandAborted" oder "Error" = TRUE), noch mehrere Zyklen an Rechenzeit, um sich in einen definierten Endzustand zu bringen und verwendete Ressourcen freizugeben. Der Baustein muss solange aufgerufen werden, bis "Shutdown" FALSE wird!
	O	CommandAborted	Ist "CommandAborted" TRUE, wurde der Funktionsbaustein unterbrochen und befindet sich in einem Endzustand. "In....", "Active" und "Error" sind FALSE! Ist zum Unterbrechungszeitpunkt der Steuerungseingang "Execute" FALSE, bleibt "CommandAborted" für genau einen SPS-Zyklus TRUE. Ist "Execute" TRUE, bleibt "CommandAborted" solange TRUE bis "Execute" deaktiviert wird. Ist ein "Shutdown"-Ausgang vorhanden, wird dieser Ausgang gleichzeitig mit "CommandAborted" TRUE. Der Baustein muss dann solange weiterhin aufgerufen werden, bis "Shutdown" FALSE wird
	O	Error	Ist "Error" TRUE, wurde der Funktionsbaustein wegen eines Fehlers abgebrochen und befindet sich in einem Endzustand. Ist bei Auftreten des Fehlers der Steuerungseingang "Execute" FALSE, bleibt "Error" für genau einen SPS-Zyklus TRUE. Ist "Execute" TRUE, bleibt "Error" solange TRUE bis "Execute" deaktiviert wird. Ist ein "Shutdown"-Ausgang vorhanden, wird dieser Ausgang gleichzeitig mit "Error" TRUE. Der Baustein muss dann solange weiterhin aufgerufen werden, bis "Shutdown" FALSE wird

Tab. 2-6: I/O flankengesteuerter Funktionsbausteine, permanente Bearbeitung



Werden bestimmte Eingänge nicht nur mit der 0/1-Flanke an "Execute" übernommen, sondern zyklisch während der Bausteinbearbeitung, so ist dies explizit zu dokumentieren und nicht über Variablennamen kenntlich zu machen!



Bei Bausteinen mit "Shutdown" muss genau definiert werden, wie sich der Baustein verhält, wenn an "Execute" eine erneute Flanke erkannt wird, solange "Shutdown" noch gesetzt ist (Retriggern).

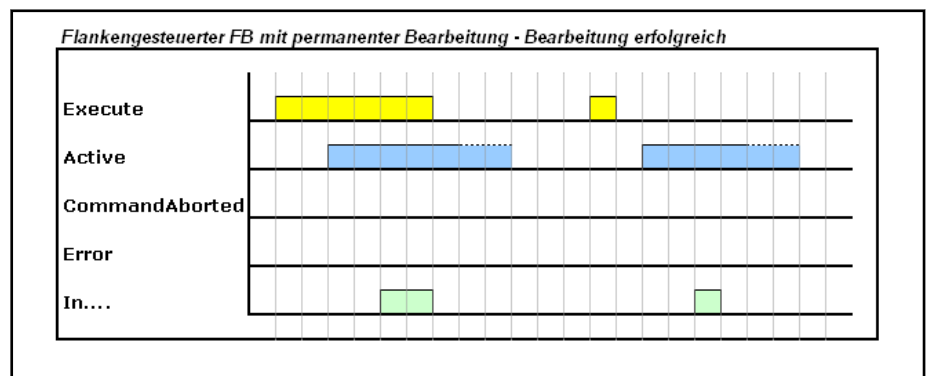


Abb. 2-19: I/O flankengesteuerter Funktionsbausteine, permanente Bearbeitung ohne "Shutdown". Bearbeitung erfolgreich

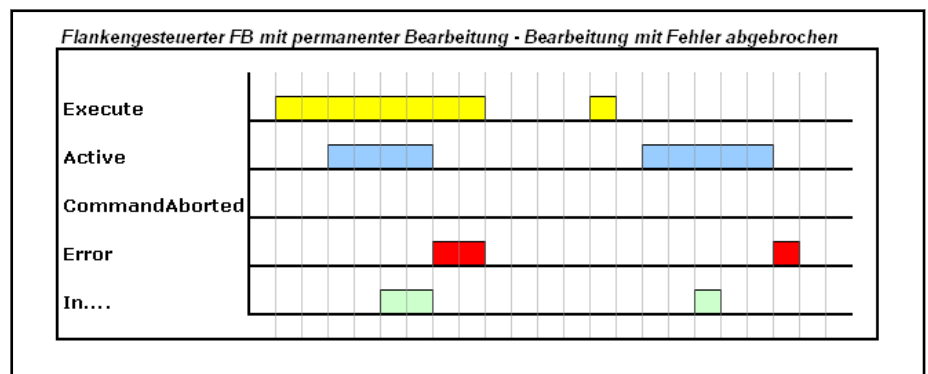


Abb. 2-20: I/O flankengesteuerter Funktionsbausteine, permanente Bearbeitung ohne "Shutdown". Bearbeitung mit Fehler abgebrochen

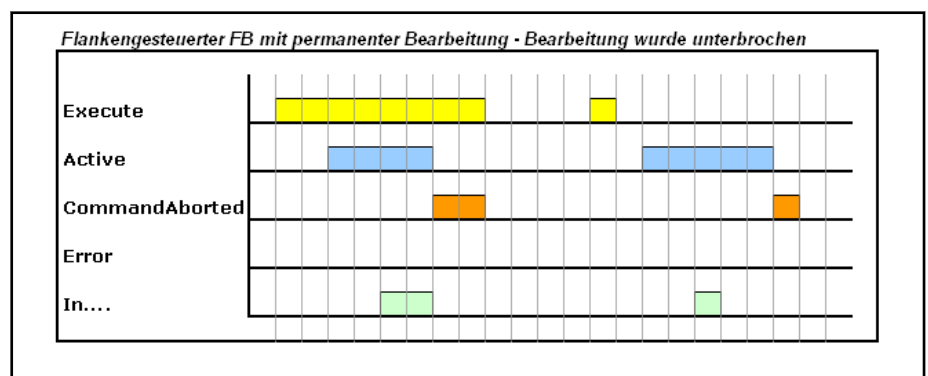


Abb. 2-21: I/O flankengesteuerter Funktionsbausteine, permanente Bearbeitung ohne "Shutdown". Bearbeitung unterbrochen

Standardschnittstellen an Funktionsbausteinen

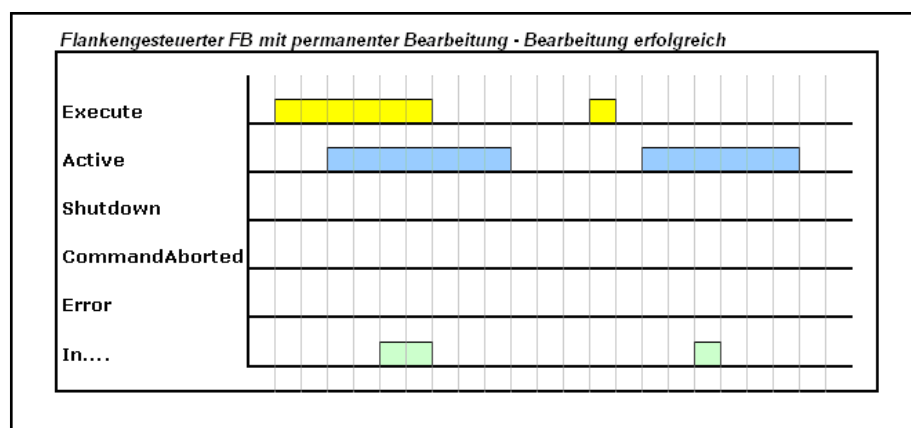


Abb. 2-22: I/O flankengesteuerter Funktionsbausteine, permanente Bearbeitung mit "Shutdown". Bearbeitung erfolgreich

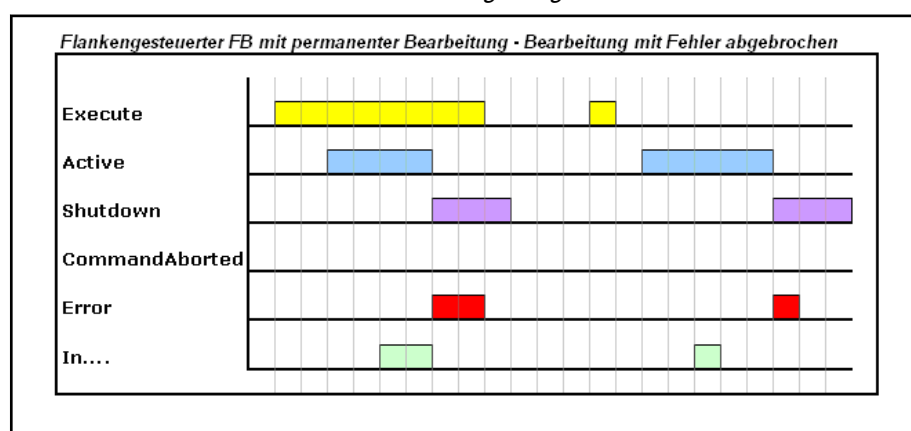


Abb. 2-23: I/O flankengesteuerter Funktionsbausteine, permanente Bearbeitung mit "Shutdown". Bearbeitung mit Fehler abgebrochen

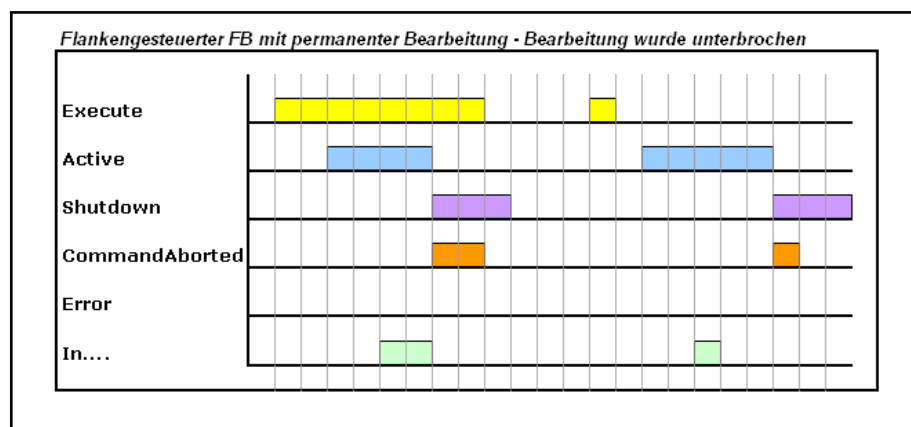


Abb. 2-24: I/O flankengesteuerter Funktionsbausteine, permanente Bearbeitung mit "Shutdown". Bearbeitung unterbrochen

3 RIL_CommonTypes.library

3.1 Einführung und Übersicht



Für die Bibliothek ist das Basis-Funktionspaket PLC des Systems MLC 15VRS ausreichend.

Die Bibliothek RIL_CommonTypes enthält die Datentypen, die Ein- und Ausgangsvariablen der Funktionsbausteine der Firma Bosch Rexroth beschreiben.

Die Bibliothek weist folgende Struktur auf:

Globale_Variablen

Globale_Variablen

- [IL_NULL](#), Seite 20, NIL-Pointer

SercosAttribute

- Die globale Variablenliste enthält die Sercos Attribute, einschließlich eines Kommentars bezüglich der Verwendung.

Datentypen

Datatypes of logical addresses

- [CONTROLS](#), Seite 21, (ENUM)
- [KIN_OBJECTS](#), Seite 24, (ENUM)
- [OBJECTS](#), Seite 22, (ENUM)
- [TP_OBJECTS](#), Seite 26, (ENUM)

Datatypes of PLCopen

- [AXIS_GROUP_REF](#), Seite 23, (STRUCT),
- [AXIS_REF](#), Seite 21, (STRUCT)
- [MC_CAM_ID](#), Seite 26, (ENUM),
- [MC_START_MODE](#), Seite 51, (ENUM)
- [MC_STEP_MODE](#), [MotionProfile](#) erweitert zu [FlexProfile](#), Seite 27, (ENUM)
- [MC_SYNC_DIRECTION](#), Seite 53, (ENUM)
- [TOUCHPROBE_REF](#), Seite 25, (STRUCT)

Datatypes of POU diagnosis

- [ERROR_STRUCT](#), Seite 55, (STRUCT)

Includes

- [ERROR_CODE](#), Seite 54, (ENUM)
- [ERROR_TABLE](#), Seite 56, (ENUM)

Datatypes of times

- [EXT_SYS_TIME_DATE](#), Seite 57, (STRUCT)
- [SYS_TIME64](#), Seite 57, (STRUCT)
- [SYS_TIME_DATE](#), Seite 58, (STRUCT)

TouchProbe

- [MB_DIFF_EVAL](#), Seite 59, (ENUM)
- [MB_EXPECT_WINDOW_MODE](#), Seite 60, (ENUM)
- [MB_INTERPOLATION_MODE](#), Seite 60, (ENUM)
- [MB_POSITION_EVAL](#), Seite 61, (ENUM)
- [MB_PROBE_DATA](#), Seite 62, (ENUM)
- [MB_PROBE_INPUT](#), Seite 61, (ENUM)
- [MB_PROBE_MODULE](#), Seite 61, (ENUM)
- [MB_PROBE_CONFIG](#), Seite 63, (ENUM)

Sie wurde vereinbart, um einen einheitlichen Zugriff auf unterschiedliche Gerätetypen für unterschiedliche Steuerungssysteme zu definieren.

3.2 Globale_Variablen

3.2.1 IL_NULL

Kurzbeschreibung Definiert wurde ein NIL-Pointer als POINTER TO BYTE.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek / Ordner
MLC 15VRS Grundpaket PLC	RIL_CommonTypes / Globale_Variablen

Tab. 3-1: Referenztabelle IL_NULL

Typdefinition

```
VAR_GLOBAL CONSTANT
  IL_NULL: POINTER TO BYTE:= 0;
END_VAR
```

3.3 Datentypen zum Referenzieren von Objekten bei Motion-Funktionsbausteinen

3.3.1 Referenzierung von Objekten, Einführung und Übersicht

Zur Referenzierung von Objekten (Steuerungen, Achsen, Kinematiken/Achsgruppen, Messtastern usw.) werden für jeden Objekttyp eine REF-Struktur mit zugeordneten Aufzählungstypen angeboten.

Die Informationsübergabe erfolgt ab Version 14VRS per **VAR_IN_OUT CONSTANT** an den jeweiligen Funktionsbaustein.

Die REF-Struktur enthält zwei 16-Bit-Adresselemente:

- die Steuerungsadresse (Element von die Steuerungsadresse (Element von [CONTROLS](#), Seite 21) und
- die Objektadresse, Element des Aufzählungstypes.



Die Informationsübergabe erfolgte bis einschließlich Version 13VRS per **VAR_IN_OUT**.

Ab Version 14VRS erfolgt die Informationsübergabe per **VAR_IN_OUT CONSTANT**.

REF-Struktur	Aufzählungstyp
AXIS_REF, Seite 21 , Achsen, z.Z. noch Nockenschaltwerke	OBJECTS, Seite 22
AXIS_GROUP_REF, Seite 23 , Achsgruppen/ Kinematiken	KIN_OBJECTS, Seite 24
TOUCHPROBE_REF, Seite 25 , Messtaster	TP_OBJECTS, Seite 26

Tab. 3-2: REF-Strukturen und ihre Aufzählungstypen



Die für den konkreten Einsatzfall verwendbaren Elemente der Aufzählungstypen sind abhängig von

- System
- Steuerungstyp / Antriebtyp und
- Firmware-Version

3.3.2 CONTROLS

Kurzbeschreibung

Die Aufzählung enthält alle Steuerungen, die im Verbund über Motion-Funktionsbausteine adressiert werden können:



Im Falle einer physikalischen Adressierung ist die "lokale Steuerung" über zwei Möglichkeiten erreichbar, Nummer "0", lokal, und über ihre eigene Adresse z.B. CTRL_3.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek /Ordner
MLC 15VRS Grundpaket PLC	RIL_CommonTypes / Datatypes of logical addresses

Tab. 3-3: Referenztablette CONTROLS

Typdefinition

```
// Enumeration of valid logical control addresses
TYPE CONTROLS :
(
    LOCAL_CNTRL := 0,    // Local control (default)
    CNTRL_1 := 1,        // Control 1
    CNTRL_2 := 2,        // Control 2
    // ...
    CNTRL_99 := 99       // Control 99
);
END_TYPE
```

Funktionsbeschreibung

Diese Aufzählung der angeschlossenen Steuerungen enthält alle Steuerungen, die im Verbund über Motion-Funktionsbausteine adressiert werden können:

- Die Nummer 0 ist freigegeben für die lokale Steuerung (die Steuerung selbst),
- Der Zahlenbereich 1 .. 99 ist reserviert für alle verbundenen Steuerungen.

3.3.3 AXIS_REF

Kurzbeschreibung

Die Struktur enthält Informationen zur gewünschten Steuerung und dem von dieser Steuerung gesteuerten Objekt (Achse, Nockenschaltwerk, Oszilloskop...).

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek / Ordner
MLC 15VRS Grundpaket PLC	RIL_CommonTypes / Datatypes of PLCopen

Tab. 3-4: Referenztablelle AXIS_REF

Typdefinition

```

(* Structure of PLCopen element AXIS_REF. The AXIS_REF is a structure that contains *)
(* information on the corresponding object. *)
(* It is used as a VAR_IN_OUT CONSTANT in all Motion Control Function Blocks. *)
TYPE AXIS_REF :
STRUCT
  CntrlNo: CONTROLS := LOCAL_CNTRL; (* control reference number *)
  AxisNo: OBJECTS := NO_OBJECT; (* Steuerungsnummer, Init: lokale Steuerung *)
  (* object reference number *)
  (* Achsreferenznummer, Init: kein Objekt *)
END_STRUCT
END_TYPE

```

Funktionsbeschreibung

Die Informationsübergabe erfolgt ab Version 14VRS per **VAR_IN_OUT CONSTANT** an den jeweiligen PLCopen-Funktionsbaustein.

Die AXIS_REF-Struktur enthält zwei 16-Bit-Adresselemente:

- die Steuerungsadresse (Element von [CONTROLS, Seite 21](#)) und
- die Objektadresse (Element von [OBJECTS, Seite 22](#))

Die Elemente können per logischer oder physischer Adresse angesprochen werden. Im Falle einer logischen Adresse zeigt das Element auf eine Parameterstruktur mit einer physikalischen Adresse (z.B. Ethernet-Adresse der Steuerung, Sercos-Adresse des Antriebs o.ä.).



Die Informationsübergabe erfolgte bis einschließlich Version 13VRS per **VAR_IN_OUT**.

Ab Version 14VRS erfolgt die Informationsübergabe per **VAR_IN_OUT CONSTANT**.

3.3.4 OBJECTS**Kurzbeschreibung**

Die Aufzählung der verfügbaren Achsen enthält alle Objekte eines Verbundes, auf die per "AXIS_REF" durch Motion-Funktionsbausteine zugegriffen werden kann.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek / Ordner
MLC 15VRS Grundpaket PLC	RIL_CommonTypes / Datatypes of logical addresses

Tab. 3-5: Referenztablelle OBJECTS

Typdefinition

```

// Enumeration of valid logical object addresses
TYPE OBJECTS :
(
  // not specified object
  NO_OBJECT := 0, // no object specified
  // drives 0001..0999
  AXIS_1 := 1, // drive 1
  // ...
  AXIS_99 := 99, // drive 99
  AXIS_100 := 100, // drive 100
  // ...
  AXIS_512 := 512, // drive 512
)

```

```

    RMA_1 := 1001,          (* real master axis 1 *)
(* RMA_2 := 1002,          (* real master axis 2 *)

    VMA_1 := 2001,          (* virtual master axis 1 *)
    VMA_2 := 2002,          (* virtual master axis 2 *)
    VMA_INT := 2003,        (* internal virtual master axis *)
    VMA_EXT := 2004,        (* external virtual master axis *)
    MA_LINK_1 := 2005,      (* link axis 1 *)
    MA_LINK_2 := 2006,      (* link axis 2 *)

// ELS Group Local Master 3001..3999
    GLMA_1 := 3001,         // ELS Group Local Master Axis 1
    // ...
    GLMA_8 := 3008,         // ELS Group Local Master Axis 8

// ELS Groups 4001..4008
    ELSG_1 := 4001,         // ELS Group 1
    // ...
    ELSG_8 := 4008,         // ELS Group 8

// ELS System Masters 5001..5006
    ELSM_1 := 5001,         // ELS System Master 1
    // ...
    ELSM_6 := 5006,         // ELS System Master 6

// FieldBus Drives 6001..6099
    FieldBusAxis_1 := 6001, //Fieldbus Drive 1
    // ...
    FieldBusAxis_99 := 6099, //Fieldbus Drive 99

// Program Limit Switch 8001..8999
    PLS_1 := 8001,          //Program Limit Switch unit 1
    // ...
    PLS_4 := 8004,          //Program Limit Switch unit 4

// Oscilloscope 9001..9999
    OSCI_CHANNEL_1 := 9001, // Oscilloscope Channel 1
    // ...
    OSCI_CHANNEL_128 := 9128 // Oscilloscope Channel 128

// Sercos address 10001..10512
    SERCOS_ADDRESS_1 := 10001, // Sercos address 1
    // ...
    SERCOS_ADDRESS_512 := 10512, // Sercos address 512

// Sercos topology 11001..11512
    SERCOS_TOPOLOGY_1 := 11001, // Sercos topology 1
    // ...
    SERCOS_TOPOLOGY_512 := 11512, // Sercos topology 512

// Logical drive 12001..12512 with native operation data *)
    AXIS_NATIVE_1 := 12001, // Logical drive 1 with native operation data
    // ...
    AXIS_NATIVE_512 := 12512 // Logical drive 512 with native operation data
);
END_TYPE

```



Die Nummer "00", "NO_OBJECT" (es wurde kein Objekt angegeben) wird für deaktivierte Achsen verwendet.



Die Vergabe des Namens und der Nummern erfolgt beim Anlegen der jeweiligen Achse/ des Objektes in der IndraWorks-Suite (IndraLogic, GlobaleMlcVariablen).

3.3.5 AXIS_GROUP_REF

Kurzbeschreibung

Die Struktur enthält Informationen zur gewünschten Steuerung und zur korrespondierenden Achsgruppe.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek / Ordner
MLC 15VRS Grundpaket PLC	RIL_CommonTypes / Datatypes of PLCopen

Tab. 3-6: Referenztable *AXIS_GROUP_REF***Typdefinition**

```
// Structure of PLCopen element AXIS_GROUP_REF.
// The AXIS_GROUP_REF is a structure that contains information of the axis-group.
// It is used as a VAR_IN_OUT CONSTANT in Motion Control Function Blocks.
TYPE AXIS_GROUP_REF:
STRUCT
  CntrlNo: CONTROLS := LOCAL_CNTRL; // control reference number
                                     // Steuerungsnummer, Init: lokale Steuerung
  GroupNo: KIN_OBJECTS := NO_KINEMATICS; // object reference number
                                     // Achsgruppenreferenznummer, Init: NO_KINEMATICS
END_STRUCT
END_TYPE
```

Funktionsbeschreibung

Die Informationsübergabe erfolgt ab Version 14VRS per **VAR_IN_OUT CONSTANT** an den jeweiligen Funktionsbaustein.

Die *AXIS_GROUP_REF*-Struktur enthält zwei 16-Bit-Adresselemente:

- die Steuerungsadresse (Element von [CONTROLS](#), Seite 21) und
- die Objektadresse (Element von [KIN_OBJECTS](#), Seite 24)

Die Elemente können per logischer oder physischer Adresse angesprochen werden. Im Falle einer logischen Adresse zeigt das Element auf eine Parameterstruktur mit einer physikalischen Adresse (z.B. Ethernet-Adresse der Steuerung).



Die Informationsübergabe erfolgte bis einschließlich Version 13VRS per **VAR_IN_OUT**.

Ab Version 14VRS erfolgt die Informationsübergabe per **VAR_IN_OUT CONSTANT**.

3.3.6 KIN_OBJECTS

Kurzbeschreibung

Die Aufzählung der verfügbaren Kinematiken (Achsgruppen) enthält Objekte eines Motion-Verbundes, auf die mit Coordinated Motion-Funktionsbausteinen zugegriffen werden kann, siehe auch ([AXIS_GROUP_REF](#), Seite 23).

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek / Ordner
MLC 15VRS Grundpaket PLC	RIL_CommonTypes / Datatypes of logical addresses

Tab. 3-7: Referenztable *KIN_OBJECTS***Typdefinition**

```
TYPE KIN_OBJECTS :
(
  (* not specified object
   NO_KINEMATICS :=0,
  (* axis groups 1..16 *)
  KINEMATICS_1 := 01,
  KINEMATICS_2 := 02,
  KINEMATICS_3 := 03,
  KINEMATICS_4 := 04,
  KINEMATICS_5 := 05,
  KINEMATICS_6 := 06,
  KINEMATICS_7 := 07,
  (* no object specified
  (* synchronize axis group 1 *)
  (* synchronize axis group 2 *)
  (* synchronize axis group 3 *)
  (* synchronize axis group 4 *)
  (* synchronize axis group 5 *)
  (* synchronize axis group 6 *)
  (* synchronize axis group 7 *)
```

```

KINEMATICS_8 := 08,      (* synchronize axis group 8 *)
KINEMATICS_9 := 09,      (* synchronize axis group 9 *)
KINEMATICS_10 := 10,     (* synchronize axis group 10 *)
KINEMATICS_11 := 11,     (* synchronize axis group 11 *)
KINEMATICS_12 := 12,     (* synchronize axis group 12 *)
KINEMATICS_13 := 13,     (* synchronize axis group 13 *)
KINEMATICS_14 := 14,     (* synchronize axis group 14 *)
KINEMATICS_15 := 15,     (* synchronize axis group 15 *)
KINEMATICS_16 := 16,     (* synchronize axis group 16 *)
);
END_TYPE

```



Die Vergabe des Namens und der Nummern erfolgt beim Anlegen des Objektes in der IndraWorks-Suite (IndraLogic, GlobalMlcVariablen).



Die Anzahl der realisierbaren Kinematiken hängt vom Steuerungstyp und der verwendeten Firmwareversion ab.

3.3.7 TOUCHPROBE_REF

Kurzbeschreibung

Die Struktur enthält Informationen zur gewünschten Steuerung und dem von dieser Steuerung gesteuerten Objekt (Messtaster).

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek / Ordner
MLC 15VRS Grundpaket PLC	RIL_CommonTypes / Datatypes of PLCopen

Tab. 3-8: Referenztablette TOUCHPROBE_REF

Typdefinition

```

(* Structure of PLCopen element TOUCHPROBE_REF. The TOUCHPROBE_REF is a structure that *)
(* contains information on the corresponding object. *)

TYPE TOUCHPROBE_REF :
STRUCT
  CntrlNo:      CONTROLS      := LOCAL_CNTRL;  (* control reference number *)
              (* Steuerungsnummer, Init: lokale Steuerung *)
  TouchProbeNo: TP_OBJECTS := NO_TOUCHPROBE  (* object reference number *)
              (* Messtastereferenznummer, Init: kein Objekt *)
END_STRUCT
END_TYPE

```

Funktionsbeschreibung

Die Informationsübergabe erfolgt ab Version 14VRS per **VAR_IN_OUT CONSTANT** an den jeweiligen PLCopen-Funktionsbaustein.

Die TOUCHPROBE_REF-Struktur enthält zwei 16-Bit-Adresselemente:

- die Steuerungsadresse (Element von **CONTROLS**, Seite 21) und
- die Objektadresse (Element von **TP_OBJECTS**, Seite 26).



Die Informationsübergabe erfolgte bis einschließlich Version 13VRS per **VAR_IN_OUT**.

Ab Version 14VRS erfolgt die Informationsübergabe per **VAR_IN_OUT CONSTANT**.

3.3.8 TP_OBJECTS

Kurzbeschreibung Die Aufzählung der verfügbaren Messtaster (Touch Probe) enthält Objekte eines Motion-Verbundes, auf die mit Messtaster-Funktionsbausteinen zugegriffen werden kann, siehe auch ([TOUCHPROBE_REF](#), Seite 25).

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek /Ordner
MLC 15VRS Grundpaket PLC	RIL_CommonTypes / Datatypes of logical addresses

Tab. 3-9: Referenztable TP_OBJECTS

Typdefinition

```
(* Enumeration of valid logical object addresses *)
TYPE TP_OBJECTS :
(
  (* not specified object *)
  NO_TOUCHPROBE :=0,      (* no object specified *)

  (* Touch Probe 1..100, realized on controller *)
  TOUCHPROBE_1   := 01,   (* Touch Probe 1 *)
  (* ... *)
  TOUCHPROBE_100 :=100    (* Touch Probe 100 *)

  (* Touch Probe 1001..1002, realized on drive *)
  TOUCHPROBE_DRIVE_1 :=1001 (* Touch Probe Drive 1 *)
  TOUCHPROBE_DRIVE_2 :=1002 (* Touch Probe Drive 2 *)
);
END_TYPE
```



Die Vergabe des Namens und der Nummern erfolgt beim manuellen Anlegen des jeweiligen Messtasters in der IndraWorks-Suite (IndraLogic, GlobaleMlcVariablen).



Die Anzahl der realisierbaren Messtaster hängt vom Steuerungstyp und der verwendeten Firmwareversion ab.

3.4 Datentypen zur Unterstützung von Kurvenscheiben, Motion- und FlexProfilen

3.4.1 Datentyp zum Umschaltung zwischen Kurvenscheiben, allgemeines

In Abhängigkeit von der Leistungsfähigkeit der Steuerelektronik können folgende Kurventabellen hinterlegt werden:

- MPDx - 4 Tabellen á 1024 Punkte (Tabelle 1...4),
- MPBx - 4 Tabellen á 1024 Punkte (Tabelle 1...4),
- MPHx - 4 Tabellen á 1024 Punkte (Tabelle 1...4), 4 Tabellen á 128 Punkte (Tabelle 5...8).

Im laufenden Betrieb kann zwischen den Kurventabellen umgeschaltet werden. Auswahl und Umschaltung können über den Funktionsbaustein MC_CamIn (CamTableID: [MC_CAM_ID](#), Seite 26) oder über Parametersteuerung "A-0-2740, ELS, CamIn, Kurventabellenvorwahl" erfolgen.

3.4.2 MC_CAM_ID

Kurzbeschreibung Der Aufzählungstyp unterstützt die Auswahl der gewünschten Kurventabelle ([Datentypen zur Unterstützung von Kurvenscheiben, Motion- und FlexProfilen](#), Seite 26).

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek / Ordner
MLC 15VRS Grundpaket PLC	RIL_CommonTypes / Datatypes of PLCopen

Tab. 3-10: Referenztable MC_CAM_ID

Typdefinition

```

TYPE MC_CAM_ID :
(
  CAM_TABLE_1  := 1, (* Kurventabelle 1 *)
  CAM_TABLE_2  := 2, (* Kurventabelle 2 *)
  (* ... *)
  CAM_TABLE_255 := 255 (* Kurventabelle 255 *)
);
END_TYPE

```

Funktionsbeschreibung

Im laufenden Betrieb kann zwischen den Kurventabellen umgeschaltet werden. Auswahl und Umschaltung können über den Funktionsbaustein MC_CamIn, ML_PLCopen.library, oder über Parametersteuerung "A-0-2740, ELS, CamIn, Kurventabellenvorwahl", erfolgen.

3.4.3 MC_STEP_MODE, MotionProfile erweitert zu FlexProfile**Kurzbeschreibung**

Der Aufzählungsdatentyp MC_STEP_MODE beschreibt die Abhängigkeit der Folgeachseposition von der Leitachse während eines Bewegungsschrittes (MotionProfile erweitert zu FlexProfile).

Die Abhängigkeit kann gegeben sein als funktionale Abhängigkeit oder in Form einer Kurventabelle.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek / Ordner
MLC 15VRS Grundpaket PLC	RIL_CommonTypes / Datatypes of PLCopen

Tab. 3-11: Referenztable MC_STEP_MODE

Typdefinition

```

TYPE MC_STEP_MODE :
(
  REST_IN_REST_INCLINEDSINE      := 16#0000, (* MotionProfile, FlexProfile *)
  REST_IN_REST_POLY5             := 16#0100, (* MotionProfile, FlexProfile *)
  REST_IN_VELOCITY_POLY5         := 16#0200, (* MotionProfile, FlexProfile *)
  REST_IN_VELOCITY_POLY7         := 16#0300, (* FlexProfile *)
  VELOCITY_IN_REST_POLY5         := 16#0400, (* MotionProfile, FlexProfile *)
  VELOCITY_IN_REST_POLY7         := 16#0500, (* FlexProfile *)
  CONSTANT_VELOCITY              := 16#0600, (* MotionProfile, FlexProfile *)
  VELOCITY_IN_VELOCITY_POLY5     := 16#0700, (* MotionProfile, FlexProfile *)

  REST_IN_REST_POLY5_VLIM        := 16#1200, (* MotionProfile, FlexProfile *)
  REST_IN_REST_PARABOLA         := 16#1300, (* FlexProfile *)
  REST_IN_REST_POLY8            := 16#1400, (* FlexProfile *)

  REST_IN_REST_LINEAR            := 16#0800, (* FlexProfile *)
  REST_IN_REST_POLY7             := 16#0900, (* FlexProfile *)
  REST_IN_REST_SINE              := 16#0A00, (* FlexProfile *)
  REST_IN_REST_GUTMANSINE        := 16#0B00, (* FlexProfile *)
  REST_IN_REST_SINEACC           := 16#0C00, (* FlexProfile *)
  REST_IN_REST_SINETORQUE        := 16#0D00, (* FlexProfile *)
  REST_IN_REST_MOD_TRAPEZE       := 16#0E00, (* FlexProfile *)
  REST_IN_REST_MOD_SINE          := 16#0F00, (* FlexProfile *)

  VELOCITY_IN_VELOCITY_POLY7     := 16#1000, (* FlexProfile *)
  VELOCITY_IN_VELOCITY_MOD_SINE  := 16#1100, (* FlexProfile *)

  MOTION_IN_MOTION_POLY5        := 16#2000, (* FlexProfile, *)
  (* auch FlexStep / also Flex motion step *)

```

RIL_CommonTypes.library

```

MOTION_IN_MOTION_POLY7          := 16#2100, (* FlexProfile, *)
(* auch FlexStep / also Flex motion step *)
COMMON_POLY5                     := 16#4000, (* FlexProfile *)
COMMON_POLY7                     := 16#4100, (* FlexProfile *)
COMMON_POLY2                     := 16#4200, (* FlexProfile *)
COMMON_POLY3                     := 16#4300, (* FlexProfile *)
COMMON_POLY4                     := 16#4400, (* FlexProfile *)
COMMON_POLY8                     := 16#4500, (* FlexProfile *)

(* Erweiterte Bewegungsgesetze / Extended motion laws *)
X_VELOCITY_IN_VELOCITY_TRAPEZE_ALIM := 16#5000, (* FlexProfile *)
X_MOTION_IN_MOTION_POLY5_VLIM      := 16#6000, (* FlexProfile *)
X_MOTION_IN_MOTION_POLY5_SLIM      := 16#6100, (* FlexProfile, *)
(* Auch FlexStep / also Flex motion step *)
X_FIT_VEL_TRAPEZE_ALIM             := 16#7000, (* FlexProfile *)
X_FIT_SINE_TRAPEZE_ALIM            := 16#7100, (* FlexProfile *)
X_FIT_ACC_TRAPEZE_JLIM             := 16#7200, (* FlexProfile *)

X_MOTION_IN_MOTION_ACAM            := 16#8000, (* FlexProfile, *)
X_MOTION_IN_MOTION_VCAM            := 16#8100, (* FlexProfile, *)
X_MOTION_IN_MOTION_VCAM2_A         := 16#8200, (* FlexProfile, *)
X_MOTION_IN_MOTION_VCAM2_B         := 16#8300, (* FlexProfile, *)

(* Punktetabellen MotionProfile / Cam tables *)
CAMTABLE_1                       := 1,          (* C-0-2001 *)
CAMTABLE_2                       := 2,          (* C-0-2002 *)
CAMTABLE_3                       := 3,          (* C-0-2003 *)
(* ..... *)
CAMTABLE_98                      := 98,         (* C-0-2098 *)
CAMTABLE_99                      := 99,         (* C-0-2099 *)
);
END_TYPE

```

Funktionsbeschreibung



Nachstehende Funktionsbeschreibung enthält die Details für Motion- und FlexProfile.

Als Modus eines Schrittes können verschiedene Normprofile oder eine Kurventabelle definiert werden.

Als Modi stehen zur Verfügung:

- R-R: Rast in Rast, Polynom 5. Ordnung,
- R-R: Rast in Rast, geneigte Sinuslinie,
- R-G: Rast in Geschwindigkeit, Polynom 5. Ordnung,
- G-R: Geschwindigkeit in Rast, Polynom 5. Ordnung,
- G: konstante Geschwindigkeit,
- G-G: Geschwindigkeit in Geschwindigkeit, Polynom 5. Ordnung,
- Kurvenscheibentabelle.

R-R: Rast in Rast

Für das Profil "Rast in Rast" stehen zehn Profile zur Verfügung. Es kann zwischen den normierten Bewegungsgesetzen "Polynom 5. Ordnung" und "Geneigte Sinuslinie" ... gewählt werden.

Diese Profile sind durch den Hub und Leitachsbereich definiert. Die Endgeschwindigkeit des Schrittes wird vom Folgeschritt bestimmt.

Das Profil für die "**Polynom 5. Ordnung**" ist durch nachstehende Funktion bestimmt:

$$f(z) = 6z^5 - 15z^4 + 10z^3$$

Abb. 3-1: Rast in Rast, Polynom 5. Ordnung

Die möglichen Randwerte für das Bewegungsgesetz sind:

Randwert	z=0	z=1
1. Ordnung (Geschwindigkeit)	0	0
2. Ordnung (Beschleunigung)	0	0
3. Ordnung (Ruck)	automatisch	automatisch

Tab. 3-12: Randwerte für das Bewegungsgesetz - Polynom 5. Ordnung

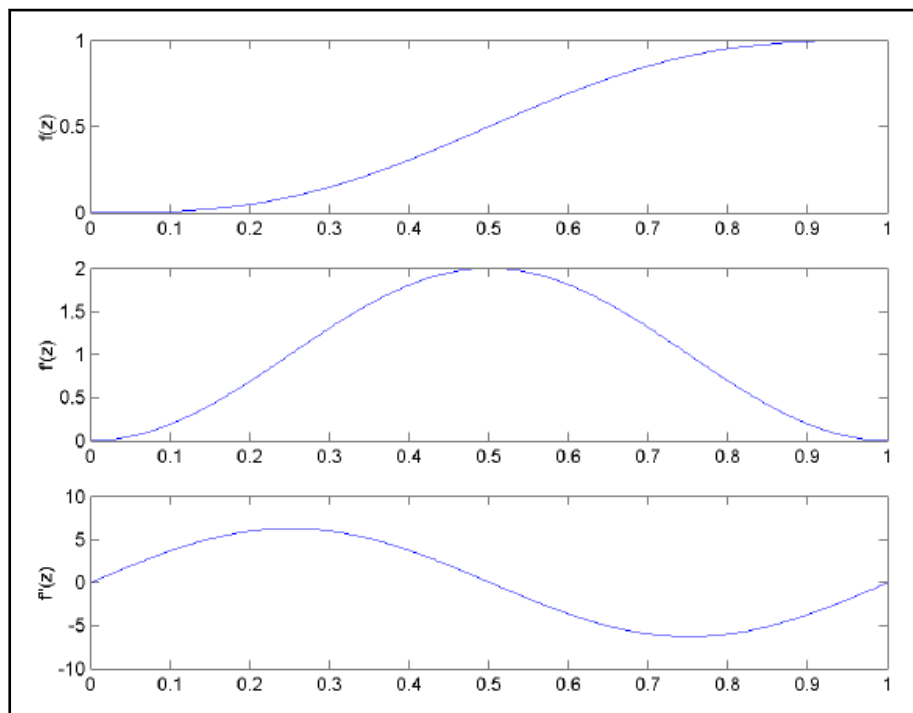


Abb. 3-2: Rast in Rast, Polynom 5. Ordnung

Für das "Polynom 7. Ordnung" gilt die Formel :

$$f(z) = -20z^7 + 70z^6 - 84z^5 - 35z^4$$

Abb. 3-3: Rast in Rast, Polynom 7. Ordnung

Die möglichen Randwerte für das Bewegungsgesetz sind:

Randwert	z=0	z=1
1. Ordnung (Geschwindigkeit)	0	0
2. Ordnung (Beschleunigung)	0	0
3. Ordnung (Ruck)	0	0

Tab. 3-13: Randwerte für das Bewegungsgesetz - Polynom 7. Ordnung

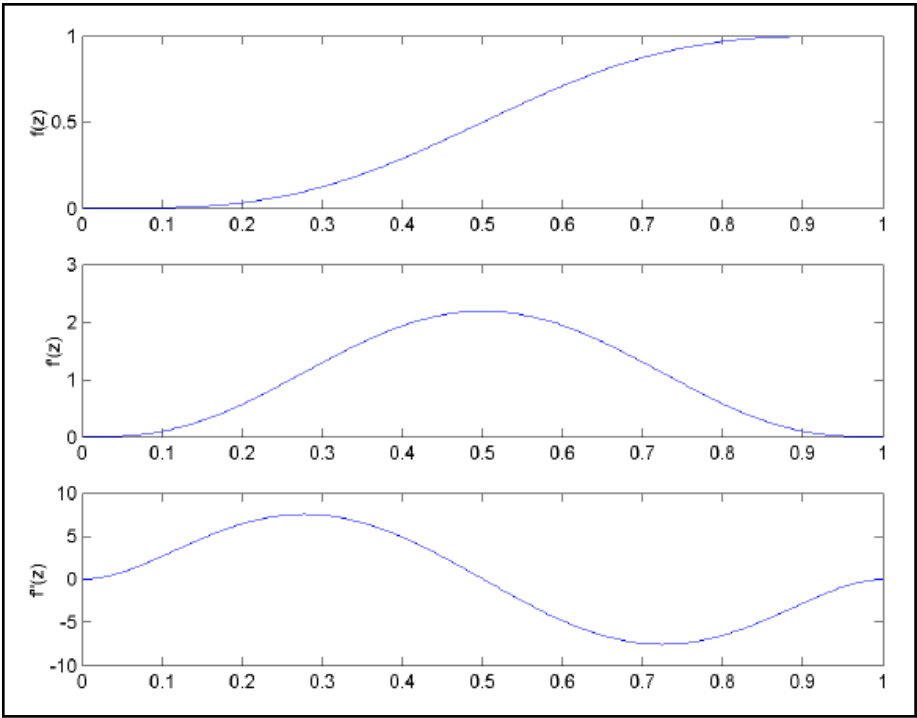


Abb. 3-4: Rast in Rast, Polynom 7. Ordnung

Das Profil für die **"Geneigte Sinuslinie"** ist durch nachstehende Funktion bestimmt:

$$f(z) = z - \frac{1}{2\pi} \sin(2\pi z)$$

Abb. 3-5: Rast in Rast, Geneigte Sinuslinie

Die möglichen Randwerte für das Bewegungsgesetz sind:

Randwert	z=0	z=1
1. Ordnung (Geschwindigkeit)	0	0
2. Ordnung (Beschleunigung)	0	0
3. Ordnung (Ruck)	automatisch	automatisch

Tab. 3-14: Randwerte für Bewegungsgesetz - geneigte Sinuslinie

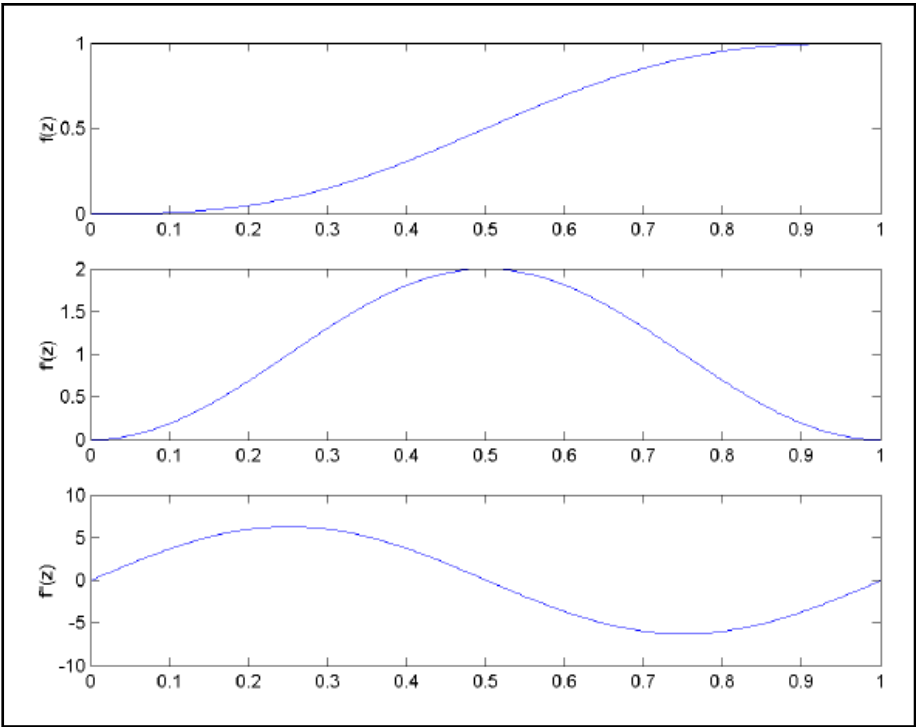


Abb. 3-6: Rast in Rast, Geneigte Sinuslinie (BESTEHORN)

Das Profil für die "Sinuslinie (einfache Sinoide)" ist durch nachstehende Funktion bestimmt:

$$f(z) = \frac{1}{2} \left[1 - \cos(\pi z) \right]$$

Abb. 3-7: Rast in Rast, Sinuslinie

Die möglichen Randwerte für das Bewegungsgesetz sind:

Randwert	z=0	z=1
1. Ordnung (Geschwindigkeit)	0	0
2. Ordnung (Beschleunigung)	automatisch	automatisch
3. Ordnung (Ruck)	automatisch	automatisch

Tab. 3-15: Randwerte für das Bewegungsgesetz Sinuslinie

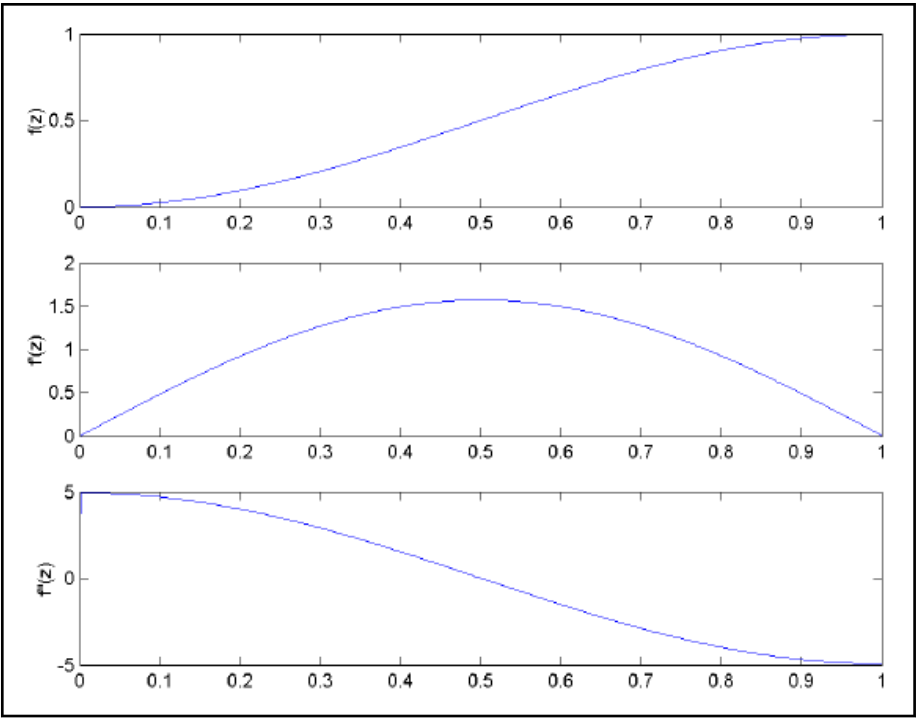


Abb. 3-8: Rast in Rast, Sinuslinie (einfache Sinoide)
Das Profil für die "Lineare Rast" ist durch nachstehende Funktion bestimmt:

$$f(z) = 0$$

Abb. 3-9: Lineare Rast
Die möglichen Randwerte für das Bewegungsgesetz sind:

Randwert	z=0	z=1
1. Ordnung (Geschwindigkeit)	0	0
2. Ordnung (Beschleunigung)	0	0
3. Ordnung (Ruck)	0	0

Tab. 3-16: Randwerte für Bewegungsgesetze - Lineare Rast

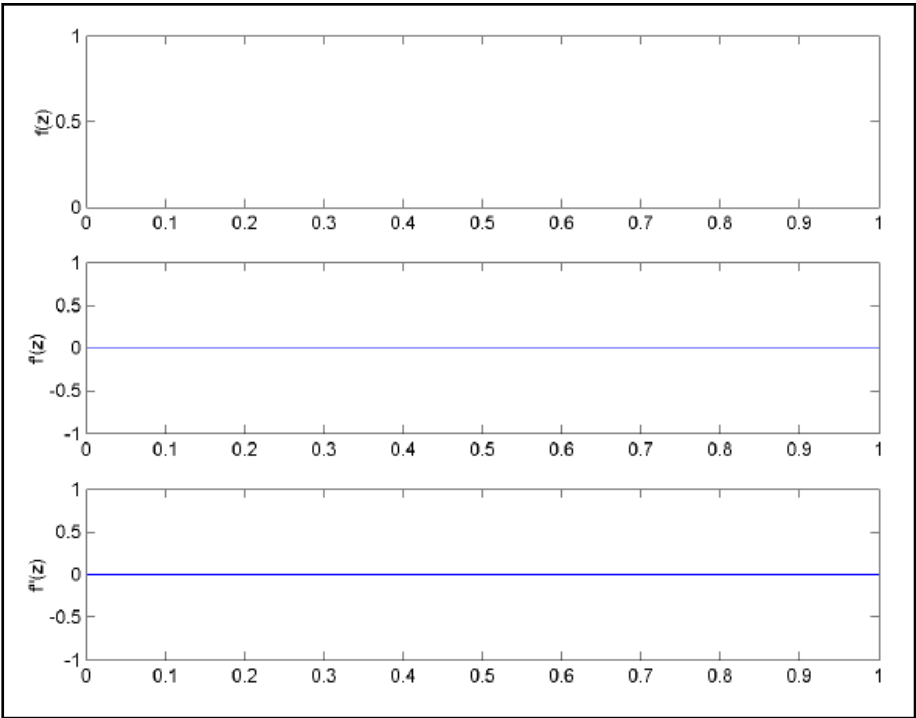


Abb. 3-10: Lineare Rast

Das Profil für die **"Sinoide von Gutman"** ist durch nachstehende Funktion bestimmt:

$$f(z) = z - \frac{15}{32\pi} \sin(2\pi z) - \frac{1}{96\pi} \sin(6\pi z)$$

Abb. 3-11: Sinoide nach Gutman

Die möglichen Randwerte für das Bewegungsgesetz sind:

Randwert	z=0	z=1
1. Ordnung (Geschwindigkeit)	0	0
2. Ordnung (Beschleunigung)	0	0
3. Ordnung (Ruck)	automatisch	automatisch

Tab. 3-17: Randwerte für das Bewegungsgesetz - Sinoide von Gutman

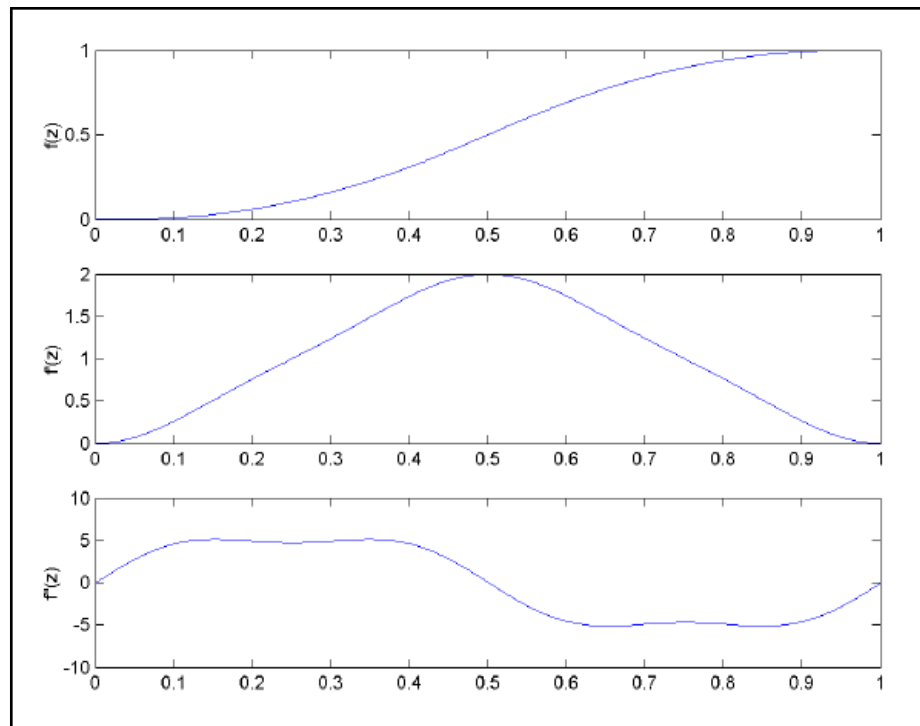


Abb. 3-12: Rast in Rast, Sinoide von Gutman

Das Profil für die **"Beschleunigungsoptimale geneigte Sinuslinie"** ist durch nachstehende Funktion bestimmt:

$$f(z) = z - \frac{1}{2\pi} \sin(2\pi z)$$

$$g(z) = z - \frac{0.41}{2\pi} \sin(2\pi z)$$

Abb. 3-13: Beschleunigungsoptimale geneigte Sinuslinie

Die möglichen Randwerte für das Bewegungsgesetz sind:

Randwert	z=0	z=1
1. Ordnung (Geschwindigkeit)	0	0
2. Ordnung (Beschleunigung)	0	0
3. Ordnung (Ruck)	automatisch	automatisch

Tab. 3-18: Randwerte für das Bewegungsgesetz - Beschleunigungsoptimale geneigte Sinuslinie

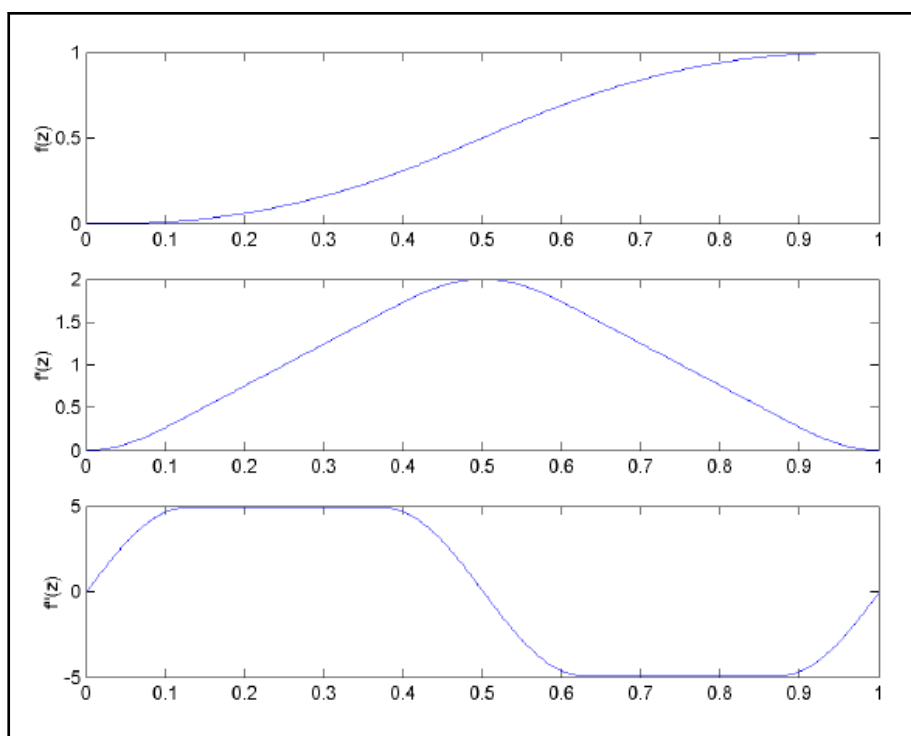


Abb. 3-14: Rast in Rast, Beschleunigungsoptimale geneigte Sinuslinie

Das Profil für die **"Momentenoptimale geneigte Sinuslinie"** ist durch nachstehende Funktion bestimmt:

$$f(z) = z - \frac{1}{2\pi} \sin(2\pi z)$$

$$g(z) = z - \frac{0.134}{2\pi} \sin(2\pi z)$$

Abb. 3-15: Momentenoptimale geneigte Sinuslinie

Die möglichen Randwerte für das Bewegungsgesetz sind:

Randwert	z=0	z=1
1. Ordnung (Geschwindigkeit)	0	0
2. Ordnung (Beschleunigung)	0	0
3. Ordnung (Ruck)	automatisch	automatisch

Tab. 3-19: Randwerte für das Bewegungsgesetz - Momentenoptimale geneigte Sinuslinie

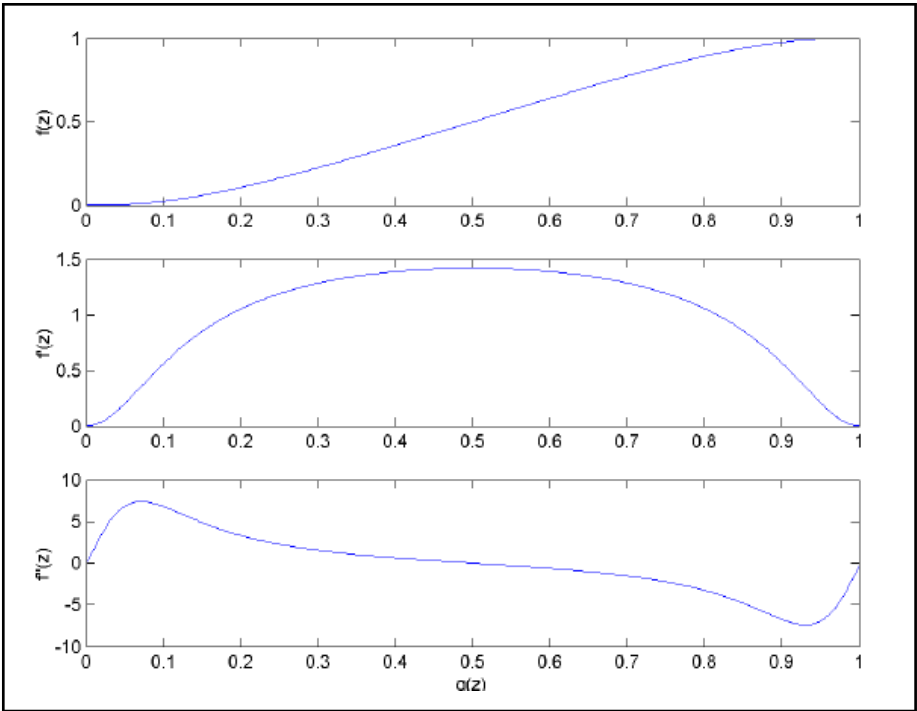


Abb. 3-16: Rast in Rast, Momentenoptimale geneigte Sinuslinie

Das Profil für die **"Modifizierte Sinoide"** ist durch nachstehende Funktion bestimmt:

$$\begin{aligned} 0 \leq z \leq \frac{1}{8} : f(z) &= \frac{1}{4 + \pi} \left[\pi z - \frac{1}{4} \sin(4\pi z) \right] \\ \frac{1}{8} \leq z \leq \frac{7}{8} : f(z) &= \frac{1}{4 + \pi} \left[2 + \pi z - \frac{4}{9} \sin\left(\frac{4\pi}{3} z + \frac{\pi}{3}\right) \right] \\ \frac{7}{8} \leq z \leq 1 : f(z) &= \frac{1}{4 + \pi} \left[4 + \pi z - \frac{1}{4} \sin(4\pi z) \right] \end{aligned}$$

Abb. 3-17: Modifizierte Sinoide

Die möglichen Randwerte für das Bewegungsgesetz sind:

Randwert	z=0	z=1
1. Ordnung (Geschwindigkeit)	0	0
2. Ordnung (Beschleunigung)	0	0
3. Ordnung (Ruck)	automatisch	automatisch

Tab. 3-20: Randwerte für das Bewegungsgesetz - Modifizierte Sinoide

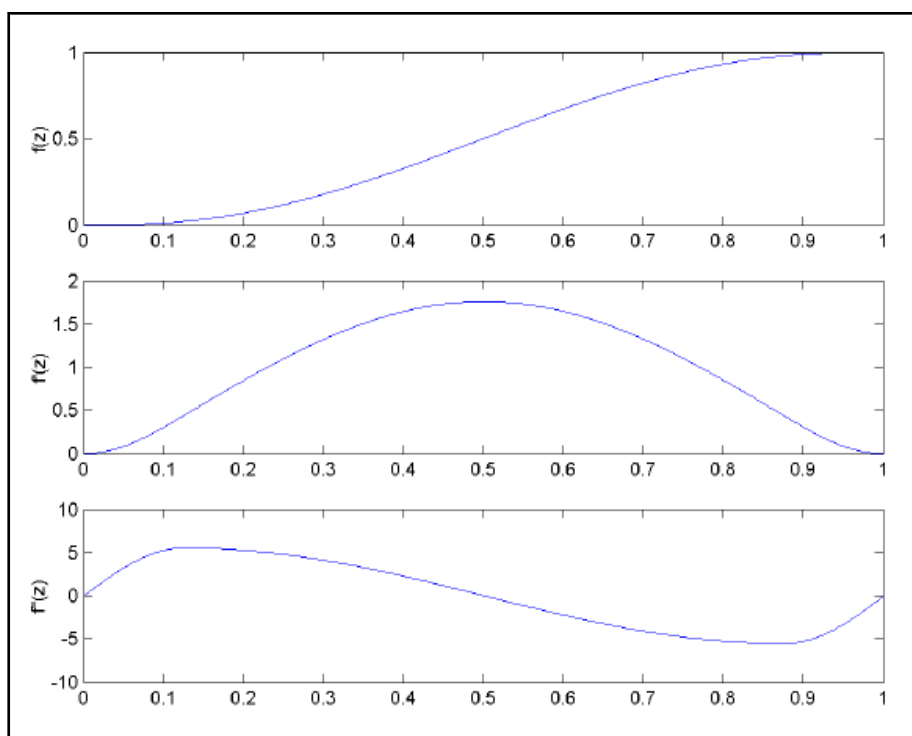


Abb. 3-18: Rast in Rast, Modifizierte Sinoide

Das Profil für die **"Modifiziertes Beschleunigungstrapez"** ist durch nachstehende Funktion bestimmt:

$$\begin{aligned}
 0 \leq z \leq \frac{1}{8}: f(z) &= \frac{1}{2+\pi} \left[2z - \frac{1}{2\pi} \sin(4\pi z) \right] \\
 \frac{1}{8} \leq z \leq \frac{3}{8}: f(z) &= \frac{1}{2+\pi} \left[-\frac{1}{2\pi} + 2z + 4\pi \left(z + \frac{1}{8} \right)^2 \right] \\
 \frac{3}{8} \leq z \leq \frac{1}{2}: f(z) &= \frac{1}{2+\pi} \left[-\frac{\pi}{2} + 2(1+\pi)z - \frac{1}{2\pi} \sin(4\pi z - \pi) \right] \\
 \frac{1}{2} \leq z \leq 1: f(z) &= 1 - f(1-z)
 \end{aligned}$$

Abb. 3-19: Modifiziertes Beschleunigungstrapez

Die möglichen Randwerte für das Bewegungsgesetz sind:

Randwert	z=0	z=1
1. Ordnung (Geschwindigkeit)	0	0
2. Ordnung (Beschleunigung)	0	0
3. Ordnung (Ruck)	automatisch	automatisch

Tab. 3-21: Randwerte für das Bewegungsgesetz - Modifiziertes Beschleunigungstrapez

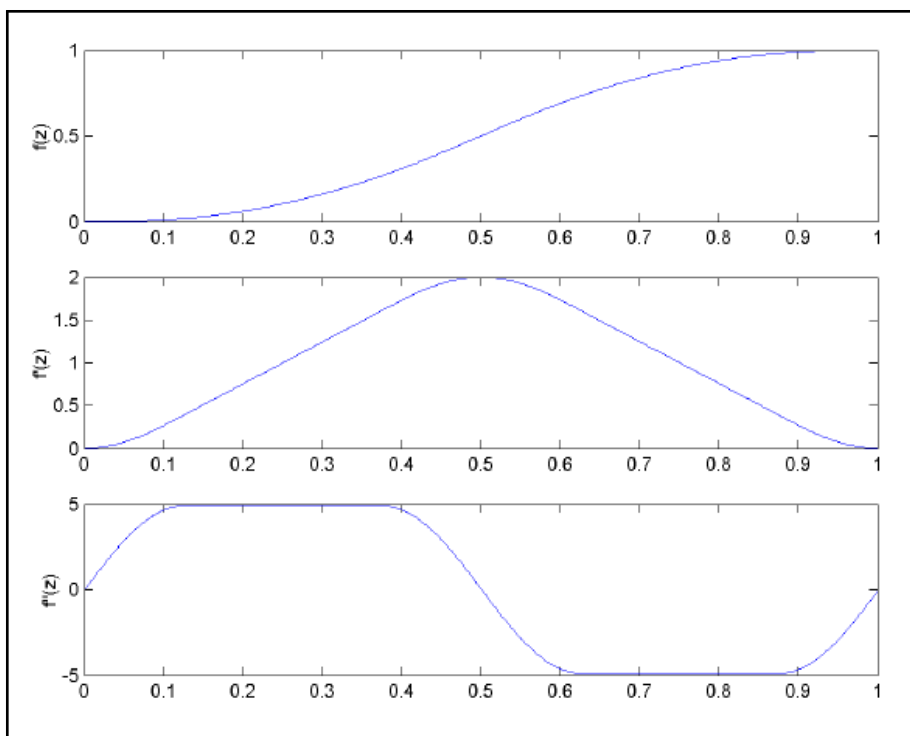


Abb. 3-20: Rast in Rast, Modifiziertes Beschleunigungstrapez

R-G: Rast in Geschwindigkeit

Dieses Profil dient zum Übergang aus dem Stillstand in eine bestimmte Geschwindigkeit.

Das Profil wird mit einem "Polynom 5. Ordnung" berechnet :

$$f(z) = A_5 z^5 + A_4 z^4 + A_3 z^3$$

Abb. 3-21: Polynom 5. Ordnung

Die möglichen Randwerte für das Bewegungsgesetz sind:

Randwert	z=0	z=1
1. Ordnung (Geschwindigkeit)	0	wählbar
2. Ordnung (Beschleunigung)	0	wählbar
3. Ordnung (Ruck)	automatisch	automatisch

Tab. 3-22: Randwerte für das Bewegungsgesetz Polynom 5. Ordnung

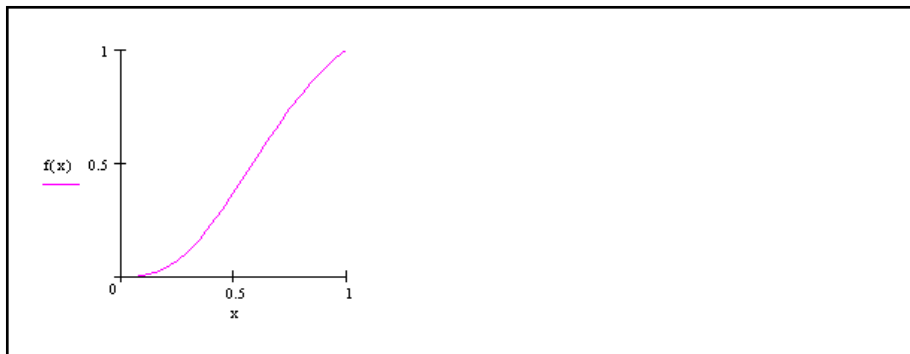


Abb. 3-22: Rast in Geschwindigkeit, Polynom 5. Ordnung

Dieses Profil dient zum Übergang aus dem Stillstand in eine bestimmte Geschwindigkeit.

Das Profil wird mit einem **"Polynom 7. Ordnung"** berechnet :

$$f(z) = A_7 z^7 + A_6 z^6 + A_5 z^5 + A_4 z^4 + A_3 z^3$$

Abb. 3-23: Polynom 7. Ordnung

Die möglichen Randwerte für das Bewegungsgesetz sind:

Randwert	z=0	z=1
1. Ordnung (Geschwindigkeit)	0	wählbar
2. Ordnung (Beschleunigung)	0	wählbar
3. Ordnung (Ruck)	automatisch	wählbar

Tab. 3-23: Randwerte für das Bewegungsgesetz - Polynom 7. Ordnung

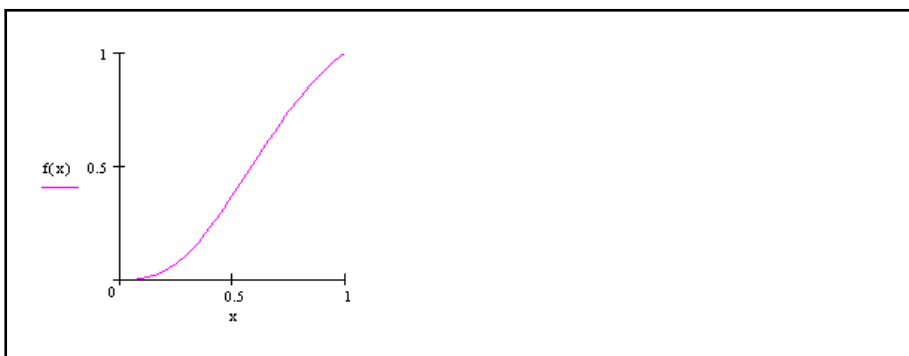


Abb. 3-24: Rast in Geschwindigkeit, Polynom 7. Ordnung

G-R: Geschwindigkeit in Rast

Das Profil wird zum Übergang aus einer definierten Geschwindigkeit in den Stillstand verwendet.

Bei diesem Bewegungsprofil wird ein Weg verfahren, der durch den Hub für diesen Bewegungsschritt festgelegt ist.

Das Profil wird mit einem **"Polynom 5. Ordnung"** berechnet :

$$f(z) = A_5 z^5 + A_4 z^4 + A_3 z^3 + A_1 z$$

Abb. 3-25: Geschwindigkeit in Rast, Polynom 5. Ordnung

Die möglichen Randwerte für das Bewegungsgesetz sind:

Randwert	z=0	z=1
1. Ordnung (Geschwindigkeit)	wählbar	0
2. Ordnung (Beschleunigung)	wählbar	0
3. Ordnung (Ruck)	automatisch	0

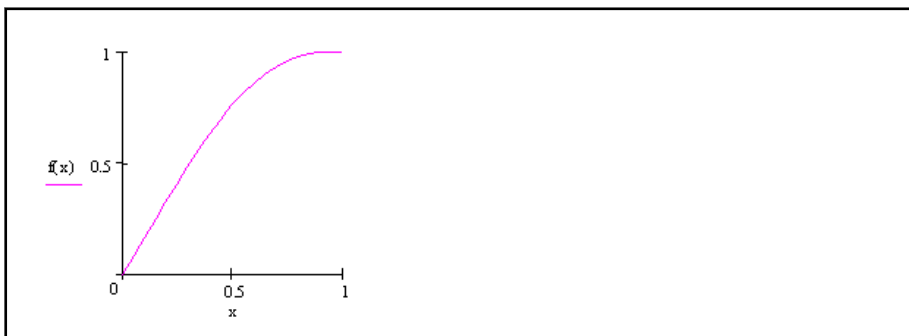


Abb. 3-26: Geschwindigkeit in Rast, Polynom 5. Ordnung

Das Profil wird zum Übergang aus einer definierten Geschwindigkeit in den Stillstand verwendet.

Das Profil wird mit einem **"Polynom 7. Ordnung"** berechnet :

$$f(z) = A_7 z^7 + A_6 z^6 + A_5 z^5 + A_4 z^4 + A_3 z^3 + A_2 z^2 + A_1 z$$

Abb. 3-27: Rast in Geschwindigkeit, Polynom 7. Ordnung

Die möglichen Randwerte für das Bewegungsgesetz sind:

Randwert	z=0	z=1
1. Ordnung (Geschwindigkeit)	wählbar	0
2. Ordnung (Beschleunigung)	wählbar	0
3. Ordnung (Ruck)	wählbar	0

Tab. 3-24: Randwerte für das Bewegungsgesetz - Polynom 7. Ordnung

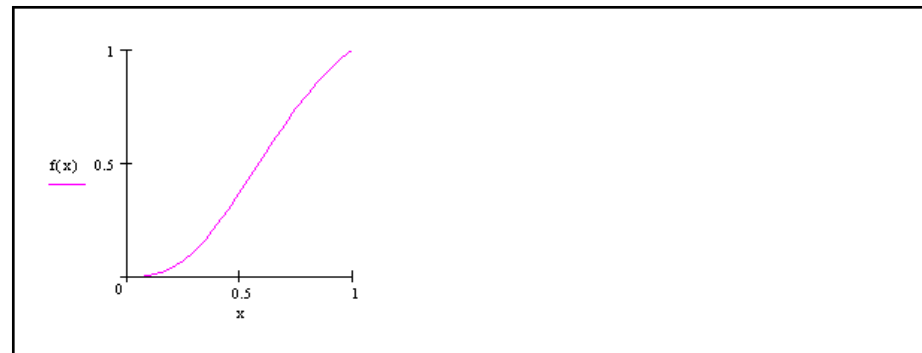


Abb. 3-28: Rast in Geschwindigkeit, Polynom 7. Ordnung

G: konstante Geschwindigkeit

Das Profil dieses Bewegungsschrittes ist eine **"Gerade"**. Die Geschwindigkeit innerhalb dieses Schrittes ist konstant. Es wird ein Weg verfahren, der durch den Hub für diesen Bewegungsschritt festgelegt ist.

Die Formel für das Profil lautet in normierter Form:

$$f(z) = z.$$

Die möglichen Randwerte für das Bewegungsgesetz sind:

Randwert	z=0	z=1
1. Ordnung (Geschwindigkeit)	automatisch	automatisch
2. Ordnung (Beschleunigung)	automatisch	automatisch
3. Ordnung (Ruck)	automatisch	automatisch

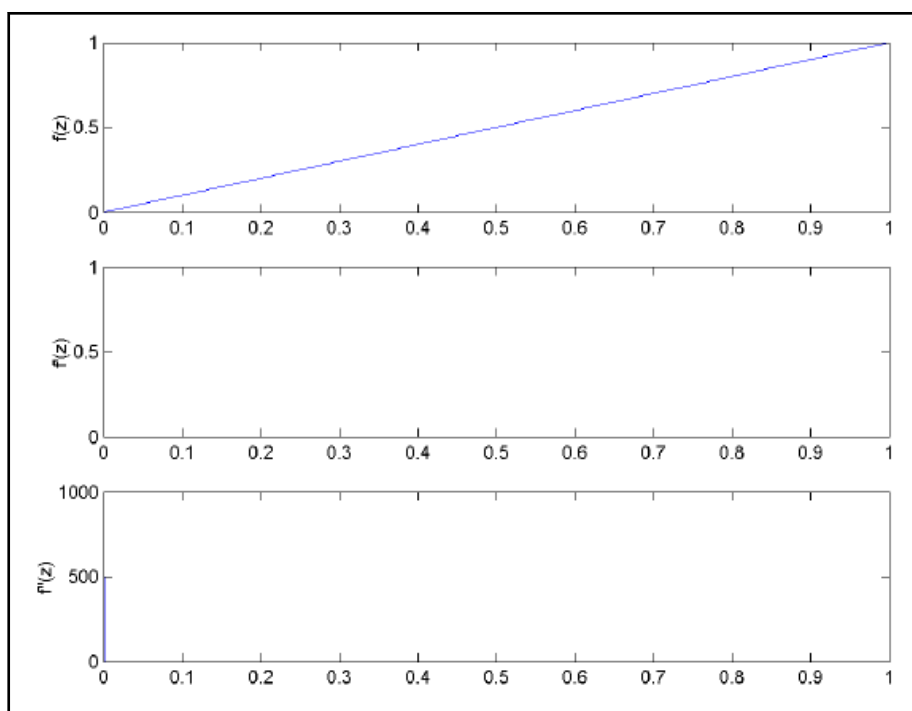


Abb. 3-29: Konstante Geschwindigkeit

Die Folgeachs-Geschwindigkeit wird aus den für diesen Bewegungsschritt vorgegebenen Werten bestimmt:

- Leitachsbereich
- Hub
- Leitachsgeschwindigkeit.

Die Eingabe eines Wertes für die Folgeachs-Geschwindigkeit ist nicht erforderlich.

G-G: Geschwindigkeit in Geschwindigkeit

Das Profil für die **"Polynom 5. Ordnung"** ist durch nachstehende Funktion bestimmt:

$$f(z) = A_5 z^5 + A_4 z^4 + A_3 z^3 + A_2 z^2 + A_1 z$$

Abb. 3-30: Polynom 5. Ordnung

Die möglichen Randwerte für das Bewegungsgesetz sind:

Randwert	z=0	z=1
1. Ordnung (Geschwindigkeit)	wählbar	wählbar
2. Ordnung (Beschleunigung)	wählbar	wählbar
3. Ordnung (Ruck)	automatisch	automatisch

Tab. 3-25: Randwerte für das Bewegungsgesetz - Polynom 5. Ordnung

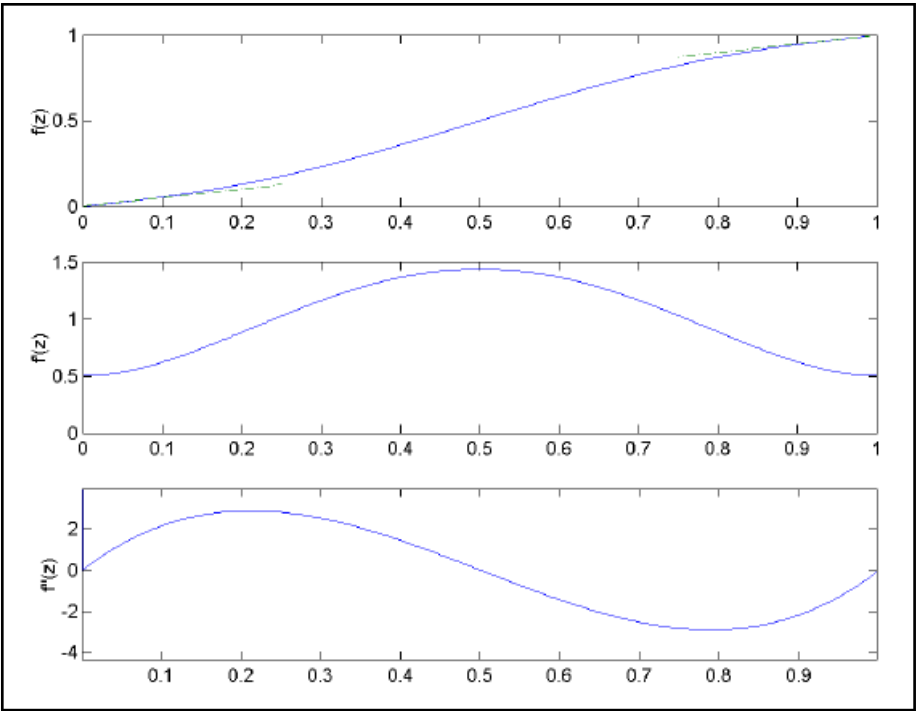


Abb. 3-31: *Geschwindigkeit in Geschwindigkeit, Polynom 5. Ordnung*
Das Profil für die **"Polynom 7. Ordnung"** ist durch nachstehende Funktion bestimmt:

$$f(z) = A_7z^7 + A_6z^6 + A_5z^5 + A_4z^4 + A_3z^3 + A_2z^2 + A_1z$$

Abb. 3-32: *Polynom 7. Ordnung*
Die möglichen Randwerte für das Bewegungsgesetz sind:

Randwert	z=0	z=1
1. Ordnung (Geschwindigkeit)	wählbar	wählbar
2. Ordnung (Beschleunigung)	wählbar	wählbar
3. Ordnung (Ruck)	wählbar	wählbar

Tab. 3-26: *Randwerte für das Bewegungsgesetz - Polynom 7. Ordnung*

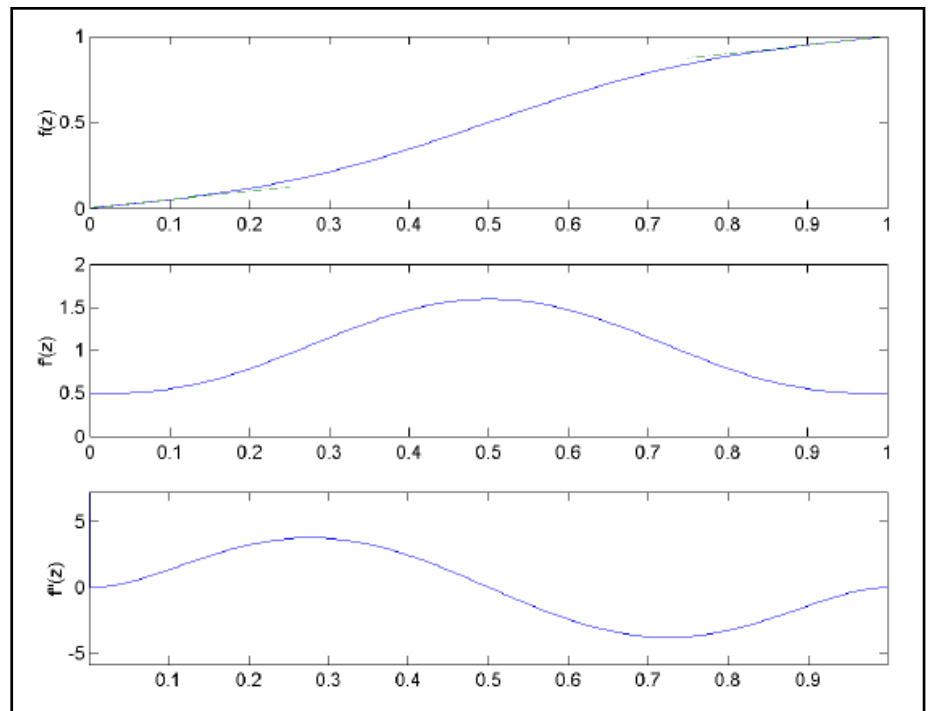


Abb. 3-33: Geschwindigkeit in Geschwindigkeit, Polynom 7. Ordnung

Das Profil für die **"Modifizierte Sinuslinie"** ist durch nachstehende Funktion bestimmt:

$$\begin{aligned}
 0 \leq z \leq \frac{\lambda}{8}: f'(z) &= C_a \frac{\lambda}{2\pi} \left(z - \frac{\lambda}{2\pi} \sin \left(2\pi \frac{z}{\lambda} \right) \right) + f'(0)z \\
 \frac{\lambda}{8} \leq z \leq \lambda: f'(z) &= C_a \frac{\lambda}{2\pi} \left(z - \frac{9\lambda}{2\pi} \sin \left(2\pi \frac{z + \frac{\lambda}{2}}{3\lambda} \right) + \frac{4\lambda}{\pi} \right) + f'(0)z \\
 \lambda \leq z \leq 1 - \frac{1-\lambda}{4}: f'(z) &= C_b \frac{1-\lambda}{2\pi} \left(z - \frac{9(1-\lambda)}{2\pi} \sin \left(2\pi \frac{z + \frac{3-5\lambda}{3(1-\lambda)}}{3\lambda} \right) + \frac{4\lambda}{\pi} \right) \\
 &\quad + f'(1) + (f'(0) - f'(1)) \left(\frac{3}{4} - \frac{1}{\pi} \right) + \lambda \\
 1 - \frac{1-\lambda}{4} \leq z \leq 1: f'(z) &= -C_b \frac{1-\lambda}{2\pi} \left(z - \frac{1-\lambda}{2\pi} \sin \left(2\pi \frac{z + \lambda}{1-\lambda} \right) - 1 \right) \\
 &\quad + 1 + f'(1) + (z-1)
 \end{aligned}$$

Abb. 3-34: Modifizierte Sinuslinie

Die möglichen Randwerte für das Bewegungsgesetz sind:

Randwert	z=0	z=1
1. Ordnung (Geschwindigkeit)	wählbar	wählbar
2. Ordnung (Beschleunigung)	0	0
3. Ordnung (Ruck)	0	0

Tab. 3-27: Randwerte für das Bewegungsgesetz - Modifizierte Sinuslinie

Erweiterte Bewegungsgesetze

Bei den erweiterten Bewegungsgesetzen handelt es sich um spezielle Bewegungsgesetze. Sie unterscheiden sich von den bekannten Bewegungsgesetzen aus den VDI Richtlinien durch neue Freiheitsgrade. Während ein Standard-Bewegungsgesetz durch die Angabe seiner linken und rechten Randwerte eindeutig definiert ist, erlauben die hier dargestellten Bewegungsgesetze noch weitere unterschiedliche Einstellungsmöglichkeiten, wie z.B. eine geforderte maximale Geschwindigkeit. In dem Fall ist der Leitachsabschnitt des Bewegungsschrittes nicht anzugeben. Stattdessen ergibt sich dieser aus der Parametrierung des Bewegungsgesetzes und der Festlegung von Kennwerten. So erlaubt beispielsweise die beschleunigungsbegrenzte Bewegung die Angabe von Hub, Anfangs- und Endgeschwindigkeit, nicht aber die Vorgabe eines Leitachsabschnitts. Dieser, und damit die Dauer des Bewegungsschrittes, ergibt sich aus den ebenfalls festlegbaren Werten einer maximalen Beschleunigung und einer maximalen Geschwindigkeit.

	Hub	Leitachsabschnitt	$v_{\max}$	$a_{\max}$	$j_{\max}$
Überschwingungsfreies Polynom 5. Ordnung	wählbar	wählbar	wählbar	automatisch	automatisch
Geschwindigkeitsbegrenztes Polynom 5. Ordnung	wählbar	wählbar	wählbar	automatisch	automatisch
Beschleunigungsbegrenzte Bewegung	wählbar	automatisch	wählbar	wählbar	automatisch
Beschleunigungsbegrenztes Sinoidenprofil	wählbar	automatisch	wählbar	wählbar	automatisch
Ruckbegrenzte Bewegung	wählbar	automatisch	wählbar	wählbar	wählbar

Tab. 3-28: Erweiterte Bewegungsgesetze

Das Profil für **"Geschwindigkeitsbegrenztes Polynom 5. Ordnung"** kann wie folgt werden:

Das aus der VDI bekannte Polynom 5. ermöglicht keine Kontrolle der maximalen Verfahrgeschwindigkeit. Zur Lösung dieses Problems wird das Bewegungsgesetz in drei Sektionen aufgeteilt. Die mittlere Sektion entspricht einer Geraden mit der geforderten Geschwindigkeit. Umgeben wird diese von zwei Polynomen 5. Grades. Diese sorgen für einen glatten Übergang von der Anfangs- bzw. Endgeschwindigkeit auf die Verfahrgeschwindigkeit. Zu ermitteln sind die Schnittpunkte zwischen der ersten und zweiten sowie der zweiten und dritten Sektion. Die Punkte werden so gewählt, dass sich für die beiden Polynome kein Wendepunkt einstellt

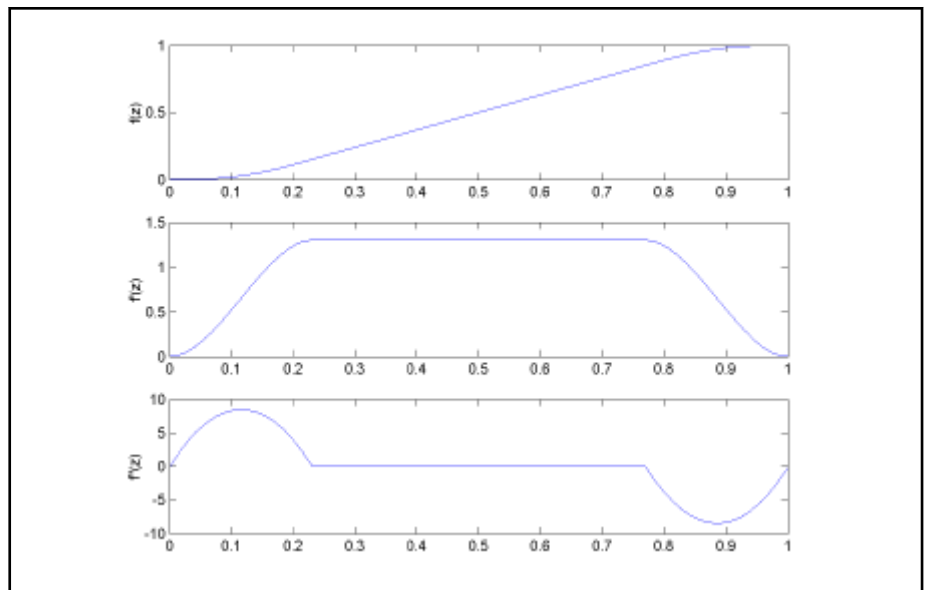


Abb. 3-35: Geschwindigkeitsbegrenztes Polynom 5. Ordnung

Das Profil für **"Überschwingungsfreies Polynom 5. Ordnung"** kann wie folgt beschrieben werden:

Das hier gezeigte Bewegungsgesetz erkennt und verhindert die mögliche Polynom-Überschwingung. Wie diese verändert wird, hängt dabei von der Kombination der normierten Geschwindigkeitsrandwerte ab. Es ergeben sich vier Fälle. Für Kombinationen in denen die normierte Anfangsgeschwindigkeit "v0" und die normierte Endgeschwindigkeit "v1" unterschiedliche Vorzeichen haben, kann dieses Bewegungsgesetz nicht verwendet werden. Dies würde bedeuten, dass die Bewegungsbahn in einem Randpunkt einen Knick hätte, wenn sie den normierten Bereich von $0 \leq z \leq 1$ und $0 \leq f(z) \leq 1$ nicht verlassen dürfte.

- Fall 1:
 $V_0 \leq 1$ und $V_1 \leq 1$

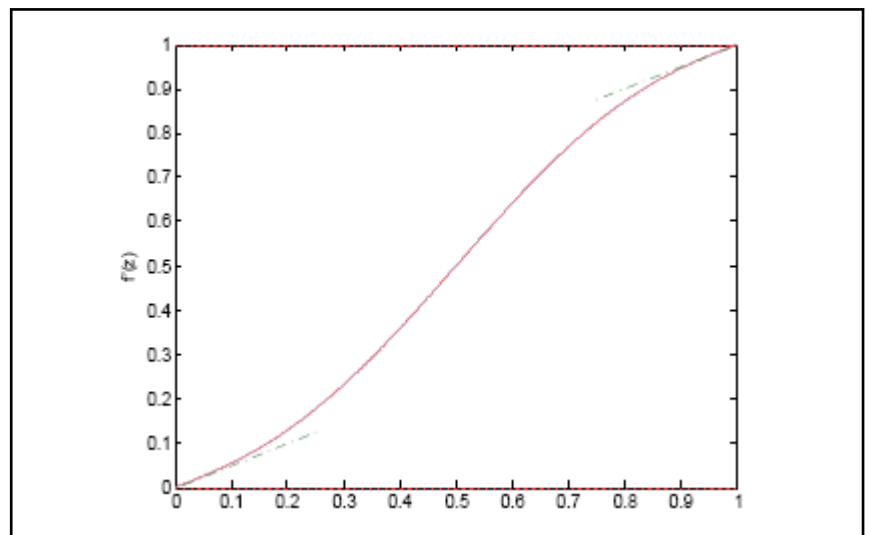


Abb. 3-36: Fall 1

- Fall 2:
 $V_0 \leq 1$ und $V_1 > 1$

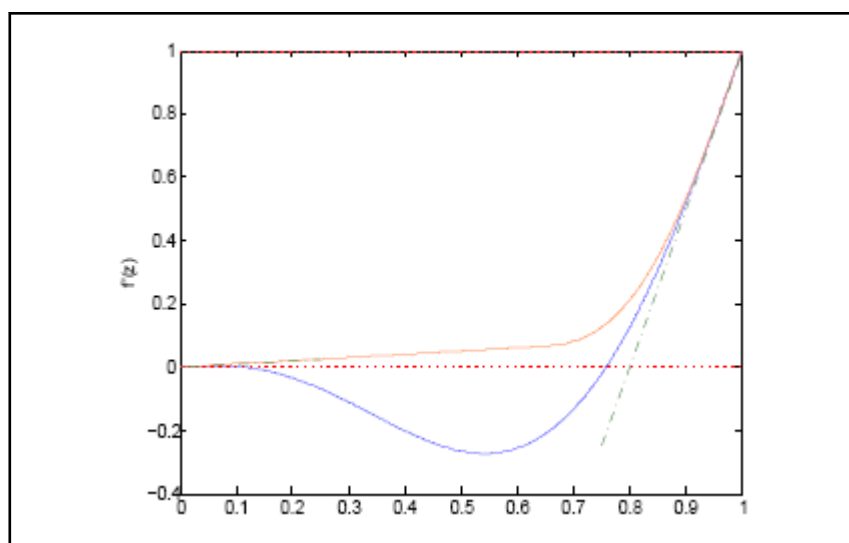


Abb. 3-37: Fall 2

- Fall 3:
 $V_0 > 1$ und $V_1 \leq 1$

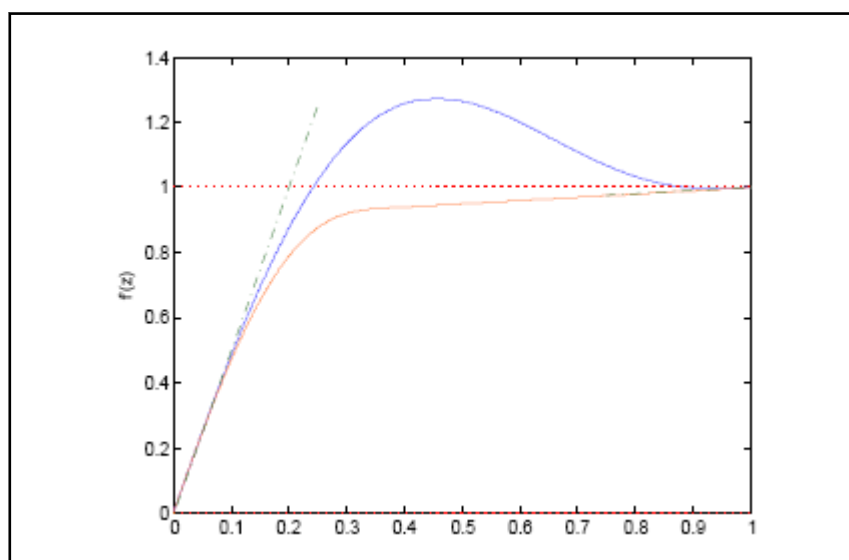


Abb. 3-38: Fall 3

- Fall 4:
 $V_0 > 1$ und $V_1 > 1$

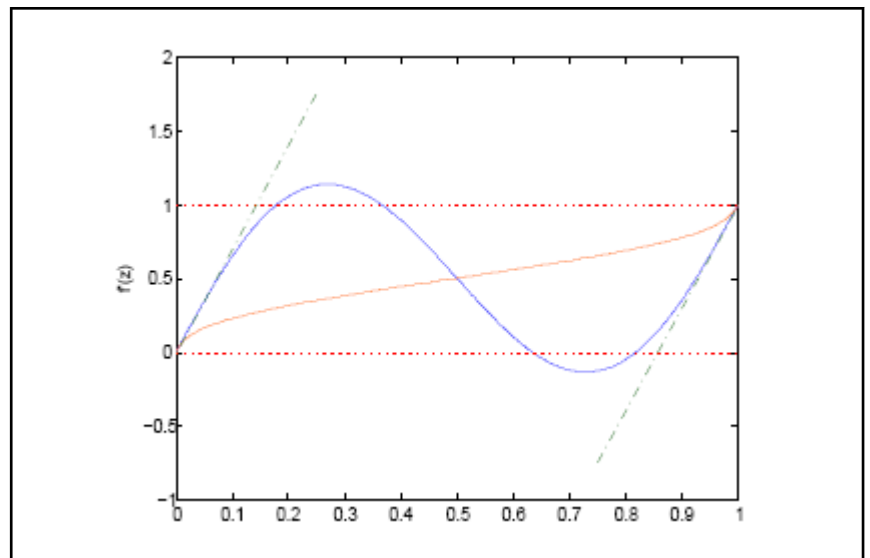


Abb. 3-39: Fall 4

Das Profil für die **"Beschleunigungsbegrenzte Bewegung"** kann wie folgt beschrieben werden:

Als einfachste Art der Bahninterpolation kann das Trapezprofil, oft auch Rampenprofil genannt, gesehen werden. Soll eine Wegstrecke "s" verfahren werden, so lässt sich hierbei der Bahnverlauf in drei Phasen aufteilen. In eine erste Phase, in der mit einem vorgegebenen Beschleunigungswert auf eine vorgegebene Verfahrgeschwindigkeit beschleunigt wird, in eine zweite Phase konstanter Geschwindigkeit und schließlich in eine Bremsphase in der ebenfalls mit einer angegebenen Verzögerung wieder abgebremst wird. Aufgabe der Bahninterpolation ist es, aus diesen vorgegebenen Beschleunigungswerten a_1 und a_3 sowie der gewünschten Geschwindigkeit v_m und der zurückzulegenden Wegstrecke "s", die Start- und Endzeitpunkte t_1 , t_2 und t_3 bzw. die einzelnen Verfahrszeiten T_1 , T_2 und T_3 zu berechnen. In der hier vorgestellten Variante wird außerdem auch eine geforderte Startgeschwindigkeit v_0 sowie Endgeschwindigkeit v_3 berücksichtigt.

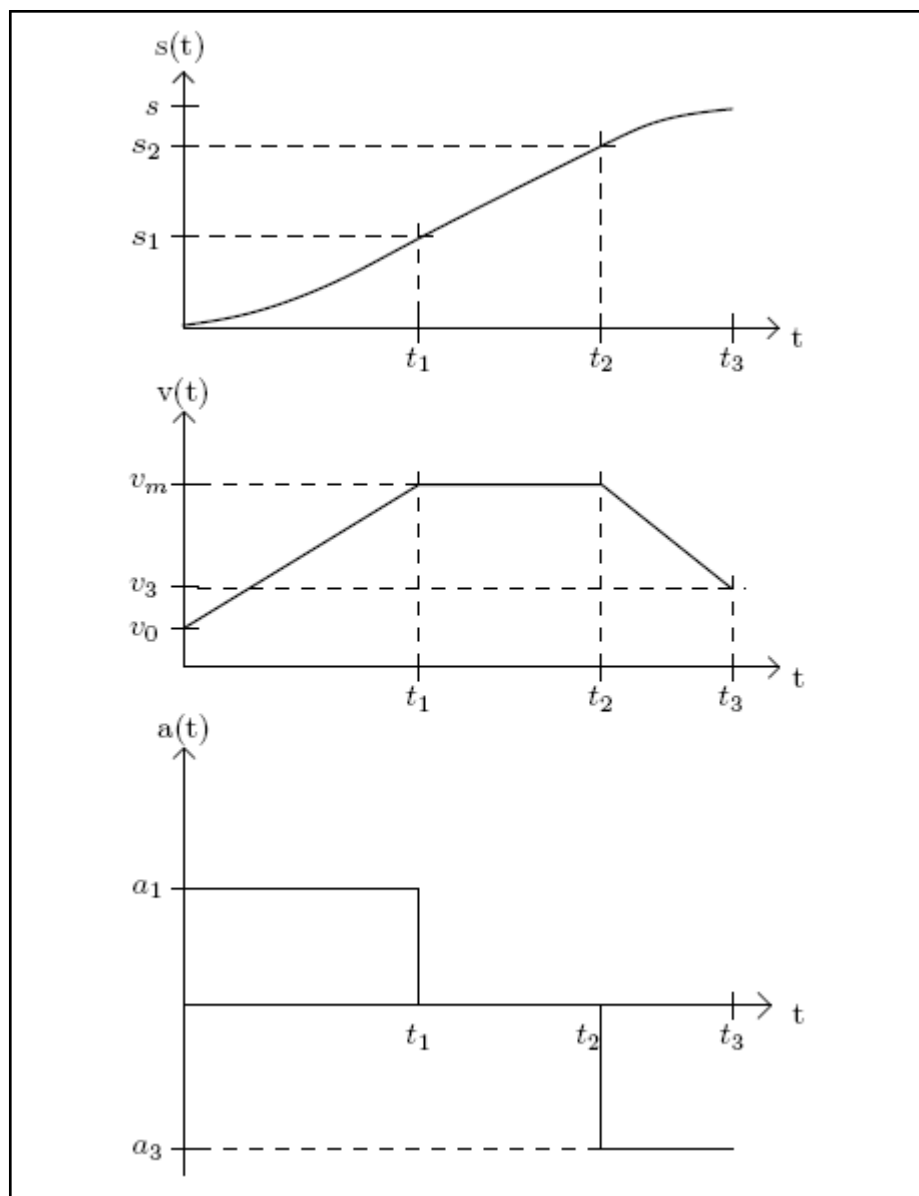


Abb. 3-40: Beschleunigungsbegrenzte Bewegung

Die möglichen Randwerte für das Bewegungsgesetz sind:

Randwert	z=0	z=1
1. Ordnung (Geschwindigkeit)	wählbar	wählbar
2. Ordnung (Beschleunigung)	automatisch	automatisch
3. Ordnung (Ruck)	automatisch	automatisch

Tab. 3-29: Randwerte für Bewegungsgesetz - Beschleunigungsbegrenzte Bewegung

Das Profil für die **"Beschleunigungsbegrenzte Bewegung mit Sinoidenprofil"** kann wie folgt beschrieben werden:

Beim Sinoidenprofil kann die Start- und Endgeschwindigkeiten ungleich 0 sein, sowie unterschiedliche Beschleunigungsgrößen a_1 und a_3 für die Beschleunigungs- und Bremsphase haben. Weiterhin werden auch die Sonderfälle berücksichtigt, in denen die Startgeschwindigkeit v_0 bereits größer als

die gewünschte Verfahrgeschwindigkeit v_m ist. Gleiches gilt auch für die Endgeschwindigkeit v_3 .

Die möglichen Randwerte für das Bewegungsgesetz sind:

Randwert	$z=0$	$z=1$
1. Ordnung (Geschwindigkeit)	wählbar	wählbar
2. Ordnung (Beschleunigung)	automatisch	automatisch
3. Ordnung (Ruck)	automatisch	automatisch

Tab. 3-30: Randwerte für Bewegungsgesetz - Beschleunigungsbegrenzte Bewegung mit Sinoidenprofil

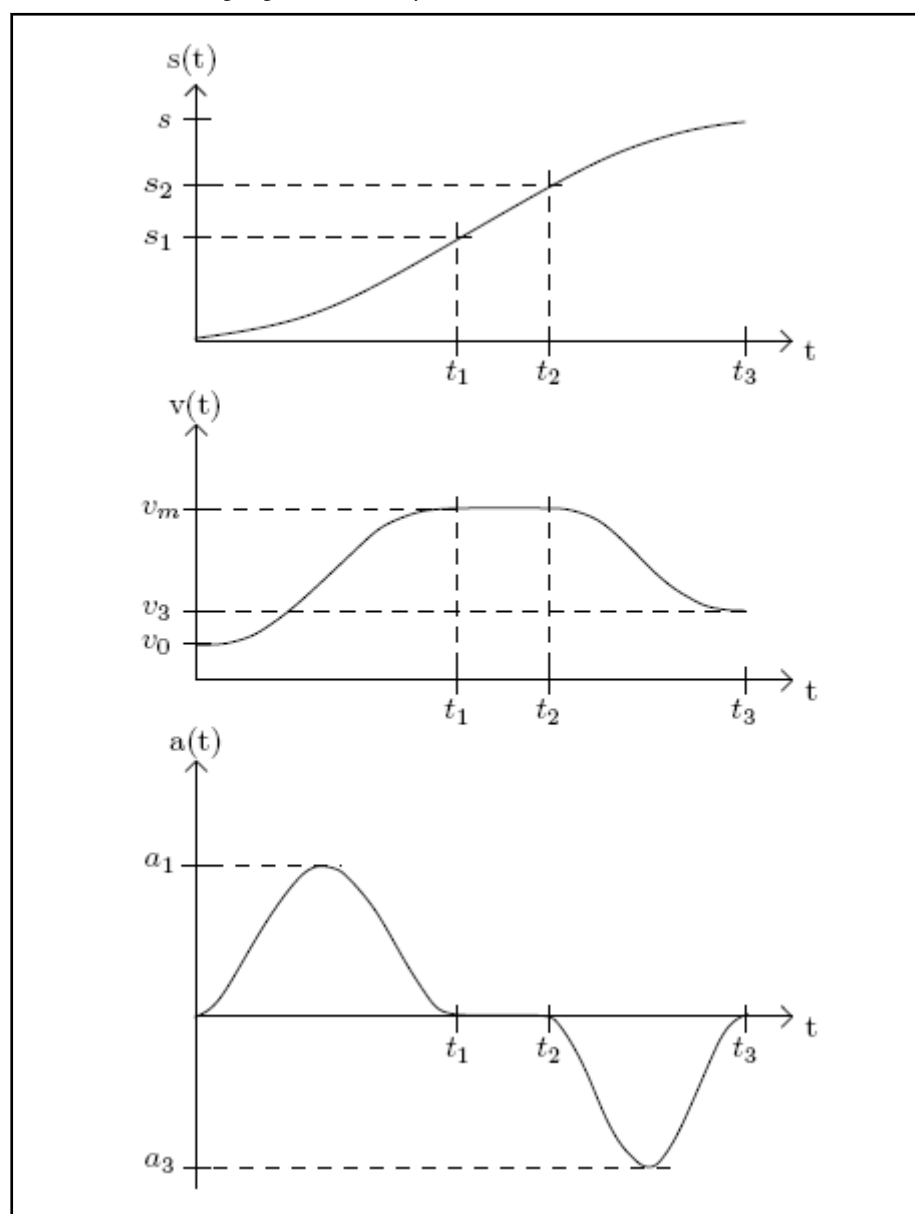


Abb. 3-41: Beschleunigungsbegrenzte Bewegung mit Sinoidenprofil

Das Profil für die **"Ruckbegrenzte Bewegung"** kann wie folgt beschrieben werden:

Für viele anspruchsvollere Bewegungsaufgaben gilt die Einhaltung einer Beschleunigung mit Ruckbegrenzung. Beispielsweise für den Transport von Flüssigkeiten oder aber auch bei leicht schwingungsfähigen Maschinen.

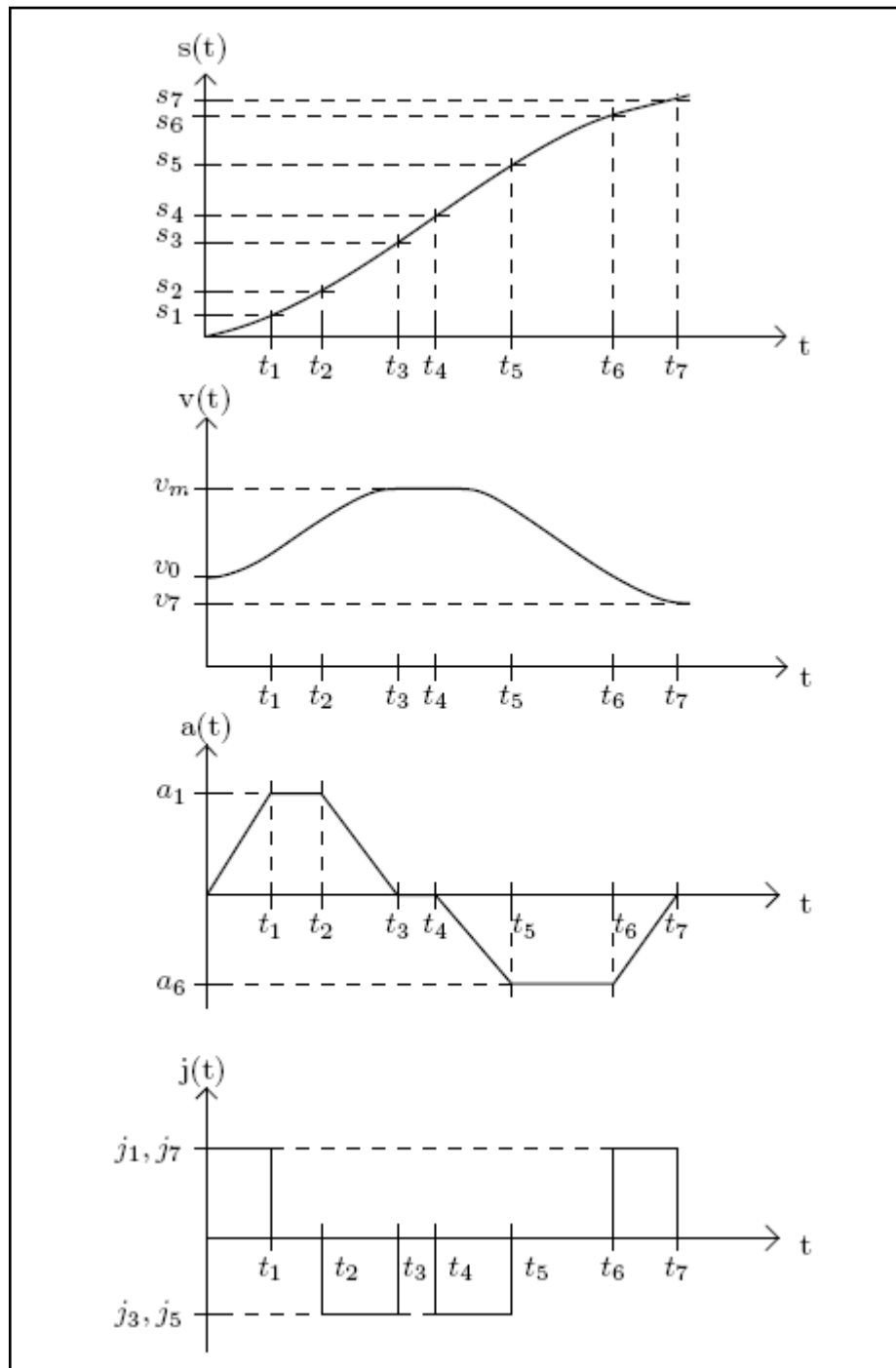


Abb. 3-42: Ruckbegrenzte Bewegung

3.5 Datentypen zum Aufsynchronisieren in den lagegeregelten Synchron-Betriebsarten

3.5.1 Synchronisation, Einführung und Übersicht

Die im folgenden beschriebenen Funktionen sind in den Betriebsarten Winkelsynchronisation und Elektronische Kurvenscheibe wirksam. Generell wird das Aufsynchronisieren nach dem Zuschalten der Synchron-Betriebsart durchgeführt. Dieser Aufsynchronisiervorgang ist in zwei Schritte unterteilt:

1. Zunächst beschleunigt bzw. verzögert der Folgeantrieb mit der "Aufsynchronisierbeschleunigung" auf die synchrone Geschwindigkeit.
2. Im zweiten Schritt wird die Positionsdifferenz abgebaut. Dazu wird der synchronen Geschwindigkeit eine "Aufsynchronisiergeschwindigkeit" überlagert.

Bezüglich der Lageanpassung im zweiten Schritt wird zwischen relativer und absoluter Synchronisation ([MC_START_MODE](#), Seite 51) unterschieden:

- Bei relativer Synchronisation ist das Aufsynchronisieren mit dem Erreichen der synchronen Geschwindigkeit abgeschlossen.
- Nur bei absoluter Synchronisation wird nach dem Erreichen der synchronen Geschwindigkeit der absolute Winkelbezug hergestellt (Aufsynchronisierichtung, [MC_SYNC_DIRECTION](#), Seite 53).

3.5.2 MC_START_MODE

Kurzbeschreibung

Dieser Parameter gibt die Art der automatischen dynamischen Synchronisation bei Phasensynchronisation an.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek / Ordner
MLC 15VRS	RIL_CommonTypes / Datatypes of PLCopen
Grundpaket PLC	

Tab. 3-31: Referenztable MC_START_MODE

Typdefinition

```

TYPE MC_START_MODE :
(
  ABSOLUTE      := 0, (* synchronization with absolute angle reference, *)
                    (* only for first design *)
  RELATIVE      := 1, (* synchronization to synchronous velocity, *)
                    (* only for first design *)
  ABSOLUTE_RAMPIN := 2, (* synchronization with absolute angle reference, *)
                    (* for every design *)
  RELATIVE_RAMPIN := 3, (* synchronization to synchronous velocity, *)
                    (* for every design *)
);
END_TYPE

```



Für das Kommando MB_GearInPos gelten nur die beiden Startmodi ABSOLUT und RELATIV.

Absolute Synchronisation

Die Folgeachseposition beim Nulldurchgang der Leitachseposition ist eindeutig definiert und ändert sich nicht bei jedem Aufruf von MB_MotionProfile/ MC_CamIn.

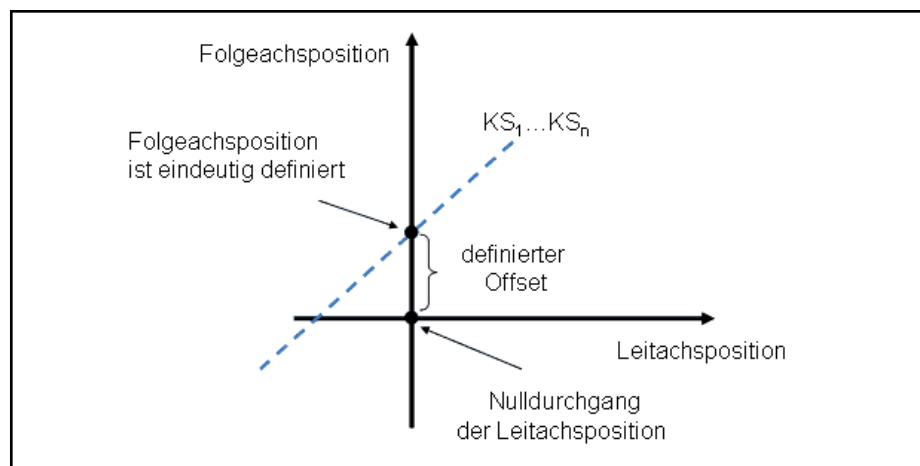


Abb. 3-43: Absolute Synchronisation

Relative Synchronisation

Die Folgeachseposition beim Nulldurchgang der Leitachseposition ergibt sich zufällig und ändert sich bei jedem Aufruf von MB_MotionProfile/ MC_CamIn.

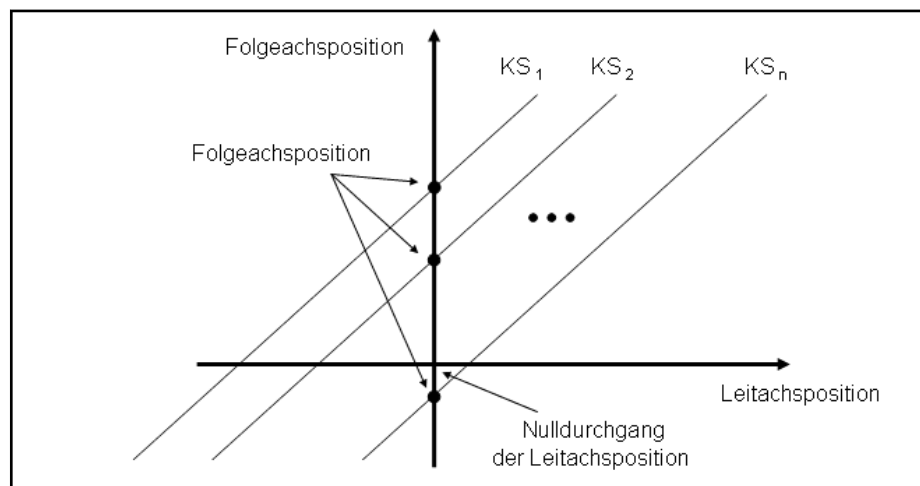


Abb. 3-44: Relative Synchronisation

"Harte" Synchronisation

- StartMode= 0: absolute Synchronisation, "hartes Umschalten" bei Profilwechsel,
- StartMode= 1: relative Synchronisation, "hartes Umschalten" bei Profilwechsel.

Die Umschaltung erfolgt nach Überfahren der Leitachs-Umschaltposition (A-0-2900, MotionProfile, Leitachs-Umschaltposition) durch die Leitachse.

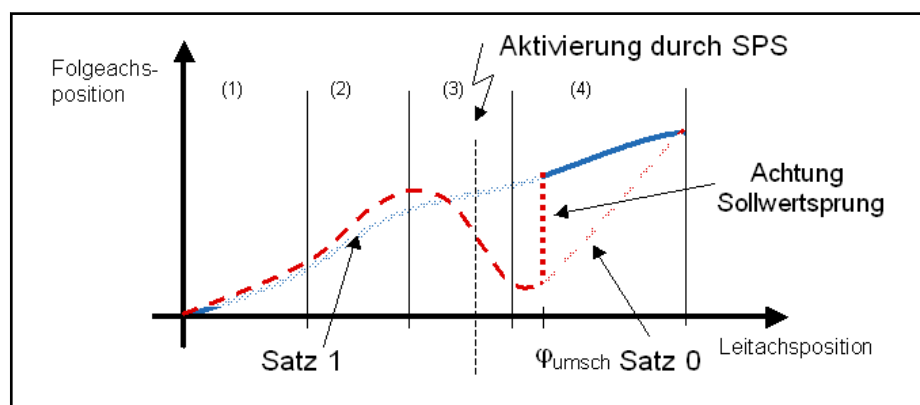


Abb. 3-45: Hartes Umschalten

"Weiche" Synchronisation

- StartMode= 2: absolute Synchronisation, "weiches Umschalten" bei Profilwechsel,
- StartMode= 3: relative Synchronisation, "weiches Umschalten" bei Profilwechsel.

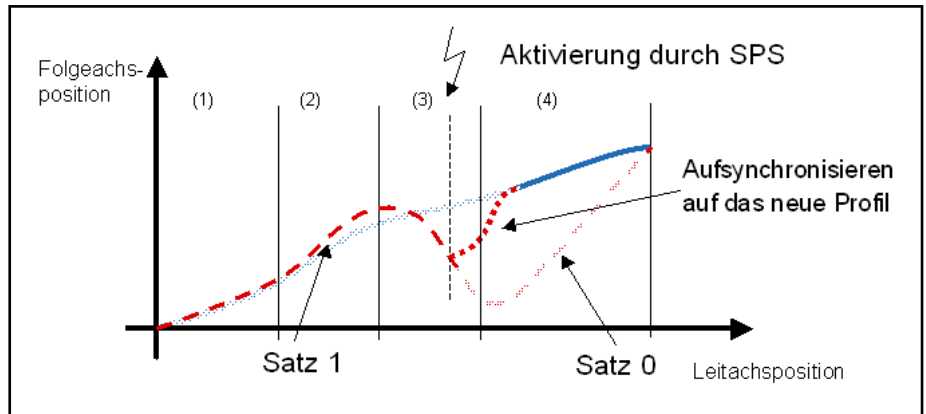


Abb. 3-46: Weiches Umschalten

Die Umschaltung erfolgt sofort nach Starten des FBs, es wird nicht auf das Überfahren der Leitachs-Umschaltposition (A-0-2900, MotionProfile, Leitachs-Umschaltposition) durch die Leitachse gewartet.



Das "Weiche Aufsynchronisieren" setzt die Berechnung auf der Steuerung voraus.

3.5.3 MC_SYNC_DIRECTION

Kurzbeschreibung

Bei Modulo-Achsen kann die Bewegung der Achse in positiver oder negativer Richtung erfolgen.

Durch die Aufsynchronisierrichtung wird festgelegt, in welche Richtung sich die Achse bewegen soll.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek / Ordner
MLC 15VRS Grundpaket PLC	RIL_CommonTypes / Datatypes of PLCopen

Tab. 3-32: Referenztable MC_SYNC_DIRECTION

Typdefinition

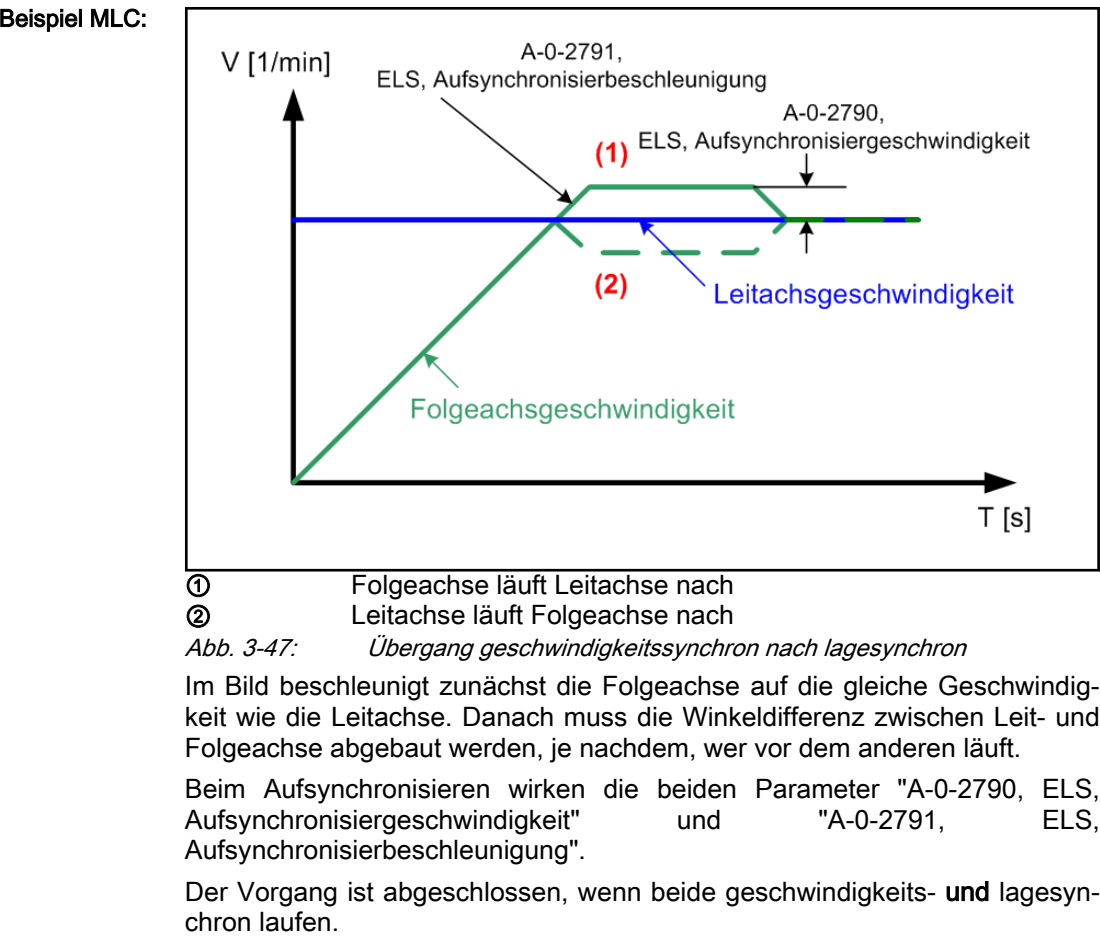
```

TYPE MC_SYNC_DIRECTION :
(
  SHORTEST_WAY    := 0,  (* kürzester Weg /shortest distance *)
  CATCH_UP        := 1,  (* immer vorwärts / always forward *)
  SLOW_DOWN       := 2
);
END_TYPE

```



Ist der kürzeste Weg zur absoluten Synchronisation jedoch kleiner als das "Aufsynchronisierfenster", bei Modulo-Format, wird der kürzeste Weg verfahren und die vorgegebene Aufsynchronisierrichtung ignoriert.



3.6

Datentypen zur Fehlerdiagnose bei Motion-Funktionsbausteinen

3.6.1

ERROR_CODE

Kurzbeschreibung Die Aufzählung enthält die Fehlerarten, die einen Fehler am Funktionsbaustein charakterisieren.

Der Errorcode soll dem Anwender einen schnellen Hinweis auf die Fehlerursache geben.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek / Ordner
MLC 15VRS	RIL_CommonTypes / Datatypes of POU
Grundpaket PLC	diagnosis

Tab. 3-33: Referenztabelle ERROR_CODE

Typdefinition

```
(* Enumeration of valid types of diagnosis/ Aufzählung gültiger Fehlerarten *)
TYPE ERROR_CODE :
(
  NONE_ERROR           := 16#0000, (* No error code available *)
  INPUT_INVALID_ERROR  := 16#0001, (* kein Fehlercode verfügbar, FB fehlerfrei *)
  COMMUNICATION_ERROR  := 16#0002, (* Invalid input *)
                          (* Fehler auf Grund ungültiger Eingabewerte *)
                          (* Error while communication *)
                          (* Fehler beim Aufbau einer Kommunikation *)
                          (* bzw.während der Kommunikation *)
)
```

```

RESOURCE_ERROR      := 16#0003, (* Resource error *)
ACCESS_ERROR        := 16#0004, (* Ressource (z.B.Parameter) ist nicht vorhanden *)
STATE_MACHINE_ERROR := 16#0005, (* Access error *)
INPUT_RANGE_ERROR   := 16#0006, (* Ressource ist zwar da, aber man kann nicht *)
CALCULATION_ERROR    := 16#0007, (* auf sie zugreifen. *)
DEVICE_ERROR         := 16#0008, (* Beispiel: Man möchte Parameter in Phase 4 be- *)
OTHER_ERROR          := 16#7FFE, (* schreiben, der nur in Phase 2 beschreibbar ist*)
SYSTEM_ERROR         := 16#7FFF (* Invalid state State - Machine *)
                        (* Der aktuelle Zustand der State-Machine *)
                        (* verhindert die weitere Abarbeitung *)
                        (* Error because Input out of range *)
                        (* Während der Verarbeitung wird festgestellt, *)
                        (* dass durch eine oder mehrere Eingangsgrößen *)
                        (* undefinierte bzw. zu große oder zu kleine *)
                        (* Sollwerte generiert werden. *)
                        (* Error while calculation *)
                        (* Ein Berechnungsfehler ist aufgetreten, *)
                        (* z.B. Division durch 0 *)
                        (* Device Error occurred *)
                        (* Übergeordneter Gerätefehler ist aufgetreten, *)
                        (* z.B IndraDrive: *)
                        (* Kurvenscheibenbetriebsart F2005, *)
                        (* Ungültige Kurvenscheibe *)
                        (* Undefined Error *)
                        (* allgemeiner (undefinierter) Fehler *)
                        (* System error occured *)
                        (* z.B. Betriebssystemfehler *)
);
END_TYPE

```

3.6.2 ERROR_STRUCT

Kurzbeschreibung Die Fehlerstruktur beinhaltet alle Detailinformationen zu einem konkreten Fehler.

Die Information ist an der jeweiligen Instanz eines PLCopen-Funktionsbausteines direkt auswertbar und ablesbar. Die Interpretation der Bestandteile muss wie nachstehend beschrieben erfolgen:

1. Ermitteln der Tabellennummer ([ERROR_TABLE](#), Seite 56). Damit steht die Herkunft des Fehlers fest.
2. Mit dieser Kenntnis wird "Additional1" ausgewertet. Der Inhalt ist die Diagnosenummer, die in den jeweiligen Bausteinbeschreibungen zu finden ist (siehe auch Hinweise zur Störungsbeseitigung, MLC). Kontrolle, ob "Additional2" = 0 ist, wenn ja, ist die Diagnose komplett, wenn nein, gehe weiter.
3. Auswertung von "Additional2" (bei MLC_Table bisher nicht genutzt).

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek / Ordner
MLC 15VRS	RIL_CommonTypes / Datatypes of POU
Grundpaket PLC	diagnosis, Includes

Tab. 3-34: Referenztable ERROR_STRUCT

Typdefinition

```

(* Structure for description of diagnosis of function blocks. *)
(* The ERROR_STRUCT is a structure that contains information on the *)
(* corresponding diagnosis, which is market by the PLCopen output 'Error' *)

TYPE ERROR_STRUCT :
STRUCT
  Table      : ERROR_TABLE; (* Additional diagnosis table *)
  Additional1 : DWORD;       (* links to list used for identification *)
  Additional2 : DWORD;       (* of the error of "Additional1" *)
                        (* Additional diagnosis number1 *)
                        (* Contains error number according to the *)
                        (* specified table. *)
                        (* Additional diagnosis number2 *)
                        (* Can contain additions to "Additional1". *)

```

END_STRUCT
END_TYPE



In **Additional1** werden die in den "MLC, Diagnosen - Hinweise zur Störungsbeseitigung" beschriebenen Meldungen, Warnungen und Fehlermeldungen angezeigt.

3.6.3 ERROR_TABLE

Kurzbeschreibung

Die ERROR_TABLE liefert das erste Element der [ERROR_STRUCT](#), Seite 55. Anhand der Nummer ist die Tabelle auszuwählen, in der der "Additional1"- Fehlercode gesucht werden muß.

Beispiel:

Für MLC-Anwendungen sollte dies z.B. die Tabelle "MLC_TABLE", 16#0030, sein.

Die in der jeweiligen Baustein-Beschreibung enthaltenen Fehlernummern unter "Additional1" beziehen ihre Informationen aus dieser Tabelle.

Bei Fehlern im Zugriff auf einen Parameter können Bausteine die Tabelle SERCOS_TABLE, 16#0010, verwenden.

Bei Fehlern in der Kommando-Abarbeitung können Bausteine die Tabellen F_RELATED_TABLE, 16#0170, oder INDRV_TABLE, 16#0070, verwenden.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek / Ordner
MLC 15VRS Grundpaket PLC	RIL_CommonTypes / Datatypes of POU diagnosis, Includes

Tab. 3-35: Referenztable ERROR_TABLE

Typdefinition

```
(* Enumeration of valid tables in additional information *)
TYPE ERROR_TABLE:
(
  NO_TABLE_USED      := 16#0000, (* reserved *)
  SERCOS_TABLE       := 16#0010, (* Sercos Error: Additional1 = Sercos Code *)
  MLD_TABLE          := 16#0020, (* Drive-based Motion - Logic *)
  MLC_TABLE          := 16#0030, (* Controller - based Motion - Logic *)
  MTX_TABLE          := 16#0040, (* CNC *)
  MLP_TABLE          := 16#0050, (* PC - based Motion Logic *)
  PLC_TABLE          := 16#0060, (* PLC *)
  INDRV_TABLE        := 16#0070, (* IndraDrive *)
  DIAX_TABLE         := 16#0080, (* DIAX *)
  ECO_TABLE          := 16#0090, (* EcoDrive *)
  PB_DP_TABLE        := 16#0130, (* Profibus *)
  DEVICENET_TABLE    := 16#0140, (* DeviceNet *)
  ETHERNET_TABLE     := 16#0150, (* Ethernet *)
  ETHERNET_IP_TABLE  := 16#0151, (* Ethernet/IP ODVA *)
  INTERBUS_TABLE     := 16#0160, (* Interbus *)
  F_RELATED_TABLE    := 16#0170, (* function related *)
  CAN_TABLE          := 16#0180, (* CAN bus *)
  INLINEIO_TABLE     := 16#0190, (* Inline IOs *)
  PROFINETIO_TABLE   := 16#0200, (* ProfinetIO, ProfinetIO Device *)

  USER1_TABLE        := 16#1000, (* free User Table *)
  ...
  USER10_TABLE       := 16#1009 (* free User Table *)
);
END_TYPE
```


3.7 Datentypen zur Wiedergabe des Datum und absoluter Zeitangaben bei Motion-Funktionsbausteinen

3.7.1 Wiedergabe des Datum und absoluter Zeitangaben, Einführung und Übersicht

Zur Wiedergabe des Datum und anderer absoluter Zeitangaben existieren drei Strukturen:

- [SYS_TIME64, Seite 57](#), Systemzeit in Mikrosekunden seit 01.01.1970, 00:00 Uhr.
- [EXT_SYS_TIME_DATE, Seite 57](#), Systemzeit in Mikrosekunden seit 01.01.1970, 00:00 Uhr, Zusatzdaten Jahr, Monat, Tag, Stunden, Minuten, Sekunden, Millisekunden und Wochentag werden aus der Systemzeit berechnet.
- [EXT_SYS_TIME_DATE, Seite 57](#), Jahr, Monat, Tag, Stunden, Minuten, Sekunden, Millisekunden und Wochentag, berechnet auf der Basis der Systemzeit in Mikrosekunden seit 01.01.1970, 00:00 Uhr.

3.7.2 SYS_TIME64

Kurzbeschreibung Struktur zur Wiedergabe der Systemzeit in Mikrosekunden seit 01.01.1970, 00:00 Uhr.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek / Ordner
MLC 15VRS Grundpaket PLC	RIL_CommonTypes / Datatypes of times

Tab. 3-36: Referenztablette SYS_TIME64

Typdefinition

```
(* Structure of system time - Microseconds since 1970 *)
TYPE SYS_TIME64 :
STRUCT
  LowValue:    DWORD;    (* Low value [us]      *)
  HighValue:   DWORD;    (* High value [us]   *)
END_STRUCT
END_TYPE
```

3.7.3 EXT_SYS_TIME_DATE

Kurzbeschreibung Struktur zur Wiedergabe der Systemzeit in Mikrosekunden seit 01.01.1970, 00:00 Uhr und der Zusatzdaten Jahr, Monat, Tag, Stunden, Minuten, Sekunden, Millisekunden und Wochentag, die aus der Systemzeit berechnet werden.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek / Ordner
MLC 15VRS Grundpaket PLC	RIL_CommonTypes / Datatypes of times

Tab. 3-37: Referenztablette EXT_SYS_TIME_DATE

Typdefinition

```
(* Extended structure of system time *)
TYPE EXT_SYS_TIME_DATE :
STRUCT
  LowValue: DWORD;    (* Low value [us]      *)
```

```

HighValue: DWORD;          (* High value [us]      *)
Year : UINT;               (* Year              *)
Month : UINT;              (* Month             *)
Day : UINT;                (* Day               *)
Hour : UINT;               (* Hour              *)
Minute : UINT;             (* Minute            *)
Second : UINT;             (* Second            *)
Milliseconds : UINT;       (* Milliseconds      *)
DayOfWeek: UINT;           (* Day of week       *)
END_STRUCT
END_TYPE

```

3.7.4 SYS_TIME_DATE

Kurzbeschreibung Struktur zur Wiedergabe der Daten Jahr, Monat, Tag, Stunden, Minuten, Sekunden, Millisekunden und Wochentag, die aus einer Systemzeit in Mikrosekunden seit 01.01.1970, 00:00 Uhr berechnet werden.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek / Ordner
MLC 15VRS Grundpaket PLC	RIL_CommonTypes / Datatypes of times

Tab. 3-38: Referenztable SYS_TIME_DATE

Typdefinition

```

(* Structure of system time *)
TYPE SYS_TIME_DATE :
STRUCT
    Year : UINT;          (* Year              *)
    Month : UINT;         (* Month             *)
    Day : UINT;           (* Day               *)
    Hour : UINT;          (* Hour              *)
    Minute : UINT;        (* Minute            *)
    Second : UINT;        (* Second            *)
    Milliseconds : UINT;  (* Milliseconds      *)
    DayOfWeek: UINT;      (* Day of week       *)
END_STRUCT
END_TYPE

```

3.8 Datentypen zur Unterstützung von Messtaster-Funktionen

3.8.1 Datentypen zur Unterstützung von Messtaster-Funktionen, allgemeines

Der Abschnitt enthält Datentypen, die die Messtaster-Funktionen sowohl bei Abarbeitung im Antrieb als auch in der Steuerung unterstützen:

- [MB_DIFF_EVAL](#), Seite 59, Einstellung der Methoden der Differenzmessung
- [MB_EXPECT_WINDOW_MODE](#), Seite 60, Konfiguration und Aktivierung des Erwartungsfensters
- [MB_INTERPOLATION_MODE](#), Seite 60, Aktivierung der Interpolation
- [MB_POSITION_EVAL](#), Seite 61, Auswahl und Aktivierung des Triggersignals
- [MB_PROBE_MODULE](#), Seite 61, Auswahl der Triggerquelle für den Messtaster
- [MB_PROBE_INPUT](#), Seite 61, Auswahl des gewünschten Einganges auf dem ausgewählten Modul
- [MB_PROBE_DATA](#), Seite 62, Auswahl der Messtaster-Bezugssignalquelle

Die zu einer Messtaster-Funktion gehörenden Daten sind in nachstehender Struktur zusammengestellt:

- [MB_PROBE_CONFIG](#), Seite 63

3.8.2 MB_DIFF_EVAL

Kurzbeschreibung Der Aufzählungstyp "MB_DIFF_EVAL" unterstützt die Einstellung der Methoden der Differenzmessung.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek / Ordner
MLC 15VRS	RIL_CommonTypes / TouchProbe
Grundpaket PLC	

Tab. 3-39: Referenztable MB_DIFF_EVAL

Typdefinition

```

TYPE MB_DIFF_EVAL :
(
  DIFF_DISABLED      := 0, (* diff.measurement is disabled *)
  DIFF_STANDARD_MODE := 1, (* diff. last two measured values*)
  DIFF_NEG_TO_NEG_EDGE:= 2, (* neg. edge to neg. edge diff. *)
  DIFF_POS_TO_NEG_EDGE:= 3, (* pos. edge to neg. edge diff. *)
  DIFF_NEG_TO_POS_EDGE:= 4, (* neg. edge to pos. edge diff. *)
  DIFF_POS_TO_POS_EDGE:= 5 (* pos. edge to pos. edge diff. *)
);
END_TYPE

```

Funktionsbeschreibung

Folgende Einstellungen werden unterstützt:

- 0 - Differenzmessung ist abgeschaltet, Differenz-Signale für ML_TouchProbe bzw. ML_TouchProbe_Continuous werden nicht berechnet.
- 1 - Differenz zwischen den letzten zwei Messwerten, je nach aktivierter Flanke.

Beispiel: beide Flanken sind aktiviert, erfasst wird Differenz pos/neg Flanke, weiter neg/pos Flanke, weiter pos/neg Flanke.

- Im nachstehenden Bild sind die erweiterten Methoden 2..5 der Differenzmessung dargestellt:

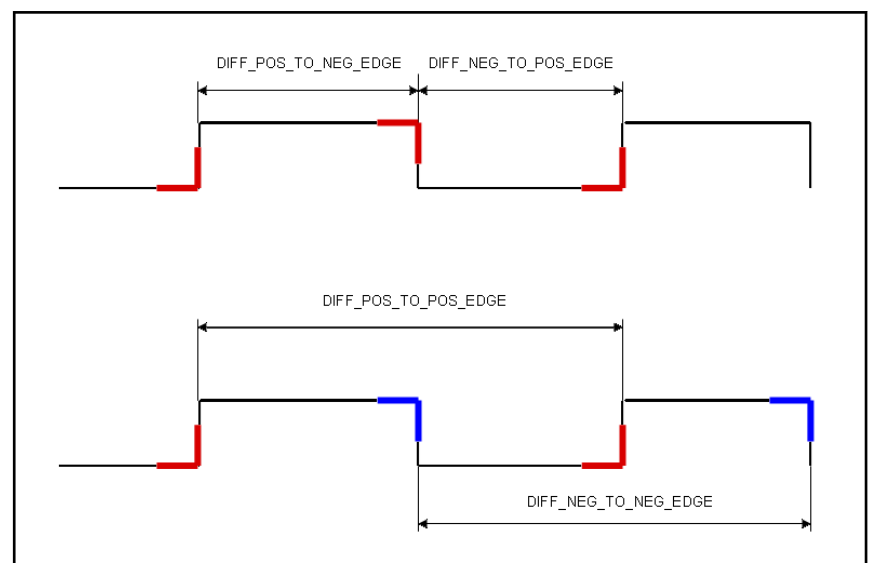


Abb. 3-48: Methoden der Differenzmessung

3.8.3 MB_EXPECT_WINDOW_MODE

Kurzbeschreibung Der Aufzählungstyp "MB_EXPECT_WINDOW_MODE" unterstützt die Konfiguration und Aktivierung des Erwartungsfensters.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek / Ordner
MLC 15VRS Grundpaket PLC	RIL_CommonTypes / TouchProbe

Tab. 3-40: Referenztable MB_EXPECT_WINDOW_MODE

Typdefinition

```

TYPE MB_EXPECT_WINDOW_MODE :
(
  EXPECT_WINDOW_DISABLED:= 0, (* expectation window disabled *)
  EXPECT_WINDOW_ABS      := 1, (* expectation window absolute *)
  EXPECT_WINDOW_REL      := 2, (* expectation window relative *)
  (* to "ExpectWindowSetPoint" *)
);
END_TYPE

```

Funktionsbeschreibung Der Anwender kann sich getrennt für die positive und die negative Flanke des Messtasterwertes für ein Erwartungsfenster entscheiden.

Entscheidet sich der Anwender für die Nutzung eines Erwartungsfensters um , so kann er dessen Grenzen auf zwei unterschiedliche Arten definieren:

1. Erwartungsfenster deaktiviert:
Die jeweilige Flanke des Messtastersignals wird ungefiltert verwendet.
2. Erwartungsfenster absolut definiert:
Die Startposition und die Endposition des Erwartungsfensters sind absolut anzugeben. Der Mittelpunktswert ist in diesem Fall bedeutungslos.
3. Erwartungsfenster relativ definiert:
Der Mittelpunktswert des Erwartungsfensters sind absolut anzugeben. Die Startposition und die Endposition geben die Entfernung zum Mittelpunktswert an.

3.8.4 MB_INTERPOLATION_MODE

Kurzbeschreibung Der Aufzählungstyp "MB_INTERPOLATION_MODE" unterstützt die Aktivierung der Interpolation.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek / Ordner
MLC 15VRS Grundpaket PLC	RIL_CommonTypes / TouchProbe

Tab. 3-41: Referenztable MB_INTERPOLATION_MODE

Typdefinition

```

TYPE MB_INTERPOLATION_MODE:
(
  NO_INTERPOLATION      := 0, (* no interpolation *)
  LINEAR_INTERPOLATION  := 1, (* linear interpolation *)
  CUBIC_3_INTERPOLATION := 2, (* cubic interpolation with 3 setpoints *)
  CUBIC_4_INTERPOLATION := 3, (* cubic interpolation with 4 setpoints *)
);
END_TYPE

```

3.8.5 MB_POSITION_EVAL

Kurzbeschreibung Der Aufzählungstyp "MB_POSITION_EVAL" unterstützt die Auswahl und Aktivierung des Triggersignals.

Mit diesen Enumerationswerten kann ausgewählt, ob auf die positive oder die negative Flanke bzw. auf beiden Flanken getriggert werden soll.



Der Defaultwert "Null" blockiert die Triggierung!

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek / Ordner
MLC 15VRS	RIL_CommonTypes / TouchProbe
Grundpaket PLC	

Tab. 3-42: Referenztable MB_POSITION_EVAL

Typdefinition

```

TYPE MB_POSITION_EVAL :
(
  POSITION_EVAL_DISABLED:= 0, (* disabled *)
  POSITION_POS_EDGE      := 1, (* positive edge is evaluated *)
  POSITION_NEG_EDGE      := 2, (* negative edge is evaluated *)
  POSITION_POS_NEG_EDGE := 3 (* pos.and neg. edge are evaluated *)
);
END_TYPE

```

3.8.6 MB_PROBE_MODULE

Kurzbeschreibung Der Aufzählungstyp "MB_PROBE_MODULE" unterstützt die Auswahl der Triggerquelle für den Messtaster.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek / Ordner
MLC 15VRS	RIL_CommonTypes / TouchProbe
Grundpaket PLC	

Tab. 3-43: Referenztable MB_PROBE_MODULE

Typdefinition

```

TYPE MB_PROBE_MODULE :
(
  ONBOARD_MODULE := 0, (* Onboard module MLC, Inputs 1...8 *)
  FUNCTION_MODULE_1:= 1, (* Function module 1 MLC, Inputs 1..16 (CFL-01.1-E2, Fast IO) *)
  FUNCTION_MODULE_2:= 2, (* Function module 2 MLC, Inputs 1..16 (CFL-01.1-E2, Fast IO) *)
  FUNCTION_MODULE_3:= 3, (* Function module 3 MLC, Inputs 1..16 (CFL-01.1-E2, Fast IO) *)
  FUNCTION_MODULE_4:= 4, (* Function module 4 MLC, Inputs 1..16 (CFL-01.1-E2, Fast IO) *)
);
END_TYPE

```

3.8.7 MB_PROBE_INPUT

Kurzbeschreibung Der Aufzählungstyp "MB_PROBE_INPUT" unterstützt die Auswahl des gewünschten Einganges auf dem ausgewählten Modul.



Für das OnBoard -Modul stehen die Eingänge 1..8 zur Verfügung.
Die Fast IO- Module haben die Eingänge 1..16 als Triggerquelle.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek / Ordner
MLC 15VRS	RIL_CommonTypes / TouchProbe
Grundpaket PLC	

Tab. 3-44: Referenztable MB_PROBE_INPUT

Typdefinition

```

TYPE MB_PROBE_INPUT :
(
  D_INPUT_1      := 1,  (* digital input 1 *)
  D_INPUT_2      := 2,  (* digital input 2 *)
  D_INPUT_3      := 3,  (* digital input 3 *)
  D_INPUT_4      := 4,  (* digital input 4 *)
  D_INPUT_5      := 5,  (* digital input 5 *)
  D_INPUT_6      := 6,  (* digital input 6 *)
  D_INPUT_7      := 7,  (* digital input 7 *)
  D_INPUT_8      := 8,  (* digital input 8 *)
  D_INPUT_9      := 9,  (* digital input 9 *)
  D_INPUT_10     := 10, (* digital input 10 *)
  D_INPUT_11     := 11, (* digital input 11 *)
  D_INPUT_12     := 12, (* digital input 12 *)
  D_INPUT_13     := 13, (* digital input 13 *)
  D_INPUT_14     := 14, (* digital input 14 *)
  D_INPUT_15     := 15, (* digital input 15 *)
  D_INPUT_16     := 16, (* digital input 16 *)
);
END_TYPE

```

3.8.8 MB_PROBE_DATA

Kurzbeschreibung

Der Aufzählungstyp "MB_PROBE_DATA" unterstützt die Auswahl der Mess-taster-Bezugssignalquelle sowohl für die Messtaster des Systems MLC (A-Parameter), als auch für die Messtaster des Antriebs (S-/P-Parameter).

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek / Ordner
MLC 15VRS	RIL_CommonTypes / TouchProbe
Grundpaket PLC	

Tab. 3-45: Referenztable MB_PROBE_DATA

Typdefinition

```

TYPE MB_PROBE_DATA :
(
  TP_ACT_POS_FEEDBACK1      := 0, // S-0-0051 / A-0-0100, Position, feedback 1 v
  TP_ACT_POS_FEEDBACK2      := 1, // S-0-0053 / -----, Position, feedback 2 v
  TP_ACT_POSIN_ACT_CYCLE     := 2, // P-0-0753 / -----, Position actual value
  TP_SYNC_CMD_POS           := 3, // P-0-0778 / A-0-2653, Synchronous position c
  TP_PROBE_TIME              := 4, // ----- / -----, time measurement
  TP_ACT_MEASURING_ENC_POS    := 5, // P-0-0052 / -----, Actual position value
  TP_RES_MASTER_AXIS_POS     := 6, // P-0-0775 / A-0-2650, Resulting master axis p
  TP_EFF_MASTER_AXIS_POS     := 7, // P-0-0776 / A-0-2651, Effective master axis p
  TP_CAM_SHAFT_PROF_ACCESS_ANGLE := 8, // P-0-0227 / A-0-2656, Cam shaft profile acce
  TP_ACT_MASTER_AXIS_POS     := 9, // not yet supported!!!
                                // P-0-0053 / A-0-0300, Master axis, actual po
  TP_ELS_CMD_POS             := 10, // ----- / A-0-2655, ELS, position command
  TP_USER_DATA_CONTAINER_A    := 11, // ----- / A-0-0521, Container of actual use
  TP_USER_DATA_CONTAINER_B    := 12, // ----- / A-0-0523, Container of actual use
  TP_USER_DATA_CONTAINER_C    := 13, // ----- / A-0-0525, Container of actual use
  TP_USER_DATA_CONTAINER_D    := 14, // ----- / A-0-0527, Container of actual use
  TP_USER_DATA_CONTAINER_E    := 15, // ----- / A-0-0529, Container of actual use
  TP_USER_DATA_CONTAINER_F    := 16, // ----- / A-0-0531, Container of actual use
);
END_TYPE

```

3.8.9 MB_PROBE_CONFIG

Kurzbeschreibung Die Struktur "MB_PROBE_CONFIG" dient der Aufnahme der Daten einer Messtaster-Funktion.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek / Ordner
MLC 15VRS Grundpaket PLC	RIL_CommonTypes / TouchProbe

Tab. 3-46: Referenztablelle MB_PROBE_CONFIG

Typdefinition

```

TYPE MB_PROBE_CONFIG :
STRUCT
  ModuloValueDrive      :REAL;      (* Modulo value *)
  ProcessingFormat      :BOOL;      (* Weighting position, abs./modulo *)
  ProbePositionEval     :MB_POSITION_EVAL; (* Enabling trigger signal, de
                                     (* pos., neg., both edges *)
  DifferenzMeasurment   :MB_DIFF_EVAL;  (* difference measurement meth
  ExpectWindowPosMode   :MB_EXPECT_WINDOW_MODE; (* Expectation window mode pos
                                     (* def.: Off, absolute / relat
  ExpectWindowNegMode   :MB_EXPECT_WINDOW_MODE; (* Expectation window mode neg
                                     (* def.: Off, absolute / relat
  ExpectWindowPosStart  :REAL;      (* Start pos. expectation window pos. edge *)
  ExpectWindowPosEnd    :REAL;      (* End pos. expectation window pos. edge *)
  ExpectWindowNegStart  :REAL;      (* Start pos. expectation window neg. edge *)
  ExpectWindowNegEnd    :REAL;      (* End pos. expectation window neg. edge *)
END_STRUCT
END_TYPE

```

"MB_PROBE_CONFIG" nutzt die folgenden Datentypen

- [MB_POSITION_EVAL, Seite 61](#)
- [MB_DIFF_EVAL, Seite 59](#)
- [MB_EXPECT_WINDOW_MODE, Seite 60](#)

3.8.10 MB_PROBE_CONFIG, Besonderheiten bei Sercos-basierten Achsen

Funktionsbeschreibung: Nachstehende Tabelle spiegelt die Verbindung zum Parametersystem der Steuerung wider:

Strukturelement	Datentyp	Parameter
ModuloValueDrive	REAL	A-0-0045, Modulowert
ProcessingFormat	BOOL	A-0-0059.7, Wichtungsfaktor für Geschwindigkeitsdaten, Bit 7 absolut/modulo
ProbePositionEval	MB_POSITION_EVAL, Seite 61	Aktivierung positive /negative Flanke, M-0-0007, Steuerwort
DifferenzMeasurment	MB_DIFF_EVAL, Seite 59	Differenzmessung, M-0-0007, Steuerwort
ExpectWindowPosMode	MB_EXPECT_WINDOW_MODE, Seite 60	Aktivierung und Konfiguration des Erwartungsfensters der positiven Flanke M-0-0007, Steuerwort
ExpectWindowNegMode	MB_EXPECT_WINDOW_MODE, Seite 60	Aktivierung und Konfiguration des Erwartungsfensters der negativen Flanke M-0-0007, Steuerwort
ExpectWindowPosStart	REAL	M-0-0202, Startposition des Erwartungsfensters, positive Flanke

Strukturelement	Datentyp	Parameter
ExpectWindowPosEnd	REAL	M-0-0203, Endposition des Erwartungsfensters, positive Flanke
ExpectWindowNegStart	REAL	M-0-0204, Startposition des Erwartungsfenster, negative Flanke
ExpectWindowNegEnd	REAL	M-0-0205, Endposition des Erwartungsfenster, negative Flanke

Tab. 3-47: Parameterzuordnung Messtaster-Funktion

4 RIL_Uilities.library

4.1 Einführung und Übersicht

Allgemeines Die Bibliothek "RIL_Uilities" erweitert die Grundfunktionalität des Systems MLC und stellt allgemeine Basis-Funktionalitäten zur Verfügung. Die hier beschriebene Bibliothek beinhaltet Basis-Funktionalitäten, die für viele verschiedene Anwendungen zum Einsatz kommen, aber auch einzeln verwendet werden können.



Für die Bibliothek ist das Basis-Funktionspaket PLC des Systems MLC 15VRS ausreichend.

Anwendungsgebiete und Funktionen / Funktionsbausteine von RIL_Uilities

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine	Beschreibung
Vertippen von Variablen (Adjustment)	
IL_ContinuousAdjustType01, Seite 72	kontinuierliche Verstellung einer REAL-Variablen über binäre Eingänge
IL_ContinuousAdjustType02, Seite 76	kontinuierliche Verstellung einer DINT-Variablen über binäre Eingänge.
IL_IncrementalAdjustType01, Seite 79	inkrementelle Verstellung einer REAL-Variablen über binäre Eingänge
Basisklassen für Funktionsbausteinentwicklung (Base FB)	
Basisfunktionsbaustein IL_EnableDoneBase, Seite 87	implementiert das Interface eines Funktionsbausteins mit Enable-Eingang und Done-, Active- und Standard-Fehlerausgängen
Basisfunktionsbaustein IL_EnableInOpBase, Seite 90	implementiert das Interface eines Funktionsbausteins mit Enable-Eingang und InOperation- und Standard-Fehlerausgängen
Basisfunktionsbaustein IL_EnableInOpShutdownBase, Seite 92	implementiert das Interface eines Funktionsbausteins mit Enable-Eingang, InOperation-, Shutdown- und Standard-Fehlerausgängen.
Basisfunktionsbaustein IL_ExecuteDoneAbortedBase, Seite 102	implementiert das Interface eines Funktionsbausteins mit Execute-Eingang und Done-, CommandAborted-, Active- und Standard-Fehlerausgängen
Basisfunktionsbaustein IL_ExecuteDoneAbortedShutdownBase, Seite 105	implementiert das Interface eines Funktionsbausteins mit Execute-Eingang und Done-, Active-, Shutdown-, CommandAborted-Ausgang und Standard-Fehlerausgängen
Basisfunktionsbaustein IL_ExecuteDoneBase, Seite 97	implementiert das Interface eines Funktionsbausteins mit Execute-Eingang und Done-, Active- und Standard-Fehlerausgängen
Basisfunktionsbaustein IL_ExecuteInOpBase, Seite 99	implementiert das Interface eines Funktionsbausteins mit Execute-Eingang und InOperation-, Active- und Standard-Fehlerausgängen
Konvertierung (Bit-Stream Connect)	
IL_BoolToByte, Seite 111	8 einzelne Bits (BOOL) werden zu einem BYTE konvertiert (FUN)
IL_BoolToDWord, Seite 112	32 einzelne Bits (BOOL) werden zu einem DWORD konvertiert (FUN)
IL_BoolToWord, Seite 115	16 einzelne Bits (BOOL) werden zu einem WORD konvertiert (FUN)
IL_ByteToDWord, Seite 116	4 BYTES werden zu einem DWORD konvertiert (FUN)
IL_ConcatByte, Seite 117	2 BYTES werden zu einem WORD zusammengefasst (FUN)
IL_ConcatWord, Seite 118	2 WORDs werden zu einem DWORD zusammengefasst

Konvertierung (Bit-Stream Split)	
IL_ByteToBool, Seite 119	FB extrahiert 8 Bits (BOOL) aus einem BYTE
IL_DWordToBool, Seite 120	FB extrahiert 32 Bits (BOOL) aus einem DWORD
IL_DWordToByte, Seite 123	FB extrahiert 4 BYTES aus einem DWORD
IL_HighByte, Seite 123	FUN extrahiert höherwertiges BYTE aus einem WORD
IL_HighWord, Seite 124	FUN extrahiert höherwertiges WORD aus einem DWORD
IL_LowByte, Seite 124	FUN extrahiert niederwertiges BYTE aus einem WORD
IL_LowWord, Seite 125	FUN extrahiert niederwertiges WORD aus einem DWORD
IL_WordToBool, Seite 125	FB extrahiert 16 Bits (BOOL) aus einem WORD
Konvertierung (ByteSwap)	
IL_SwapDWord, Seite 127	FUN tauscht BYTES eines DWORDs (3 2 1 0 -> 0 1 2 3) Achtung, nicht PLCopen-kompatibel!
IL_SwapWord, Seite 128	FUN tauscht BYTES eines WORDs (1 0 -> 0 1)
Prüfsummen (CRC16)	
IL_CRC16, Seite 129	FB berechnet CRC16-Prüfsumme über ein Datum
Konvertierung (Datatype Change)	
IL_AsciiToByte auf Seite 130	FUN wandelt beliebiges ASCII-Zeichen in Bytecode um
IL_ByteToAscii auf Seite 131	FUN wandelt beliebigen Bytecode in ein ASCII-Zeichen um
IL_RealToString auf Seite 131	FUN wandelt REAL-Wert in einen STRING um
IL_LRealToString auf Seite 132	FUN wandelt (L)REAL-Wert formatgesteuert in einen STRING um
Hochauflösende Zeitunterstützung (Debugging)	
IL_HighResTimeDiff, Seite 134	berechnet die Zeitdifferenz zweier hochaufgelöster Zeittakte des Systems in Mikrosekunden (vor und nach Codeabschnitt)
IL_HighResTimeTick, Seite 134	liest den hochaufgelösten Zeittakt des Systems
Trigger (FlipFlops)	
IL_Toggle, Seite 136	setz- und rücksetzbares FlipFlop
Mathematische Funktionen (Mathematical Functions)	
IL_InterpolationLinear, Seite 137,	dient zur Berechnung von Zwischenpunkten, kann für Linearisierungen verwendet werden
IL_MaxValue, Seite 139,	gibt mit jedem Aufruf den Maximalwert aller bisher aufgetreten Wert an
Zeichenfolgen ohne Längenbegrenzung (StringFun_NoLimitStrLength)	
IL_StrLower, Seite 144	Funktion konvertiert eine Zeichenfolge in Kleinbuchstaben
IL_StrUpper, Seite 144	Funktion konvertiert eine Zeichenfolge in Großbuchstaben
IL_StrCompare, Seite 145	Funktion vergleicht zwei Zeichenfolgen und liefert die Differenz zurück. Groß-/Kleinschreibung berücksichtigt
IL_StrCompareNoCase, Seite 146	Funktion vergleicht zwei Zeichenfolgen und gibt die Differenz zurück. Groß-/Kleinschreibung nicht berücksichtigt

IL_StrNCompare, Seite 146	Funktion vergleicht die eingegebene Anzahl Zeichen von zwei Zeichenfolgen und gibt die Differenz zurück. Groß-/Kleinschreibung berücksichtigt
IL_StrNCompareNoCase, Seite 147	Funktion vergleicht die eingegebene Anzahl Zeichen von zwei Zeichenfolgen und gibt die Differenz zurück. Groß-/Kleinschreibung nicht berücksichtigt
IL_AsciiStreamToBase64, Seite 148	Funktion konvertiert eine im ASCII-Format abgelegte Zeichenkette ins BASE64-Format
IL_Base64ToAsciiStream, Seite 149	Funktion konvertiert eine im BASE64-Format abgelegte Zeichenkette ins ASCII-Format
IL_BufferToString, Seite 151	Funktion kopiert eine ASCII-Zeichenkette aus einem Byte-Array
IL_StringToBuffer, Seite 153	Funktion kopiert eine ASCII-Zeichenkette in ein Byte-Array
IL_StrCopy, Seite 154	Funktion kopiert eine Zeichenfolge von der Quell- zur Zieladresse
IL_StrNCopy, Seite 155	Die Funktion kopiert einen Teil-Zeichenfolge aus einer Zeichenfolge in eine Ziel-Zeichenfolge. Die Länge des Ziel-Zeichenfolge ist festgelegt. Wenn die Quell-Zeichenfolge kürzer als die geforderte Länge ist, wird die Ziel-Zeichenfolge durch Füllzeichen ergänzt
IL_CONCAT, Seite 156	Funktion fügt Zeichenfolge 2 hinter Zeichenfolge 1 an
IL_DELETE, Seite 157	Funktion löscht eine Teil-Zeichenfolge aus der Zeichenfolge ab einer vorgegebenen Position
IL_FIND, Seite 158	Funktion sucht die Position der eingegebenen Teil-Zeichenfolge in einer Zeichenfolge
IL_INSERT, Seite 158	Funktion fügt eine Teil-Zeichenfolge ab der eingegebenen Position in eine Zeichenfolge ein
IL_LEFT, Seite 159	Funktion liefert das x Zeichen lange linke Teil einer Zeichenfolge
IL_LEN, Seite 160	Funktion liefert die Länge einer Zeichenfolge
IL_MID, Seite 161	Funktion liefert eine Teil-Zeichenfolge ab der eingegebenen Position aus einer Zeichenfolge
IL_REPLACE, Seite 162	Funktion ersetzt eine Teil-Zeichenfolge einer Zeichenfolge durch eine andere Teil-Zeichenfolge
IL_RIGHT, Seite 163	Funktion liefert das x Zeichen lange rechte Teil einer Zeichenfolge
IL_Ascii_To_Char, Seite 164	Funktion konvertiert Ascii-Code in ein Zeichen
IL_Char_To_Ascii, Seite 164	Funktion konvertiert ein Zeichen in Ascii-Code
IL_Int_To_Str, Seite 164	Funktion konvertierte eine Ganzzahl formatabhängig in eine Zeichenfolge
IL_Real_To_Str, Seite 165	Funktion konvertierte eine REAL-Zahl formatabhängig in eine Zeichenfolge
IL_String_To_Unicode, Seite 166	Funktion konvertiert eine Zeichenfolge entsprechend vorgegebener Option (utf8 / utf16) in Unicode und speichert die Werte in einem DWORD-Array ab
IL_Unicode_To_String, Seite 167	Funktion konvertiert eine UniCode-Zeichenfolge vorgegebener Option (utf8 / utf16) in einen STRING

IL_BufferGetElementCount, Seite 168	Funktion bestimmt die Anzahl der durch ein "Separator"-Zeichen getrennten Objekte in einer Zeichenkette
IL_Code, Seite 169	Funktion ermittelt den ASCII-Code eines vorgegebenen Zeichens in einer Zeichenkette in dezimaler Darstellung
IL_StrInverse, Seite 170	Funktion invertiert eine Zeichenfolge
IL_StrSetEndOfString, Seite 171	Funktion setzt das Ende einer Zeichenfolge
IL_StrGetElement, Seite 171	Zeichenfolge wird durch die eingegebene Trenn-Zeichenfolge in einige Elementen aufgeteilt. Die Funktion ruft eines der Elemente ab
IL_StrLChr, Seite 172	Funktion isoliert eine Teil-Zeichenfolge aus der Zeichenfolge bis zum ersten Auftreten des eingegebenen Trennzeichens
IL_StrRChr, Seite 173	Funktion isoliert eine Teil-Zeichenfolge aus der Zeichenfolge ab dem letzten Auftreten des eingegebenen Trennzeichens
IL_StrSetElement, Seite 174	Zeichenfolge wird durch die eingegebene Trenn-Zeichenfolge in Elementen aufgeteilt. Die Funktion setzt ein neues Element an die gewünschte Position in der Zeichenfolge
IL_StrSplit, Seite 175	Zeichenfolge wird durch ein vorgegebenes Trennzeichen aufgeteilt. Die Funktion legt die Adresse jeder Teil-Zeichenfolge in ein Array
IL_StrTrimAll, Seite 176	Funktion löscht alle vorgegebene Zeichen in einer Zeichenfolge
IL_StrTrimSpace, Seite 177	Funktion löscht die Leerzeichen in einer Zeichenfolge je nach gewünschter Option
Zeitfunktionen (TIME)	
IL_Date, Seite 179	liest das aktuelle Datum des Systems
IL_DateAndTime, Seite 180	liest das aktuelle Datum und die aktuelle Uhrzeit des Systems
IL_ExtSysTimeDate, Seite 180	liest das aktuelle Datum und die aktuelle Uhrzeit des Systems im erweiterten Systemformat
IL_LocalTime, Seite 181	rechnet steuerungsinterne UTC-Zeit auf lokale Zeit um
IL_SysTime64, Seite 183	liest das aktuelle Datum und die aktuelle Uhrzeit des Systems in Mikrosekunden seit 01.01.1970
IL_SysTime64ToSysTimeDate, Seite 184	dient der Formatumwandlung des Systemdatums und der Systemzeit
IL_SysTimeDate, Seite 184	liest das aktuelle Datum und die aktuelle Uhrzeit des Systems
IL_SysTimeDateToSysTime64, Seite 185	dient der Formatumwandlung des Systemdatums und der Systemzeit
IL_TimeOfDay, Seite 185	FUN liest die aktuelle Uhrzeit des Systems
Doppelt-verkettete Listen (Utilities/ Logic)	
IL_LinkedList, Seite 190,	verwaltet doppelt-verkettete Listen (verwendet für Ringpuffer, FIFOs und LIFOs)
Modulwert-Berechnung (Utilities/ Logic)	
IL_GetModuloType01, Seite 196	berechnen einen Modulowert, Wertebereich liegt zwischen 0.0 und + Modulowert (REAL)
IL_GetModuloType02, Seite 197	berechnen einen Modulowert, Wertebereich liegt zwischen 0 und + Modulowert (DINT)

IL_GetModuloType03, Seite 198	berechnen einen Modulwert, Wertebereich liegt zwischen - Modulwert und + Modulwert (REAL)
IL_GetModuloType04, Seite 199	berechnen einen Modulwert., Wertebereich liegt zwischen - modulo / 2 und + modulo / 2 (REAL)

Tab. 4-1: RIL_Uilities-Bibliothek

4.2 Vertippen von Variablen (Adjustment)

4.2.1 Einführung und Übersicht

Mit den nachstehenden Funktionsbausteinen können SPS-Variable über binäre Eingänge kontinuierlich oder inkrementell geändert (vertippt) werden:

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine	Beschreibung
Vertippen von Variablen (Adjustment)	
IL_ContinuousAdjustType01, Seite 72	kontinuierliche Verstellung einer REAL-Variablen über binäre Eingänge
IL_ContinuousAdjustType02, Seite 76	kontinuierliche Verstellung einer DINT-Variablen über binäre Eingänge
IL_IncrementalAdjustType01, Seite 79	inkrementelle Verstellung einer REAL-Variablen über binäre Eingänge

Tab. 4-2: RIL_Uilities, Adjustment



Die genannten Funktionsbausteinen haben beim Übergang zur Version 14 ihre Namen von MB_* nach IL_* geändert und sind von "ML_TechBase" nach "RIL_Uilities" übergegangen.

Bei Verwendung der oben aufgeführten Funktionsbausteine zum Vertippen von SPS-Variablen sind die folgenden Bedingungen zu berücksichtigen:

- Der Betrieb muss über einen binären Eingang ("Enable") gestoppt werden
- Mit den Funktionsbausteinen IL_ContinuousAdjustType01 und IL_ContinuousAdjustType02 kann die zu beeinflussende Variable kontinuierlich verändert werden (ähnlich dem langen Tippen bei SYNAX)
- Mit dem Funktionsbaustein IL_IncrementalAdjustType01 kann die zu beeinflussende Variable schrittweise verändert werden (ähnlich dem kurzen Tippen bei SYNAX)
- Die gewählte Variable kann innerhalb der Grenzwerte in aufsteigender oder absteigender Schrittfolge verändert werden
- Mit Erreichen eines Grenzwertes ("HighLimitActive" = TRUE oder "LowLimitActive" = TRUE) wird die kontinuierliche Verstellung deaktiviert, d.h., der Funktionsbaustein beeinflusst die Variable nicht weiter. In diesem Fall wird der entsprechende Grenzwert am Eingang "HighLimit" bzw. "LowLimit" ausgegeben

Allgemeine Grundlagen

Besitzen bei einem angegebenen Modulwert die Grenzen "HighLimit" und "LowLimit" den gleichen Wert, wird "LowLimit" gleich Null und "HighLimit" gleich dem Modulwert gesetzt. Somit ist es möglich, die Grenzwerte über den gesamten Modulbereich zu verstellen.

Die Änderungsgeschwindigkeit beschreibt die Rate, mit der sich die beeinflusste Variable während des Verstellvorgangs ändert. Diese Rate ist abhängig von der Schrittweite und der Anzahl der Schritte pro Sekunde. Die Änderungsgeschwindigkeit errechnet sich z.B. beim Ändern einer Position wie folgt:

$$\text{Änderungsgeschwindigkeit} \left[\frac{\text{Degree}}{s} \right] = \text{Increments} [\text{Degree}] * \text{Increments per Second} \left[\frac{1}{s} \right]$$

Abb. 4-1: Geschwindigkeitsänderung bei Positionsänderung

Wird z.B. die Drehzahl der virtuellen Leitachse verändert, errechnet sich die Änderungsgeschwindigkeit wie folgt:

$$\text{Änderungsgeschwindigkeit} \left[\frac{r.p.m.}{s} \right] = \text{Increments} [r.p.m.] * \text{Increments per Second} \left[\frac{1}{s} \right]$$

Abb. 4-2: Geschwindigkeitsänderung beim Ändern einer Achsdrehzahl

Solange das Stellsignal anliegt, wird die beeinflusste Variable mit der definierten Geschwindigkeit verändert.

Stellgrenzen Die Verstellung kann nur über die entsprechenden Eingänge innerhalb der festgelegten Grenzwerte erfolgen.

Ausnahme:

Befindet sich der Anfangswert der zu verändernden Größe außerhalb der vorgegebenen Grenzen, kann die beeinflusste Variable nur in Richtung des Zielbereiches verändert werden. Wird ein Modulowert angegeben, kann die beeinflusste Variable in beiden Richtungen verändert werden. Nach dem Erreichen des Zielbereichs ist es nicht mehr möglich, diesen zu verlassen.

Preset Mit Hilfe der Eingänge "Preset" und "PresetValue" kann die Ausgangsgröße auf einen definierten Wert vorbesetzt werden. Der Ausgang "PresetAck" wird gesetzt, sobald der "Presetvalue" übernommen wurde. Die Ausgangsgröße wird auf dem "PresetValue" gehalten, solange der Eingang "Preset" gesetzt ist.

Grundregeln

- Sind beide Stellsignale ("Inc"/"Dec") gleichzeitig TRUE, werden beide als FALSE bewertet
- Ein Stellsignal ändert die Polarität und löst einen sofortigen Richtungswechsel aus. Ebenso wird die aktuelle Verstellung nicht beendet
- Wird die Stellgeschwindigkeit während einer Bewegung umgeschaltet, verhalten sich die Funktionsbausteine

IL_ContinuousAdjustType01/IL_ContinuousAdjustType02 und

IL_IncrementalAdjustType01

unterschiedlich.

Bei IL_ContinuousAdjustType01 und IL_ContinuousAdjustType02 ist die Änderung sofort wirksam. Im Gegensatz dazu wird die Bewegung bei IL_IncrementalAdjustType01 beendet, bevor die neue Schrittgeschwindigkeit übernommen wird. Ein nachfolgender Stellvorgang wird dann mit den geänderten Größen durchgeführt.

Für den Funktionsbaustein IL_IncrementalAdjustType01 gelten ferner folgende Regeln:

- Die Eingänge "Inc" und "Dec" werden flankenbezogen ausgewertet. Eine steigende Flanke löst eine Anpassung um die vorgegebene Schrittweite ("StepWidth") aus
- Ein während einer laufenden Bewegung erkannt neues Stellsignal (positive Flanke) mit gleicher Polarität bewirkt eine weitere Bewegung im Anschluss an die Aktuelle

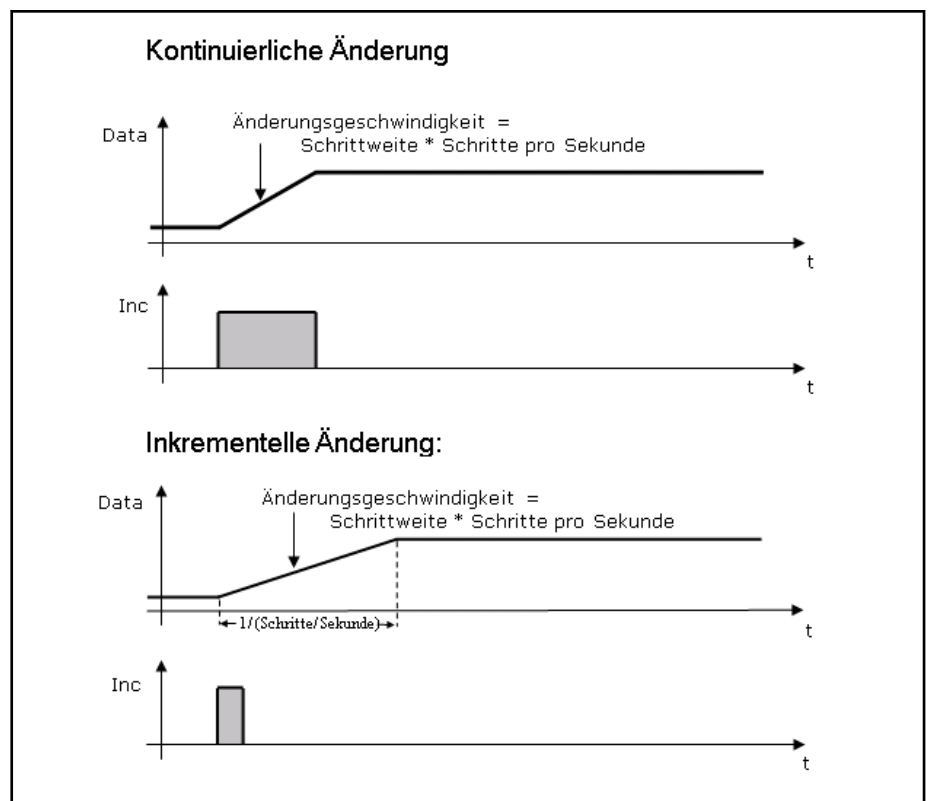


Abb. 4-3: Vergleich des Verhaltens von kontinuierlicher und inkrementeller Änderung

Die Funktionsbausteine IL_ContinuousAdjustType01, IL_ContinuousAdjustType02 und IL_IncrementalAdjustType01 initialisieren bei der ersten Aktivierung ("Enable" = TRUE) unabhängig von den anstehenden Eingängen die zur Bestimmung der Zykluszeit notwendigen Variablen. Im zweiten Zyklus steht dann eine berechnete Zykluszeit zur Verfügung, die notwendig ist, um die korrekte Änderungsgeschwindigkeit zu berechnen. Anschließend wird die Zykluszeit fortlaufend aktualisiert.

Durch die Angabe eines Modulwertes liefert der Funktionsbaustein eine Ausgangsgröße, die sich zwischen Null und dem angegebenen Modulwert bewegt. Bei Überschreiten des Modulwertes wird lediglich die Ausgangsgröße auf Null zurückgesetzt, der Verstellvorgang jedoch fortgesetzt. Der vorgegebene gültige Bereich kann dementsprechend zwischen Null und dem Modulwert liegen. Der Bereich kann um den Nullwert verstellt werden.

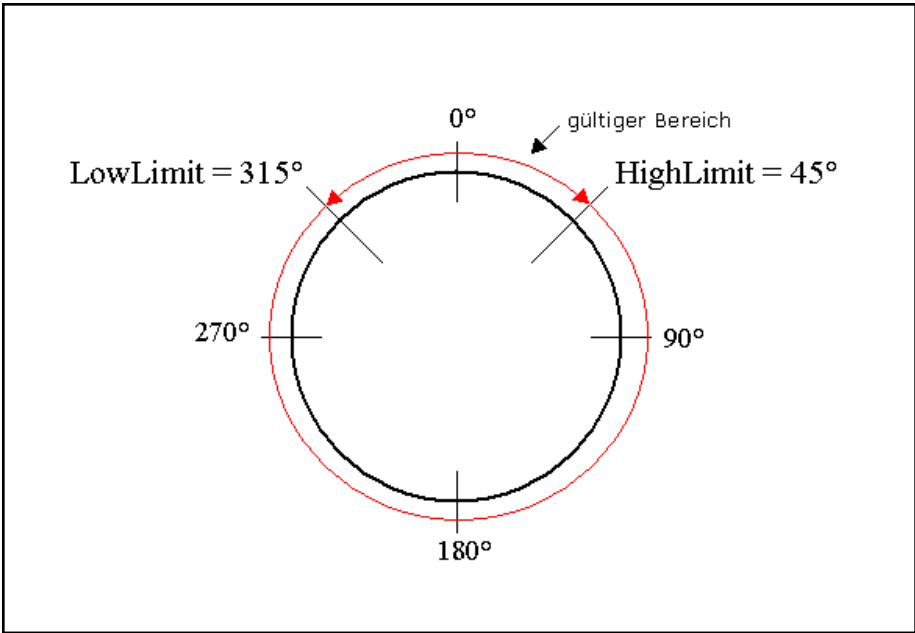


Abb. 4-4: Mögliche Zuordnung eines gültigen Bereichs bei Angabe eines Modulowertes von 360°

Zusätzlich bietet der Funktionsbaustein IL_ContinuousAdjustType01 bzw. IL_ContinuousAdjustType02 die Möglichkeit, die Änderungsgeschwindigkeit durch die Angabe von Wichtungsfaktoren und Zeitintervallen zeitabhängig zu verändern.

4.2.2 IL_ContinuousAdjustType01

Kurzbeschreibung

Der Funktionsbaustein IL_ContinuousAdjustType01 ermöglicht die kontinuierliche Verstellung einer REAL-Variablen über binäre Eingänge.

Der Funktionsbaustein "MB_ContinuousAdjustType01 (ML_Tech-Base-Bibliothek)" wurde mit Version 14VRS in "IL_ContinuousAdjustType01" umbenannt.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek
MLC 15VRS	RIL_Utilities

Tab. 4-3: Referenztablelle Funktionsbaustein IL_ContinuousAdjustType01

Schnittstellenbeschreibung

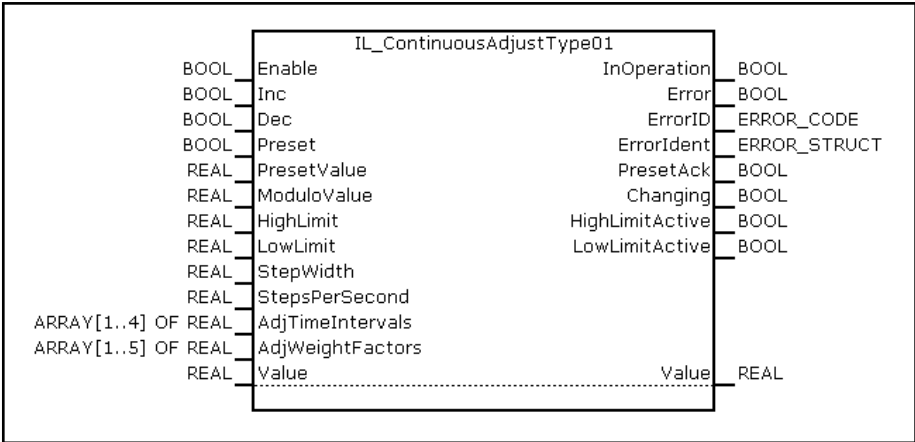


Abb. 4-5: Funktionsbaustein IL_ContinuousAdjustType01

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Freigabe des Funktionsbausteins (zyklisch, zustands-gesteuert)
	Inc	BOOL	Beeinflusste Variable erhöhen
	Dec	BOOL	Beeinflusste Variable verringern
	Preset	BOOL	Voreinstellwert (PresetValue) setzen
	PresetValue	REAL	Vorgabewert für den Voreinstellwert
	ModuloValue	REAL	Modulowert (wenn 0, dann absolute Verarbeitung)
	HighLimit	REAL	Maximaler Ausgangswert der zu beeinflussenden Va-riablen
	LowLimit	REAL	Minimaler Ausgangswert der zu beeinflussenden Va-riablen
	StepWidth	REAL	Schrittweite
	StepsPerSecond	REAL	Anzahl der Schritte pro Sekunde
	AdjTimeIntervals	ARRAY [1..4] OF REAL	Zeitintervalle in Sekunden zur Wichtung der Ände-rungsgeschwindigkeit
	AdjWeightFactors	ARRAY [1..5] OF REAL	Wichtungsfaktoren der Änderungsgeschwindigkeit
VAR_OUTPUT	InOperation	BOOL	Funktionsbaustein aktiv. Die Ausgänge "HighLimitActi-ve" und "LowLimitActive" sind gültig
	Error	BOOL	Berechnung der zu beeinflussenden Variablen "Value" wurde mit Fehler abgeschlossen; die Ausgangsvariab-le "ErrorIdent" ist gültig
	ErrorID	ERROR_CODE	Fehlerkurzbeschreibung
	ErrorIdent	ERROR_STRUCT	Detailbeschreibung der Diagnose im Fehlerfall Fehlerbehandlung, Seite 75
	PresetAck	BOOL	Der Wert des Eingangs "PresetValue" wurde über-nommen
	Changing	BOOL	Berechnung der zu beeinflussenden Variablen erfolgt. Ausgangsvariable "Value" ist gültig
	HighLimitActive	BOOL	Maximaler Ausgangswert der zu beeinflussenden Va-riablen wurde erreicht
VAR_IN_OUT	LowLimitActive	BOOL	Minimaler Ausgangswert der zu beeinflussenden Va-riablen wurde erreicht
	Value	REAL	Die zu beeinflussende Variable

Tab. 4-4: Schnittstellenvariablen FB IL_ContinuousAdjustType01

Signal-Zeit-Diagramm

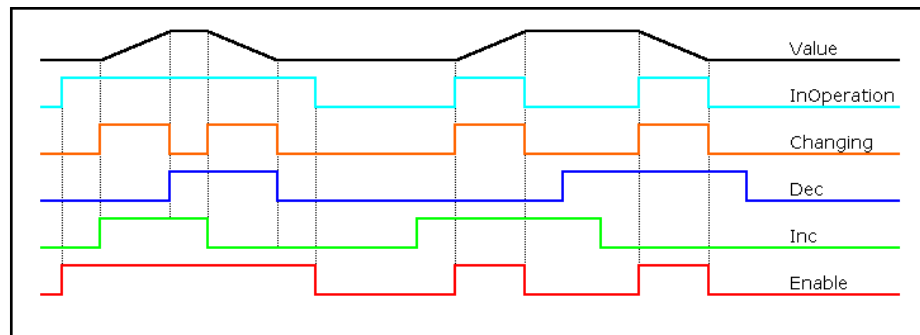


Abb. 4-6: Kontinuierliche Parametereinstellung

Funktionsbeschreibung

Die Verstellung der beeinflussten Variablen "Value" kann sowohl in positiver als auch in negativer Richtung erfolgen. Durch die Grenzwerte wird ein Gültigkeitsbereich für die Beeinflussung der Variablen festgelegt. Die Vorgabe eines Modulo-Wertes erlaubt die Verstellung um innerhalb des Bezugssystems der Achse zu sein. Durch Festlegen der Schrittweite und der Anzahl der Schritte pro Sekunde wird die Änderungsgeschwindigkeit der beeinflussten Variablen bestimmt. Zudem besteht die Möglichkeit, einen vorgegebenen Wert ("PresetValue"), der sich im zulässigen Betriebsbereich befindet, direkt auf die beeinflusste Variable zu schreiben.

Nach der Bearbeitungsfreigabe über "Enable" kann die zu beeinflussende Variable über die Eingänge "Inc" (in positiver Richtung) und "Dec" (in negativer Richtung) verändert werden. Solange der Eingang "Inc" bzw. "Dec" gesetzt ist, wird die Größe der einstellbaren Variablen mit der vorgegebenen Geschwindigkeit kontinuierlich erhöht bzw. verringert. ("StepsPerSecond" * "StepWidth").

Bei der Initialisierung des Funktionsbausteines muss sichergestellt werden, dass die Eingangsgrößen "ModuloValue", "HighLimit" und "LowLimit" gültige Werte besitzen.

Durch die Festlegung von Wichtungsfaktoren ("AdjWeightFactors[]") und dazugehörigen Zeitintervallen ("AdjTimeIntervals[]") lässt sich die Änderungsgeschwindigkeit zeitabhängig verändern. Die Wichtungsfaktoren und die Zeitintervalle müssen zusätzlich zu den Eingängen "StepsPerSecond" und "StepWidth" gesetzt werden. Die Änderungsgeschwindigkeit wird dann, abhängig von dem aktuellen Zeitintervall, mit dem entsprechenden Wichtungsfaktor multipliziert. Die nachfolgende Abbildung stellt das Verhalten des Funktionsbausteines anhand einer Beispielkonfiguration dar: Die Eingangsgrößen von "AdjTimeIntervals" ([T1; T2; T3; T4]) werden in Sekunden angegeben. T5 bezeichnet dabei den Zeitraum nach Abschluss des Zeitintervalls T4 bis zum Abbruch des Stellvorganges. Die Eingangsgrößen von "AdjWeightFactors" ([0,5; 1; 1,5; 2; 4]) bestimmen die Wichtung der Änderungsgeschwindigkeit. Der letzte Wert von "AdjWeightFactors" ("AdjWeightFactors [5]") wird dem Zeitintervall T5 zugeordnet.

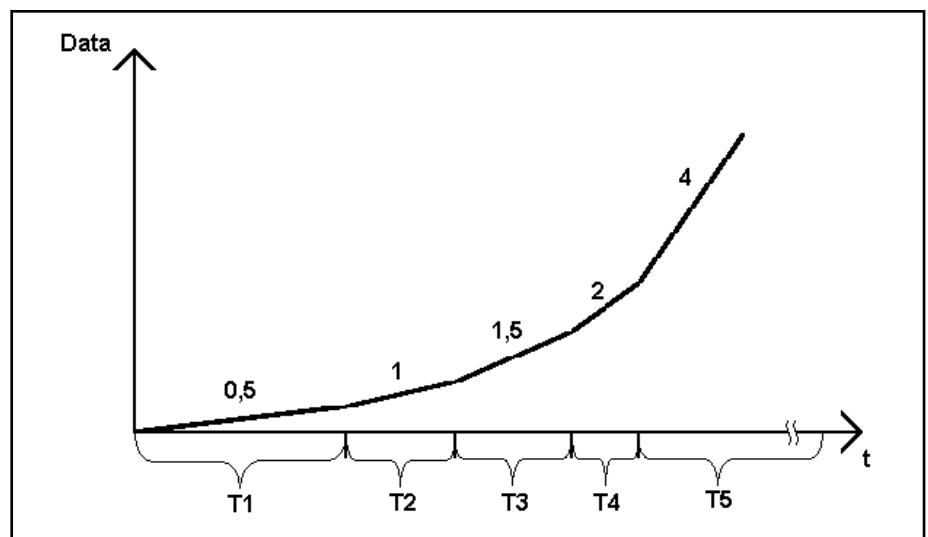


Abb. 4-7: Gewichtete Verstellung



Wird ein Zeitintervall von Null gewählt, wird der darauffolgende Wichtungsfaktor bis zum Abbruch des Verstellvorganges beibehalten.

Beispiel:

Ist "AdjTimeIntervals[2]" = 0, wird der Wichtungsfaktor von "AdjWeightFactors[3]" bis zum Abbruch beibehalten.

Die Berechnung der abgelaufenen Zeit zur Bestimmung des korrekten Zeitintervalls der Wichtungsfaktoren wird mit jedem neu beginnenden Verstellvorgang rückgesetzt. Sind die Eingänge "AdjWeightFactors" und "AdjTimeIntervals" nicht gesetzt, wird eine kontinuierliche Verstellung mit der durch die Eingänge "StepsPerSecond" und "StepWidth" vorgegebenen Änderungsgeschwindigkeit durchgeführt.



Für die Eingangswerte der Zeitintervalle ("AdjTimeIntervals") müssen positive Werte angegeben werden. Die Wichtungsfaktoren können auch negative Werte erhalten.

Die zu beeinflussende Variable kann nur auf den vorgegebenen Voreinstellwert gesetzt werden, wenn sich dieser innerhalb des gültigen, durch "HighLimit" und "LowLimit" vorgegebenen Bereichs befindet.

Der Ausgang "InOperation" signalisiert, dass der Funktionsbaustein in Betrieb ist. Eine Verstellung findet jedoch nicht statt. Der Ausgang "Changing" zeigt an, dass der Verstellvorgang fehlerfrei abgearbeitet wurde. Der Wert der verstellten Variablen "Value" ist gültig.

Der Ausgang "Error" zeigt das Auftreten eines Fehlers während des Verstellvorgangs an. Detailangaben zu dem Fehler sind der Struktur "ErrorIdent" zu entnehmen.

Fehlerbehandlung

Der Funktionsbaustein erzeugt in Additional1/Additional2 der Tabelle „F_RELATED_TABLE“, 16#0170, die folgenden Fehlermeldungen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR (16#0006)	16#0002	16#0000	Eingänge liegen außerhalb des zulässigen Bereichs
RESOURCE_ERROR (16#0003)	16#0004	16#0000	Antriebs-Firmware wird nicht unterstützt
STATE_MACHINE_ERROR (16#0005)	16#0006	16#0000	Ungültiger Zustand des Funktionsbausteins

Tab. 4-5: Fehlercodes FB IL_ContinuousAdjustType01

4.2.3 IL_ContinuousAdjustType02

Kurzbeschreibung Der Funktionsbaustein IL_ContinuousAdjustType02 ermöglicht die kontinuierliche Verstellung einer zu beeinflussenden DINT-Variablen über binäre Eingänge.



Der Funktionsbaustein "MB_ContinuousAdjustType02 (ML_Tech-Base-Bibliothek)" wurde mit Version 14VRS in "IL_ContinuousAdjustType02" umbenannt.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek
MLC 15VRS	RIL_Uilities

Tab. 4-6: Referenztable Funktionbaustein IL_ContinuousAdjustType02

Schnittstellenbeschreibung

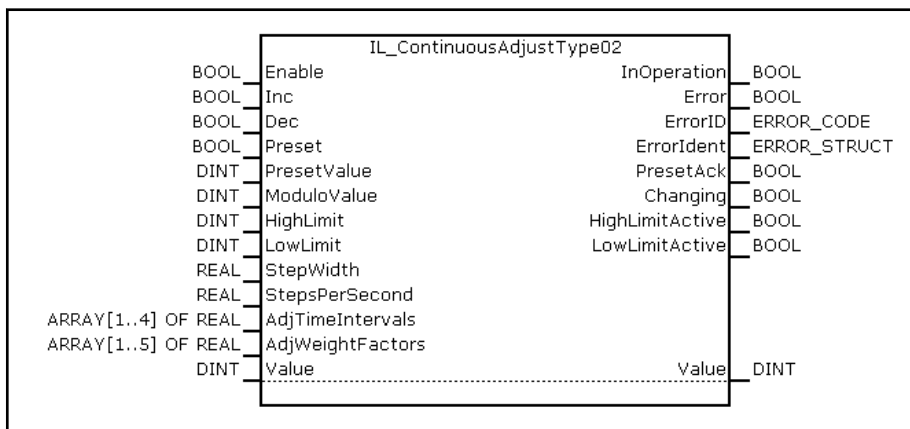


Abb. 4-8: Funktionsbaustein IL_ContinuousAdjustType02

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Freigabe des Funktionsbausteins (zyklisch, zustands-gesteuert)
	Inc	BOOL	Beeinflusste Variable erhöhen
	Dec	BOOL	Beeinflusste Variable verringern
	Preset	BOOL	Voreinstellwert (PresetValue) setzen
	PresetValue	DINT	Vorgabewert für den Voreinstellwert
	ModuloValue	DINT	Modulowert (wenn 0, dann absolute Verarbeitung)
	HighLimit	DINT	Maximaler Ausgangswert der zu beeinflussenden Variablen

E/A-Typ	Name	Datentyp	Beschreibung
	LowLimit	DINT	Minimaler Ausgangswert der zu beeinflussenden Variablen
	StepWidth	REAL	Schrittweite
	StepsPerSecond	REAL	Schritte pro Sekunde
	AdjTimeIntervals	ARRAY [1..4] OF REAL	Zeitintervalle in Sekunden zur Wichtung der Änderungsgeschwindigkeit
	AdjWeightFactors	ARRAY [1..5] OF REAL	Wichtungsfaktoren der Änderungsgeschwindigkeit
VAR_OUTPUT	InOperation	BOOL	Funktionsbaustein aktiv. Die Ausgänge HighLimitActive und LowLimitActive sind gültig
	Error	BOOL	Berechnung der zu beeinflussenden Variablen "Value" wurde mit Fehler abgeschlossen; die Ausgangsvariable „ErrorIdent“ ist gültig
	ErrorID	ERROR_CODE	Fehlerkurzbeschreibung
	ErrorIdent	ERROR_STRUCT	Detailbeschreibung der Diagnose im Fehlerfall Fehlerbehandlung, Seite 79
	PresetAck	BOOL	Der Wert des Eingangs "PresetValue" wurde übernommen
	Changing	BOOL	Berechnung der zu beeinflussenden Variablen erfolgt. Die Ausgangsvariable "Value" ist gültig
	HighLimitActive	BOOL	Maximaler Ausgangswert der zu beeinflussenden Variablen wurde erreicht
	LowLimitActive	BOOL	Minimaler Ausgangswert der zu beeinflussenden Variablen wurde erreicht
VAR_IN_OUT	Value	DINT	Die zu beeinflussende Variable

Tab. 4-7: Schnittstellenvariablen FB IL_ContinuousAdjustType02

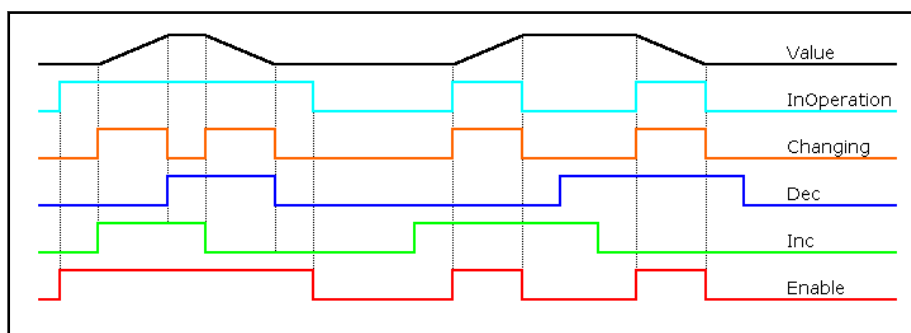
Signal-Zeit-Diagramm

Abb. 4-9: Kontinuierliche Parametereinstellung

Funktionsbeschreibung

Die Verstellung der beeinflussten Variablen "Value" kann sowohl in positiver als auch in negativer Richtung erfolgen. Durch die Grenzwerte wird ein Gültigkeitsbereich für die Beeinflussung der Variablen festgelegt. Die Vorgabe eines Modulo-Wertes erlaubt die Verstellung um innerhalb des Bezugssystems der Achse zu sein. Durch Festlegen der Schrittweite und der Anzahl der Schritte pro Sekunde wird die Änderungsgeschwindigkeit der beeinflussten Variablen bestimmt. Zudem besteht die Möglichkeit, einen vorgegeben Wert (PresetValue), der sich im zulässigen Betriebsbereich befindet, direkt auf die beeinflusste Variable zu schreiben.

Nach der Bearbeitungsfreigabe über "Enable" kann die zu beeinflussende Variable über die Eingänge "Inc" (in positiver Richtung) und "Dec" (in negativer Richtung) verändert werden. Solange der Eingang "Inc" bzw. "Dec" gesetzt ist, wird die Größe der einstellbaren Variablen mit der vorgegebenen Geschwindigkeit kontinuierlich erhöht bzw. verringert. ("StepsPerSecond" * "StepWidth").

Bei der Initialisierung des Funktionsbausteines muss sichergestellt werden, dass die Eingangsgrößen "ModuloValue", "HighLimit" und "LowLimit" gültige Werte besitzen.

Durch die Festlegung von Wichtungsfaktoren ("AdjWeightFactors[]") und dazugehörigen Zeitintervallen ("AdjTimeIntervals[]") lässt sich die Änderungsgeschwindigkeit zeitabhängig verändern. Die Wichtungsfaktoren und die Zeitintervalle müssen zusätzlich zu den Eingängen "StepsPerSecond" und "StepWidth" gesetzt werden. Die Änderungsgeschwindigkeit wird dann, abhängig von dem aktuellen Zeitintervall, mit dem entsprechenden Wichtungsfaktor multipliziert. Die nachstehende Abbildung stellt das Verhalten des Funktionsbausteins anhand einer Beispielkonfiguration dar: Die Eingangsgrößen von "AdjTimeIntervals" ([T1; T2; T3; T4]) werden in Sekunden angegeben. T5 bezeichnet dabei den Zeitraum nach Abschluss des Zeitintervalls T4 bis zum Abbruch des Verstellvorganges. Die Eingangsgrößen von "AdjWeightFactors" ([0,5; 1; 1,5; 2; 4]) bestimmen die Wichtung der Änderungsgeschwindigkeit. Der letzte Wert von "AdjWeightFactors" ("AdjWeightFactors [5]") wird dem Zeitintervall T5 zugeordnet.

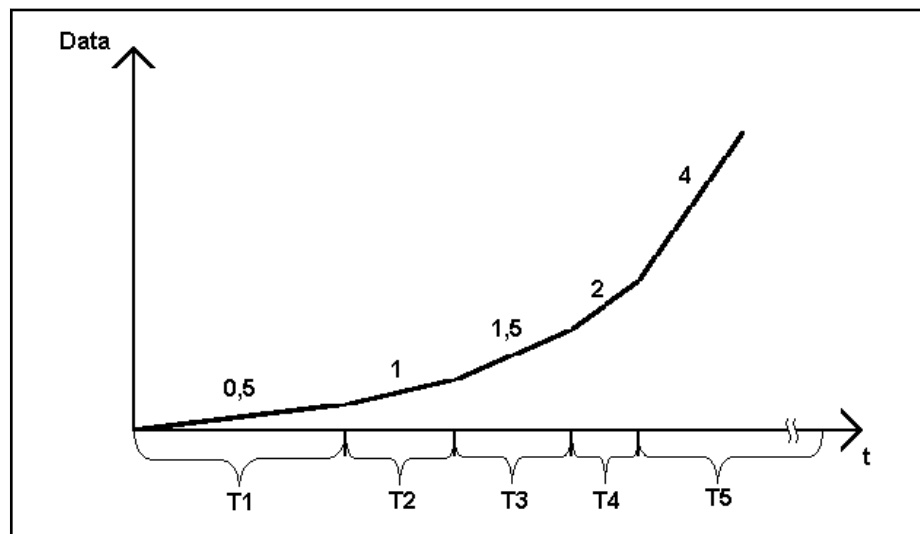


Abb. 4-10: Gewichtete Verstellung



Wird ein Zeitintervall von Null gewählt, wird der darauffolgende Wichtungsfaktor bis zum Abbruch des Verstellvorganges beibehalten.

Beispiel:

Ist "AdjTimeIntervals[2]" = 0, wird der Wichtungsfaktor von "AdjWeightFactors[3]" bis zum Abbruch beibehalten.

Die Berechnung der abgelaufenen Zeit zur Bestimmung des korrekten Zeitintervalls der Wichtungsfaktoren wird mit jedem neu beginnenden Verstellvorgang rückgesetzt. Sind die Eingänge "AdjWeightFactors" und "AdjTimeIntervals" nicht gesetzt, wird eine kontinuierliche Verstellung mit der durch die Eingänge "StepsPerSecond" und "StepWidth" vorgegebenen Änderungsgeschwindigkeit durchgeführt.



Für die Eingangswerte der Zeitintervalle (AdjTimeIntervals) müssen positive Werte angegeben werden. Die Wichtungsfaktoren können auch negative Werte erhalten.

Die zu beeinflussende Variable kann nur auf den vorgegebenen Voreinstellwert gesetzt werden, wenn sich dieser innerhalb des gültigen, durch "HighLimit" und "LowLimit" vorgegebenen Bereichs befindet.

Der Ausgang "InOperation" signalisiert, dass der Funktionsbaustein in Betrieb ist. Eine Verstellung findet jedoch nicht statt. Der Ausgang "Changing" zeigt an, dass der Verstellvorgang fehlerfrei abgearbeitet wurde. Der Wert der verstellten Variablen "Value" ist gültig.

Der Ausgang "Error" zeigt das Auftreten eines Fehlers während des Verstellvorgangs an. Detailangaben zu dem Fehler sind der Struktur "ErrorIdent" zu entnehmen.

Fehlerbehandlung

Der Funktionsbaustein erzeugt in Additional1/Additional2 der Tabelle "F_RELATED_TABLE", 16#0170, die folgenden Fehlermeldungen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR (16#0006)	16#0002	16#0000	Eingänge liegen außerhalb des zulässigen Bereichs
RESOURCE_ERROR (16#0003)	16#0004	16#0000	Antriebs-Firmware wird nicht unterstützt
STATE_MACHINE_ERROR (16#0005)	16#0006	16#0000	Ungültiger Zustand des Funktionsbausteins

Tab. 4-8: Fehlercodes FB IL_IncrementalAdjustType02

4.2.4 IL_IncrementalAdjustType01

Kurzbeschreibung

Der Funktionsbaustein IL_IncrementalAdjustType01 ermöglicht die inkrementelle Verstellung einer REAL-Variablen über binäre Eingänge.



Der Funktionsbaustein "MB_IncrementalAdjustType01 (ML_Tech-Base-Bibliothek)" wurde mit Version 14VRS in "IL_IncrementalAdjustType01" umbenannt.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek
MLC 15VRS	RIL_Uilities

Tab. 4-9: Referenztablette Funktionsbaustein IL_IncrementalAdjustType01

Schnittstellenbeschreibung

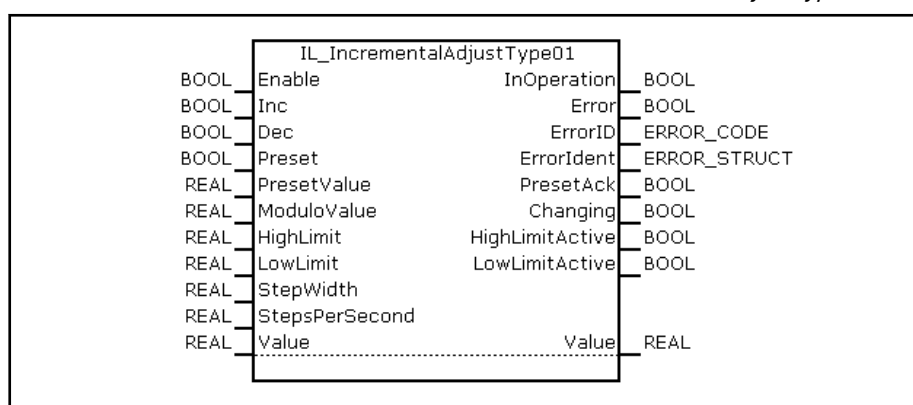


Abb. 4-11: Funktionsbaustein IL_IncrementalAdjustType01

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Freigabe des Funktionsbausteins (zyklisch, zustands-gesteuert)
	Inc	BOOL	Beeinflusste Variable erhöhen
	Dec	BOOL	Beeinflusste Variable verringern
	Preset	BOOL	Voreinstellwert (PresetValue) setzen
	PresetValue	REAL	Vorgabewert für den Voreinstellwert
	ModuloValue	REAL	Modulowert zur rotatorischen Berechnung
	HighLimit	REAL	Maximaler Ausgangswert der zu beeinflussenden Va-riablen
	LowLimit	REAL	Minimaler Ausgangswert der zu beeinflussenden Va-riablen
	StepWidth	REAL	Schrittweite
	StepsPerSecond	REAL	Schritte pro Sekunde
VAR_OUTPUT	InOperation	BOOL	Funktionsbaustein aktiv. Die Ausgänge "HighLimitActive" und "LowLimitActive" sind gültig
	Error	BOOL	Berechnung der zu beeinflussenden Variablen "Value" wurde mit Fehler abgeschlossen; die Ausgangsvariable "ErrorIdent" ist gültig
	ErrorID	ERROR_CODE	Fehlerkurzbeschreibung
	ErrorIdent	ERROR_STRUCT	Detailbeschreibung der Diagnose im Fehlerfall
	PresetAck	BOOL	Der Wert des Eingangs "PresetValue" wurde über-nommen
	Changing	BOOL	Berechnung der zu beeinflussenden Variablen erfolgt. Ausgangsvariable "Value" ist gültig
	HighLimitActive	BOOL	Maximaler Ausgangswert der zu beeinflussenden Va-riablen wurde erreicht
VAR_IN_OUT	Value	REAL	Die zu beeinflussende Variable

Tab. 4-10: Schnittstellenvariablen FB IL_IncrementalAdjustType01

Signal-Zeit-Diagramm

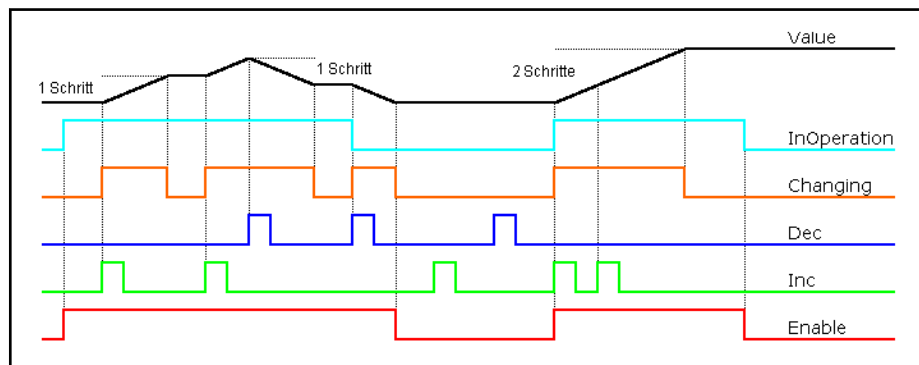


Abb. 4-12: Inkrementelle Parametervstellung

Funktionsbeschreibung Die Verstellung der beeinflussten Variablen "Value" kann sowohl in positiver als auch in negativer Richtung erfolgen. Durch die Grenzwerte wird ein Gültigkeitsbereich für die Beeinflussung der Variablen festgelegt. Die Vorgabe eines Modulo-Wertes erlaubt die Verstellung um innerhalb des Bezugssystems der Achse zu sein. Durch Festlegen der Schrittweite und der Anzahl der Schritte pro Sekunde wird die Änderungsgeschwindigkeit der beeinflussten Variablen bestimmt. Zudem besteht die Möglichkeit, einen vorgegebenen Wert ("PresetValue"), der sich im zulässigen Betriebsbereich befindet, direkt auf die beeinflusste Variable zu schreiben.

Nach der Bearbeitungsfreigabe über "Enable" kann die zu beeinflussende Variable über die Eingänge "Inc" (in positiver Richtung) und "Dec" (in negativer Richtung) verändert werden. Solange der Eingang "Inc" bzw. "Dec" gesetzt ist, wird die Größe der einstellbaren Variablen mit der vorgegebenen Geschwindigkeit kontinuierlich erhöht bzw. verringert. ("StepsPerSecond" * "StepWidth").



Die zu beeinflussende Variable kann nur auf den vorgegebenen Voreinstellwert gesetzt werden, wenn sich dieser innerhalb des gültigen, durch "HighLimit" und "LowLimit" vorgegebenen Bereichs befindet.

Der Ausgang "InOperation" signalisiert, dass der Funktionsbaustein in Betrieb ist. Eine Verstellung findet jedoch nicht statt. Der Ausgang "Changing" zeigt an, dass der Verstellvorgang fehlerfrei abgearbeitet wurde. Der Wert der verstellten Variablen "Value" ist gültig.

Ein während der Bearbeitung des Funktionsbausteins auftretender Fehler wird durch den Ausgang "Error" = TRUE signalisiert. Die Detailangaben zu dem Fehler sind der Ausgangsstruktur "ErrorIdent" zu entnehmen.

Fehlerbehandlung Der Funktionsbaustein erzeugt in Additional1/Additional2 der Tabelle "F_RELATED_TABLE", 16#0170, die folgenden Fehlermeldungen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR (16#0006)	16#0002	16#0000	Eingänge liegen außerhalb des zulässigen Bereichs
RESOURCE_ERROR (16#0003)	16#0004	16#0000	Antriebs-Firmware wird nicht unterstützt
STATE_MACHINE_ERROR (16#0005)	16#0006	16#0000	Ungültiger Zustand des Funktionsbausteins

Tab. 4-11: Fehlercodes FB IL_IncrementalAdjustType01

4.2.5 Parametrierung der Funktionsbausteine zum Vertippen von Variablen

Erforderliche Hardware Die folgenden Rexroth-Hardware-Komponenten werden benötigt:

- MLC Hardware

Erforderliche Software Folgende PC-Software ist zu verwenden:

- IndraWorks MLC 15VRS

Parametrierung der Funktionsbausteine zum Vertippen von Variablen

Es ist keine besondere Parametrierung erforderlich.

4.3 Basisklassen für Funktionsbausteinentwicklung (Base FB)

4.3.1 Einführung und Übersicht

Anwendungsgebiete und Funktionen / Funktionsbausteine von RIL_Uilities

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine	Beschreibung
Basisklassen für Funktionsbausteinentwicklung (Base FB)	
Basisfunktionsbaustein IL_EnableDoneBase , Seite 87	implementiert das Interface eines Funktionsbausteins mit Enable-Eingang und Done-, Active- und Standard-Fehlerausgängen
Basisfunktionsbaustein IL_EnableInOpBase , Seite 90	implementiert das Interface eines Funktionsbausteins mit Enable-Eingang und InOperation- und Standard-Fehlerausgängen
Basisfunktionsbaustein IL_EnableInOpShutdownBase , Seite 92	implementiert das Interface eines Funktionsbausteins mit Enable-Eingang und InOperation-, Shutdown- und Standard-Fehlerausgängen
Basisfunktionsbaustein IL_ExecuteDoneBase , Seite 97	implementiert das Interface eines Funktionsbausteins mit Execute-Eingang und Done-, Active- und Standard-Fehlerausgängen
Basisfunktionsbaustein IL_ExecuteInOpBase , Seite 99	implementiert das Interface eines Funktionsbausteins mit Execute-Eingang und InOperation-, Active- und Standard-Fehlerausgängen
Basisfunktionsbaustein IL_ExecuteDoneAbortedBase , Seite 102	implementiert das Interface eines Funktionsbausteins mit Execute-Eingang und Done-, Active-, CommandAborted- und Standard-Fehlerausgängen
Basisfunktionsbaustein IL_ExecuteDoneAbortedShutdownBase , Seite 105	implementiert das Interface eines Funktionsbausteins mit Execute-Eingang und Done-, Active-, Shutdown-, CommandAborted- und Standard-Fehlerausgängen

Tab. 4-12: *RIL_Uilities, Base FBs*

Die Basisfunktionsbausteine

- `IL_EnableDoneBase`
- `IL_EnableInOpBase`
- `IL_EnableInOpShutdownBase`
- `IL_ExecuteDoneBase`
- `IL_ExecuteInOpBase`
- `IL_ExecuteDoneAbortedBase`
- `IL_ExecuteDoneAbortedShutdownBase`

erleichtern die Entwicklung von Funktionsbausteinen. Sie implementieren das Grundgerüst des Funktionsbausteins sowie das Verhalten der Standard-Ein- und Ausgänge.

Für beide Funktionsblocktypen ist jeweils eine State-Machine vorgegeben, die die Standard-Ein- und Ausgänge bedient. Der Anwendercode kann durch Implementierung von namentlich vorgegebenen Methoden in die zugehörige State-Machine eingefügt werden. Zusätzlich sind Hilfsmethoden implementiert, die das Setzen von Fehlercodes erleichtern.

Um einen Basisfunktionsbaustein zu verwenden, muss dieser bei der Definition des eigenen Funktionsbausteins über das Schlüsselwort `EXTENDS` refe-

renziert werden. Dieses Vorgehen nennt man auch einen Funktionsbaustein ableiten. Der eigene Funktionsbaustein enthält dann die Ein- und Ausgänge des Basisfunktionsbausteins sowie nachfolgend alle Ein- und Ausgänge des eigenen Funktionsbausteins. Sämtliche im Basisfunktionsbaustein implementierte Methoden können im eigenen (abgeleiteten) Funktionsbaustein genutzt werden. Dies nennt man auch Vererben von Methoden.

Überschreibbare Methoden

Die Basisfunktionsbausteine definieren einige Methoden, die durch die interne Zustandsmaschine aufgerufen werden. Diese Methoden können neu definiert werden, um eigene Funktionalität zu implementieren. Man nennt dies auch Überschreiben von Methoden.

Im folgenden sind alle überschreibbaren Methoden der Basisfunktionsbausteine aufgeführt.

Methode	Beschreibung
mReset()	In dieser Methode können interne Funktionsbausteine und selbst definierte Ausgänge zurückgesetzt werden. Sie wird beim Übergang in den Zustand STATE_IDLE aufgerufen. Die Ausgänge "Active", "InOperation" und "Error" werden intern zurückgesetzt. Diese Methode ist optional zu implementieren.
mInit(Reset: BOOL): BOOL	In dieser Methode werden die Vorbereitungen für die eigentliche Aufgabe des Funktionsbausteins implementiert. Diese Methode wird im bzw. beim Übergang in den Zustand STATE_INIT aufgerufen. Beim Übergang ist "Reset" TRUE. In mInit() erfolgt die Initialisierung des Funktionsbausteins. Solange die Initialisierung nicht abgeschlossen wurde, muss mInit() TRUE zurückgeben. Ist die Initialisierung abgeschlossen, muss mInit() mit FALSE zurückkehren. In mInit() kann eine untergeordnete Zustandsmaschine implementiert werden. Diese Methode ist optional zu implementieren.
mInOp(Reset: BOOL): BOOL	In dieser Methode wird die eigentliche Aufgabe des Funktionsbausteins implementiert. Diese Methode wird im bzw. beim Übergang in den Zustand STATE_INOP aufgerufen. Beim Übergang in den Zustand STATE_INOP ist das Argument "Reset" TRUE. Bei einem Funktionsbaustein mit "Execute"-Eingang wird mInOp(FALSE) solange aufgerufen, bis FALSE zurückgegeben wird. Dann wird automatisch der Zustand STATE_DONE aktiviert, falls es sich um eine Ableitung eines Funktionsbausteins mit "Done"-Ausgang handelt.
mError()	Diese Methode wird im Fehlerfall von mSetError() aufgerufen. Hier können boolesche Ausgänge zurückgesetzt werden. "Active", "InOperation" und die Fehlerausgänge werden schon durch mSetError() bedient. Diese Methode ist optional zu implementieren.

Methode	Beschreibung
mIdle()	Diese Methode wird in den Zuständen STATE_IDLE, STATE_INIT und STATE_DONE, STATE_ERROR, jedoch nicht in STATE_INOP aufgerufen. Hier können Funktionsbausteine aufgerufen werden, die über eine kontinuierliche Zeitmessung verfügen und deshalb immer aufgerufen werden müssen.
mShutdown(Reset: BOOL): BOOL	In dieser Methode wird der Endprozess des Funktionsbausteins implementiert. Diese Methode wird im bzw. beim Übergang in den Zustand STATE_SHUTDOWN oder STATE_ERROR aufgerufen. Beim Übergang ist das Argument "Reset" TRUE. Solange der Endprozess nicht abgeschlossen wird, muss mShutdown() aufgerufen werden und TRUE zurückgeben. Ist der Endprozess abgeschlossen, muss mShutdown() mit FALSE zurückkehren.

Hilfsfunktionen Hilfsfunktionen sind durch die Basisfunktionsbausteine bereit gestellte Methoden, die in eigenen Implementierungen genutzt werden können. Sie dürfen nicht durch eigene Implementierungen überschrieben werden. Derzeit sind zwei Methoden zum Setzen des Fehlerzustands implementiert.

Methode	Beschreibung
mSetError(enErrorID: ERROR_CODE, enTable: ERROR_TABLE, dwAdditional1: DWORD, dwAdditional2: DWORD)	Diese Methode setzt den angegebenen Fehlercode an den Ausgängen "ErrorID" und "ErrorIdent" und aktiviert den "Error"-Ausgang. Danach wird mError() aufgerufen. Der Funktionsblock wechselt in den Zustand STATE_ERROR. Nach dem Aufruf von mSetError() sollte ein Rücksprung mittels RETURN erfolgen, um die weitere Bearbeitung des Codes nach einem Fehler zu unterbinden.
mSetErrorIdent(enErrorID: ERROR_CODE, stErrorIdent: ERROR_STRUCT)	Diese Methode setzt den angegebenen Fehlercode an den Ausgängen "ErrorID" und "ErrorIdent" und aktiviert den "Error"-Ausgang. Danach wird mError() aufgerufen. Der Funktionsblock wechselt in den Zustand STATE_ERROR. Diese Funktion eignet sich zum Übernehmen des Fehlercodes aus einem anderen Funktionsblock. Nach dem Aufruf von mSetErrorIdent() sollte ein Rücksprung mittels RETURN erfolgen, um die weitere Bearbeitung des Codes nach einem Fehler zu unterbinden.
mSetCommandAborted()	Diese Methode ist nur für den Basisfunktionsbaustein IL_ExecuteDoneAborted-Base verfügbar. Der Aufruf dieser Methode setzt den Ausgang "Command-Aborted" und versetzt den Funktionsbaustein in den Zustand Command-Aborted.

Anwendung der Basisfunktionsbausteine

Die Basisfunktionsbausteine können durch Ableitung in eigenen Funktionsbausteinen erweitert werden. Hierfür muss bei der Erstellung des neuen Funktionsbausteins der Punkt "Erweitert:" aktiviert sein und der Basisfunktionsbaustein im entsprechenden Textfeld eingetragen werden.

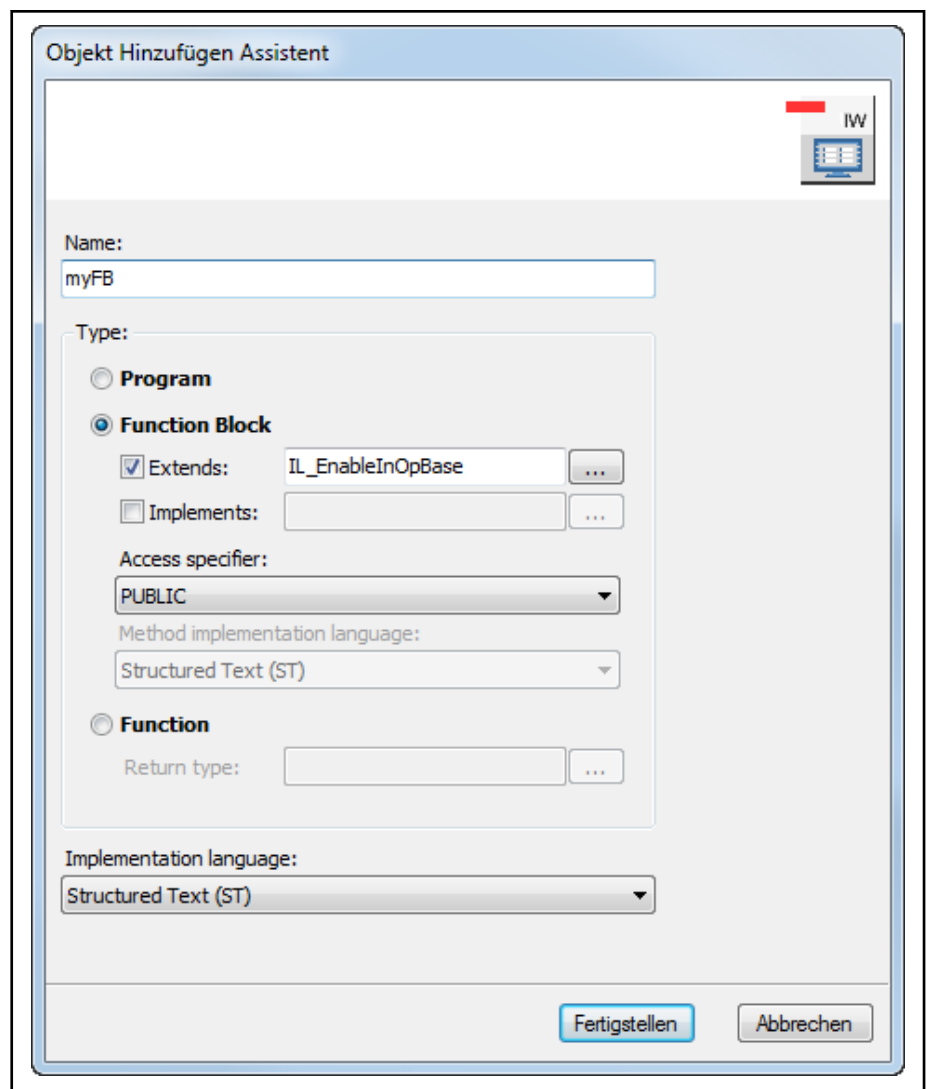


Abb. 4-13: Nutzung eines Basis-Funktionsbausteins

Das Grundprinzip ist die Vererbung. Der neue Funktionsbaustein enthält alle bereits im Basisbaustein implementierten Variablen und Methoden. Durch Neuimplementierung der vorhandenen Methoden kann eigene Funktionalität hinzugefügt werden. Die im vom Basisbaustein abgeleiteten Funktionsbaustein neu deklarierten Ein- und Ausgänge erscheinen unterhalb der bereits im Basisbaustein definierten Schnittstellen. Es müssen also nur noch die neuen Ein- und Ausgänge definiert werden.

Im Implementationsteil des abgeleiteten Funktionsbausteins muss zwingend der Aufruf von "FB_Main()" erfolgen. Weitere Implementierungen sind an dieser Stelle nicht erlaubt.

```

1  FUNCTION_BLOCK myFB EXTENDS IL_EnableInOpBase
2  VAR_INPUT
3      MyInput: REAL;
4  END_VAR
5  VAR_OUTPUT
6      MyOutput: REAL;
7  END_VAR
8  VAR
9      iInitState: INT;
10 END_VAR
11
1  FB_Main();

```

Abb. 4-14: Implementierung eines Funktionsbausteins auf Basis des Basisfunktionsbausteins IL_EnableInOpBase

Daraus ergibt sich die Schnittstelle des erstellten Funktionsbausteins:

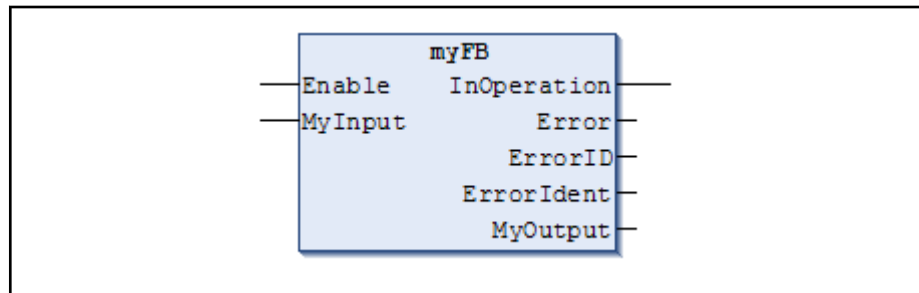


Abb. 4-15: Schnittstelle des Beispielfunktionsbausteins auf Basis von IL_EnableInOpBase

Die Implementierung der eigentlichen Funktion erfolgt in den überschreibbaren Methoden "mInit()" und "mInOp()". Nicht benötigte Methoden müssen auch nicht implementiert werden.

Die folgende Grafik demonstriert die Implementierung der mInit()-Methode. Die im Beispiel dargestellte Zustandsmaschine ist optional.

```

1  METHOD mInit : BOOL
2  VAR_INPUT
3      Reset: BOOL;
4  END_VAR
5  VAR_CONSTANT
6      // define error codes
7      ADD1_STATEMACHINE: DWORD := 16#00000006;
8      ADD2_STATEMACHINE: DWORD := 16#00000000;
9  END_VAR

1 IF Reset=TRUE THEN
2     // First call => entering Init state
3     iInitState := 0; // Reset state machine
4 END_IF

5
6 mInit := TRUE; // Call me again
7
8 CASE iInitState OF
9     0:
10        // Write your own code here
11        iInitState := iInitState + 1;
12     1:
13        // Write your own code here
14        mInit := FALSE; // Init finished => don't call me again
15 ELSE
16     // Undefined state => error
17     mSetError(STATE_MACHINE_ERROR,
18             F_RELATED_TABLE,
19             ADD1_STATEMACHINE,
20             ADD2_STATEMACHINE);
21 END_CASE

```

Abb. 4-16: Implementierung einer mInit()-Methode

4.3.2 Basisfunktionsbaustein IL_EnableDoneBase

Kurzbeschreibung

Dieser Basisfunktionsbaustein (Basisklasse) implementiert das Interface eines Funktionsbausteins mit "Enable"-Eingang und "Done"- sowie "Active"-Ausgang. Weiterhin werden die Standard-Fehlerausgänge angelegt. Der Baustein implementiert eine zu den Standard-Ein- und Ausgängen passende Zustandsmaschine.

Funktionsbausteine dieses Typs werden vorrangig für zyklische Prozesse, wie z.B. zyklische Abfrage eines externen Geräts, verwendet.

Dieser Funktionsbaustein dient als Basisklasse für eigene Implementierungen. Er muss über das Schlüsselwort "EXTENDS" eingebunden werden.

Schnittstellenbeschreibung

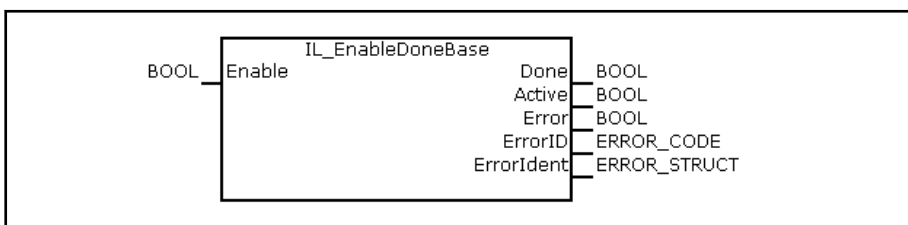


Abb. 4-17: Funktionsbaustein IL_EnableDoneBase

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe (zyklisch, zustandsgesteuert)
VAR_OUTPUT	Done	BOOL	Der Baustein hat ein Ergebnis ausgegeben
	Active	BOOL	Der Funktionsbaustein arbeitet
	Error	BOOL	Zeigt an, dass ein Fehler aufgetreten ist
	ErrorID	ERROR_CODE	Fehlerursache
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerursache

Tab. 4-13: Funktionsbaustein IL_EnableDoneBase

Funktionsbeschreibung

Diese Basisklasse implementiert das Interface eines Funktionsbausteins mit "Enable"-Eingang und "Done"- sowie "Active"-Ausgang. Weiterhin werden die Standard-Fehlerausgänge angelegt. Der Baustein implementiert eine zu den Standard Ein- und Ausgängen passende Zustandsmaschine.

Dieser Funktionsbaustein dient als Basisklasse für eigene Implementierungen. Er muss über das Schlüsselwort "EXTENDS" eingebunden werden.

Funktionsbausteine mit "Enable"-Eingang werden vorrangig für sich zyklisch wiederholende Aufgaben, eingesetzt.

Dieser Basisfunktionsbaustein implementiert den Eingang "Enable" sowie die Ausgänge "Done", "Active", "Error", "ErrorID" und "ErrorIdent". Die Abarbeitung startet bei steigender Flanke an "Enable" und läuft nach der Initialisierungsphase so lange, bis entweder "Enable" zurückgesetzt wird, oder ein Fehler auftritt. Nach Durchlaufen der Initialisierungsphase wird "Active" gesetzt, um anzuzeigen, dass der Funktionsbaustein seine Aufgabe durchführt. Nach Abschluss der Aufgabe wird "Done" für einen Zyklus gesetzt. Danach beginnt eine erneute Abarbeitung.

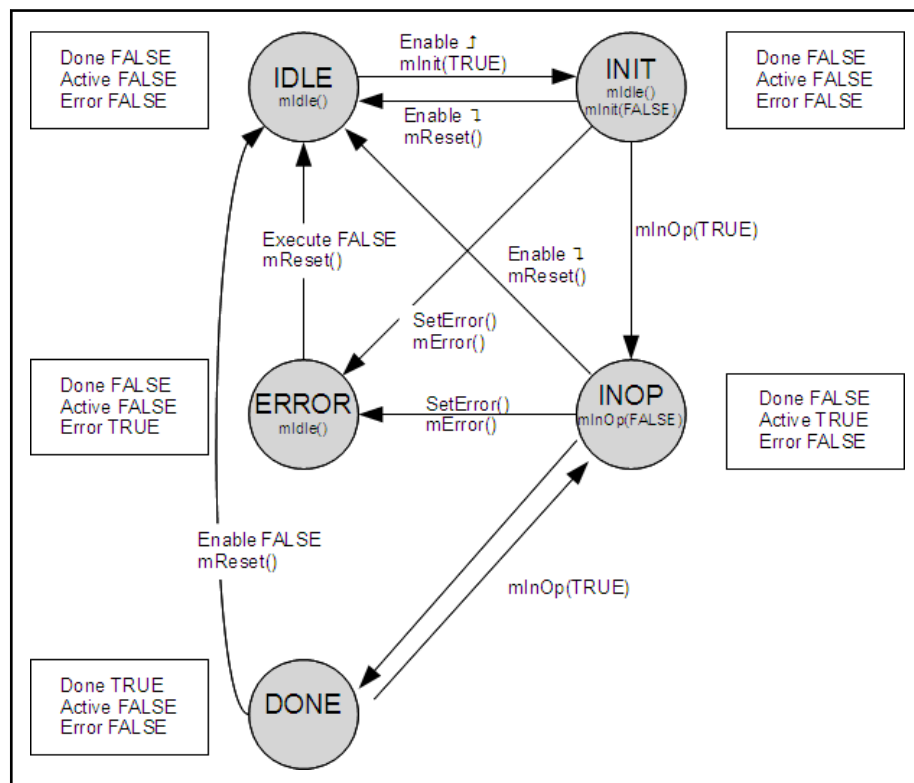


Abb. 4-18: Zustandsmaschine eines Funktionsbausteins mit Enable-Eingang und Done-Ausgang

Bausteine dieses Typs haben fünf interne Zustände (siehe auch [IL_FB_STATES](#) auf Seite 110):

STATE_IDLE	Dieser Zustand stellt den Startzustand des Funktionsbausteins dar. Er wird aktiviert, wenn "Enable" False ist. Beim Betreten des Zustands wird mReset() ausgeführt. Im Zustand wird mIdle() ausgeführt. Wird "Enable" gesetzt, wird in den Zustand STATE_INIT gewechselt.
STATE_INIT	Zustand dient dem Hochlauf des Funktionsbausteins. Er wird vor der eigentlichen zyklischen Operation durchlaufen. Beim Betreten des Zustands wird mInit(Reset:=TRUE) aufgerufen. Im Zustand werden mInit(Reset:=FALSE) und mIdle() aufgerufen. Gibt mInit() False zurück, wird in den Zustand STATE_INOP gewechselt. Wird mSetError() oder mSetErrorIdent() aufgerufen, wird in den Zustand STATE_ERROR gewechselt. Wird "Enable" zurückgesetzt, wird in den Zustand STATE_IDLE gewechselt.
STATE_INOP	Hier wird die zyklische Aufgabe ausgeführt. Der Ausgang "Active" ist True, solange dieser Zustand aktiv ist. Beim Betreten des Zustands wird mInOp(Reset:=TRUE) aufgerufen. Im Zustand wird mInOp(Reset:=FALSE) aufgerufen. Wird mSetError() oder mSetErrorIdent() aufgerufen, wird in den Zustand STATE_ERROR gewechselt. Wird "Enable" zurückgesetzt, wird in den Zustand STATE_IDLE gewechselt.
STATE_DONE	Zustand ist nach erfolgreicher Abarbeitung der Aufgabe aktiv. Er wird nach einem Zyklus wieder verlassen. Danach wird wieder STATE_INOP aktiv.
STATE_ERROR	Dies ist der Fehlerzustand des Funktionsbausteins. Der Fehlerausgang "Error" ist in diesem Zustand aktiv und der Fehlercode wird an "ErrorID" bzw. "ErrorIdent" angezeigt. Beim Betreten des Zustands wird mError() aufgerufen. Im Zustand wird mIdle() aufgerufen. Wird "Enable" zurückgesetzt, wird in den Zustand STATE_IDLE gewechselt.

Signal-Zeit Diagramm Aus der oben beschriebenen State-Machine ergibt sich das folgende Zeitverhalten:

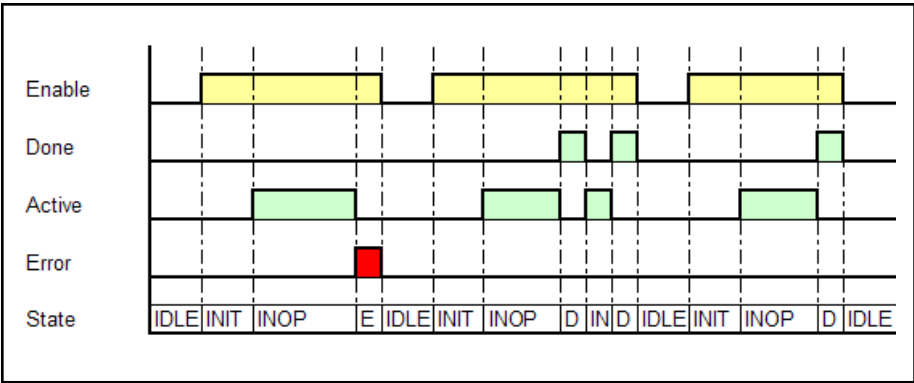


Abb. 4-19: Zeitverhalten eines Funktionsbausteins auf Basis von IL_EnableDoneBase

Fehlercodes Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann die folgenden Fehlermeldungen in Additional1/Additional2 erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#00000006	16#0000FFFF	Ungültiger Zustand der Zustandsmaschine

Tab. 4-14: Fehlercodes, die von IL_EnableDoneBase generiert werden

4.3.3 Basisfunktionsbaustein IL_EnableInOpBase

Kurzbeschreibung Dieser Basisfunktionsbaustein (Basisklasse) implementiert das Interface eines Funktionsbausteins mit "Enable"-Eingang und "InOperation"-Ausgang. Weiterhin werden die Standard-Fehlerausgänge angelegt. Der Baustein implementiert eine zu den Standard-Ein- und Ausgängen passende Zustandsmaschine.

Funktionsbausteine dieses Typs werden vorrangig für kontinuierliche Prozesse, wie z.B. Regler, verwendet.

Dieser Funktionsbaustein dient als Basisklasse für eigene Implementierungen. Er muss über das Schlüsselwort "EXTENDS" eingebunden werden.

Schnittstellenbeschreibung

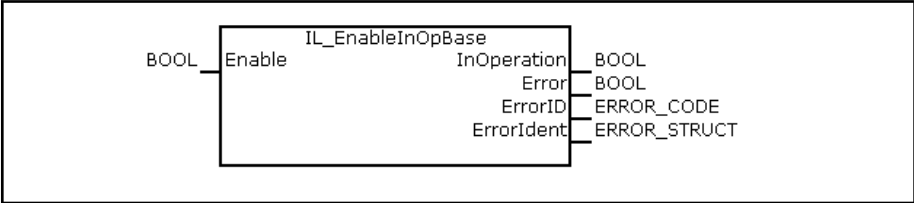


Abb. 4-20: Funktionsbaustein IL_EnableInOpBase

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe (zyklisch, zustandsgesteuert)
VAR_OUTPUT	InOperation	BOOL	Der Baustein arbeitet an seiner eigentlichen Aufgabe
	Error	BOOL	Zeigt an, dass ein Fehler aufgetreten ist
	ErrorID	ERROR_CODE	Fehlerursache
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerursache

Tab. 4-15: Funktionsbaustein IL_EnableInOpBase

Funktionsbeschreibung Diese Basisklasse implementiert das Interface eines Funktionsbausteins mit "Enable"-Eingang und "InOperation"-Ausgang. Weiterhin werden die Stan-

dard-Fehlerausgänge angelegt. Der Baustein implementiert eine zu den Standard Ein- und Ausgängen passende Zustandsmaschine.

Dieser Funktionsbaustein dient als Basisklasse für eigene Implementierungen. Er muss über das Schlüsselwort "EXTENDS" eingebunden werden.

Dieser Basisfunktionsbaustein implementiert den Eingang "Enable" sowie die Ausgänge "InOperation", "Error", "ErrorID" und "ErrorIdent". Die Abarbeitung startet bei steigender Flanke an "Enable" und läuft nach der Initialisierungsphase so lange, bis entweder "Enable" zurückgesetzt wird, oder ein Fehler auftritt. Nach Durchlaufen der Initialisierungsphase wird "InOperation" gesetzt, um anzuzeigen, dass der Funktionsbaustein seine zyklische Aufgabe durchführt.

Funktionsbausteine mit "Enable"-Eingang und "InOperation"-Ausgang werden vorrangig für sich zyklisch wiederholende Aufgaben, wie z.B. Regelung, eingesetzt.

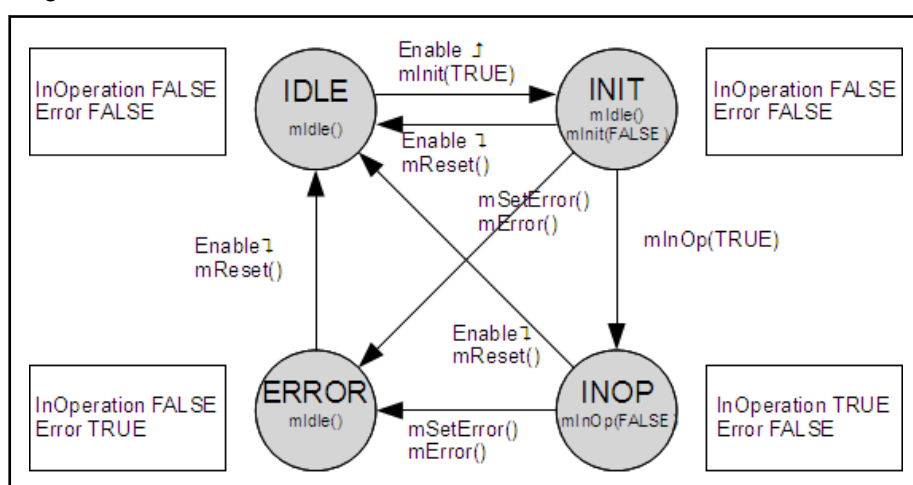


Abb. 4-21: Zustandsmaschine eines Funktionsbausteins mit Enable-Eingang und InOperation-Ausgang

Bausteine dieses Typs haben vier interne Zustände (siehe auch [IL_FB_STATES](#) auf Seite 110):

STATE_IDLE Dieser Zustand stellt den Startzustand des Funktionsbausteins dar. Er wird aktiviert, wenn "Enable" False ist. Beim Betreten des Zustands wird `mReset()` ausgeführt. Im Zustand wird `mIdle()` ausgeführt.

Wird "Enable" gesetzt, wird in den Zustand STATE_INIT gewechselt.

STATE_INIT Dieser Zustand dient dem Hochlauf des Funktionsbausteins. Er wird vor der eigentlichen zyklischen Operation durchlaufen. Beim Betreten des Zustands wird `mInit(Reset:=TRUE)` aufgerufen. Im Zustand werden `mInit(Reset:=FALSE)` und `mIdle()` aufgerufen.

Gibt `mInit()` False zurück, wird in den Zustand STATE_INOP gewechselt. Wird `mSetError()` oder `mSetErrorIdent()` aufgerufen, wird in den Zustand STATE_ERROR gewechselt. Wird "Enable" zurückgesetzt, wird in den Zustand STATE_IDLE gewechselt.

STATE_INOP

Hier wird die zyklische Aufgabe ausgeführt. Der Ausgang "In-Operation" ist True, solange dieser Zustand aktiv ist. Beim Betreten des Zustands wird `mInOp(Reset:=TRUE)` aufgerufen. Im Zustand wird `mInOp(Reset:=FALSE)` aufgerufen.

Wird `mSetError()` oder `mSetErrorIdent()` aufgerufen, wird in den Zustand **STATE_ERROR** gewechselt. Wird "Enable" zurückgesetzt, wird in den Zustand **STATE_IDLE** gewechselt.

STATE_ERROR

Dies ist der Fehlerzustand des Funktionsbausteins. Der Fehlerausgang "Error" ist in diesem Zustand aktiv und der Fehlercode wird an "ErrorID" bzw. "ErrorIdent" angezeigt. Beim Betreten des Zustands wird `mError()` aufgerufen. Im Zustand wird `mIdle()` aufgerufen.

Wird "Enable" zurückgesetzt, wird in den Zustand **STATE_IDLE** gewechselt.

Signal-Zeit Diagramm

Aus der oben beschriebenen State-Machine ergibt sich das folgende Zeitverhalten:

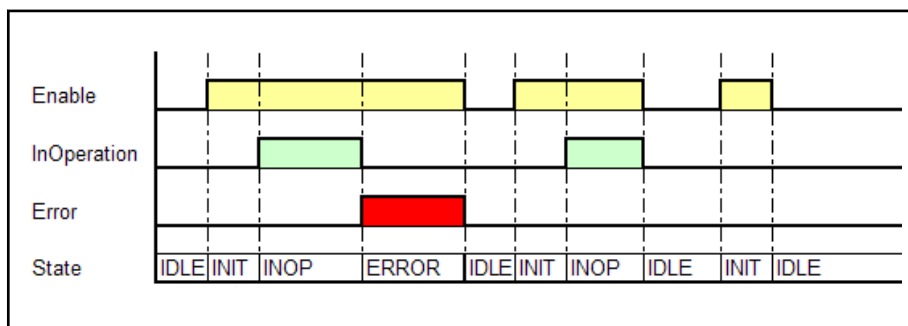


Abb. 4-22: Zeitverhalten eines Funktionsbausteins auf Basis von `IL_EnableInOpBase`

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann die folgenden Fehlermeldungen in Additional1/Additional2 erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#00000006	16#0000FFFF	Ungültiger Zustand der Zustandsmaschine

Tab. 4-16: Fehlercodes, die von `IL_EnableInOpBase` generiert werden

4.3.4 Basisfunktionsbaustein `IL_EnableInOpShutdownBase`

Kurzbeschreibung

Dieser Basisfunktionsbaustein (Basisklasse) implementiert das Interface eines Funktionsbausteins mit "Enable"-Eingang und "InOperation"- und "Shutdown"-Ausgang. Weiterhin werden die Standard-Fehlerausgänge angelegt. Der Baustein implementiert eine zu den Standard-Ein- und Ausgängen passende Zustandsmaschine.

Funktionsbausteine dieses Typs werden vorrangig für kontinuierliche Prozesse, wie z.B. Regler, verwendet.

Dieser Funktionsbaustein dient als Basisklasse für eigene Implementierungen. Er muss über das Schlüsselwort "EXTENDS" eingebunden werden.

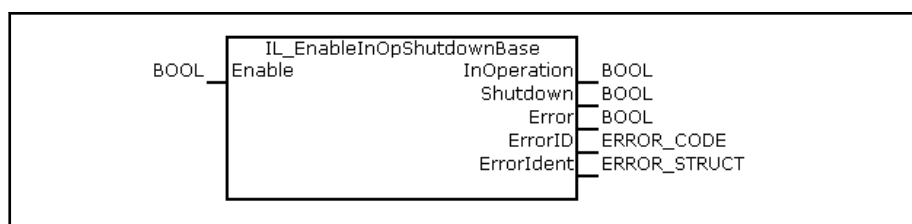
Schnittstellenbeschreibung

Abb. 4-23: Funktionsbaustein IL_EnableInOpShutdownBase

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe (zyklisch, zustandsgesteuert)
VAR_OUTPUT	InOperation	BOOL	Baustein arbeitet an seiner eigentlichen Aufgabe
	Shutdown	BOOL	Zeigt an, dass sich der Baustein gerade in einem definierten Endzustand befindet
	Error	BOOL	Zeigt an, dass ein Fehler aufgetreten ist
	ErrorID	ERROR_CODE	Fehlerursache
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerursache

Tab. 4-17: Funktionsbaustein IL_EnableInOpShutdownBase

Funktionsbeschreibung

Diese Basisklasse implementiert das Interface eines Funktionsbausteins mit "Enable"-Eingang und "InOperation"- und "Shutdown"-Ausgang. Weiterhin werden die Standard-Fehlerausgänge angelegt. Der Baustein implementiert eine zu den Standard Ein- und Ausgängen passende Zustandsmaschine.

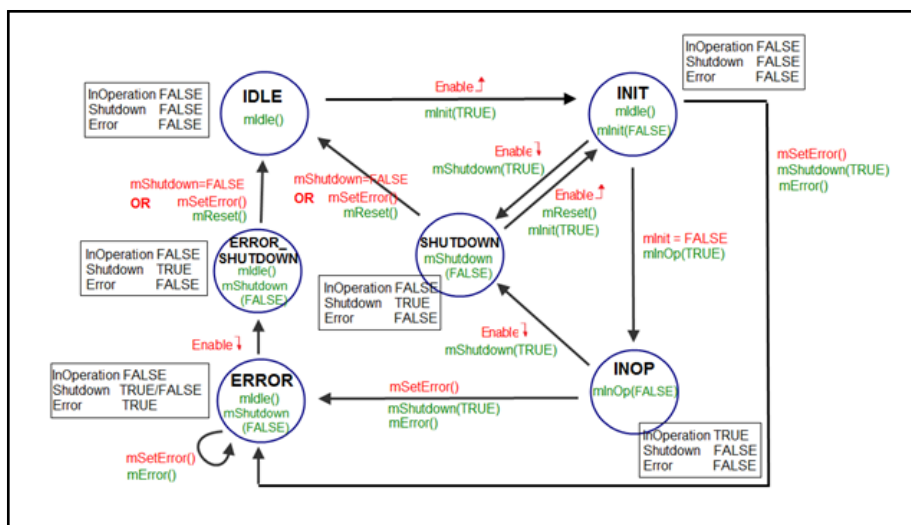
Dieser Funktionsbaustein dient als Basisklasse für eigene Implementierungen. Er muss über das Schlüsselwort "EXTENDS" eingebunden werden.

Dieser Basisfunktionsbaustein implementiert den Eingang "Enable" sowie die Ausgänge "InOperation", "Shutdown", "Error", "ErrorID" und "ErrorIdent".

Die Abarbeitung startet bei steigender Flanke an "Enable" und läuft nach der Initialisierungsphase so lange, bis entweder "Enable" zurückgesetzt wird, oder ein Fehler auftritt. Nach Durchlaufen der Initialisierungsphase wird "InOperation" gesetzt, um anzuzeigen, dass der Funktionsbaustein seine zyklische Aufgabe durchführt.

Der Ausgang "Shutdown" signalisiert, dass der Funktionsbaustein in den Zustand "SHUTDOWN" wechselt. Solange der Ausgang "Shutdown" gesetzt ist, muss der FB aufgerufen werden.

Funktionsbausteine mit "Enable"-Eingang und "InOperation"-Ausgang werden vorrangig für sich zyklisch wiederholende Aufgaben, wie z.B. Regelung, eingesetzt.



rot/oben Bedingungen der Zustandsumschaltung
 grün/unten Aufgerufene Methoden bei der Zustandsumschaltung

Abb. 4-24: Zustandsmaschine eines Funktionsbausteins mit Enable-Eingang und InOperation/Shutdown-Ausgang

Bausteine dieses Typs haben sechs interne Zustände (siehe auch IL_FB_STATES auf Seite 110):

- STATE_IDLE** Dieser Zustand stellt den Startzustand des Funktionsbausteins dar. Er wird aktiviert, wenn "Enable" False ist. Beim Betreten des Zustands wird mReset() ausgeführt. Im Zustand wird mIdle() ausgeführt.
- Wird "Enable" gesetzt, wird in den Zustand STATE_INIT gewechselt.
- STATE_INIT** Dieser Zustand dient dem Hochlauf des Funktionsbausteins. Er wird vor der eigentlichen zyklischen Operation durchlaufen. Beim Betreten des Zustands wird mInit(Reset:=TRUE) aufgerufen. Im Zustand werden mInit(Reset:=FALSE) und mIdle() aufgerufen.
- Gibt mInit() False zurück, wird in den Zustand STATE_INOP gewechselt. Wird mSetError() oder mSetErrorIdent() aufgerufen, wird in den Zustand STATE_ERROR gewechselt. Wird "Enable" zurückgesetzt, wird in den Zustand STATE_SHUTDOWN gewechselt.
- STATE_INOP** Hier wird die zyklische Aufgabe ausgeführt.
- Der Ausgang "InOperation" ist True, solange dieser Zustand aktiv ist. Beim Betreten des Zustands wird mInOp(Reset:=TRUE) aufgerufen. Im Zustand wird mInOp(Reset:=FALSE) aufgerufen.
- Wird mSetError() oder mSetErrorIdent() aufgerufen, wird in den Zustand STATE_ERROR gewechselt. Wird "Enable" zurückgesetzt, wird in den Zustand STATE_SHUTDOWN gewechselt.

STATE_SHUT-DOWN	Dies ist der Endzustand des Funktionsbausteins. Der Ausgang "Shutdown" ist TRUE, solange dieser Zustand aktiv ist. Beim Betreten des Zustands wird mShutDown(Reset:=TRUE) aufgerufen. Im Zustand wird mShutDown(Reset:=FALSE) aufgerufen. Gibt mShutDown() False zurück, wird in den Zustand STATE_IDLE gewechselt.
STATE_ERROR	Dies ist der Fehlerzustand des Funktionsbausteins. Der Fehlerausgang "Error" ist in diesem Zustand aktiv und der Fehlercode wird an "ErrorID" bzw. "ErrorIdent" angezeigt. Der Shutdown-Vorgang wird in diesem Zustand durchgeführt. Beim Betreten des Zustands wird mError() und mShutDown(Reset := TRUE) aufgerufen. Im Zustand wird mIdle() und mShutDown(Reset := FALSE) aufgerufen. Solange mShutDown() „TRUE“ zurückliefert, bleibt der Ausgang "Shutdown" "TRUE". Falls ein Fehler im Shutdown-Vorgang tritt auf, wird der Shutdown-Vorgang abgebrochen und der Ausgang „ShutDown“ zurückgesetzt. Wird "Enable" zurückgesetzt, wird in den Zustand STATE_ERROR_SHUTDOWN gewechselt
STATE_ERROR_SHUTDOWN	Die Voraussetzung für den Eintritt in diesem Zustand ist, dass der vorherige Zustand „STATE_ERROR“ war und eine fallende Flanke an "Enable". Falls der Shutdown-Vorgang noch nicht fertig ist, bleibt das Programm in diesem Zustand und wird mShutDown (Reset := FALSE) aufgerufen. Der Ausgang „ShutDown“ ist „TRUE“, solange dieser Zustand aktiv ist. Gibt mShutDown() False zurück, wird in den Zustand STATE_IDLE gewechselt. Falls ein Fehler im Shutdown-Vorgang auftritt, wird der Shutdown-Vorgang abgebrochen. Der Ausgang "Error" wird für einen Zyklus gesetzt. Der Ausgang "ShutDown" wird zurückgesetzt. Die State-Machine wechselt in den Zustand STATE_IDLE. Falls der Shutdown-Vorgang schon in STATE_ERROR durchgeführt wurde, wird mShutDown() in diesem Zustand nicht aufgerufen. Es wird sofort in den STATE_IDLE gewechselt.

Signal-Zeit Diagramm Aus der oben beschriebenen State-Machine ergibt sich das folgende Zeitverhalten:

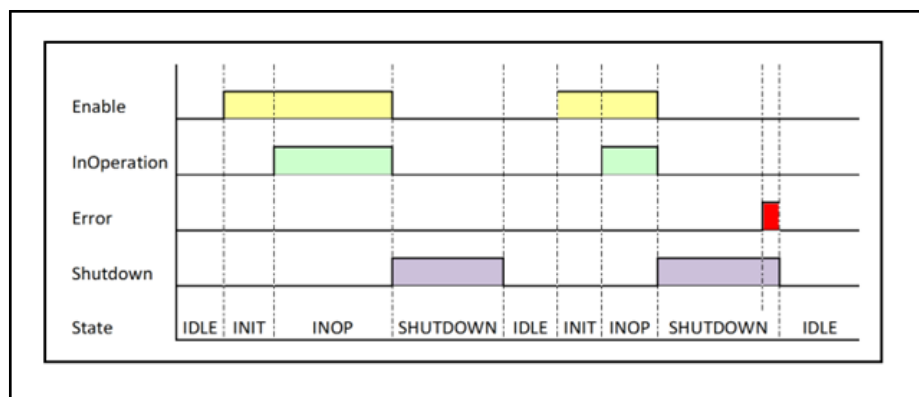


Abb. 4-25: Zeitverhalten eines Funktionsbausteins auf Basis von IL_EnableInOpShutBase (I)

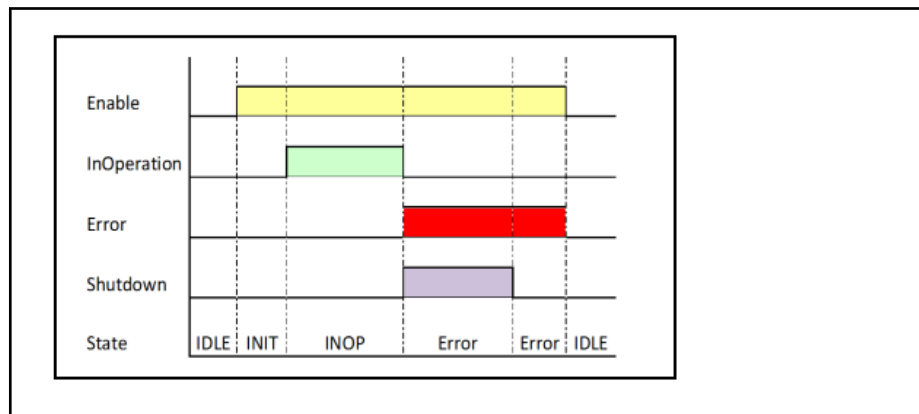


Abb. 4-26: Zeitverhalten eines Funktionsbausteins auf Basis von IL_EnableInOpShutBase (II)

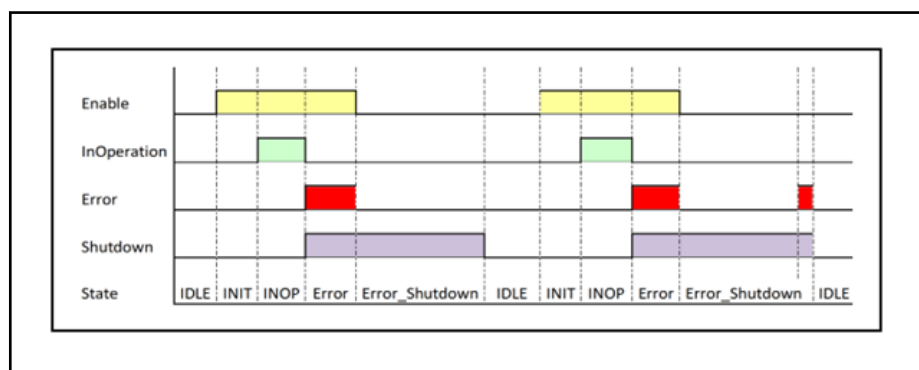


Abb. 4-27: Zeitverhalten eines Funktionsbausteins auf Basis von IL_EnableInOpShutBase (III)

Fehlercodes Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann die folgenden Fehlermeldungen in Additional1/Additional2 erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#00000006	16#0000FFFF	Ungültiger Zustand der Zustandsmaschine

Tab. 4-18: Fehlercodes, die von IL_EnableInOpShutdownBase generiert werden

4.3.5 Basisfunktionsbaustein IL_ExecuteDoneBase

Kurzbeschreibung Dieser Basisfunktionsbaustein (Basisklasse) implementiert das Interface eines Funktionsbausteins mit "Execute"-Eingang und "Done"- sowie "Active"-Ausgang. Weiterhin werden die Standard-Fehlerausgänge angelegt. Der Baustein implementiert eine zu den Standard-Ein- und Ausgängen passende Zustandsmaschine.

Funktionsbausteine dieses Typs werden für einmalige Aufgaben, die mehrere Zyklen Bearbeitungszeit benötigen, verwendet. Ein Beispiel ist die Initialisierung einer externen Hardware.

Dieser Funktionsbaustein dient als Basisklasse für eigene Implementierungen. Er muss über das Schlüsselwort "EXTENDS" eingebunden werden.

Schnittstellenbeschreibung

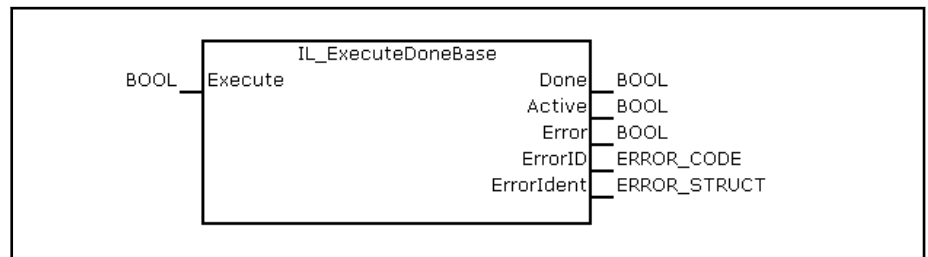


Abb. 4-28: Funktionsbaustein IL_ExecuteDoneBase

	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe (zyklisch, flankengesteuert)
VAR_OUTPUT	Done	BOOL	Der Baustein hat die Bearbeitung ohne Fehler abgeschlossen
	Active	BOOL	Der Funktionsbaustein wurde gestartet
	Error	BOOL	Zeigt an, dass ein Fehler aufgetreten ist
	ErrorID	ERROR_CODE	Fehlerursache
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerursache

Tab. 4-19: Funktionsbaustein IL_ExecuteDoneBase

Funktionsbeschreibung Diese Basisklasse implementiert das Interface eines Funktionsbausteins mit "Execute"-Eingang und "Done"- sowie "Active"-Ausgang. Weiterhin werden die Standard-Fehlerausgänge angelegt. Der Baustein implementiert eine zu den Standard Ein- und Ausgängen passende Zustandsmaschine.

Dieser Funktionsbaustein dient als Basisklasse für eigene Implementierungen. Er muss über das Schlüsselwort "EXTENDS" eingebunden werden.

Dieser Basisfunktionsbaustein implementiert den Eingang "Execute" sowie die Ausgänge "Done", "Active", "Error", "ErrorID" und "ErrorIdent".

Die Abarbeitung startet bei steigender Flanke an "Execute" und läuft nach der Initialisierungsphase so lange, bis entweder der Zustand STATE_DONE erreicht wird, oder ein Fehler auftritt.

Eine neue steigende Flanke an "Execute" bewirkt ein Rücksetzen und Neustart des Funktionsbausteins.

Funktionsbausteine mit "Execute"-Eingang und "Done"-Ausgang werden vorrangig für einmalige Aufgaben mit definiertem Abschluss, wie z.B. Initialisierung einer externen Hardware, eingesetzt.

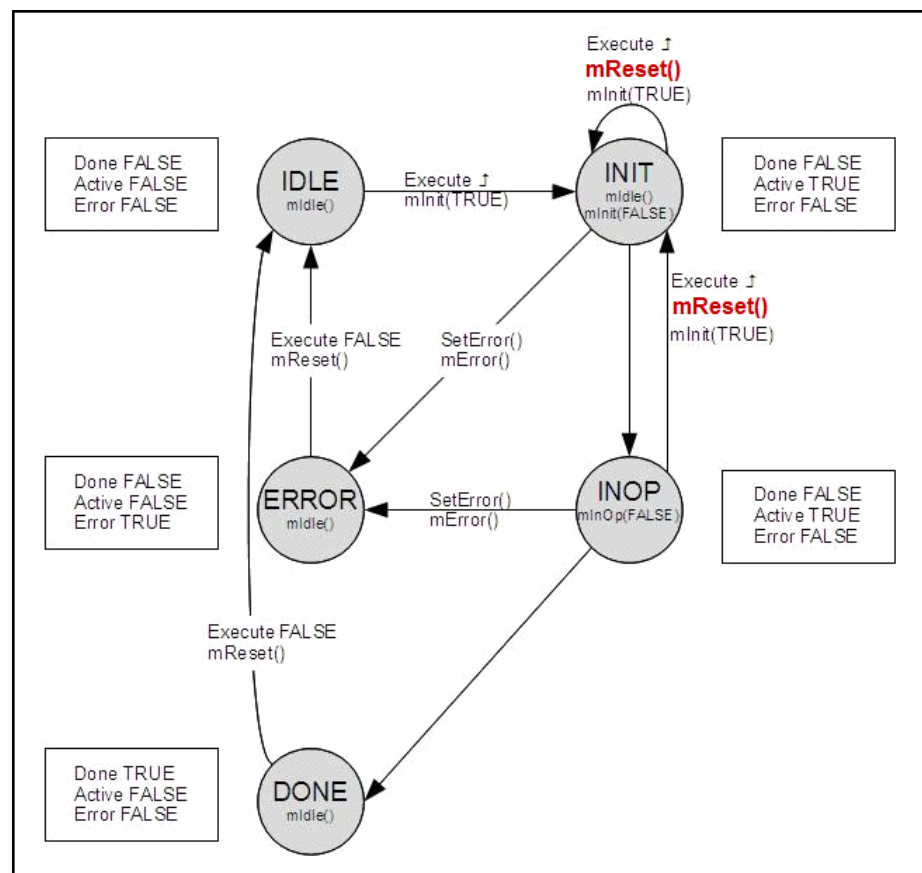


Abb. 4-29: State-Machine eines Funktionsbausteins mit Execute-Eingang und Done- sowie Active-Ausgang

Bausteine dieses Typs haben fünf interne Zustände (siehe auch [IL_FB_STATES](#) auf Seite 110):

STATE_IDLE Dieser Zustand stellt den Startzustand des Funktionsbausteins dar. Er wird aktiviert, wenn "Execute" False und sich der Baustein in den Zuständen STATE_DONE oder STATE_ERROR befindet. Beim Betreten des Zustands wird `mReset()` ausgeführt. Im Zustand wird `mdle()` ausgeführt.

Wird "Execute" gesetzt, wird in den Zustand STATE_INIT gewechselt.

STATE_INIT Zustand dient dem Hochlauf des Funktionsbausteins. Er wird vor der eigentlichen zyklischen Operation durchlaufen. Der Ausgang "Active" ist True, solange dieser Zustand aktiv ist. Beim Betreten des Zustands wird `mInit(Reset:=TRUE)` aufgerufen. Im Zustand werden `mInit(Reset:=FALSE)` und `mdle()` aufgerufen.

Gibt `mInit()` False zurück, wird in den Zustand STATE_INOP gewechselt. Wird `mSetError()` oder `mSetErrorIdent()` aufgerufen, wird in den Zustand STATE_ERROR gewechselt.

STATE_INOP

Hier wird die zyklische Aufgabe ausgeführt. Der Ausgang "Active" ist True, solange dieser Zustand aktiv ist. Beim Betreten des Zustands wird `mInOp(Reset:=TRUE)` aufgerufen. Im Zustand wird `mInOp(Reset:=FALSE)` aufgerufen.

Gibt `mInOp()` False zurück, so ist die Aufgabe beendet und der Funktionsbaustein wechselt in den Zustand STATE_DONE.

Wird `mSetError()` oder `mSetErrorIdent()` aufgerufen, wird in den Zustand STATE_ERROR gewechselt.

STATE_DONE

Zustand ist nach erfolgreicher Abarbeitung der Aufgabe aktiv. Er wird verlassen, wenn "Execute" False ist, ist jedoch mindestens einen Zyklus lang aktiv. Im Zustand wird `mdle()` aufgerufen.

STATE_ERROR

Dies ist der Fehlerzustand des Funktionsbausteins. Der Fehlerausgang "Error" ist in diesem Zustand aktiv und der Fehlercode wird an "ErrorID" bzw. "ErrorIdent" angezeigt. Beim Betreten des Zustands wird `mError()` aufgerufen. Im Zustand wird `mdle()` aufgerufen.

Wird "Execute" zurückgesetzt, wird in den Zustand STATE_IDLE gewechselt.

Signal-Zeit Diagramm

Aus der oben beschriebenen State-Machine ergibt sich das folgende Zeitverhalten:

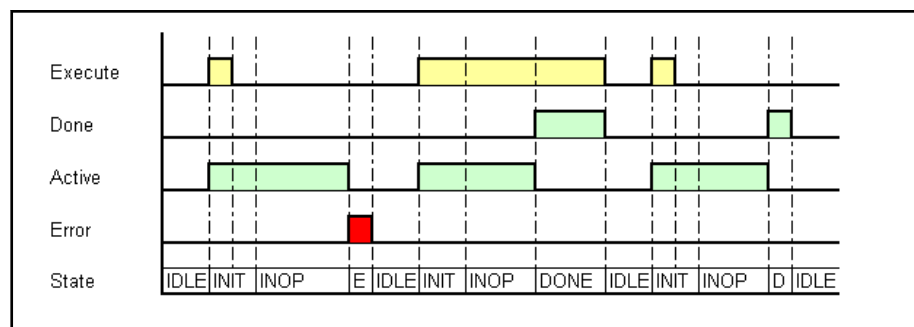


Abb. 4-30: Zeitverhalten eines Funktionsbausteins auf Basis von `IL_ExecuteDoneBase`

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann die folgenden Fehlermeldungen in Additional1/Additional2 erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#00000006	16#0000FFFF	Ungültiger Zustand der Zustandsmaschine

Tab. 4-20: Fehlercodes, die von `IL_ExecuteDoneBase` generiert werden

4.3.6 Basisfunktionsbaustein `IL_ExecuteInOpBase`

Kurzbeschreibung

Dieser Basisfunktionsbaustein (Basisklasse) implementiert das Interface eines Funktionsbausteins mit "Execute"-Eingang und "InOperation"- sowie "Active"-Ausgang. Weiterhin werden die Standard-Fehlerausgänge angelegt. Der Baustein implementiert eine zu den Standard-Ein- und Ausgängen passende Zustandsmaschine.

Funktionsbausteine dieses Typs werden für andauernde Aufgaben ohne definierten Abschluss verwendet.

Dieser Funktionsbaustein dient als Basisklasse für eigene Implementierungen. Er muss über das Schlüsselwort "EXTENDS" eingebunden werden.

Schnittstellenbeschreibung

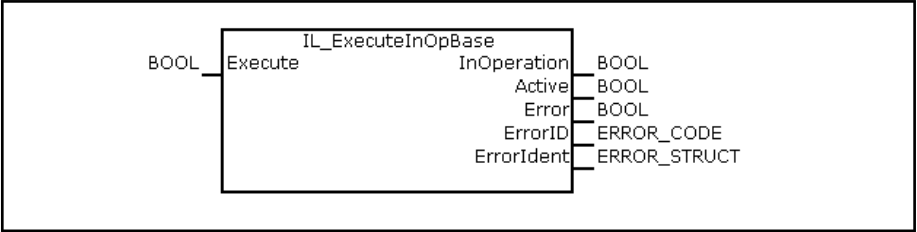


Abb. 4-31: Funktionsbaustein IL_ExecuteInOpBase

	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe (zyklisch, flankengesteuert)
VAR_OUTPUT	InOperation	BOOL	Der Baustein bearbeitet seine Aufgabe
	Active	BOOL	Der Funktionsbaustein wurde gestartet
	Error	BOOL	Zeigt an, dass ein Fehler aufgetreten ist
	ErrorID	ERROR_CODE	Fehlerursache
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerursache

Tab. 4-21: Funktionsbaustein IL_ExecuteInOpBase

Funktionsbeschreibung

Diese Basisklasse implementiert das Interface eines Funktionsbausteins mit "Execute"-Eingang und "InOperation"- sowie "Active"-Ausgang. Weiterhin werden die Standard-Fehlerausgänge angelegt. Der Baustein implementiert eine zu den Standard Ein- und Ausgängen passende Zustandsmaschine.

Dieser Funktionsbaustein dient als Basisklasse für eigene Implementierungen. Er muss über das Schlüsselwort "EXTENDS" eingebunden werden.

Dieser Basisfunktionsbaustein implementiert den Eingang "Execute" sowie die Ausgänge "InOperation", "Active", "Error", "ErrorID" und "ErrorIdent". Die Abarbeitung startet bei steigender Flanke an "Execute" und läuft nach der Initialisierungsphase so lange, bis ein Fehler auftritt. Eine neue steigende Flanke an "Execute" bewirkt ein Rücksetzen und Neustart des Funktionsbausteins.

Funktionsbausteine mit "Execute"-Eingang und "InOperation"-Ausgang werden vorrangig für kontinuierliche Aufgaben ohne definiertem Abschluss, wie z.B. Geschwindigkeitsregelung, eingesetzt.

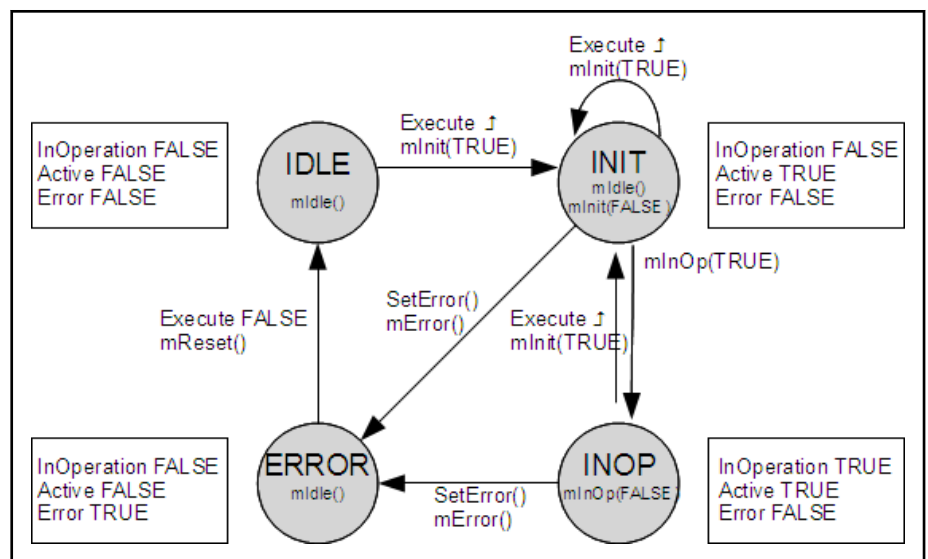


Abb. 4-32: State-Machine eines Funktionsbausteins mit Execute-Eingang und InOperation- sowie Active-Ausgang

Bausteine dieses Typs haben vier interne Zustände (siehe auch [IL_FB_STATES](#) auf Seite 110):

STATE_IDLE

Zustand stellt den Startzustand des Funktionsbausteins dar. Er wird aktiviert, wenn "Execute" False und sich der Baustein im Zustand STATE_ERROR befindet. Beim Betreten des Zustands wird mReset() ausgeführt. Im Zustand wird mIdle() ausgeführt.

Wird "Execute" gesetzt, wird in den Zustand STATE_INIT gewechselt.

STATE_INIT

Zustand dient dem Hochlauf des Funktionsbausteins. Er wird vor der eigentlichen zyklischen Operation durchlaufen. Der Ausgang "Active" ist True, solange dieser Zustand aktiv ist. Beim Betreten des Zustands wird mInit(Reset:=TRUE) aufgerufen. Im Zustand werden mInit(Reset:=FALSE) und mIdle() aufgerufen.

Gibt mInit() False zurück, wird in den Zustand STATE_INOP gewechselt. Wird mSetError() oder mSetErrorIdent() aufgerufen, wird in den Zustand STATE_ERROR gewechselt.

- STATE_INOP

Hier wird die zyklische Aufgabe ausgeführt. Der Ausgang "Active" ist True, solange dieser Zustand aktiv ist. Beim Betreten des Zustands wird mInOp(Reset:=TRUE) aufgerufen. Im Zustand wird mInOp(Reset:=FALSE) aufgerufen.

Wird mSetError() oder mSetErrorIdent() aufgerufen, wird in den Zustand STATE_ERROR gewechselt.
- STATE_ERROR

Dies ist der Fehlerzustand des Funktionsbausteins. Der Fehlerausgang "Error" ist in diesem Zustand aktiv und der Fehlercode wird an "ErrorID" bzw. "ErrorIdent" angezeigt. Beim Betreten des Zustands wird mError() aufgerufen. Im Zustand wird mIdle() aufgerufen.

Wird "Execute" zurückgesetzt, wird in den Zustand STATE_IDLE gewechselt.

Signal-Zeit Diagramm

Aus der oben beschriebenen State-Machine ergibt sich das folgende Zeitverhalten:

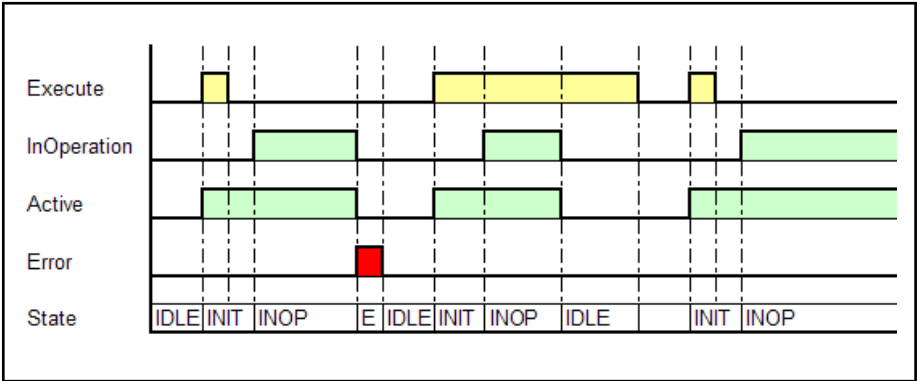


Abb. 4-33: Zeitverhalten eines Funktionsbausteins auf Basis von IL_ExecuteInOpBase

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann die folgenden Fehlermeldungen in Additional1/Additional2 erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#00000006	16#0000FFFF	Ungültiger Zustand der Zustandsmaschine

Tab. 4-22: Fehlercodes, die von IL_ExecuteInOpBase generiert werden

4.3.7

Basisfunktionsbaustein IL_ExecuteDoneAbortedBase

- Kurzbeschreibung

Dieser Basisfunktionsbaustein (Basisklasse) implementiert das Interface eines Funktionsbausteins mit "Execute"-Eingang und "Done"-, "Command-Aborted"- sowie "Active"-Ausgang. Weiterhin werden die Standard-Fehlerausgänge angelegt. Der Baustein implementiert eine zu den Standard-Ein- und Ausgängen passende Zustandsmaschine.

Funktionsbausteine dieses Typs werden für einmalige Aufgaben, die mehrere Zyklen Bearbeitungszeit benötigen, verwendet. Ein Beispiel ist die Initialisierung einer externen Hardware.

Dieser Funktionsbaustein dient als Basisklasse für eigene Implementierungen. Er muss über das Schlüsselwort "EXTENDS" eingebunden werden.

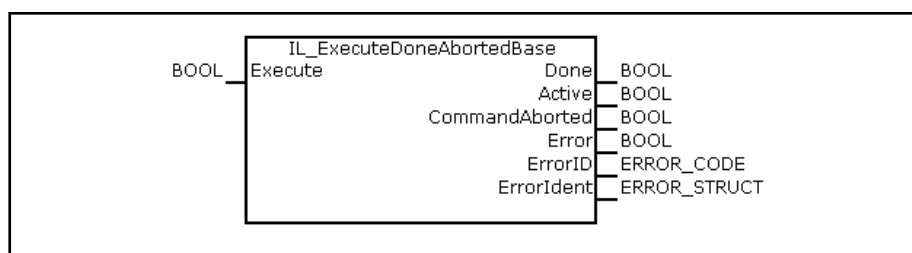
Schnittstellenbeschreibung

Abb. 4-34: Funktionsbaustein IL_ExecuteDoneAbortedBase

	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe (zyklisch, flankengesteuert)
VAR_OUTPUT	Done	BOOL	Der Baustein hat die Bearbeitung ohne Fehler abgeschlossen
	Active	BOOL	Der Funktionsbaustein wurde gestartet
	CommandAborted	BOOL	Die Abarbeitung wurde durch einen anderen Funktionsbaustein unterbrochen
	Error	BOOL	Zeigt an, dass ein Fehler aufgetreten ist
	ErrorID	ERROR_CODE	Fehlerursache
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerursache

Tab. 4-23: Funktionsbaustein IL_ExecuteDoneAbortedBase

Funktionsbeschreibung

Diese Basisklasse implementiert das Interface eines Funktionsbausteins mit "Execute"-Eingang und "Done"-, "CommandAborted"- sowie "Active"-Ausgang. Weiterhin werden die Standard-Fehlerausgänge angelegt. Der Baustein implementiert eine zu den Standard Ein- und Ausgängen passende Zustandsmaschine.

Dieser Funktionsbaustein dient als Basisklasse für eigene Implementierungen. Er muss über das Schlüsselwort "EXTENDS" eingebunden werden.

Funktionsbausteine mit "Execute"-Eingang werden vorrangig für einmalige Aufgaben mit definiertem Abschluss, wie z.B. Initialisierung einer externen Hardware, eingesetzt. Dieser Basisfunktionsbaustein implementiert den Eingang "Execute", sowie die Ausgänge "Done", "Active", "Error", "ErrorID" und "ErrorIdent". Die Abarbeitung startet bei steigender Flanke an "Execute" und läuft nach der Initialisierungsphase so lange, bis entweder der Zustand STATE_DONE erreicht wird, ein Fehler auftritt, oder die Bearbeitung durch einen anderen Funktionsbaustein abgebrochen wird.

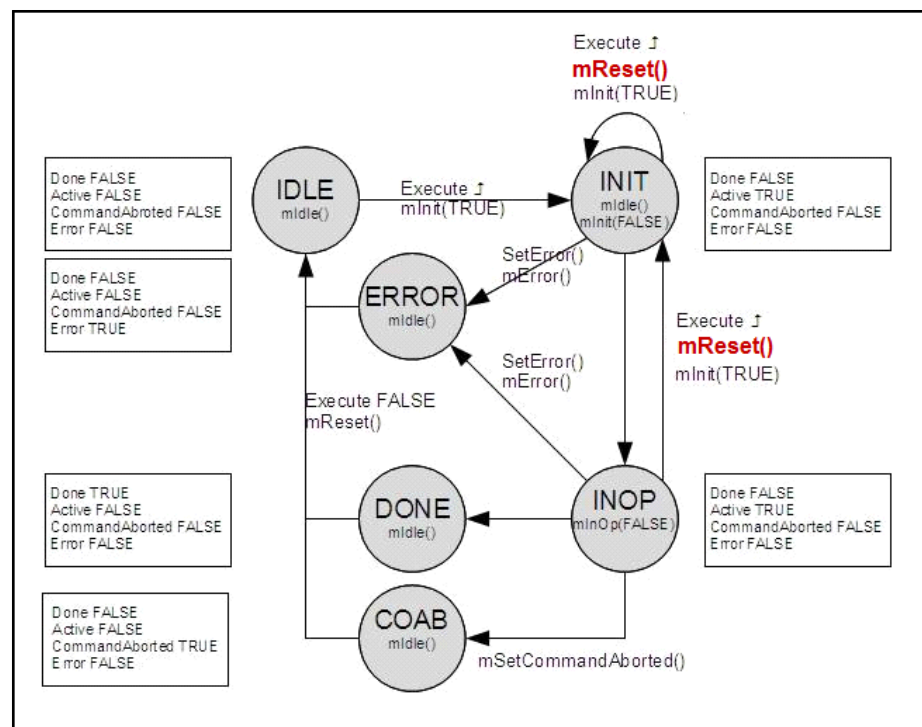


Abb. 4-35: State-Machine eines FBs mit Execute-Eingang und Done-, CommandAborted- sowie Active-Ausgang

Bausteine dieses Typs haben sechs interne Zustände (siehe auch [IL_FB_STATES](#) auf Seite 110):

STATE_IDLE Zustand stellt den Startzustand des Funktionsbausteins dar. Er wird aktiviert, wenn "Execute" False und sich der Baustein in den Zuständen STATE_DONE oder STATE_ERROR befindet. Beim Betreten des Zustands wird mReset() ausgeführt. Im Zustand wird mIdle() ausgeführt.

Wird "Execute" gesetzt, wird in den Zustand STATE_INIT gewechselt.

STATE_INIT Zustand dient dem Hochlauf des Funktionsbausteins. Er wird vor der eigentlichen zyklischen Operation durchlaufen. Der Ausgang "Active" ist True, solange dieser Zustand aktiv ist. Beim Betreten des Zustands wird mInit(Reset:=TRUE) aufgerufen. Im Zustand werden mInit(Reset:=FALSE) und mIdle() aufgerufen.

Gibt mInit() False zurück, wird in den Zustand STATE_INOP gewechselt. Wird mSetError() oder mSetErrorIdent() aufgerufen, wird in den Zustand STATE_ERROR gewechselt. Wird "Execute" zurückgesetzt, wird in den Zustand STATE_IDLE gewechselt.

STATE_INOP

Hier wird die zyklische Aufgabe ausgeführt. Der Ausgang "Active" ist True, solange dieser Zustand aktiv ist. Beim Betreten des Zustands wird `mInOp(Reset:=TRUE)` aufgerufen. Im Zustand wird `mInOp(Reset:=FALSE)` aufgerufen.

Gibt `mInOp()` False zurück, so ist die Aufgabe beendet und der Funktionsbaustein wechselt in den Zustand STATE_DONE.

Wird `mSetError()` oder `mSetErrorIdent()` aufgerufen, wird in den Zustand STATE_ERROR gewechselt.

STATE_DONE

Zustand ist nach erfolgreicher Abarbeitung der Aufgabe aktiv. Er wird verlassen, wenn "Execute" False ist, ist jedoch mindestens einen Zyklus lang aktiv. Im Zustand wird `mdle()` aufgerufen.

STATE_COAB

Zustand wird nach Abarbeiten der Hilfsmethode `mSetCommandAborted` aktiv. Er wird verlassen, wenn "Execute" False ist, ist jedoch mindestens einen Zyklus lang aktiv. Im Zustand wird `mdle()` aufgerufen.

STATE_ERROR

Dies ist der Fehlerzustand des Funktionsbausteins. Der Fehlerausgang "Error" ist in diesem Zustand aktiv und der Fehlercode wird an "ErrorID" bzw. "ErrorIdent" angezeigt. Beim Betreten des Zustands wird `mError()` aufgerufen. Im Zustand wird `mdle()` aufgerufen.

Wird "Execute" zurückgesetzt, wird in den Zustand STATE_IDLE gewechselt.

Signal-Zeit Diagramm

Aus der oben beschriebenen State-Machine ergibt sich das folgende Zeitverhalten:

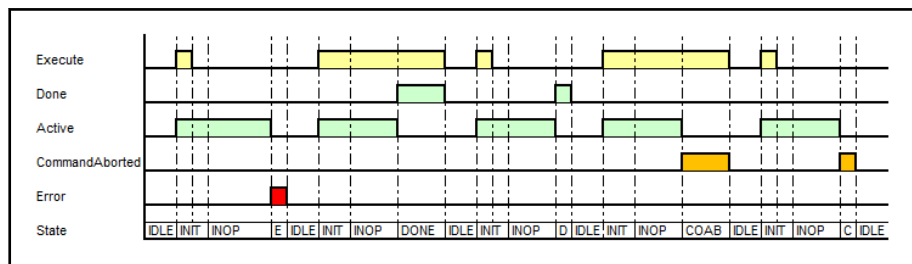


Abb. 4-36: Zeitverhalten eines Funktionsbausteins auf Basis von `IL_ExecuteDoneAbortedBase`

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann die folgenden Fehlermeldungen in Additional1/Additional2 erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#00000006	16#0000FFFF	Ungültiger Zustand der Zustandsmaschine

Tab. 4-24: Fehlercodes, die von `IL_ExecuteDoneAbortedBase` generiert werden

4.3.8 Basisfunktionsbaustein `IL_ExecuteDoneAbortedShutdownBase`

Kurzbeschreibung

Dieser Basisfunktionsbaustein (Basisklasse) implementiert das Interface eines Funktionsbausteins mit "Execute"-Eingang und "Done"-, "Active"-, "Shutdown"-, sowie "CommandAborted"-Ausgang. Weiterhin werden die Standard-

Fehlerausgänge angelegt. Der Baustein implementiert eine zu den Standard-Ein- und -Ausgängen passende Zustandsmaschine.

Funktionsbausteine dieses Typs werden für einmalige Aufgaben, die mehrere Zyklen Bearbeitungszeit benötigen, verwendet. Ein Beispiel ist die Initialisierung einer externen Hardware.

Dieser Funktionsbaustein dient als Basisklasse für eigene Implementierungen. Er muss über das Schlüsselwort "EXTENDS" eingebunden werden.

Schnittstellenbeschreibung

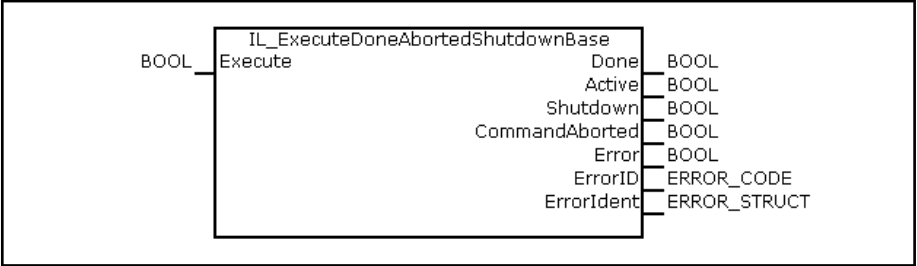


Abb. 4-37: Funktionsbaustein IL_ExecuteDoneAbortedShutdownBase

	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe (zyklisch, flankengesteuert)
VAR_OUTPUT	Done	BOOL	Der Baustein hat die Bearbeitung ohne Fehler abgeschlossen
	Active	BOOL	Der Funktionsbaustein wurde gestartet
	Shutdown	BOOL	Die Bearbeitung ist beendet. Sie wurde vom Anwender (mAbort()) oder durch einen Fehler (mSetError() oder mSetErrorIdent()) abgebrochen, der Baustein muss aber noch Aktivitäten ordnungsgemäß beenden
	CommandAborted	BOOL	Die Abarbeitung wurde durch einen anderen Funktionsbaustein unterbrochen
	Error	BOOL	Zeigt an, dass ein Fehler aufgetreten ist
	ErrorID	ERROR_CODE	Fehlerursache
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerursache

Tab. 4-25: Funktionsbaustein IL_ExecuteDoneAbortedShutdownBase

Funktionsbeschreibung

Diese Basisklasse implementiert das Interface eines Funktionsbausteins mit "Execute"-Eingang und "Done"-, "Active"-, "Shutdown"- sowie "Command-Aborted"-Ausgang. Weiterhin werden die Standard-Fehlerausgänge angelegt.

Der Baustein implementiert eine zu den Standard-Ein- und -Ausgängen passende Zustandsmaschine.

Dieser Funktionsbaustein dient als Basisklasse für eigene Implementierungen. Er muss über das Schlüsselwort "EXTENDS" eingebunden werden.

Die Abarbeitung startet bei steigender Flanke an "Execute" und läuft nach der Initialisierungsphase so lange, bis entweder der Zustand STATE_DONE erreicht wird, ein Fehler auftritt oder die Bearbeitung durch einen anderen Funktionsbaustein abgebrochen wird.

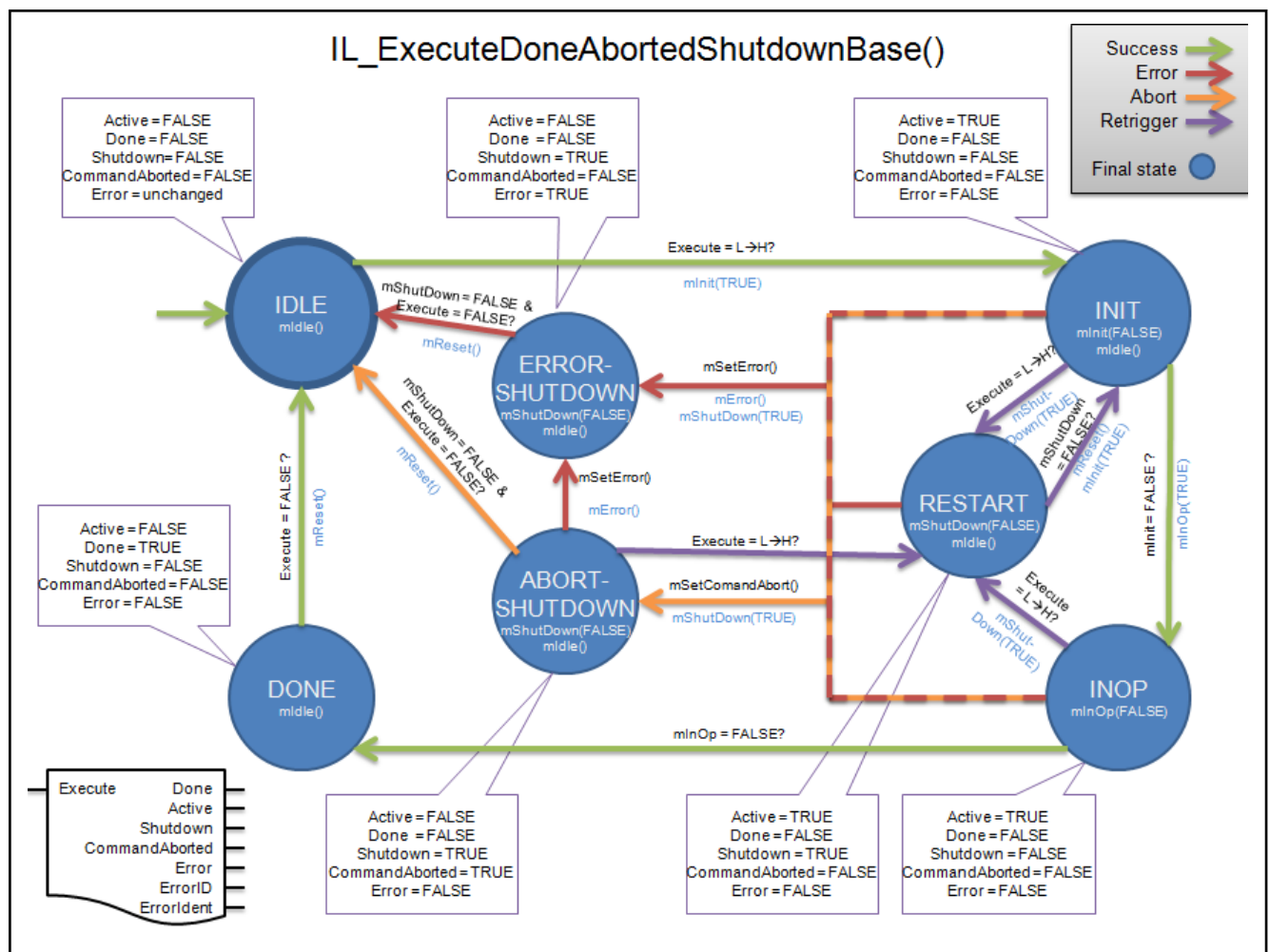


Abb. 4-38: State-Machine eines FBs mit Execute-Eingang und Done-, Active-, Shutdown-, sowie CommandAborted-Ausgang

Bausteine dieses Typs haben sieben interne Zustände (siehe auch IL_FB_STATES auf Seite 110):

STATE_IDLE	Dieser Zustand stellt den Startzustand des Funktionsbausteins dar. Er ist beim erstmaligen Aufruf aktiv und wird aktiviert, wenn "Execute" False ist und sich der Baustein in den Zuständen STATE_DONE, STATE_ABORT_SHUTDOWN oder STATE_ERROR_SHUTDOWN befindet. Beim (erneuten) Betreten des Zustands wird mReset() ausgeführt. Im Zustand wird mIdle() ausgeführt. Wird "Execute" gesetzt, wird in den Zustand STATE_INIT gewechselt.
STATE_INIT	Dieser Zustand dient dem Hochlauf des Funktionsbausteins. Er wird vor der eigentlichen zyklischen Operation durchlaufen. Der Ausgang "Active" ist True, solange dieser Zustand aktiv ist. Beim Betreten des Zustands wird mInit(Reset:=TRUE) aufgerufen. Im Zustand werden mInit(Reset:=FALSE) und mIdle() aufgerufen. Gibt mInit() False zurück, wird in den Zustand STATE_INOP gewechselt. Wird mSetError() oder mSetErrorIdent() aufgerufen, wird in den Zustand STATE_ERROR_SHUTDOWN gewechselt.
STATE_INOP	Hier wird die zyklische Aufgabe ausgeführt. Der Ausgang "Active" ist True, solange dieser Zustand aktiv ist. Beim Betreten des Zustands wird mInOp(Reset:=TRUE) aufgerufen. Im Zustand wird mInOp(Reset:=FALSE) aufgerufen. Gibt mInOp() False zurück, so ist die Aufgabe beendet und der Funktionsbaustein wechselt in den Zustand STATE_DONE. Wird mSetError() oder mSetErrorIdent() aufgerufen, wird in den Zustand STATE_ERROR_SHUTDOWN gewechselt.
STATE_DONE	Der Zustand ist nach erfolgreicher Abarbeitung der Aufgabe aktiv. Er wird verlassen, wenn "Execute" False ist, ist jedoch mindestens einen Zyklus lang aktiv. Im Zustand wird mIdle() aufgerufen.
STATE_RESTART	Wird während der Ausführung eine positive Flanke am Eingang "Execute" erkannt, so wird die aktuelle Bearbeitung beendet und die Funktion wird erneut gestartet. Dazu wird mit Eintritt in den Zustand die Methode mShutDown(TRUE) aufgerufen. Im Zustand wird die Methode mShutDown(FALSE) aufgerufen. Während der Ausführung sind die Ausgänge "Active" und "Shutdown" aktiv. Liefert die Methode mShutDown() = FALSE zurück, so wird in den Zustand STATE_INIT gewechselt.

- STATE_ABORT_SHUTDOWN** Dies ist der Abbruchzustand des Funktionsbausteins.
Die Ausgänge "Shutdown" und "CommandAborted" sind in diesem Zustand aktiv.
Beim Betreten des Zustands wird mShutDown(TRUE) aufgerufen. Im Zustand werden mShutDown(FALSE) und mIdle() aufgerufen. Liefert die Methode mShutDown() = FALSE zurück und ist "Execute" zurückgesetzt, so wird in den Zustand STATE_IDLE gewechselt.
- STATE_ERROR_SHUTDOWN** Dies ist der Fehlerzustand des Funktionsbausteins.
In diesem Zustand sind die Ausgänge "Shutdown" und "Error" aktiv, und der Fehlercode wird an "ErrorID" bzw. "ErrorIdent" angezeigt.
Beim Betreten des Zustands wird mError() aufgerufen. Im Zustand wird mShutDown(TRUE) aufgerufen. Im Zustand werden mShutDown(FALSE) und mIdle() aufgerufen. Liefert die Methode mShutDown() = FALSE zurück und ist "Execute" zurückgesetzt, so wird in den Zustand STATE_IDLE gewechselt. Der Ausgang "Error" bleibt dabei gesetzt.

Methoden Dem Anwender stehen folgende Methoden zur Verfügung:

Methode	Funktion
mIdle()	Ruhecode solange der FB nicht in INOP ist
BOOL:= mInit(Reset)	Initialisierungscode des FBs. Wird bearbeitet bis mInit = FALSE liefert. Reset ist beim ersten Aufruf TRUE
BOOL:= mInOp(Reset)	Operationscode des FBs. Wird bearbeitet bis mInOp = FALSE liefert. Reset ist beim ersten Aufruf TRUE
BOOL:= mShutDown(Reset)	Beendigungscode des FBs durch Abbruch/Fehler/Reset. Wird bearbeitet bis mShutDown = FALSE liefert. Reset ist beim ersten Aufruf TRUE
mAbort()	Abbrechen der aktuellen Aufgabe → Übergang in Shutdown
mSetError()	Aufruf durch den Anwender, wenn ein Fehler erkannt wurde
nSetErrorIdent()	Alternativer Aufruf durch den Anwender, wenn ein Fehler erkannt wurde

Tab. 4-26: Öffentliche Methoden des FB IL_ExecuteDoneAbortedShutdownBase

Darüber hinaus sind noch folgende Methoden vorhanden:

Methode	Funktion
mError()	Einmalige Ausführung beim Eintritt in den ERROR-Zustand
mReset()	Einmalige Ausführung beim Eintritt in die Zustände IDLE bzw. INIT nach Restart
mSetOutputs()	Setzen der Ausgänge abhängig vom aktuellen Zustand

Tab. 4-27: Interne Methoden des FB IL_ExecuteDoneAbortedShutdownBase

Signal-Zeit Diagramm Aus der oben beschriebenen State-Machine ergibt sich das folgende Zeitverhalten:

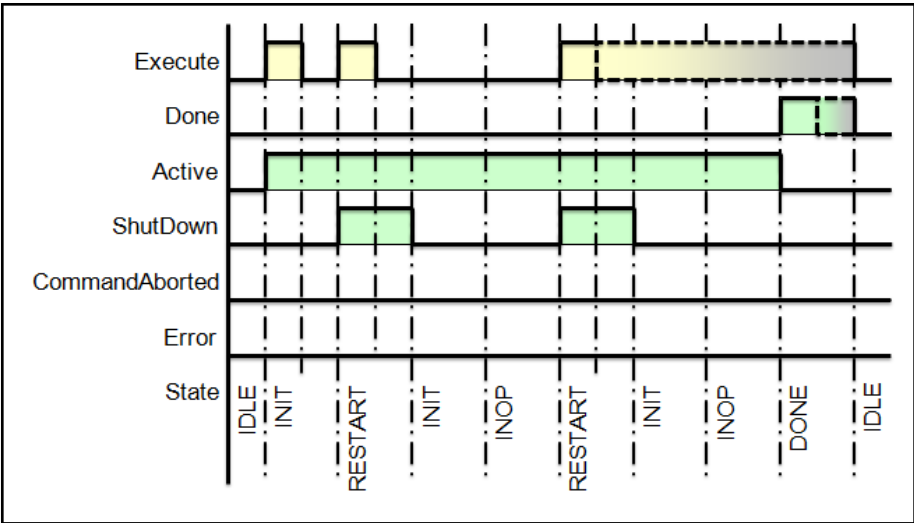


Abb. 4-39: Starten und fehlerfreies Beenden von IL_ExecuteDoneAbortedShutdownBase

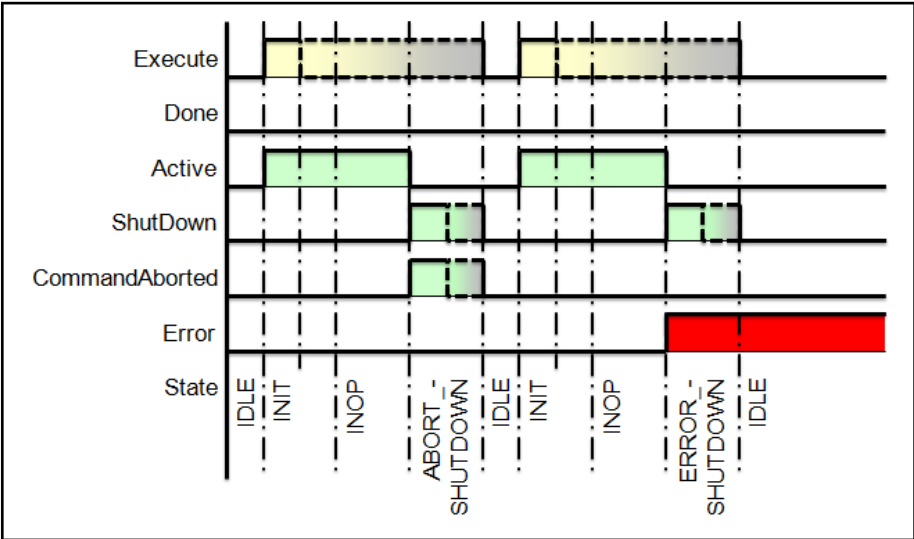


Abb. 4-40: Starten und Beenden durch Abbruch oder Fehler von IL_ExecuteDoneAbortedShutdownBase

Fehlercodes Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann die folgenden Fehlermeldungen in Additional1/Additional2 erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#00000006	16#0000FFFF	Ungültiger Zustand der Zustandsmaschine

Tab. 4-28: Fehlercodes, die von IL_ExecuteDoneAbortedShutdownBase generiert werden

4.3.9 IL_FB_STATES

IL_FB_STATES definiert einen Aufzählungstyp, der die Zustände der Standard State-Machine enthält. Die verwendeten Zustände sind vom Typ des Funktionsbausteins abhängig. Als Startzustand wird STATE_IDLE gesetzt.

Zustand		Beschreibung
STATE_IDLE	0	Startzustand eines Funktionsbausteins
STATE_INIT	1	Initialisierung des Funktionsbausteins Zustand wird nach dem Start des Funktionsbausteins durchlaufen. Hier werden alle für die eigentliche Operation notwendigen Vorbereitungen getroffen. Eine Unter-Zustandsmaschine ist möglich.
STATE_INOP	2	Arbeitszustand des Funktionsbausteins Im Zustand erfolgt die Abarbeitung der eigentlichen Aufgabe des Funktionsbausteins.
STATE_SHUTDOWN	3	Beenden der Abarbeitung Zustand ist nur bei Funktionsbausteinen mit "Shutdown"-Ausgang vorhanden. Hier werden laufende Prozesse abgeschlossen um den Funktionsbaustein wieder in den Zustand STATE_IDLE zu versetzen.
STATE_DONE	4	Abarbeitung abgeschlossen Zustand zeigt an, dass die Abarbeitung der Aufgabe erfolgreich abgeschlossen wurde.
STATE_ERROR	5	Fehlerzustand Zustand ist nach Auftreten eines Fehlers aktiv. Er wird durch Rücksetzen von "Enable" bzw. "Execute" verlassen.
STATE_COAB	6	Kommando durch anderen FB abgebrochen (Ausgang "CommandAborted" ist TRUE)

Tab. 4-29: Definierte Zustände in IL_FB_STATES

4.4 Konvertierung (Bit-Stream Connect)

4.4.1 Einführung und Übersicht

Anwendungsgebiete und Funktionen / Funktionsbausteine von RIL_Uilities

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine	Beschreibung
Konvertierungsfunktionen und -funktionsbausteine (Bit-Stream Connect)	
IL_BoolToByte, Seite 111	8 einzelne Bits (BOOL) werden zu einem BYTE konvertiert (FUN)
IL_BoolToDWord, Seite 112	32 einzelne Bits (BOOL) werden zu einem DWORD konvertiert (FUN)
IL_BoolToWord, Seite 115	16 einzelne Bits (BOOL) werden zu einem WORD konvertiert (FUN)
IL_ByteToDWord, Seite 116	4 BYTES werden zu einem DWORD konvertiert (FUN)
IL_ConcatByte, Seite 117	2 BYTES werden zu einem WORD zusammengefasst (FUN)
IL_ConcatWord, Seite 118	2 WORDs werden zu einem DWORD zusammengefasst

Tab. 4-30: RIL_Uilities, Bit-Stream Connect

Bit-Stream Connect: Hier werden kleinere Datentypen zu größeren zusammengefasst.

Ein Beispiel ist die Funktion IL_BoolToByte, die 8 Bits zu einem Byte konvertiert.

4.4.2 IL_BoolToByte

Kurzbeschreibung Diese Funktion beschreibt die einzelnen Bits eines Byte. Es werden 8 einzelne Bits zu einem Byte konvertiert.

Schnittstellenbeschreibung

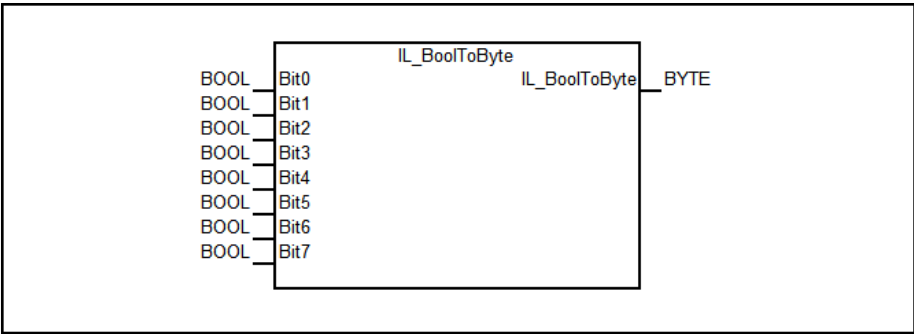


Abb. 4-41: Funktion IL_BoolToByte

	Name	Datentyp	Beschreibung
VAR_INPUT	Bit0	BOOL	Beschreibt die Position des Bit 0 im Ausgangsbyte
	Bit1	BOOL	Beschreibt die Position des Bit 1 im Ausgangsbyte
	Bit2	BOOL	Beschreibt die Position des Bit 2 im Ausgangsbyte
	Bit3	BOOL	Beschreibt die Position des Bit 3 im Ausgangsbyte
	Bit4	BOOL	Beschreibt die Position des Bit 4 im Ausgangsbyte
	Bit5	BOOL	Beschreibt die Position des Bit 5 im Ausgangsbyte
	Bit6	BOOL	Beschreibt die Position des Bit 6 im Ausgangsbyte
	Bit7	BOOL	Beschreibt die Position des Bit 7 im Ausgangsbyte
RETURN	IL_BoolToByte	BYTE	Ausgangsbyte

Tab. 4-31: Schnittstelle IL_BoolToByte

Funktionsbeschreibung

Diese Funktion schreibt die Bits der Eingänge entsprechend ihrer Wertigkeit in das Ausgangsbyte. Entsprechend des Zustandes aller Bits am Eingang der Funktion wird das Byte beschrieben. Das bedeutet, der Zustand des Eingangs "Bit0" wird auf das Bit 0 des Ausgangsbytes kopiert, der Zustand des Eingangs "Bit1" wird auf das Bit 1 des Ausgangsbytes kopiert usw.

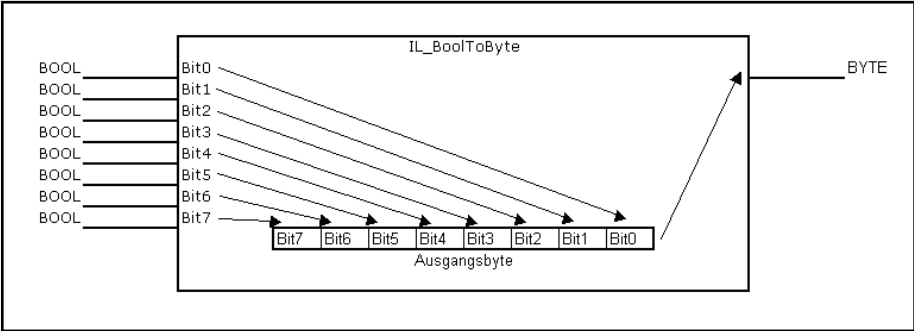


Abb. 4-42: Funktionsbeschreibung IL_BoolToByte

Fehlerbehandlung

Für diese Funktion ist keine Fehlerbehandlung verfügbar.

4.4.3 IL_BoolToDWord

Kurzbeschreibung

Diese Funktion beschreibt die einzelnen Bits eines DWords. Es werden 32 einzelne Bits zu einem Doppelwort konvertiert.

Schnittstellenbeschreibung

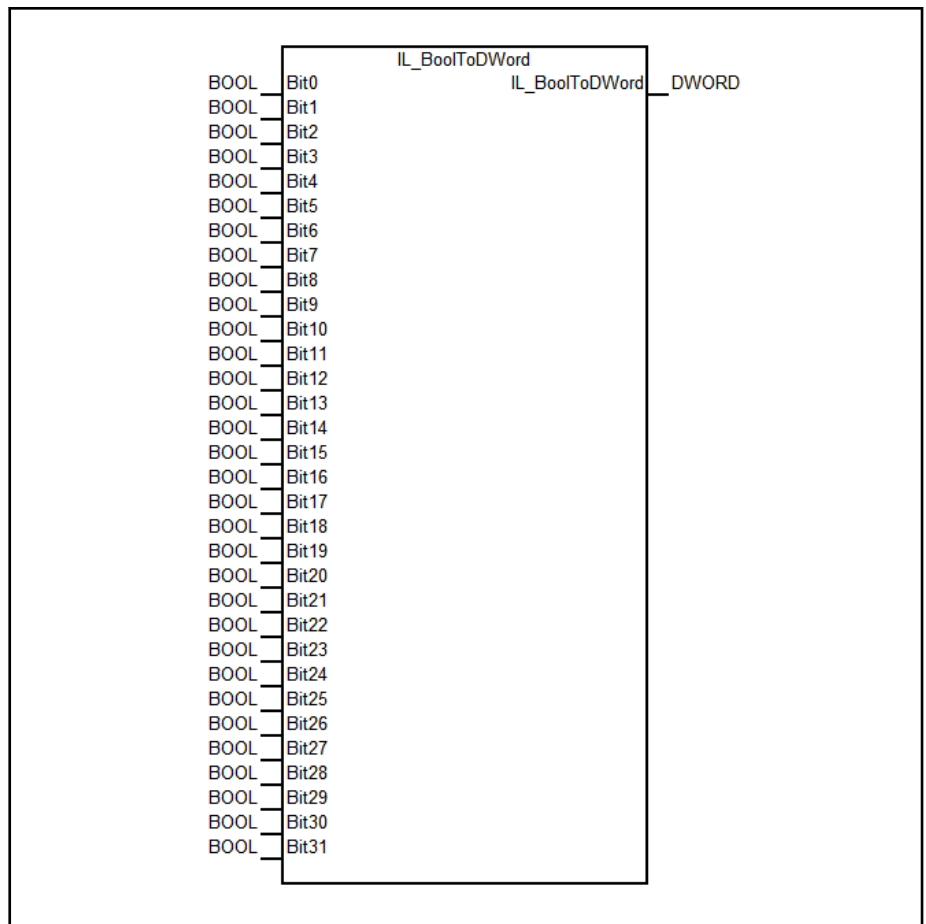


Abb. 4-43: Funktion IL_BoolToDWord

	Name	Datentyp	Beschreibung
VAR_INPUT	Bit0	BOOL	Beschreibt die Position des Bit 0 im Ausgangsdoppelwort
	Bit1	BOOL	Beschreibt die Position des Bit 1 im Ausgangsdoppelwort
	Bit2	BOOL	Beschreibt die Position des Bit 2 im Ausgangsdoppelwort
	Bit3	BOOL	Beschreibt die Position des Bit 3 im Ausgangsdoppelwort
	Bit4	BOOL	Beschreibt die Position des Bit 4 im Ausgangsdoppelwort
	Bit5	BOOL	Beschreibt die Position des Bit 5 im Ausgangsdoppelwort
	Bit6	BOOL	Beschreibt die Position des Bit 6 im Ausgangsdoppelwort
	Bit7	BOOL	Beschreibt die Position des Bit 7 im Ausgangsdoppelwort
	Bit8	BOOL	Beschreibt die Position des Bit 8 im Ausgangsdoppelwort
	Bit9	BOOL	Beschreibt die Position des Bit 9 im Ausgangsdoppelwort
	Bit10	BOOL	Beschreibt die Position des Bit 10 im Ausgangsdoppelwort
	Bit11	BOOL	Beschreibt die Position des Bit 11 im Ausgangsdoppelwort
	Bit12	BOOL	Beschreibt die Position des Bit 12 im Ausgangsdoppelwort
	Bit13	BOOL	Beschreibt die Position des Bit 13 im Ausgangsdoppelwort
	Bit14	BOOL	Beschreibt die Position des Bit 14 im Ausgangsdoppelwort

	Name	Datentyp	Beschreibung
	Bit15	BOOL	Beschreibt die Position des Bit 15 im Ausgangsdoppelwort
	Bit16	BOOL	Beschreibt die Position des Bit 16 im Ausgangsdoppelwort
	Bit17	BOOL	Beschreibt die Position des Bit 17 im Ausgangsdoppelwort
	Bit18	BOOL	Beschreibt die Position des Bit 18 im Ausgangsdoppelwort
	Bit19	BOOL	Beschreibt die Position des Bit 19 im Ausgangsdoppelwort
	Bit20	BOOL	Beschreibt die Position des Bit 20 im Ausgangsdoppelwort
	Bit21	BOOL	Beschreibt die Position des Bit 21 im Ausgangsdoppelwort
	Bit22	BOOL	Beschreibt die Position des Bit 22 im Ausgangsdoppelwort
	Bit23	BOOL	Beschreibt die Position des Bit 23 im Ausgangsdoppelwort
	Bit24	BOOL	Beschreibt die Position des Bit 24 im Ausgangsdoppelwort
	Bit25	BOOL	Beschreibt die Position des Bit 25 im Ausgangsdoppelwort
	Bit26	BOOL	Beschreibt die Position des Bit 26 im Ausgangsdoppelwort
	Bit27	BOOL	Beschreibt die Position des Bit 27 im Ausgangsdoppelwort
	Bit28	BOOL	Beschreibt die Position des Bit 28 im Ausgangsdoppelwort
	Bit29	BOOL	Beschreibt die Position des Bit 29 im Ausgangsdoppelwort
	Bit30	BOOL	Beschreibt die Position des Bit 30 im Ausgangsdoppelwort
	Bit31	BOOL	Beschreibt die Position des Bit 31 im Ausgangsdoppelwort
RETURN	IL_BoolToDWord	DWORD	Ausgangsdoppelwort

Tab. 4-32: Schnittstelle IL_BoolToDWord

Funktionsbeschreibung

Diese Funktion schreibt die Bits der Eingänge entsprechend ihrer Wertigkeit in das Ausgangsdoppelwort. Entsprechend des Zustandes aller Bits am Eingang der Funktion wird das DWord beschrieben. Das bedeutet, der Zustand des Eingangs "Bit0" wird auf das Bit 0 des Ausgangsdoppelworts kopiert, der Zustand des Eingangs "Bit1" wird auf das Bit 1 des Ausgangsdoppelworts kopiert usw.

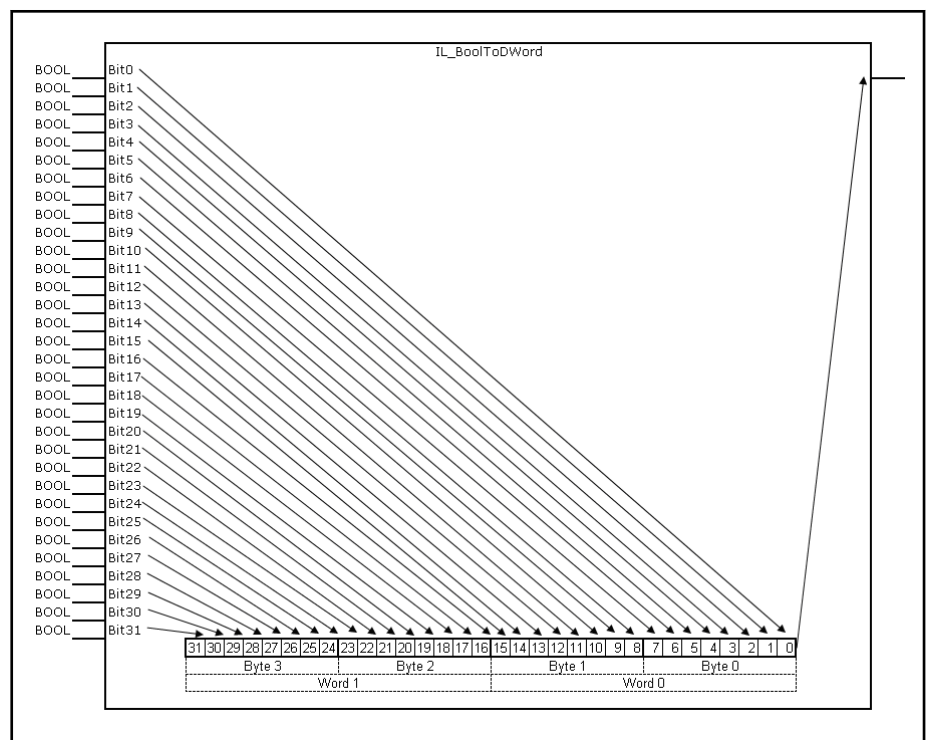


Abb. 4-44: Funktionsbeschreibung IL_BoolToDWord

Fehlerbehandlung Für diese Funktion ist keine Fehlerbehandlung verfügbar.

4.4.4 IL_BoolToWord

Kurzbeschreibung Diese Funktion beschreibt die einzelnen Bits eines Words. Es werden 16 einzelne Bits zu einem Wort konvertiert.

Schnittstellenbeschreibung

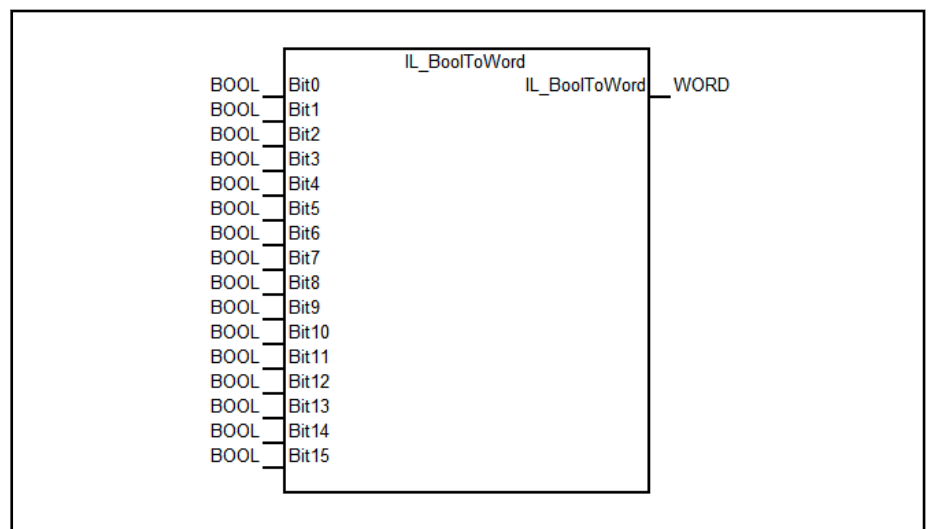


Abb. 4-45: Funktion IL_BoolToWord

	Name	Datentyp	Beschreibung
VAR_INPUT	Bit0	BOOL	Beschreibt die Position des Bit 0 im Ausgangswort
	Bit1	BOOL	Beschreibt die Position des Bit 1 im Ausgangswort
	Bit2	BOOL	Beschreibt die Position des Bit 2 im Ausgangswort

	Name	Datentyp	Beschreibung
	Bit3	BOOL	Beschreibt die Position des Bit 3 im Ausgangswort
	Bit4	BOOL	Beschreibt die Position des Bit 4 im Ausgangswort
	Bit5	BOOL	Beschreibt die Position des Bit 5 im Ausgangswort
	Bit6	BOOL	Beschreibt die Position des Bit 6 im Ausgangswort
	Bit7	BOOL	Beschreibt die Position des Bit 7 im Ausgangswort
	Bit8	BOOL	Beschreibt die Position des Bit 8 im Ausgangswort
	Bit9	BOOL	Beschreibt die Position des Bit 9 im Ausgangswort
	Bit10	BOOL	Beschreibt die Position des Bit 10 im Ausgangswort
	Bit11	BOOL	Beschreibt die Position des Bit 11 im Ausgangswort
	Bit12	BOOL	Beschreibt die Position des Bit 12 im Ausgangswort
	Bit13	BOOL	Beschreibt die Position des Bit 13 im Ausgangswort
	Bit14	BOOL	Beschreibt die Position des Bit 14 im Ausgangswort
	Bit15	BOOL	Beschreibt die Position des Bit 15 im Ausgangswort
RETURN	IL_BoolToWorld	WORD	Ausgangswort

Tab. 4-33: Schnittstelle IL_BoolToWorld

Funktionsbeschreibung

Diese Funktion schreibt die Bits der Eingänge entsprechend ihrer Wertigkeit in das Ausgangswort. Entsprechend des Zustands aller Bits am Eingang der Funktion wird das Wort beschrieben. Das bedeutet, der Zustand des Eingangs "Bit0" wird auf das Bit 0 des Ausgangsworts kopiert, der Zustand des Eingangs "Bit1" wird auf das Bit 1 des Ausgangsworts kopiert usw.

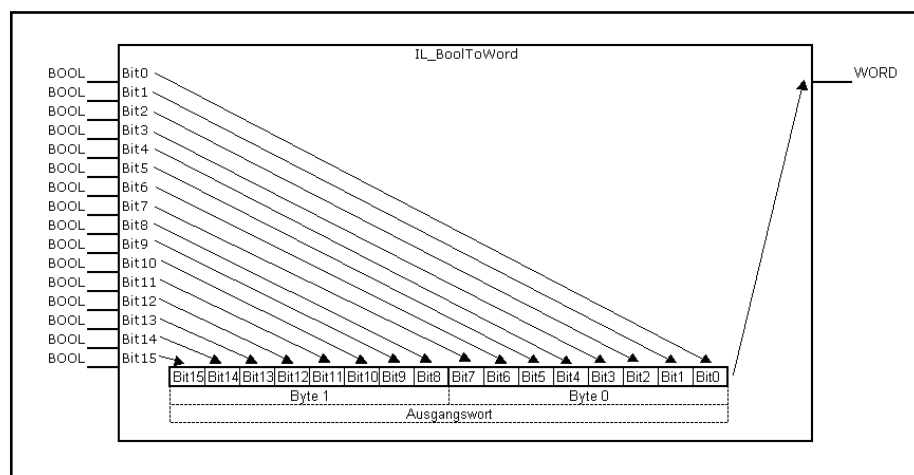


Abb. 4-46: Funktionsbeschreibung IL_BoolToWorld

Fehlerbehandlung

Für diese Funktion ist keine Fehlerbehandlung verfügbar.

4.4.5 IL_ByteToDWord

Kurzbeschreibung

Diese Funktion beschreibt die einzelnen Bytes eines DWords. Es werden 4 einzelne Bytes zu einem Doppelwort konvertiert.

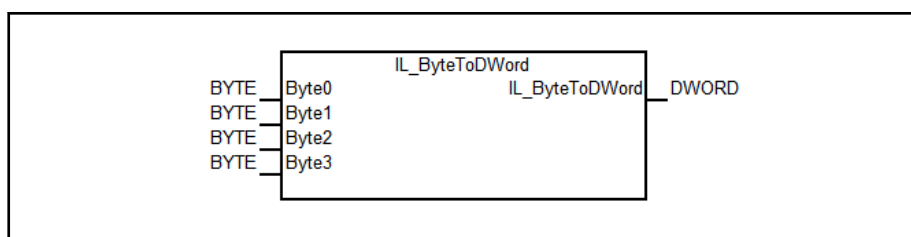
Schnittstellenbeschreibung

Abb. 4-47: Funktion IL_ByteToDWord

	Name	Datentyp	Beschreibung
VAR_INPUT	Byte0	BYTE	Beschreibt die Position des Byte 0 im Ausgangsdoppelwort
	Byte1	BYTE	Beschreibt die Position des Byte 1 im Ausgangsdoppelwort
	Byte2	BYTE	Beschreibt die Position des Byte 2 im Ausgangsdoppelwort
	Byte3	BYTE	Beschreibt die Position des Byte 3 im Ausgangsdoppelwort
RETURN	IL_ByteToDWord	DWORD	Ausgangsdoppelwort

Tab. 4-34: Schnittstelle IL_ByteToDWord

Funktionsbeschreibung

Diese Funktion schreibt die Bytes der Eingänge entsprechend ihrer Wertigkeit in das Ausgangsdoppelwort. Entsprechend des Zustandes aller Bytes am Eingang der Funktion wird das Doppelwort beschrieben. Das bedeutet, der Zustand des Eingangs "Byte0" wird auf das Byte 0 des Ausgangsdoppelworts kopiert, der Zustand des Eingangs "Byte1" wird auf das Byte 1 des Ausgangsdoppelworts kopiert usw.

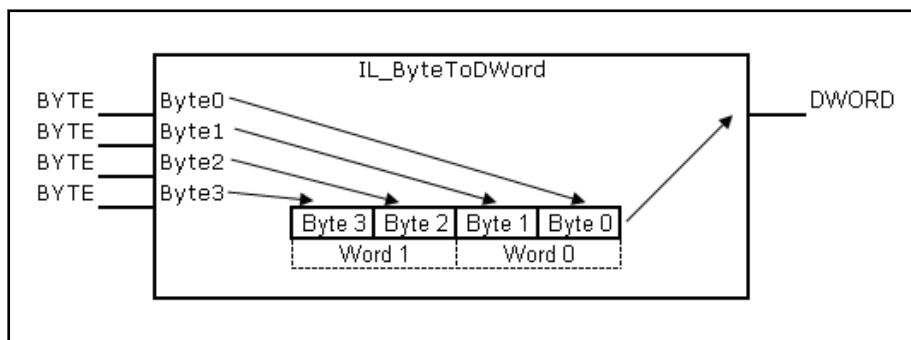


Abb. 4-48: Funktionsbeschreibung IL_ByteToDWord

Fehlerbehandlung

Für diese Funktion ist keine Fehlerbehandlung verfügbar.

4.4.6 IL_ConcatByte

Kurzbeschreibung

Diese Funktion beschreibt die einzelnen Bytes eines Words. Es werden 2 Bytes zu einem Wort zusammengefasst.

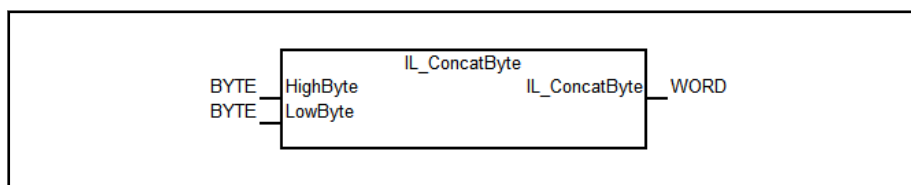
Schnittstellenbeschreibung

Abb. 4-49: Funktion IL_ConcatByte

	Name	Datentyp	Beschreibung
VAR_INPUT	HighByte	BYTE	Beschreibt die Position des Byte 1 im Ausgangswort
	LowByte	BYTE	Beschreibt die Position des Byte 0 im Ausgangswort
RETURN	IL_ConcatByte	WORD	Ausgangswort

Tab. 4-35: Schnittstelle IL_ConcatByte

Funktionsbeschreibung

Diese Funktion schreibt die Bytes der Eingänge entsprechend ihrer Wertigkeit in das Ausgangswort. Entsprechend des Zustandes der Bytes am Eingang der Funktion wird das Wort beschrieben. Das bedeutet, der Zustand des Eingangs "HighByte" wird auf das Byte 1 des Ausgangsdoppelworts kopiert, der Zustand des Eingangs "LowByte" wird auf das Byte 0 des Ausgangsworts kopiert.

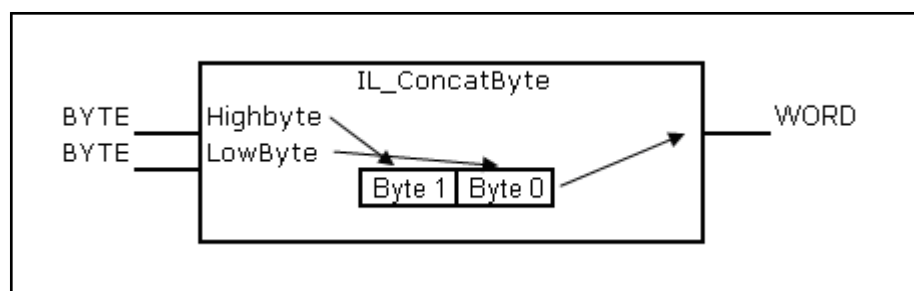


Abb. 4-50: Funktionsbeschreibung IL_ConcatByte

Fehlerbehandlung

Für diese Funktion ist keine Fehlerbehandlung verfügbar.

4.4.7 IL_ConcatWord

Kurzbeschreibung

Diese Funktion beschreibt die einzelnen Worte eines Doppelworts. Es werden 2 Worte zu einem Doppelwort zusammengefasst.

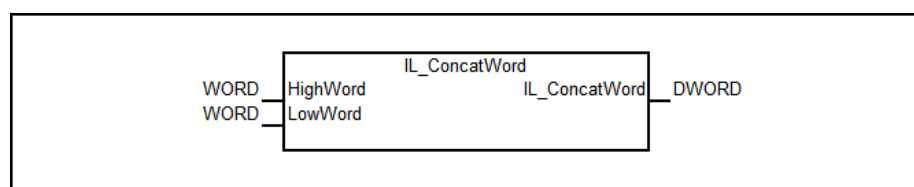
Schnittstellenbeschreibung

Abb. 4-51: Funktion IL_ConcatWord

	Name	Datentyp	Beschreibung
VAR_INPUT	HighWord	WORD	Beschreibt die Position des Word 1 im Ausgangsdoppelwort
	LowWord	WORD	Beschreibt die Position des Word 0 im Ausgangsdoppelwort
RETURN	IL_ConcatWord	DWORD	Ausgangsdoppelwort

Tab. 4-36: Schnittstelle IL_ConcatWord

Funktionsbeschreibung

Diese Funktion schreibt die Worte der Eingänge entsprechend ihrer Wertigkeit in das Ausgangsdoppelwort. Entsprechend des Zustandes der Worte am Eingang der Funktion wird das Doppelwort beschrieben. Das bedeutet, der Zustand des Eingangs "HighWord" wird auf das Wort 1 des Ausgangsdoppelworts kopiert, der Zustand des Eingangs "LowWord" wird auf das Wort 0 des Ausgangsdoppelworts kopiert.

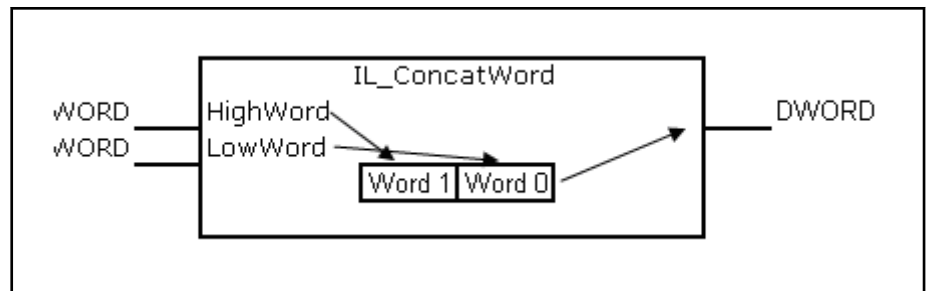


Abb. 4-52: Funktionsbeschreibung IL_ConcatWord

Fehlerbehandlung Für diese Funktion ist keine Fehlerbehandlung verfügbar.

4.5 Konvertierung (Bit-Stream Split)

4.5.1 Einführung und Übersicht

Anwendungsgebiete und Funktionen / Funktionsbausteine von RIL_Uilities

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine	Beschreibung
Konvertierungsfunktionen und -funktionsbausteine (Bit-Stream Split)	
IL_ByteToBool, Seite 119	FB extrahiert 8 Bits (BOOL) aus einem BYTE
IL_DWordToBool, Seite 120	FB extrahiert 32 Bits (BOOL) aus einem DWORD
IL_DWordToByte, Seite 123	FB extrahiert 4 BYTES aus einem DWORD
IL_HighByte, Seite 123	FUN extrahiert höherwertiges BYTE aus einem WORD
IL_HighWord, Seite 124	FUN extrahiert höherwertiges WORD aus einem DWORD
IL_LowByte, Seite 124	FUN extrahiert niederwertiges BYTE aus einem WORD
IL_LowWord, Seite 125	FUN extrahiert niederwertiges WORD aus einem DWORD
IL_WordToBool, Seite 125	FB extrahiert 16 Bits (BOOL) aus einem WORD

Tab. 4-37: RIL_Uilities, Bit-Stream Split

Bit-Stream Split: Hier wird ein Datentypen in kleinere Datentypen aufgeteilt.

Ein Beispiel ist der Funktionsbaustein IL_DWordToByte, der ein Doppelwort in 4 Bytes aufspaltet.

4.5.2 IL_ByteToBool

Kurzbeschreibung

Dieser Funktionsbaustein extrahiert einzelne Bits aus einem Byte. Es werden 8 einzelne Bits aus einem Byte extrahiert.

Schnittstellenbeschreibung

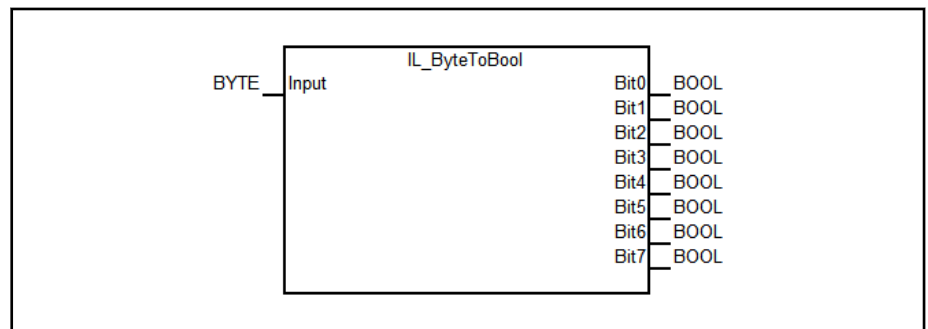


Abb. 4-53: Funktionsbaustein IL_ByteToBool

	Name	Datentyp	Beschreibung
VAR_INPUT	Input	BYTE	Eingangsbyte aus dem die Bits extrahiert werden
VAR_OUTPUT	Bit0	BOOL	Bit 0 des Eingangsbytes
	Bit1	BOOL	Bit 1 des Eingangsbytes
	Bit2	BOOL	Bit 2 des Eingangsbytes
	Bit3	BOOL	Bit 3 des Eingangsbytes
	Bit4	BOOL	Bit 4 des Eingangsbytes
	Bit5	BOOL	Bit 5 des Eingangsbytes
	Bit6	BOOL	Bit 6 des Eingangsbytes
	Bit7	BOOL	Bit 7 des Eingangsbytes

Tab. 4-38: Schnittstelle IL_ByteToBool

Funktionsbeschreibung

Mit diesem Funktionsbaustein werden die einzelnen Bits eines Bytes ermittelt. Der Zustand des Eingangs „Input“ wird erfasst und der Status der einzelnen Bits auf die Ausgangsvariablen "Bit0" .. "Bit7" kopiert.

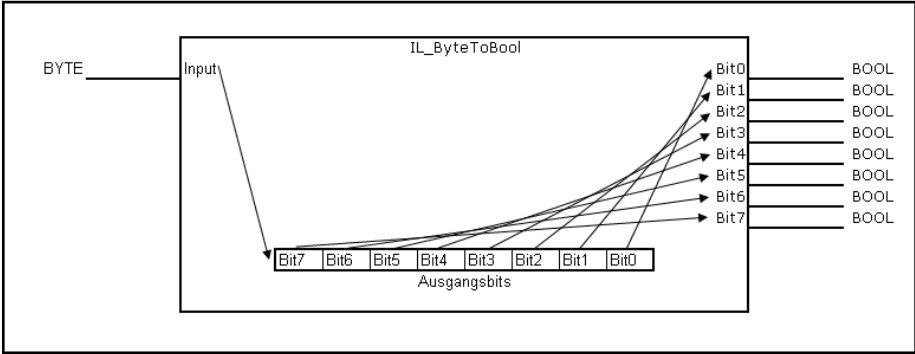


Abb. 4-54: Funktionsbeschreibung IL_ByteToBool

Fehlerbehandlung

Für diesen Funktionsbaustein ist keine Fehlerbehandlung verfügbar.

4.5.3

IL_DWordToBool

Kurzbeschreibung

Dieser Funktionsbaustein extrahiert einzelne Bits aus einem Doppelwort. Es werden 32 einzelne Bits aus einem DWord extrahiert.

Schnittstellenbeschreibung

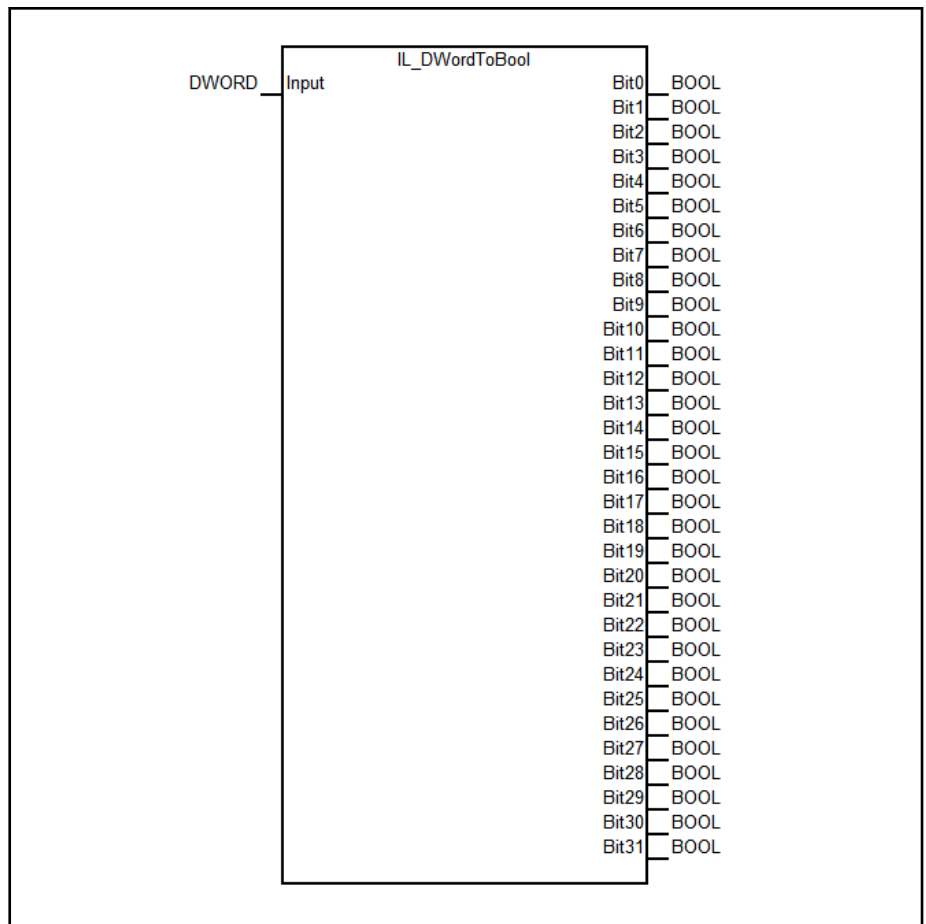


Abb. 4-55: Funktionsbaustein IL_DWordToBool

	Name	Datentyp	Beschreibung
VAR_INPUT	Input	DWORD	Eingangsdoppelwort aus dem die Bits extrahiert werden
VAR_OUPUT	Bit0	BOOL	Bit 0 des Eingangsdoppelworts
	Bit1	BOOL	Bit 1 des Eingangsdoppelworts
	Bit2	BOOL	Bit 2 des Eingangsdoppelworts
	Bit3	BOOL	Bit 3 des Eingangsdoppelworts
	Bit4	BOOL	Bit 4 des Eingangsdoppelworts
	Bit5	BOOL	Bit 5 des Eingangsdoppelworts
	Bit6	BOOL	Bit 6 des Eingangsdoppelworts
	Bit7	BOOL	Bit 7 des Eingangsdoppelworts
	Bit8	BOOL	Bit 8 des Eingangsdoppelworts
	Bit9	BOOL	Bit 9 des Eingangsdoppelworts
	Bit10	BOOL	Bit 10 des Eingangsdoppelworts
	Bit11	BOOL	Bit 11 des Eingangsdoppelworts
	Bit12	BOOL	Bit 12 des Eingangsdoppelworts
	Bit13	BOOL	Bit 13 des Eingangsdoppelworts

	Name	Datentyp	Beschreibung
	Bit14	BOOL	Bit 14 des Eingangsdoppelworts
	Bit15	BOOL	Bit 15 des Eingangsdoppelworts
	Bit16	BOOL	Bit 16 des Eingangsdoppelworts
	Bit17	BOOL	Bit 17 des Eingangsdoppelworts
	Bit18	BOOL	Bit 18 des Eingangsdoppelworts
	Bit19	BOOL	Bit 19 des Eingangsdoppelworts
	Bit20	BOOL	Bit 20 des Eingangsdoppelworts
	Bit21	BOOL	Bit 21 des Eingangsdoppelworts
	Bit22	BOOL	Bit 22 des Eingangsdoppelworts
	Bit23	BOOL	Bit 23 des Eingangsdoppelworts
	Bit24	BOOL	Bit 24 des Eingangsdoppelworts
	Bit25	BOOL	Bit 25 des Eingangsdoppelworts
	Bit26	BOOL	Bit 26 des Eingangsdoppelworts
	Bit27	BOOL	Bit 27 des Eingangsdoppelworts
	Bit28	BOOL	Bit 28 des Eingangsdoppelworts
	Bit29	BOOL	Bit 29 des Eingangsdoppelworts
	Bit30	BOOL	Bit 30 des Eingangsdoppelworts
	Bit31	BOOL	Bit 31 des Eingangsdoppelworts

Tab. 4-39: Schnittstelle IL_DWordToBool

Funktionsbeschreibung

Mit diesem Funktionsbaustein werden die einzelnen Bits eines Doppelworts ermittelt. Der Zustand des Eingangs „Input“ wird erfasst und der Status der einzelnen Bits auf die Ausgangsvariablen "Bit0" .. "Bit31" kopiert.

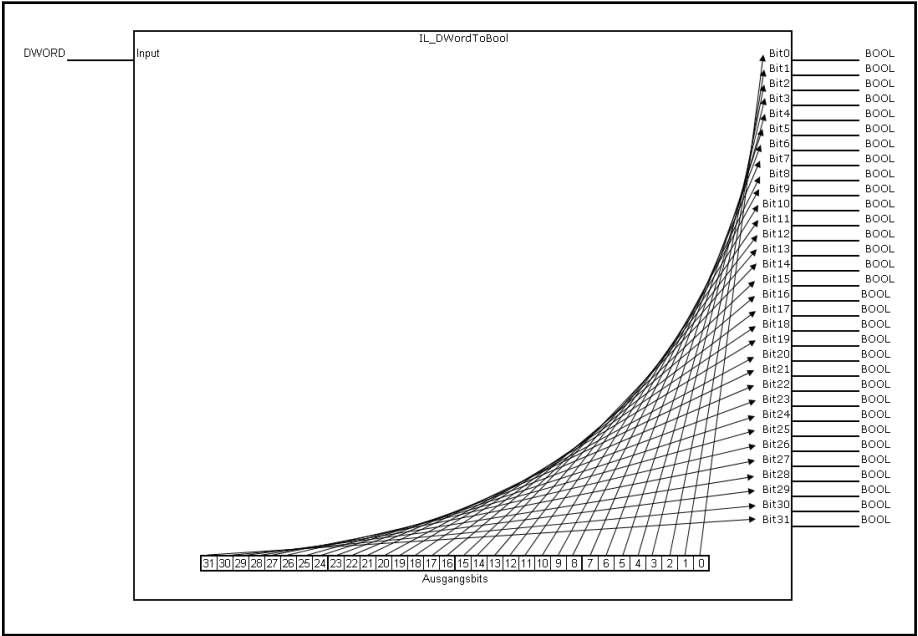


Abb. 4-56: Funktionsbeschreibung IL_DWordToBool

Fehlerbehandlung

Für diesen Funktionsbaustein ist keine Fehlerbehandlung verfügbar.

4.5.4 IL_DWordToByte

Kurzbeschreibung Dieser Funktionsbaustein extrahiert einzelne Bytes aus einem DWord. Es werden 4 einzelne Bytes aus einem Doppelwort extrahiert.

Schnittstellenbeschreibung

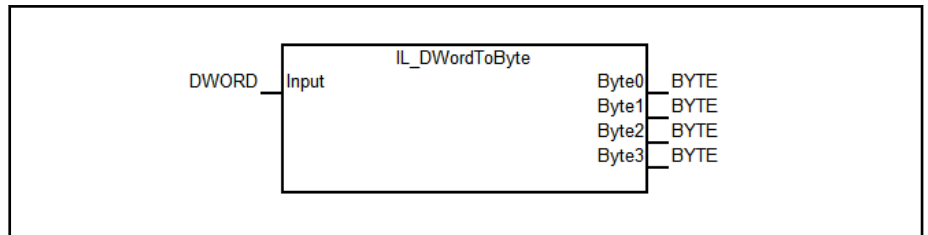


Abb. 4-57: Funktionsbaustein IL_DWordToByte

	Name	Datentyp	Beschreibung
VAR_INPUT	Input	DWORD	Eingangsdoppelwort aus dem die Bytes extrahiert werden
VAR_OUTPUT	Byte0	BYTE	Byte 0 des Eingangsdoppelworts
	Byte1	BYTE	Byte 1 des Eingangsdoppelworts
	Byte2	BYTE	Byte 2 des Eingangsdoppelworts
	Byte3	BYTE	Byte 3 des Eingangsdoppelworts

Tab. 4-40: Schnittstelle IL_DWordToByte

Funktionsbeschreibung Mit diesem Funktionsbaustein werden die einzelnen Bytes eines Doppelworts ermittelt. Der Zustand des Eingangs „Input“ wird erfasst und der Status der einzelnen Bytes auf die Ausgangsvariablen "Byte0" .. "Byte3" kopiert.

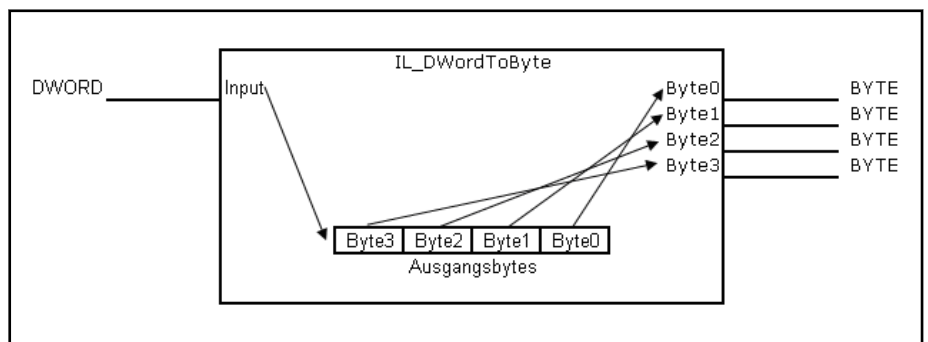


Abb. 4-58: Funktionsbeschreibung IL_DWordToByte

Fehlerbehandlung Für diesen Funktionsbaustein ist keine Fehlerbehandlung verfügbar.

4.5.5 IL_HighByte

Kurzbeschreibung Diese Funktion extrahiert das höherwertige Byte aus einem Wort.

Schnittstellenbeschreibung

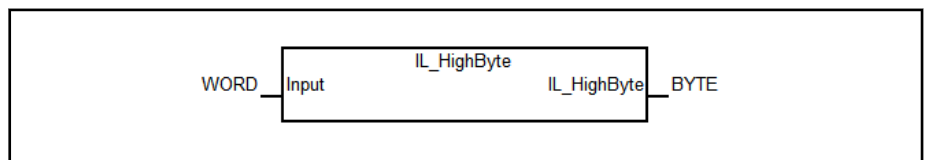


Abb. 4-59: Funktion IL_HighByte

	Name	Datentyp	Beschreibung
VAR_INPUT	Input	WORD	Eingangswort aus dem das Byte extrahiert wird
RETURN	IL_HighByte	BYTE	höherwertiges Byte des Eingangsworts

Tab. 4-41: Schnittstelle IL_HighByte

Funktionsbeschreibung

Mit dieser Funktion wird das höherwertige Byte eines Worts ermittelt. Der Zustand des Eingangs „Input“ wird erfasst und der Status des höherwertigen Bytes auf den Funktionsausgang kopiert.

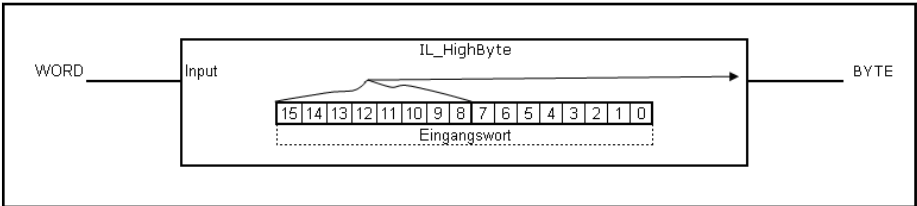


Abb. 4-60: Funktionsbeschreibung IL_HighByte

Fehlerbehandlung

Für diese Funktion ist keine Fehlerbehandlung verfügbar.

4.5.6

IL_HighWord

Kurzbeschreibung

Diese Funktion extrahiert das höherwertige Wort aus einem Doppelwort.

Schnittstellenbeschreibung

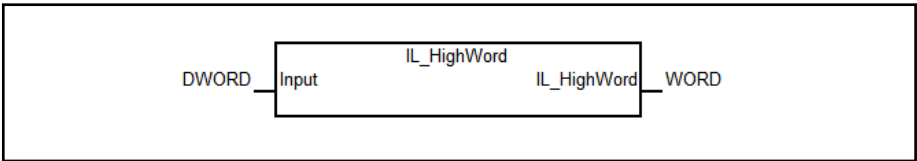


Abb. 4-61: Funktion IL_HighWord

	Name	Datentyp	Beschreibung
VAR_INPUT	Input	DWORD	Eingangsdoppelwort aus dem das Wort extrahiert wird
RETURN	IL_HighWord	WORD	höherwertiges Wort des Eingangsdoppelworts

Tab. 4-42: Schnittstelle IL_HighWord

Funktionsbeschreibung

Mit dieser Funktion wird das höherwertige Wort eines Doppelworts ermittelt. Der Zustand des Eingangs „Input“ wird erfasst und der Status des höherwertigen Worts auf den Funktionsausgang kopiert.

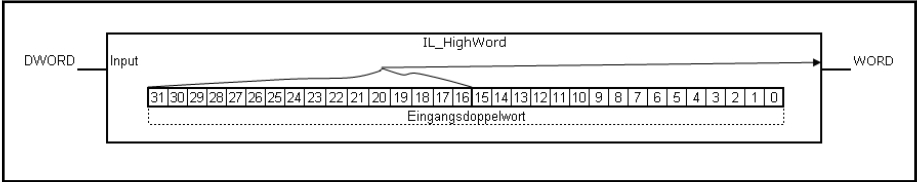


Abb. 4-62: Funktionsbeschreibung IL_HighWord

Fehlerbehandlung

Für diese Funktion ist keine Fehlerbehandlung verfügbar.

4.5.7

IL_LowByte

Kurzbeschreibung

Diese Funktion extrahiert das niederwertige Byte aus einem Wort.

Schnittstellenbeschreibung

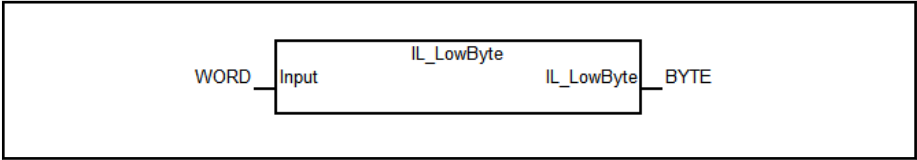


Abb. 4-63: Funktion IL_LowByte

	Name	Datentyp	Beschreibung
VAR_INPUT	Input	WORD	Eingangswort aus dem das Byte extrahiert wird
RETURN	IL_LowByte	BYTE	niederwertiges Byte des Eingangsworts

Tab. 4-43: Schnittstelle IL_LowByte

Funktionsbeschreibung Mit dieser Funktion wird das niederwertige Byte eines Worts ermittelt. Der Zustand des Eingangs „Input“ wird erfasst und der Status des niederwertigen Bytes auf den Funktionsausgang kopiert.

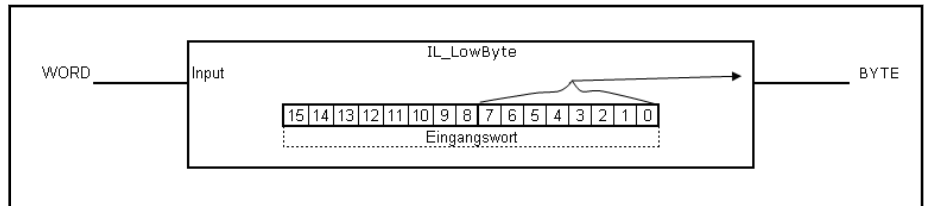


Abb. 4-64: Funktionsbeschreibung IL_LowByte

Fehlerbehandlung Für diese Funktion ist keine Fehlerbehandlung verfügbar.

4.5.8 IL_Word

Kurzbeschreibung Diese Funktion extrahiert das niederwertige Wort aus einem Doppelwort.

Schnittstellenbeschreibung

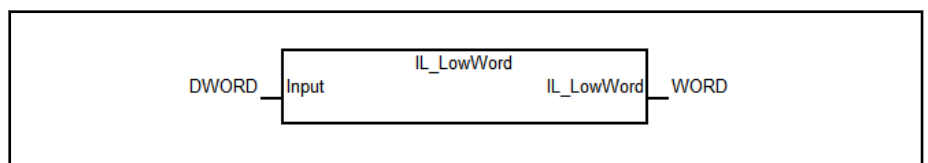


Abb. 4-65: Funktion IL_Word

	Name	Datentyp	Beschreibung
VAR_INPUT	Input	DWORD	Eingangsdoppelwort aus dem das Wort extrahiert wird
RETURN	IL_Word	WORD	niederwertiges Wort des Eingangsdoppelworts

Tab. 4-44: Schnittstelle IL_Word

Funktionsbeschreibung Mit dieser Funktion wird das niederwertige Wort eines Doppelworts ermittelt. Der Zustand des Eingangs „Input“ wird erfasst und der Status des niederwertigen Worts auf den Funktionsausgang kopiert.

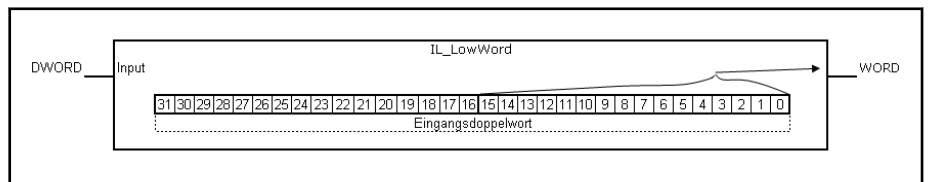


Abb. 4-66: Funktionsbeschreibung IL_Word

Fehlerbehandlung Für diese Funktion ist keine Fehlerbehandlung verfügbar.

4.5.9 IL_WordToBool

Kurzbeschreibung Dieser Funktionsbaustein extrahiert einzelne Bits aus einem Wort. Es werden 16 einzelne Bits aus einem Wort extrahiert.

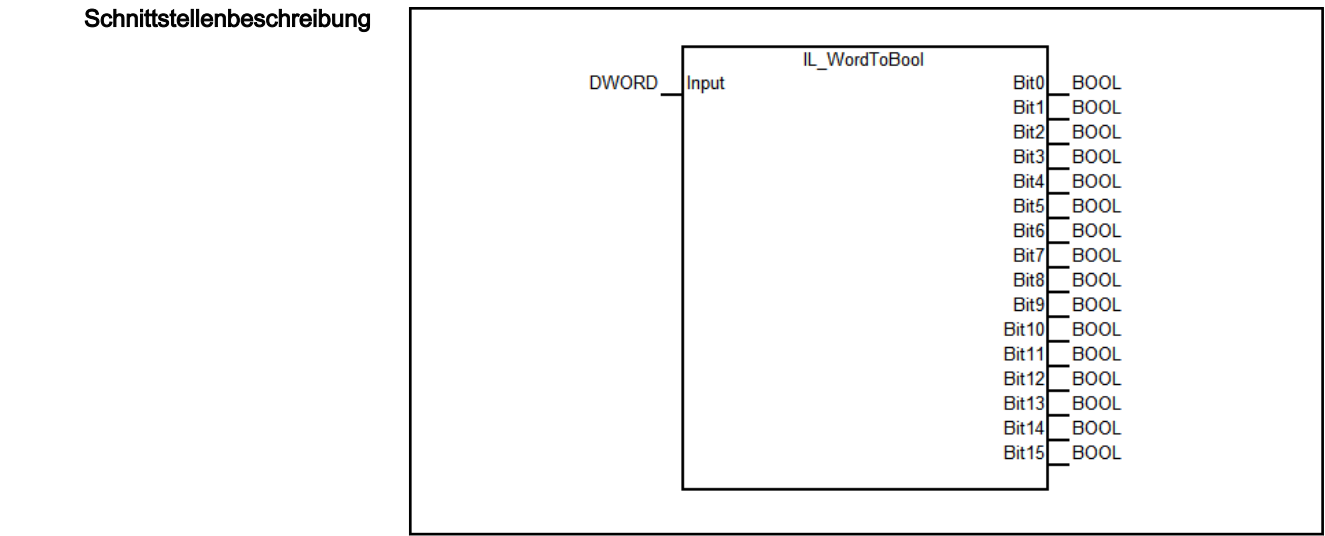


Abb. 4-67: Funktionsbaustein IL_WordToBool

	Name	Datentyp	Beschreibung
VAR_INPUT	Input	WORD	Eingangswort aus dem die Bits extrahiert werden
VAR_OUPUT	Bit0	BOOL	Bit 0 des Eingangsworts
	Bit1	BOOL	Bit 1 des Eingangsworts
	Bit2	BOOL	Bit 2 des Eingangsworts
	Bit3	BOOL	Bit 3 des Eingangsworts
	Bit4	BOOL	Bit 4 des Eingangsworts
	Bit5	BOOL	Bit 5 des Eingangsworts
	Bit6	BOOL	Bit 6 des Eingangsworts
	Bit7	BOOL	Bit 7 des Eingangsworts
	Bit8	BOOL	Bit 8 des Eingangsworts
	Bit9	BOOL	Bit 9 des Eingangsworts
	Bit10	BOOL	Bit 10 des Eingangsworts
	Bit11	BOOL	Bit 11 des Eingangsworts
	Bit12	BOOL	Bit 12 des Eingangsworts
	Bit13	BOOL	Bit 13 des Eingangsworts
	Bit14	BOOL	Bit 14 des Eingangsworts
	Bit15	BOOL	Bit 15 des Eingangsworts

Tab. 4-45: Schnittstelle IL_WordToBool

Funktionsbeschreibung

Mit diesem Funktionsbaustein werden die einzelnen Bits eines Worts ermittelt. Der Zustand des Eingangs „Input“ wird erfasst und der Status der einzelnen Bits auf die Ausgangsvariablen "Bit0" .. "Bit15" kopiert.

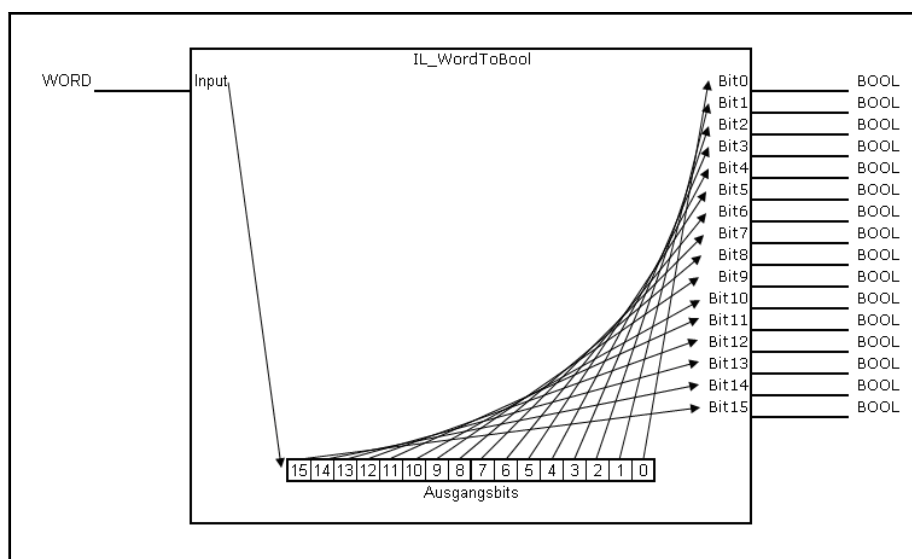


Abb. 4-68: Funktionsbeschreibung IL_WordToBool

Fehlerbehandlung Für diesen Funktionsbaustein ist keine Fehlerbehandlung verfügbar.

4.6 Konvertierung (ByteSwap)

4.6.1 Einführung und Übersicht

Anwendungsgebiete und Funktionen / Funktionsbausteine von RIL_Utilities

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine	Beschreibung
Konvertierungsfunktionen und -funktionsbausteine (ByteSwap)	
IL_SwapDWord, Seite 127	FUN tauscht BYTEs eines DWORDs (3 2 1 0 -> 0 1 2 3) Achtung, nicht PLCopen-kompatibel!
IL_SwapWord, Seite 128	FUN tauscht BYTEs eines WORDs (1 0 -> 0 1)

Tab. 4-46: RIL_Utilities, ByteSwap

Byte-Swap: Hier wird die untere Hälfte des Datentyps gegen die obere vertauscht.

Ein Beispiel ist die Funktion IL_SwapWord, die das niederwertige Byte des Eingangsworts gegen das höherwertige austauscht.

4.6.2 IL_SwapDWord

Kurzbeschreibung

Diese Funktion vertauscht die Bytes eines Doppelworts 3 2 1 0 --> 0 1 2 3.

Schnittstellenbeschreibung

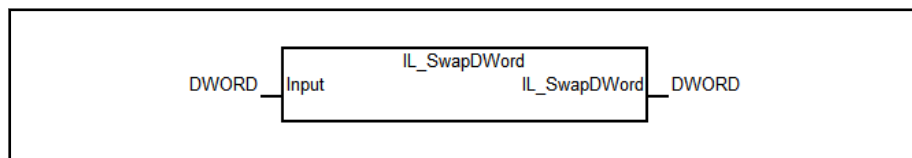


Abb. 4-69: Funktion IL_SwapDWord

	Name	Datentyp	Beschreibung
VAR_INPUT	Input	DWORD	Eingangsdoppelwort
RETURN	IL_Swap-Word	DWORD	Ausgangsdoppelwort, bei dem die Bytes vertauscht wurden.

Tab. 4-47: Schnittstelle IL_SwapDWord

Funktionsbeschreibung Mit dieser Funktion werden die Bytes eines Doppelworts in ihrer Reihenfolge vertauscht.
Der Zustand des Eingangs "Input" wird erfasst.
Die Reihenfolge der Bytes wird verändert von 3 2 1 0 nach 0 1 2 3.
Das Ergebnis wird auf den Funktionsausgang kopiert.

 Der Name der Funktion entspricht nicht den Erwartungen nach IEC 61131-3.

Fehlerbehandlung Für diese Funktion ist keine Fehlerbehandlung verfügbar.

4.6.3 IL_SwapWord

Kurzbeschreibung Diese Funktion vertauscht das höherwertige und das niederwertige Byte eines Wortes.

Schnittstellenbeschreibung

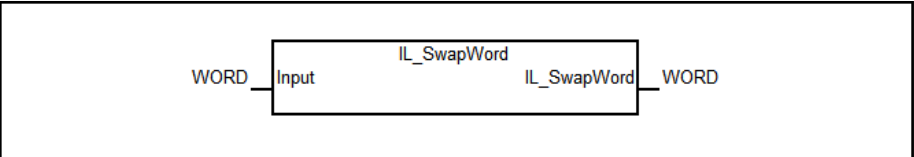


Abb. 4-70: Funktion IL_SwapWord

	Name	Datentyp	Beschreibung
VAR_INPUT	Input	WORD	Eingangswort
RETURN	IL_SwapWord	WORD	Ausgangswort, bei dem das höherwertige Byte mit dem niederwertigen Byte vertauscht wurde.

Tab. 4-48: Schnittstelle IL_SwapWord

Funktionsbeschreibung Mit dieser Funktion wird das höherwertige Byte eines Wortes mit dem niederwertigen Byte vertauscht. Der Zustand des Eingangs "Input" wird erfasst und das höherwertige und niederwertige Byte werden vertauscht auf den Funktionsausgang kopiert.

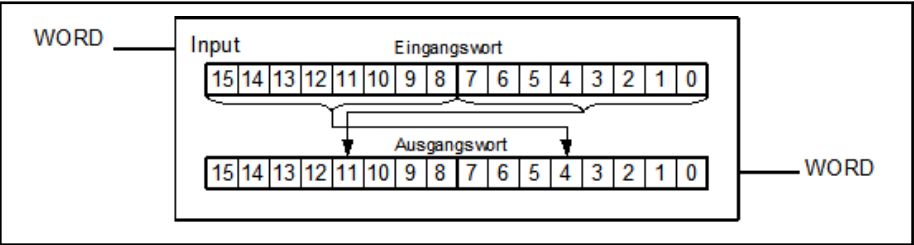


Abb. 4-71: Funktionsbeschreibung IL_SwapWord

Fehlerbehandlung Für diese Funktion ist keine Fehlerbehandlung verfügbar.

4.7 Prüfsummen

4.7.1 Einführung und Übersicht

Anwendungsgebiete und Funktionen / Funktionsbausteine von RIL_Uilities

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine	Beschreibung
Prüfsummen (CRC16)	
IL_CRC16, Seite 129	FB berechnet CRC16-Prüfsumme über ein Datum

Tab. 4-49: RIL_Uilities, CRC16

Dieser Abschnitt enthält Funktionen und Funktionsbausteine, die Prüfsummen berechnen.

Derzeit ist nur ein Funktionsbaustein in dieser Funktionsgruppe implementiert.

4.7.2 IL_CRC16

Kurzbeschreibung

Der Funktionsbaustein IL_CRC16 dient der Berechnung einer CRC16-Checksumme über ein Datum.

Schnittstellenbeschreibung

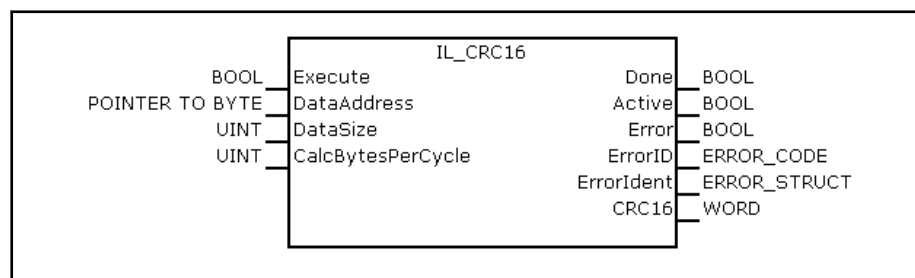


Abb. 4-72: Funktionsbaustein IL_CRC16

	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe (zyklisch, flankengesteuert)
	DataAddress	POINTER TO BYTE	Adresse des Datums über dessen die Checksumme berechnet werden soll
	DataSize	UINT	Größe des Datums in Bytes
	CalcBytesPerCycle	UINT	Größe des Teildatenblocks in Bytes, der pro Aufruf berechnet werden soll.
VAR_OUTPUT	Done	BOOL	Der Baustein hat die Bearbeitung abgeschlossen
	Active	BOOL	Der Funktionsbaustein arbeitet
	Error	BOOL	Zeigt an, dass ein Fehler aufgetreten ist
	ErrorID	ERROR_CODE	Fehlerursache
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerursache Fehlercodes, Seite 130
	CRC16	WORD	Berechnete Prüfsumme

Tab. 4-50: Funktionsbaustein IL_CRC16

Funktionsbeschreibung

Mit einer steigenden Flanke des Enable Einganges werden die Eingangswerte übernommen und die Berechnung der Prüfsumme gestartet.
Die Adresse des Datums ("DataAddress") und dessen Größe ("DataSize") müssen gültige Werte haben.
Ist der Eingang "CalcBytesPerCycle" 0 wird die Prüfsumme beim ersten Aufruf vollständig berechnet. Andernfalls werden nur die in "CalcBytesPerCylce" angegebene Anzahl Bytes berechnet, bevor der Funktionsbaustein zurückkehrt. Dadurch lässt sich eine zu hohe Rechenlast vermeiden.
Der Wert am Ausgang "CRC16" ist nur gültig, wenn der Ausgang "Done" gesetzt ist.

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann die folgenden Fehlermeldungen in Additional1/Additional2 erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#00000002	16#00000001	Der Eingang "DataAddress" ist 0.
INPUT_RANGE_ERROR	16#00000002	16#00000002	Der Eingang "DataSize" ist 0.

Tab. 4-51: Fehlercodes, die von IL_CRC16 generiert werden

4.8

Konvertierung (Datatype Change)

4.8.1

Einführung und Übersicht

Anwendungsgebiete und Funktionen / Funktionsbausteine von RIL_Uutilities

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine	Beschreibung
Konvertierung (Datatype Change)	
IL_AsciiToByte auf Seite 130	FUN wandelt beliebiges ASCII-Zeichen in Bytecode um
IL_ByteToAscii auf Seite 131	FUN wandelt beliebigen Bytecode in ein ASCII-Zeichen um
IL_RealToString auf Seite 131	FUN wandelt REAL-Wert in einen STRING um
IL_LRealToString auf Seite 132	FUN wandelt (L)REAL-Wert formatgesteuert in einen STRING um

Tab. 4-52: RIL_Uutilities, Datatype Change

Die Funktionen interpretieren ihre Eingangsdaten entsprechend der in der Beschreibung vorgegebenen Richtlinie.

4.8.2

IL_AsciiToByte

Kurzbeschreibung

Diese Funktion wandelt ein beliebiges ASCII-Zeichen in den entsprechenden Bytecode um.

Schnittstellenbeschreibung

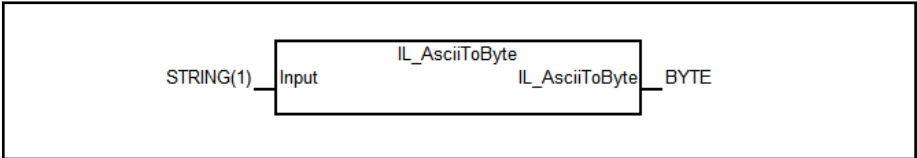


Abb. 4-73: Funktion IL_AsciiToByte

	Name	Datentyp	Beschreibung
VAR_INPUT	Input	BYTE	Eingangsbyte das den Bytecode enthält
RETURN	IL_AsciiToByte	STRING[1]	Der Funktionsausgang enthält das ASCII-Zeichen, das dem Bytecode im Eingangsstring entspricht.

Tab. 4-53: Schnittstelle IL_AsciiToByte

Funktionsbeschreibung Mit dieser Funktion wird das ASCII-Zeichen des Bytecodes im Eingangsbyte ermittelt. Der Zustand des Eingangs „Input“ wird erfasst und das entsprechende ASCII-Zeichen auf den Funktionsausgang kopiert.

Fehlerbehandlung Für diese Funktion ist keine Fehlerbehandlung verfügbar.

4.8.3 IL_ByteToAscii

Kurzbeschreibung Diese Funktion wandelt einen beliebigen Bytecode in das entsprechende ASCII-Zeichen um.

Schnittstellenbeschreibung

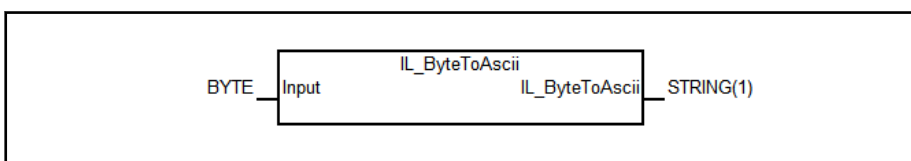


Abb. 4-74: Funktion IL_ByteToAscii

	Name	Datentyp	Beschreibung
VAR_INPUT	Input	BYTE	Eingangsbyte das den zu wandelnden Bytecode enthält
RETURN	IL_ByteToAscii	STRING[1]	Funktionsausgang enthält das ASCII-Zeichen des Bytecodes im Eingangsstring

Tab. 4-54: Schnittstelle IL_ByteToAscii

Funktionsbeschreibung Mit dieser Funktion wird der Bytecode des ASCII-Zeichen im Eingangsstring ermittelt. Der Zustand des Eingangs „Input“ wird erfasst und der entsprechende ASCII-Wert des Zeichens auf den Funktionsausgang kopiert.

Fehlerbehandlung Für diese Funktion ist keine Fehlerbehandlung verfügbar.

4.8.4 IL_RealToString

Kurzbeschreibung Diese Funktion wandelt einen Realwert in einen String um.

Schnittstellenbeschreibung

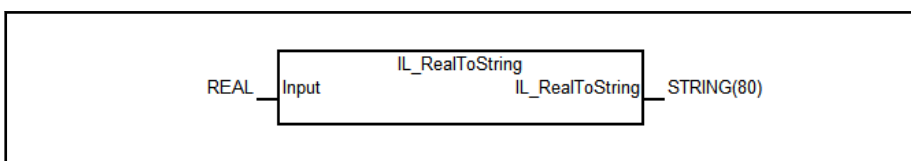


Abb. 4-75: Funktion IL_RealToString

	Name	Datentyp	Beschreibung
VAR_INPUT	Input	REAL	Eingangsbyte das die zu wandelnde Zahl enthält
RETURN	IL_RealToString	STRING	Funktionsausgang enthält den Realwert als String.

Tab. 4-55: Schnittstelle IL_RealToString

Funktionsbeschreibung Mit dieser Funktion wird der Inhalt des Eingangs "Input" erfasst, in einen String konvertiert und auf den Ausgang der Funktion kopiert. Ab Werten kleiner "0.1" verwendet IndraLogic automatisch die Exponentialschreibweise (z.B. 0.0126 wird als 1.26e-002 dargestellt). Die Funktion "IL_RealToString"

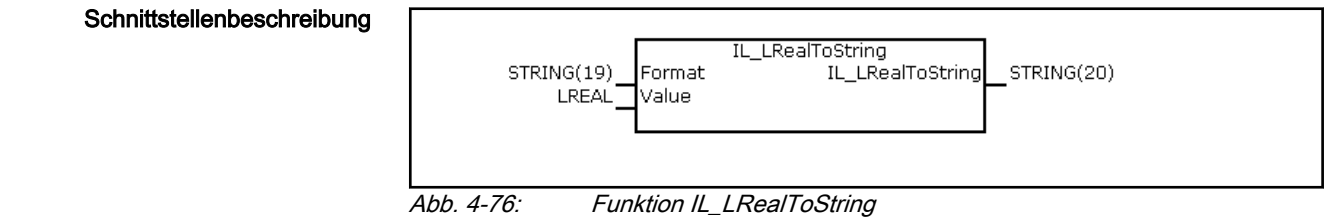
ignoriert dies und gibt die Dezimaldarstellung des Realwert als String aus. Die Funktion "IL_RealToString" berücksichtigt aber nur Werte bis zur vierten Nachkommastelle (z.B. aus Real 0.0123789 wird der String 0.0124)!

Fehlerbehandlung Für diese Funktion ist keine Fehlerbehandlung verfügbar.

4.8.5 IL_LRealToString

Kurzbeschreibung Diese Funktion wandelt einen (L)REAL-Wert in einen STRING um. Dabei wird die Zahl abhängig vom Inhalt des Formatstrings gewandelt.

 Wird eine REAL-Zahl an einen LREAL-Eingang angelegt, erfolgt eine implizite Konvertierung von REAL nach LREAL.



	Name	Datentyp	Beschreibung
VAR_INPUT	Format	STRING(19)	Formatstring für die Darstellung von Value
	Value	LREAL	zu wandelnde Zahl
Rückgabewert	IL_LRealToSt-ring	STRING(20)	Funktionsausgang enthält den LREAL-Wert als STRING

Tab. 4-56: Schnittstelle IL_LRealToString

Funktionsbeschreibung Diese Funktion wandelt einen LREAL-Wert in einen STRING um. Dabei wird die Zahl abhängig vom Inhalt des Formatstrings formatiert.

Der Formatstring bestimmt die Anzahl der Stellen vor und nach dem Komma. Mit ihm können führende Nullen ausgegeben, oder zwischen Dezimalpunkt und -komma gewählt werden.

Syntax des Formatstrings:

[0|#] [{#}] [.,|,] [{#}]

- '#': Auszugebende Stelle(n) vor bzw. nach dem Komma
- '0': Ausgabe führender Nullen
- '.': Dezimaltrennzeichen (US)
- ',' : Dezimaltrennzeichen (DE)

Wirkung des Formatstrings

LREAL-Zahl	Format	Ergebnis	Bemerkung
-1.23456789E02	###.###	-123.457	Die letzte Stelle wird gerundet
1.23456789E02	###.###	_123.457	Positive Zahlen haben immer ein führendes Leerzeichen
1.23456789E02	###,###	_123,457	Verwendet ein Komma als Dezimaltrennzeichen
1.2	###.###	___1.200	Das Format wird immer rechts und links ausgefüllt

LREAL-Zahl	Format	Ergebnis	Bemerkung
-1.2345678E-06	###.###	__-0.000	Wird ein negativer Wert zu Null gerundet, so erscheint '-0'
1.23456789E05	###.###	_12345.679	Kann die Zahl nicht in dem Format ausgegeben werden, dann wird das Format erweitert
-1.23456789E02	#####	__-123	Ausgabe als natürliche Zahl
1.23456789E02	0#####	_00123	Ausgabe mit führenden Nullen
-1.23456789E02	0#####	-00123	Wie oben
1.2345E02	(leer)	1.2345E02	Ein leeres Format wird mit der Standard LREAL_TO_STRING() Funktion gewandelt
1.23456789E02	##0.##	Format error	Bei fehlerhaftem Formatstring wird 'Format error' ausgegeben

Bemerkungen

Zulässige Werte für LREAL:

$\pm (2.2250738585072014e-308 \dots 1.7976931348623158e+308)$

Werte $> 9.0007199254741e+15$ oder $< -9.0007199254741e+15$ ($\pm (2e+53 - 1)$) werden ausschließlich mit LREAL_TO_STRING() umgewandelt.

LREAL	Format	STRING
-2.2250738585072014e-308	'###.###'	__-0.000
-999999.999999	'0#####'	-0001000000
0.1	'###.###'	__0.100
0.0314e+4	'#####.###'	__314.000
3.402823466e+38	"	3.402823466e+38

Tab. 4-57: Konvertierungsbeispiele

Fehlerbehandlung Kann ein Wert nicht in dem vorgegebenen Format ausgegeben werden, so wird das Format entsprechend erweitert.

Wird ein fehlerhaftes Format angegeben so wird 'Format Error' ausgegeben.

4.9 Debuggen (Debugging)

4.9.1 Einführung und Übersicht

Es stehen Funktionen zur Verarbeitung der hochaufgelösten CPU-Ticks zur Verfügung. Mit diesen Funktionen kann die Laufzeit von Programmmodulen bestimmt werden:

Anwendungsgebiete und Funktionen / Funktionsbausteine von RIL_Uutilities

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine	Beschreibung
Debuggen (Debugging)	

IL_HighResTimeDiff, Seite 134	berechnet die Zeitdifferenz zweier hochaufgelöster Zeittakte des Systems in Mikrosekunden
IL_HighResTimeTick, Seite 134	liest den hochaufgelösten Zeittakt des Systems

Tab. 4-58: RIL_Uutilities, Debugging

4.9.2 IL_HighResTimeDiff

Kurzbeschreibung Die Funktion IL_HighResTimeDiff berechnet die Zeitdifferenz zweier hochaufgelöster Zeittakte des Systems in Mikrosekunden.

Schnittstellenbeschreibung

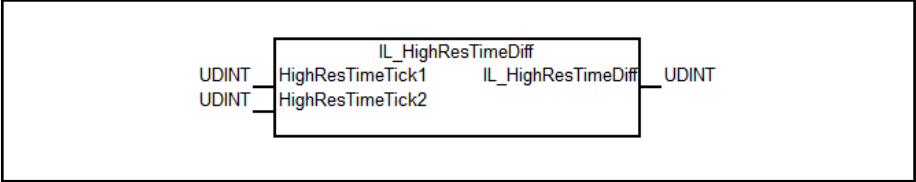


Abb. 4-77: Funktion IL_HighResTimeDiff

	Name	Datentyp	Beschreibung
VAR_INPUT	HighResTimeTick1	UDINT	Hochaufgelöster Zeittakt des Systems vor dem Codeabschnitt
	HighResTimeTick2	UDINT	Hochaufgelöster Zeittakt des Systems nach dem Codeabschnitt
RETURN	IL_HighResTimeDiff	UDINT	Zeitdifferenz zweier hochaufgelöster Zeittakte des Systems in µs

Tab. 4-59: Schnittstelle IL_HighResTimeDiff

Spezifikation Die Funktion IL_HighResTimeDiff wird zusammen mit der Funktion IL_HighResTimeTick zur Bestimmung der Laufzeit eines Codeabschnittes verwendet



Das Auslesen des hochaufgelösten Zeittaktes benötigt systemabhängig jeweils ca. ein bis zwei Mikrosekunden, die ggf. bei der Ermittlung der Zeitdifferenz durch den Anwender zusätzlich berücksichtigt werden können.

Funktionsbeschreibung Die Funktion IL_HighResTimeDiff berechnet die Zeitdifferenz zweier hochaufgelöster Zeittakte des Systems in Mikrosekunden.

Fehlerbehandlung Für diese Funktion ist keine Fehlerbehandlung verfügbar.

Anwendungsbeispiel siehe [IL_HighResTimeTick, Seite 135](#).

4.9.3 IL_HighResTimeTick

Kurzbeschreibung Die Funktion IL_HighResTimeTick liest den hochaufgelösten Zeittakt des Systems.

Schnittstellenbeschreibung

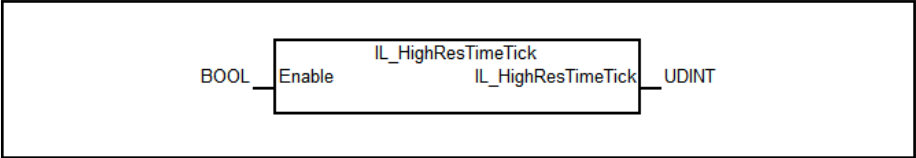


Abb. 4-78: Funktion IL_HighResTimeTick

	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe der Funktion (zyklisch, zustandsgesteuert)
RETURN	IL_HighResTimeTick	UDINT	Hochaufgelöster Zeittakt des Systems

Tab. 4-60: Schnittstelle IL_HighResTimeTick

Spezifikation Die Funktion IL_HighResTimeTick wird zusammen mit der Funktion IL_HighResTimeDiff zur Bestimmung der Laufzeit eines Codeabschnittes verwendet



Der Rückgabewert der Funktion IL_HighResTimeTick sollte auf Grund der speziellen, systemabhängigen Zeitbasis nicht direkt als Zeitwert verwendet werden.

Funktionsbeschreibung Die Funktion IL_HighResTimeTick liest nach der Bearbeitungsfreigabe der Funktion mittels "Enable" zyklisch den hochaufgelösten Zeittakt des Systems aus.

Fehlerbehandlung Für diese Funktion ist keine Fehlerbehandlung verfügbar.

Anwendungsbeispiel In diesem Programmbeispiel wird die Laufzeit eines Programmblocks gemessen.

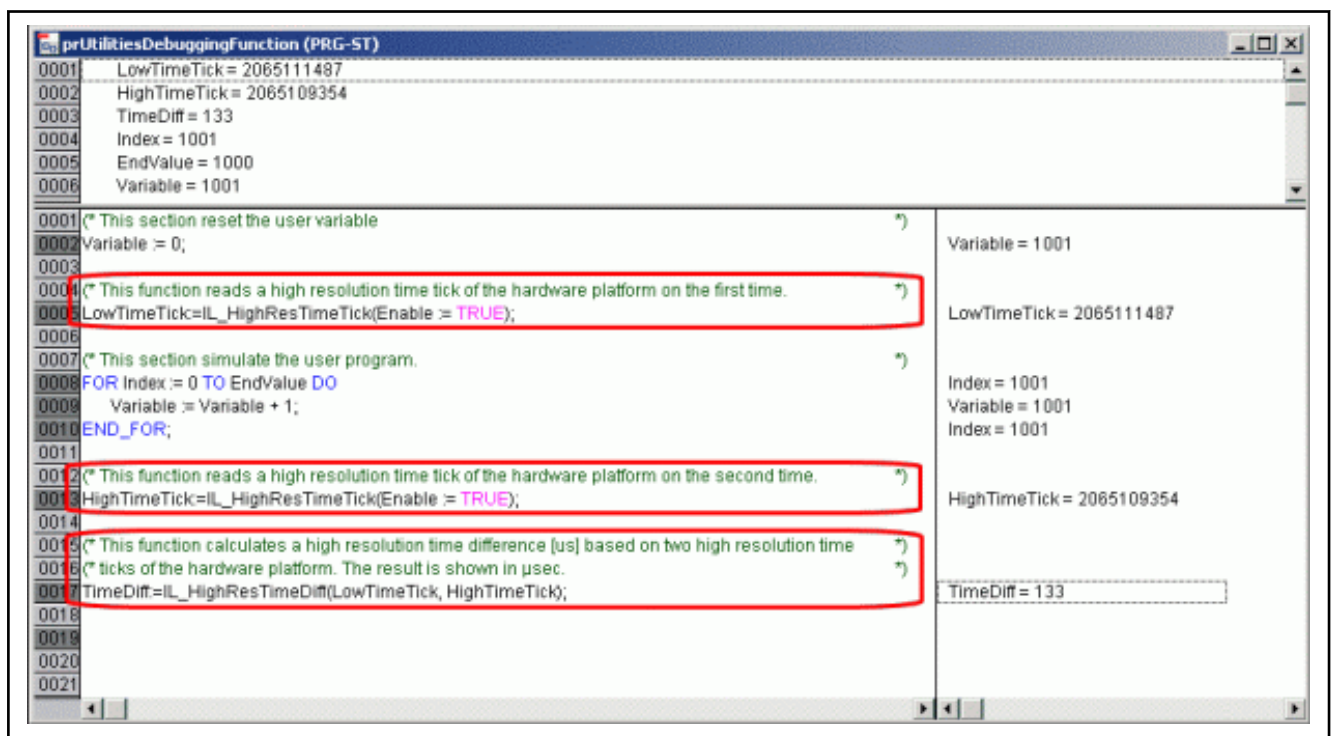


Abb. 4-79: Anwendungsbeispiel IL_HighResTimeTick

4.10 FlipFlops (FlipFlops)

4.10.1 Einführung und Übersicht

Dieser Funktionsbaustein erweitert die durch die Standard-Bibliothek angebotenen Triggerbausteine.

Anwendungsgebiete und Funktionen / Funktionsbausteine von RIL_Uilities

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine	Beschreibung
FlipFlops (FlipFlops)	
IL_Toggle auf Seite 136	setz- und rücksetzbares FlipFlop

Tab. 4-61: RIL_Uilities, FlipFlops

4.10.2 IL_Toggle

Kurzbeschreibung Dieser Funktionsbaustein wechselt den Status seines booleschen Ausgang mit jeder steigenden Flanke am "Execute" Eingang. Zwei zusätzliche Eingänge bieten die Möglichkeit, den booleschen Ausgang fest auf TRUE oder FALSE zu setzen.

Schnittstellenbeschreibung

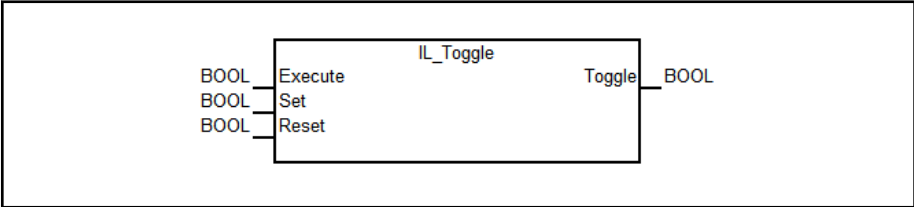


Abb. 4-80: Funktionsbaustein IL_Toggle

	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Steigende Flanke wechselt den Status des Ausgangs "Toggle"
	Set	BOOL	Setzt den Ausgang Toggle auf TRUE
	Reset	BOOL	Setzt den Ausgang Toggle auf FALSE
VAR_OUPUT	Toggle	BOOL	aktueller Zustand

Tab. 4-62: Schnittstelle IL_Toggle

Signal-Zeit-Diagramm

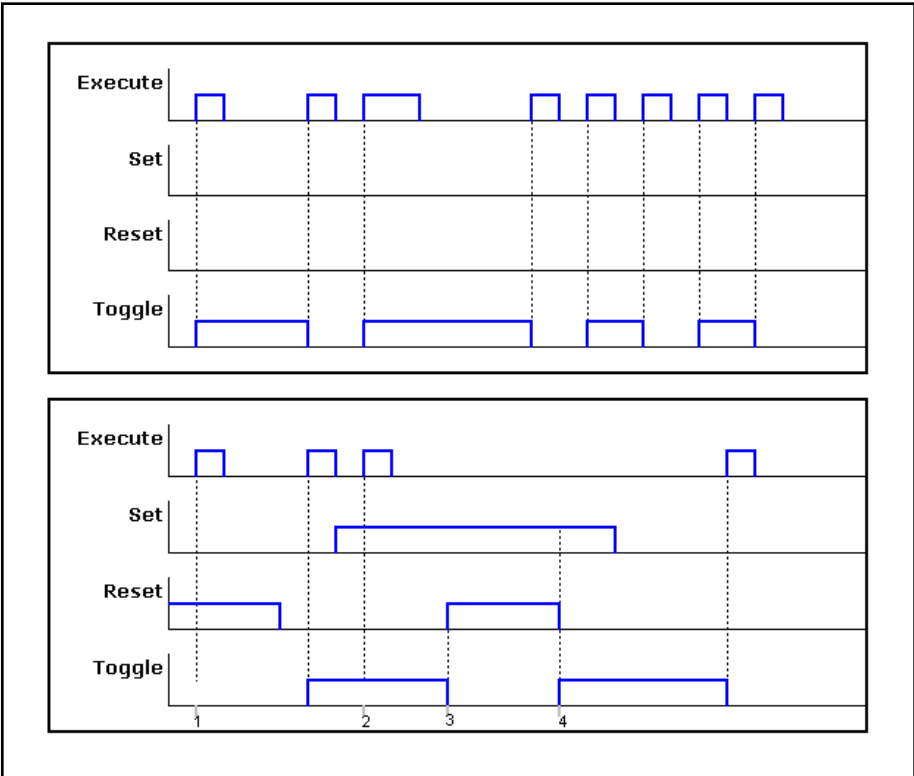


Abb. 4-81: Signal-Zeit-Diagramm des Funktionsbausteins IL_Toggle

Oben: Verhalten von "Toggle" wenn "Set" und "Reset" FALSE bleiben
Unten: Verhalten von "Toggle" wenn "Set", "Reset" und "Execute" zusammenspielen

- 1. "Toggle" bleibt FALSE, weil "Reset" TRUE ist
- 2. "Toggle" bleibt TRUE, weil "Set" TRUE ist

	3. "Toggle" wird FALSE, da "Reset" dominiert
	4. "Toggle" wird TRUE, weil "Set" TRUE ist
Funktionsbeschreibung	Der Funktionsbaustein IL_Toggle wechselt den Status seines Ausgangs "Toggle" zwischen TRUE und FALSE mit jeder steigenden Flanke am Eingang "Execute". Der Eingang "Set" fixiert den Ausgang "Toggle" auf TRUE. Der Eingang "Reset" fixiert den Ausgang "Toggle" auf FALSE. Sind beide Eingänge "Set" und "Reset" TRUE so dominiert "Reset" sodass der Ausgang "Toggle" fest auf FALSE gesetzt wird. Solange "Set" oder "Reset" TRUE sind haben Signaländerungen an "Execute" keine Auswirkungen auf den Ausgang "Toggle".
Fehlerbehandlung	Für diesen Funktionsbaustein ist keine Fehlerbehandlung verfügbar.

4.11 Mathematische Funktionen (Mathematical Functions)

4.11.1 Einführung und Übersicht

Die nachfolgenden Bausteine stellen mathematische Funktionen zur Verfügung.



Die genannten Funktionsbausteinen haben beim Übergang zur Version 14 ihre Namen von ML_* nach IL_* geändert und sind von "ML_TechBase" nach "RIL_Uilities" übergegangen.

Anwendungsgebiete und Funktionen / Funktionsbausteine von RIL_Uilities

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine	Beschreibung
Mathematische Funktionen (Mathematical Functions)	
IL_InterpolationLinear, Seite 137,	dient zur Berechnung von Zwischenpunkten und kann z.B. für Linearisierungen verwendet werden. Der Funktionsbaustein verwendet die Struktur IL_COORD_PAIR_TYPE01, Seite 140
IL_MaxValue, Seite 139,	gibt mit jedem Aufruf den Maximalwert aller bisher aufgetreten Wert an

Tab. 4-63: RIL_Uilities, Mathematische Funktionen

4.11.2 IL_InterpolationLinear

Kurzbeschreibung Der Funktionsbaustein IL_InterpolationLinear berechnet auf der Grundlage einer definierten X-Y-Stützstellentabelle zu einem gegebenen X-Wert durch lineare Interpolation zyklisch den zugehörigen Y-Wert.



Der Funktionsbaustein "ML_InterpolationLinear (ML_TechBase-Bibliothek)" wurde mit Version 14VRS in "IL_InterpolationLinear" umbenannt.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek
MLC 15VRS	RIL_Uilities

Tab. 4-64: Referenztablelle Funktionsbaustein IL_InterpolationLinear

Schnittstellenbeschreibung

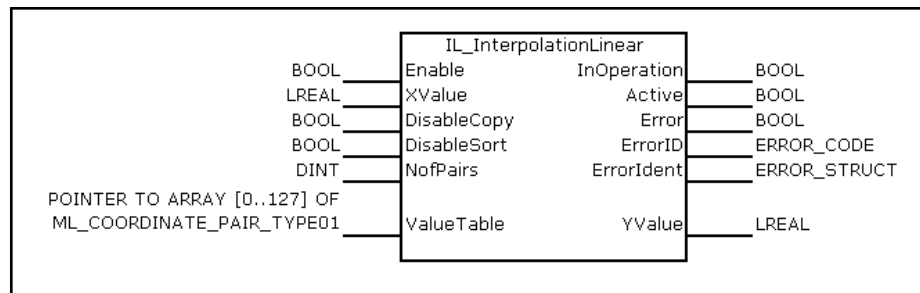


Abb. 4-82: Funktionsbaustein IL_InterpolationLinear

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Freigabe des Funktionsbausteins (zyklisch, zustands-gesteuert)
	XValue	LREAL	X-Wert
	DisableCopy	BOOL	Deaktivierung des Kopierens der Stützstellentabelle
	DisableSort	BOOL	Deaktivierung des Sortierens der Stützstellentabelle
	NofPairs	DINT	Anzahl der zusammengehörigen Wertepaare in der Stützstellentabelle
	ValueTable	POINTER TO ARRAY [0..127] OF IL_COORDINATE_PAIR_TYPE01	Adresse der Stützstellentabelle IL_COORD_PAIR_TYPE01, Seite 140
VAR_OUTPUT	InOperation	BOOL	Zyklische Berechnung eines Y-Wertes ohne Fehler abgeschlossen, Ausgangsvariable "YValue" gültig
	Active	BOOL	Berechnung eines Y-Wertes aktiv, Ausgangsvariablen ungültig
	Error	BOOL	Berechnung eines Y-Wertes mit Fehler abgeschlos-sen, Fehlerausgänge "ErrorID" und "ErrorIdent" gültig
	ErrorID	ERROR_CODE	Bei gesetztem "Error"-Ausgang enthält dieser Aus-gang eine grobe Klassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bei gesetztem "Error"-Ausgang enthält dieser Aus-gang detaillierte Informationen über den Fehler Fehlerbehandlung, Seite 139
	YValue	LREAL	Y-Wert

Tab. 4-65: Schnittstellenvariablen Funktionsbaustein IL_InterpolationLinear

Funktionsbeschreibung

Der Funktionsbaustein IL_InterpolationLinear berechnet nach der Bearbeitungsfreigabe des Funktionsbausteins mittels "Enable" zyklisch zum Eingangswert "XValue" den zugehörigen Y-Wert "YValue" auf Basis einer vorgegebenen Stützpunkttable.

Die Stützpunkttable wird abhängig von den Eingängen "DisableCopy" und "DisableSort" bei steigender Flanke auf ein internes Datenarray gesichert und anschließend aufsteigend sortiert. Die Zeit für die Berechnung eines zugehörigen Y-Wertes "YValue" kann minimiert werden, wenn bei Bearbeitungsfreigabe des Bausteins eine bereits aufsteigend sortierte Stützstellentabelle übergeben und dies durch den Eingang "DisableSort" gekennzeichnet wird. In diesem Falle wird das Datenarray nicht sortiert. Soll bei einer erneuten Abarbeitung des Bausteins die intern vorhandene und bereits verwendete Stützstellentabelle erneut angewendet werden, so muss sowohl der Eingang "DisableCopy" als auch der Eingang "Disable-Sort" auf aktiviert gesetzt werden.

Der zu berechnende Y-Wert wird im Falle der fehlerfreien Bearbeitung des Funktionsbausteins auf dem Ausgang "YValue" abgelegt, wobei die Gültigkeit des Wertes durch den Ausgang "InOperation" signalisiert wird.

Kann der zu berechnende Y-Wert nach Aktivierung des Funktionsbausteins nicht unmittelbar zur Verfügung gestellt werden, wird dies durch den Ausgang "Active" angezeigt.

Tritt während der Bearbeitung des Funktionsbausteins ein Fehler auf, so wird dieser durch den Ausgang "Error" signalisiert. Der Ausgang "YValue" wird im Fehlerfall nicht aktualisiert.

Einfluss der Eingänge "DisableCopy" und "DisableSort" auf das Laufzeitverhalten des FB

DisableCopy	DisableSort	Anwendungsfall
FALSE	FALSE	Eine unsortierte Stützstellentabelle wird vorgegeben, die in dieser Art in der vorherigen Berechnung noch nicht verwendet worden ist. Die Stützstellentabelle wird kopiert und sortiert
FALSE	TRUE	Eine bereits aufsteigend sortierte Stützstellentabelle wird vorgegeben, die in dieser Art in der vorherigen Berechnung noch nicht verwendet worden ist. Die Stützstellentabelle wird kopiert, aber intern nicht sortiert
TRUE	FALSE	Diese Kombination bildet keinen sinnvollen Zustand
TRUE	TRUE	Die gleiche Stützstellentabelle wurde in vorherigen Berechnungen bereits verwendet. Sie ist daher intern bereits bekannt und aufsteigend sortiert

Tab. 4-66: Belegung der Eingänge DisableCopy und DisableSort



Die Sortierung der Stützstellentabelle nach Aktivierung des Funktionsbausteins bedingt unter Umständen, dass der Funktionsbaustein auf Grund der Einschränkung auf eine max. zulässigen Laufzeit den gewünschten Ausgangswert nicht im ersten bzw. Folgezyklus liefern kann. Diese Zeitdauer zu Beginn der Aktivierung wird durch "Active" signalisiert.

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle **MLC_TABLE**, 16#0030. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR (16#0006)	16#F0260030	0	Eingänge ungültig (XValue außerhalb der Stützpunkte)
STATE_MACHINE_ERROR (16#0005)	16#F0260031	xx	ungültiger Zustand der Zustandsmaschine. xx gibt den ungültigen Zustand an

Tab. 4-67: Fehlercodes Funktionsbaustein IL_InterpolationLinear

4.11.3 IL_MaxValue

Kurzbeschreibung Der Funktionsbaustein IL_MaxValue ermittelt aus dem zeitlichen Verlauf des Eingangssignals dessen bisherigen Maximalwert.



Der Funktionsbaustein "ML_MaxValue (ML_TechBase-Bibliothek)" wurde mit Version 14VRS in "IL_MaxValue" umbenannt.

Zuordnung Zielsystem/Bibliothek	Zielsystem	Bibliothek
	MLC 15VRS	RIL_Uilities

Tab. 4-68: Referenztabelle Funktionsbaustein IL_MaxValue

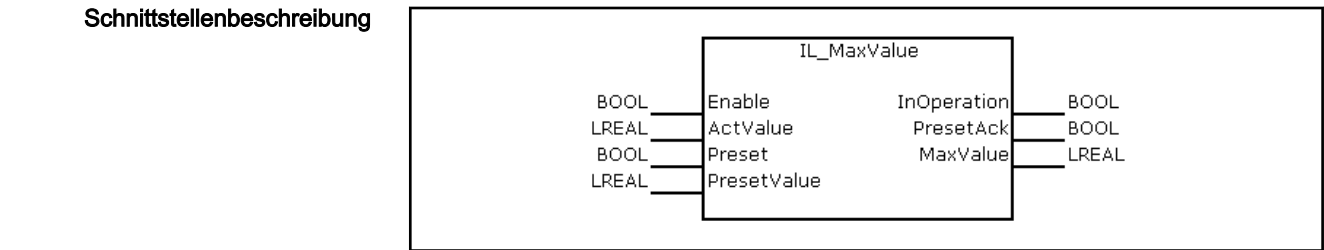


Abb. 4-83: Funktionsbaustein IL_MaxValue

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (zyklisch, zustandsgesteuert)
	ActValue	LREAL	aktueller Wert
	Preset	BOOL	Aktivierung des Vorgabewertes
	PresetValue	LREAL	Vorgabewert
VAR_OUTPUT	InOperation	BOOL	Ermittlung des Maximalwertes abgeschlossen
	PresetAck	BOOL	Bestätigung für die Übernahme des "PresetValue"-Wertes
	MaxValue	LREAL	maximaler Wert

Tab. 4-69: Schnittstellenvariablen Funktionsbaustein IL_MaxValue

Funktionsbeschreibung

Der Funktionsbaustein IL_MaxValue vergleicht nach der Freigabe des Funktionsbausteins mittels "Enable" den aktuellen Wert der Eingangsvariable "ActValue" mit dem Wert in "MaxValue". Er kopiert den jeweils größeren Wert auf die Ausgangsvariable "MaxValue".

Wird zur Laufzeit des Funktionsbausteins der Eingang "Preset" aktiviert, wird der Wert der Eingangsvariablen "PresetValue" auf die Ausgangsvariable "MaxValue" kopiert. Der Eingang ist damit zustandsgesteuert, d.h. solange der "Preset" Eingang aktiv ist, wird der "PresetValue" auf die Ausgangsvariable "MaxValue" kopiert.



Soll mit dem Aktivieren des Bausteins auch gleichzeitig ein "PresetValue"-Wert vorgegeben werden, muss der Eingang "Preset" gleichzeitig mit dem Eingang "Enable" gesetzt werden.

Fehlercodes Der Funktionsbaustein IL_MaxValue generiert keinen Fehler.

4.11.4 IL_COORD_PAIR_TYPE01

Diese Datenstruktur definiert eine zweidimensionale Koordinate mit X und Y LREAL-Werten.

Typvereinbarung

```
TYPE IL_COORD_PAIR_TYPE01 :  
STRUCT  
  X: LREAL := 0.0;           // X coord  
  Y: LREAL := 0.0;           // Y coord
```

END_STRUCT
END_TYPE

Anwendung in

- [IL_InterpolationLinear](#), Seite 137

4.12 Zeichenfolgen ohne Längenbegrenzung (StringFun_NoLi-mitStrLength)

4.12.1 Einführung und Übersicht

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek
MLC 15VRS	RIL_Uilities
MTX 15VRS	

Tab. 4-70: Referenztablelle

Die nachfolgenden Funktionen erlauben eine Textverarbeitung / Zeichenfolgenverarbeitung ohne Längenbegrenzung.

Übergeben werden dazu z. B. die Adresse und die Länge der Zeichenfolge. Die Bearbeitung erfolgt byteweise.

Folgende grundsätzliche Festlegungen gelten:

Festlegung 1 Funktionen zur Zeichenfolge-Verarbeitung, z.B. [IL_LEFT](#), Seite 159, haben zwei Eingänge "SourceStr" und "DstStr".

"SourceStr" ist die Anfangsadresse der Quell-Zeichenfolge.

"DstStr" ist die Anfangsadresse der Ziel-Zeichenfolge (das Ergebnis).

Wenn "SourceStr" und "DstStr" identisch sind, wird die Quell-Zeichenfolge geändert und keine Kopie erzeugt. Die Funktion liefert die Adresse zurück, in der das Ergebnis gespeichert wird.

Beispiel: IL_LEFT(SourceStr, DstStr, length, MaxLen)

Str1 := 'ABCDE';

L_LEFT(ADR(Str1), ADR(Str2), 3, 5); Rückgabewert der Funktion: ADR(Str2)
Str1 = 'ABCDE'
Str2 = 'ABC'

Str1 := 'ABCDE';

IL_LEFT(ADR (Str1), ADR (Str1), 3, 5); Rückgabewert der Funktion: ADR(Str1)
Str1 = 'ABC'

Festlegung 2 Ein Zeichenfolge-Zugriff wird mittels Pointer realisiert.

Über den Eingang "MaxLen" wird die Länge der Zeichenfolge begrenzt, um den Zugriff auf ungültigen Speicher zu vermeiden.

Wenn das erlaubte Maximum der Zeichenanzahl "MaxLen" erreicht ist, wird "DstStr" mit "NUL" abgeschlossen. Die Adresse von "DstStr" wird von der Funktion zurückgegeben.

Beispiel: IL_LEFT(SourceStr, DstStr, length, MaxLen)

Str1 := 'ABCDEFGHJKLMNOP';

IL_LEFT(ADR(Str1), ADR(Str2), 10, 5); Rückgabewert der Funktion: ADR(Str2)

Str1 = 'ABCDEFGHJKLMNOP'

Str2 = 'ABCDE'

Str1 := 'ABCDEFGHJKLMNOP';

IL_LEFT(ADR (Str1), ADR (Str1), 10, 5); Rückgabewert der Funktion: ADR(Str1)

Str1 = 'ABCDE'

Anwendungsgebiete und Funktionen / Funktionsbausteine von RIL_Uilities

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine	Beschreibung
Großschreibung (Capitalization)	
IL_StrLower, Seite 144	Funktion konvertiert eine Zeichenfolge in Kleinbuchstaben
IL_StrUpper, Seite 144	Funktion konvertiert eine Zeichenfolge in Großbuchstaben
Vergleich (Compare)	
IL_StrCompare, Seite 145	Funktion vergleicht zwei Zeichenfolgen und liefert die Differenz zurück. Groß-/Kleinschreibung berücksichtigt
IL_StrCompareNoCase, Seite 146	Funktion vergleicht zwei Zeichenfolgen und gibt die Differenz zurück. Groß-/Kleinschreibung nicht berücksichtigt
IL_StrNCompare, Seite 146	Funktion vergleicht die eingegebene Anzahl Zeichen von zwei Zeichenfolgen und gibt die Differenz zurück. Groß-/Kleinschreibung berücksichtigt
IL_StrNCompareNoCase, Seite 147	Funktion vergleicht die eingegebene Anzahl Zeichen von zwei Zeichenfolgen und gibt die Differenz zurück. Groß-/Kleinschreibung nicht berücksichtigt
Konvertieren (Conversions)	
IL_AsciiStreamToBase64, Seite 148	Funktion konvertiert eine im ASCII-Format abgelegte Zeichenkette ins BASE64-Format
IL_Base64ToAsciiStream, Seite 149	Funktion konvertiert eine im BASE64-Format abgelegte Zeichenkette ins ASCII-Format
IL_BufferToString, Seite 151	Funktion kopiert eine ASCII-Zeichenkette aus einem Byte-Array
IL_StringToBuffer, Seite 153	Funktion kopiert eine ASCII-Zeichenkette in ein Byte-Array
Kopieren (Copy)	
IL_StrCopy, Seite 154	Funktion kopiert eine Zeichenfolge von der Quell- zur Zieladresse
IL_StrNCopy, Seite 155	Die Funktion kopiert einen Teil-Zeichenfolge aus einer Zeichenfolge in eine Ziel-Zeichenfolge. Die Länge des Ziel-Zeichenfolge ist festgelegt. Wenn die Quell-Zeichenfolge kürzer als die geforderte Länge ist, wird die Ziel-Zeichenfolge durch Füllzeichen ergänzt
IEC (IEC)	
IL_CONCAT, Seite 156	Funktion fügt Zeichenfolge 2 hinter Zeichenfolge 1 an

IL_DELETE, Seite 157	Funktion löscht eine Teil-Zeichenfolge aus der Zeichenfolge ab einer vorgegebenen Position
IL_FIND, Seite 158	Funktion sucht die Position der eingegebenen Teil-Zeichenfolge in einer Zeichenfolge
IL_INSERT, Seite 158	Funktion fügt eine Teil-Zeichenfolge ab der eingegebenen Position in eine Zeichenfolge ein
IL_LEFT, Seite 159	Funktion liefert das x Zeichen lange linke Teil einer Zeichenfolge
IL_LEN, Seite 160	Funktion liefert die Länge einer Zeichenfolge
IL_MID, Seite 161	Funktion liefert eine Teil-Zeichenfolge ab der eingegebenen Position aus einer Zeichenfolge
IL_REPLACE, Seite 162	Funktion ersetzt eine Teil-Zeichenfolge einer Zeichenfolge durch eine andere Teil-Zeichenfolge
IL_RIGHT, Seite 163	Funktion liefert das x Zeichen lange rechte Teil einer Zeichenfolge
Typkonvertierung (Numeric conversions)	
IL_Ascii_To_Char, Seite 164	Funktion konvertiert Ascii-Code in ein Zeichen
IL_Char_To_Ascii, Seite 164	Funktion konvertiert ein Zeichen in Ascii-Code
IL_Int_To_Str, Seite 164	Funktion konvertierte eine Ganzzahl formatabhängig in eine Zeichenfolge
IL_Real_To_Str, Seite 165	Funktion konvertierte eine REAL-Zahl formatabhängig in eine Zeichenfolge
IL_String_To_Unicode, Seite 166	Funktion konvertiert eine Zeichenfolge entsprechend vorgegebener Option (utf8 / utf16) in Unicode und speichert die Werte in einem DWORD-Array ab
IL_Unicode_To_String, Seite 167	Funktion konvertiert eine UniCode-Zeichenfolge vorgegebener Option (utf8 / utf16) in einen STRING
Andere (Other)	
IL_BufferGetElementCount, Seite 168	Funktion bestimmt die Anzahl der durch ein "Separator"-Zeichen getrennten Objekte in einer Zeichenkette
IL_Code, Seite 169	Funktion ermittelt den ASCII-Code eines vorgegebenen Zeichens in einer Zeichenkette in dezimaler Darstellung
IL_StrInverse, Seite 170	Funktion invertiert eine Zeichenfolge
IL_StrSetEndOfString, Seite 171	Funktion setzt das Ende einer Zeichenfolge
Aufteilen (Split)	
IL_StrGetElement, Seite 171	Zeichenfolge wird durch die eingegebene Trenn-Zeichenfolge in einige Elementen aufgeteilt. Die Funktion ruft eines der Elemente ab
IL_StrLChr, Seite 172	Funktion isoliert eine Teil-Zeichenfolge aus der Zeichenfolge bis zum ersten Auftreten des eingegebenen Trennzeichens
IL_StrRChr, Seite 173	Funktion isoliert eine Teil-Zeichenfolge aus der Zeichenfolge ab dem letzten Auftreten des eingegebenen Trennzeichens
IL_StrSetElement, Seite 174	Zeichenfolge wird durch die eingegebene Trenn-Zeichenfolge in Elementen aufgeteilt. Die Funktion setzt ein neues Element an die gewünschte Position in der Zeichenfolge
IL_StrSplit, Seite 175	Zeichenfolge wird durch ein vorgegebenes Trennzeichen aufgeteilt. Die Funktion legt die Adresse jeder Teil-Zeichenfolge in ein Array
Beschneiden (Trim)	

IL_StrTrimAll, Seite 176	Funktion löscht alle vorgegebene Zeichen in einer Zeichenfolge
IL_StrTrimSpace, Seite 177	Funktion löscht die Leerzeichen in einer Zeichenfolge je nach gewünschter Option

Tab. 4-71: RIL_Uutilities, StringFun_NoLimitStrLength

4.12.2 IL_StrLower

Kurzbeschreibung

Die Funktion **IL_StrLower** konvertiert alle Zeichen einer Zeichenfolge in Kleinbuchstaben.
Rückgabewert: Adresse der Ziel-Zeichenfolge.

Schnittstellenbeschreibung

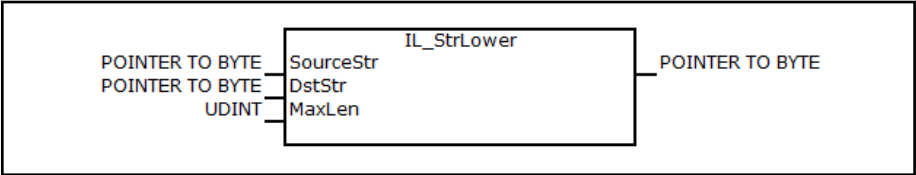


Abb. 4-84: FUN IL_StrLower

Name	Typ	Kommentar
SourceStr	POINTER TO BYTE	Adresse der Quell-Zeichenfolge
DstStr	POINTER TO BYTE	Adresse der Ziel-Zeichenfolge
MaxLen	UDINT	Max. Länge der Ziel-Zeichenfolge (in Byte/Char)

Tab. 4-72: Schnittstelle der FUN IL_StrLower

Beispiel 1

```
Str1: STRING := 'AbCDE';
Str2: STRING;
IL_StrLower(ADR(Str1), ADR(Str2), 5);
```

Rückgabewert: ADR(Str2)
Str2: 'abcde'

Beispiel 2

```
Str1: STRING := 'AbCDE';
Str2: STRING;
IL_StrLower(ADR(Str1), ADR(Str2), 2);
```

Rückgabewert: ADR(Str2)
Str2: 'ab'

Beispiel 3

```
Str1: STRING := 'AbCDE';
Str2: STRING;
IL_StrLower(ADR(Str1), ADR(Str1), 5);
```

Rückgabewert: ADR(Str1)
Str1: 'abcde'

4.12.3 IL_StrUpper

Kurzbeschreibung

Die Funktion **IL_StrUpper** konvertiert alle Zeichen einer Zeichenfolge in Großbuchstaben.
Rückgabewert: Adresse der Ziel-Zeichenfolge.

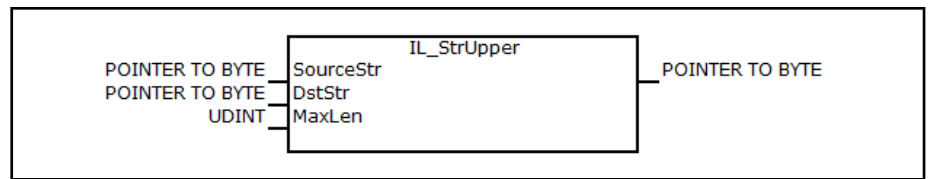
Schnittstellenbeschreibung

Abb. 4-85: FUN IL_StrUpper

Name	Typ	Kommentar
SourceStr	POINTER TO BYTE	Adresse der Quell-Zeichenfolge
DstStr	POINTER TO BYTE	Adresse der Ziel-Zeichenfolge
MaxLen	UDINT	Max. Länge der Ziel-Zeichenfolge (in Byte/Char)

Tab. 4-73: Schnittstelle der FUN IL_StrUpper

Beispiel 1

Str1: STRING := 'aBcde';

Str2: STRING;

IL_StrUpper(ADR(Str1), ADR(Str2), 5);

Rückgabewert: ADR(Str2)

Str2: 'ABCDE'

Beispiel 2

Str1: STRING := 'aBcde';

Str2: STRING;

IL_StrUpper(ADR(Str1), ADR(Str2), 2);

Rückgabewert: ADR(Str2)

Str2: 'AB'

Beispiel 3

Str1: STRING := 'aBcde';

IL_StrUpper(ADR(Str1), ADR(Str1), 5);

Rückgabewert: ADR(Str1)

Str1: 'ABCDE'

4.12.4 IL_StrCompare

Kurzbeschreibung

Die Funktion **IL_StrCompare** vergleicht zwei Zeichenfolgen und gibt die Differenz zurück.

Groß-/Kleinschreibung wird berücksichtigt. Die Differenz wird berechnet von den ersten nicht identischen Zeichen.

Falls die zwei Zeichenfolgen identisch sind, wird 0 zurückgegeben.

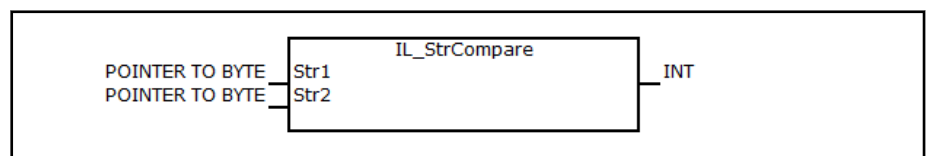
Schnittstellenbeschreibung

Abb. 4-86: FUN IL_StrCompare

Name	Typ	Kommentar
Str1	POINTER TO BYTE	Adresse der Zeichenfolge 1
Str2	POINTER TO BYTE	Adresse der Zeichenfolge 2

Tab. 4-74: Schnittstelle der FUN IL_StrCompare

Beispiel

```
Str1: STRING := 'abcde';
Str2: STRING := 'abCde';
IL_StrCompare (ADR(Str1), ADR(Str2));           Rückgabewert: 32
```

4.12.5 IL_StrCompareNoCase

Kurzbeschreibung Die Funktion **IL_StrCompare** vergleicht zwei Zeichenfolgen und gibt die Differenz zurück.

Groß-/Kleinschreibung wird **nicht** berücksichtigt. Die Differenz wird berechnet von den ersten nicht identischen Zeichen.

Falls die zwei Zeichenfolgen identisch sind, wird 0 zurückgegeben.

Schnittstellenbeschreibung

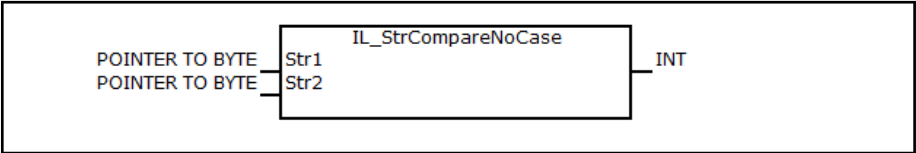


Abb. 4-87: FUN IL_StrCompareNoCase

Name	Typ	Kommentar
Str1	POINTER TO BYTE	Adresse der Zeichenfolge 1
Str2	POINTER TO BYTE	Adresse der Zeichenfolge 2

Tab. 4-75: Schnittstelle der FUN IL_StrCompareNoCase

Beispiel

```
Str1: STRING := 'abcde';
Str2: STRING := 'ABEDE'
IL_StrCompareNoCase (ADR(Str1), ADR(Str2));     Rückgabewert: -2
```

4.12.6 IL_StrNCompare

Kurzbeschreibung Die Funktion **IL_StrNCompare** vergleicht die Teil-Zeichenfolgen von zwei Zeichenfolgen und gibt die Differenz zurück.

Groß-/Kleinschreibung wird berücksichtigt.

Die Differenz wird berechnet von den ersten nicht identischen Zeichen.

Falls die zwei Zeichenfolgen identisch sind, wird 0 zurückgegeben.

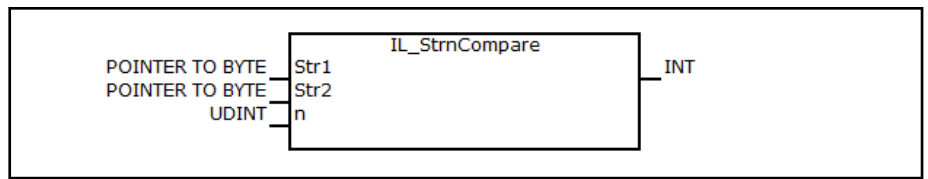
Schnittstellenbeschreibung

Abb. 4-88: FUN IL_StrNCompare

Name	Typ	Kommentar
Str1	POINTER TO BYTE	Adresse Zeichenfolge 1
Str2	POINTER TO BYTE	Adresse Zeichenfolge 2
n	UDINT	Anzahl der zu vergleichenden Zeichen

Tab. 4-76: Schnittstelle der FUN IL_StrNCompare

Beispiel 1

Str1: STRING := 'abcde';

Str2: STRING := 'abCde'

IL_StrNCompare (ADR(Str1), ADR(Str2), 3);

Rückgabewert: 32

Beispiel 2

Str1: STRING := 'abcde';

Str2: STRING := 'abCde';

IL_StrNCompare (ADR(Str1), ADR(Str2), 2);

Rückgabewert: 0

4.12.7 IL_StrNCompareNoCase**Kurzbeschreibung**

Die Funktion **IL_StrNCompareNoCase** vergleicht die Teil-Zeichenfolgen von zwei Zeichenfolgen und gibt die Differenz zurück.

Groß-/Kleinschreibung wird **nicht** berücksichtigt.

Die Differenz wird berechnet von den ersten nicht identischen Zeichen.

Falls die zwei Zeichenfolgen identisch sind, wird 0 zurückgegeben.

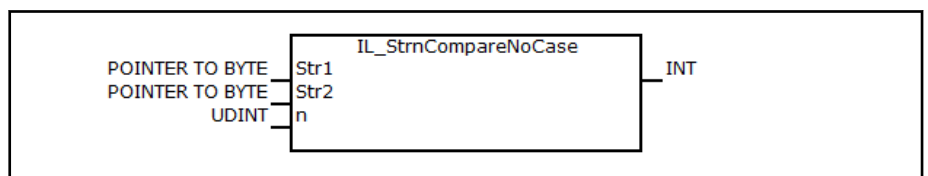
Schnittstellenbeschreibung

Abb. 4-89: FUN IL_StrNCompareNoCase

Name	Typ	Kommentar
Str1	POINTER TO BYTE	Adresse Zeichenfolge 1
Str2	POINTER TO BYTE	Adresse Zeichenfolge 2
n	UDINT	Anzahl der zu vergleichenden Zeichen

Tab. 4-77: Schnittstelle der FUN IL_StrNCompareNoCase

Beispiel 1

```
Str1: STRING := 'abcde';
Str2: STRING := 'abEde';
IL_StrNCompareNoCase (ADR(Str1), ADR(Str2), 3);      Rückgabewert: -2
```

Beispiel 2

```
Str1: STRING := 'abcde';
Str2: STRING := 'abCde';
IL_StrNCompareNoCase (ADR(Str1), ADR(Str2), 3);      Rückgabewert: 0
```

4.12.8 IL_AsciiStreamToBase64

Kurzbeschreibung Die Funktion **IL_AsciiStreamToBase64** wird verwendet, um eine im ASCII-Format abgelegte Zeichenkette ins **BASE64**-Format zu konvertieren.

Zuweisung: Zielsystem/Bibliothek

Zielsystem	Bibliothek
MLC 15VRS	RIL_Uutilities

Tab. 4-78: Referenztabelle des IL_AsciiStreamToBase64

Schnittstellenbeschreibung

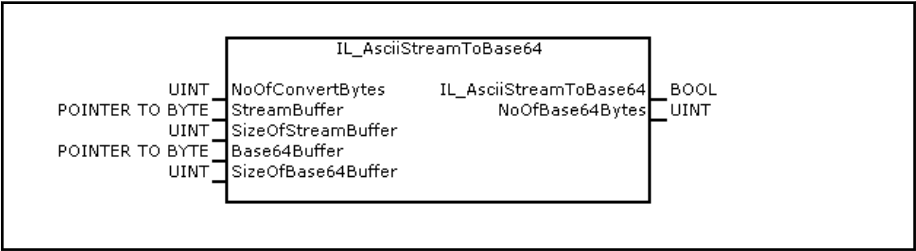


Abb. 4-90: Funktion IL_AsciiStreamToBase64

	Name	Typ	Kommentar
VAR_INPUT	NoOfConvertBytes	UINT	Anzahl der zu konvertierenden Bytes im "StreamBuffer"
	StreamBuffer	POINTER TO BYTE	Zeiger auf zu konvertierende ASCII-Zeichenkette Verwenden Sie den ADR ()-Operator
	SizeOfStreamBuffer	UINT	(Maximale) Größe des Eingangs-"StreamBuffer" Verwenden Sie den SIZEOF ()-Operator.
	Base64Buffer	POINTER TO BYTE	Zeiger auf Puffer der BASE64-konvertierten Daten Verwenden Sie den ADR ()-Operator
	SizeOfBase64Buffer	UINT	(Maximale) Größe des Ausgangs-BASE64-Puffers Verwenden Sie den SIZEOF ()-Operator.
VAR_OUTPUT	NoOfBase64Bytes	UINT	Anzahl an Bytes im Ausgangs-BASE64-Puffer als konvertiertes Ergebnis
RETURN		BOOL	Abarbeitung ausgeführt.

Tab. 4-79: Funktion IL_AsciiStreamToBase64

Funktionsbeschreibung Die Funktion **IL_AsciiStreamToBase64** wird verwendet, um eine im ASCII-Format abgelegte Zeichenkette ins **BASE64**-Format zu konvertieren.
Die Quelldaten werden im "StreamBuffer" übergeben und die konvertierten Daten im "Base64Buffer" gespeichert.

"NoOfConvertBytes" enthält die Anzahl der zu konvertierenden Bytes, die im "StreamBuffer" bereitgestellt sind.

Die (maximale) Größe des "StreamBuffer" ist in "SizeOfStreamBuffer" vorgegeben.

Nach der Konvertierung steht das Ergebnis im "Base64Buffer" zur Verfügung.

Der Benutzer muss die weitere Bearbeitung der "Base64Buffer"-Daten sicherstellen, bevor der nächste Datensatz konvertiert wird.

Die (maximale) Größe des "Base64Buffer" ist in "SizeOfBase64Buffer" vorgegeben.

Der Ausgang "NoOfBase64Bytes" stellt die Anzahl der konvertierten und gespeicherten Bytes im "Base64Buffer" bereit.

Der Rückgabewert "IL_AsciiStreamToBase64" vom Typ BOOL enthält die Information zum Erfolg des Funktionsaufrufes:

TRUE wird ausgegeben, wenn die Konvertierung erfolgreich war.

FALSE wird ausgegeben, wenn die Daten nicht vollständig konvertiert werden konnten oder "SizeOfStreamBuffer" und "SizeOfBase64Buffer" nicht zueinander passen.

Nur kompatible Daten werden konvertiert.

Beispiele

Die Funktion **IL_AsciiStreamToBase64** wird verwendet, um eine ASCII-Zeichenkette ins BASE64-Format zu konvertieren.

Der Rückgabewert vom Typ BOOL gibt Informationen zum Erfolg der Operation.

```
VAR
  bIL_AsciiStreamToBase64: BOOL;
  uiNoOfConvertBytes: UINT := UINT#3;
  arStreamBuffer: STRING(96) := 'Man';
  uiNoOfBase64Bytes : UINT;
  arBase64Buffer : STRING(128);
END_VAR

bIL_AsciiStreamToBase64 := IL_AsciiStreamToBase64(
  NoOfConvertBytes:= uiNoOfConvertBytes,
  StreamBuffer:= ADR(arStreamBuffer),
  SizeOfStreamBuffer:= SIZEOF(arStreamBuffer),
  Base64Buffer:= ADR(arBase64Buffer),
  SizeOfBase64Buffer:= SIZEOF(arBase64Buffer),
  NoOfBase64Bytes=> uiNoOfBase64Bytes);
```

Das Ergebnis des Beispiels "IL_AsciiStreamToBase64" wird nachfolgend gezeigt:

bIL_AsciiStreamToBase64	BOOL	TRUE
uiNoOfConvertBytes	UINT	3
arStreamBuffer	STRING(96)	'Man'
uiNoOfBase64Bytes	UINT	4
arBase64Buffer	STRING(128)	'TWFu'

Abb. 4-91: Beispiel "IL_AsciiStreamToBase64"

4.12.9 IL_Base64ToAsciiStream

Kurzbeschreibung

Die Funktion **IL_Base64ToAsciiStream** wird verwendet, um eine im **BASE64**-Format abgelegte Zeichenkette ins ASCII-Format zu konvertieren.

Zuweisung: Zielsystem/Bibliothek

Zielsystem	Bibliothek
MLC 15VRS	RIL_Utilities

Tab. 4-80: Referenztable des IL_Base64ToAsciiStream

Schnittstellenbeschreibung

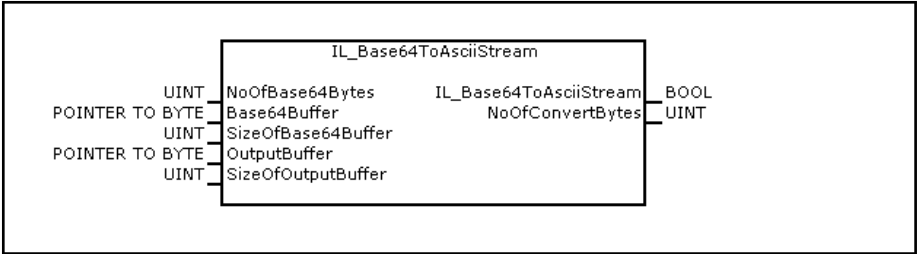


Abb. 4-92: Funktion IL_Base64ToAsciiStream

	Name	Typ	Kommentar
VAR_INPUT	NoOfBase64Bytes	UINT	Anzahl der zu konvertierenden Bytes im Base64Buffer
	Base64Buffer	POINTER TO BYTE	Zeiger auf zu konvertierende BASE64-Zeichenkette Verwenden Sie den ADR ()-Operator.
	SizeOfBase64Buffer	UINT	(Maximale) Größe des Eingangs-"Base64Buffer" Verwenden Sie den SIZEOF ()-Operator
	OutputBuffer	POINTER TO BYTE	Zeiger auf Puffer der ASCII-konvertierten Daten Verwenden Sie den ADR ()-Operator.
	SizeOfOutputBuffer	UINT	Größe des OutputBuffer Verwenden Sie den SIZEOF ()-Operator
VAR_OUTPUT	NoOfConvertBytes	UINT	Anzahl an Bytes im Ausgangs-OutputBuffer als konvertiertes Ergebnis
RETURN		BOOL	Abarbeitung ausgeführt.

Tab. 4-81: Funktion IL_Base64ToAsciiStream

Funktionsbeschreibung

Die Funktion **IL_Base64ToAsciiStream** wird verwendet um eine im **BASE64**-Format abgelegte Zeichenkette ins ASCII-Format zu konvertieren.

Die Quelldaten werden im "Base64Buffer" übergeben und die konvertierten Daten im "OutputBuffer" gespeichert.

"NoOfBase64Bytes" enthält die Anzahl der zu konvertierenden Bytes, die im "Base64Buffer" bereitgestellt sind.

Die (maximale) Größe des "Base64Buffer" ist in "SizeOfBase64Buffer" vorgegeben.

Nach der Konvertierung steht das Ergebnis im "OutputBuffer" zur Verfügung. Der Benutzer muss die weitere Bearbeitung der "OutputBuffer"-Daten sicherstellen, bevor der nächste Datensatz konvertiert wird.

Die (maximale) Größe des "OutputBuffer" ist in "SizeOfOutputBuffer" vorgegeben.

Der Ausgang "NoOfConvertBytes" stellt die Anzahl der konvertierten und gespeicherten Bytes im "OutputBuffer" bereit.

Der Rückgabewert "IL_Base64ToAsciiStream" vom Typ BOOL enthält die Information zum Erfolg des Funktionsaufrufes:

TRUE wird ausgegeben, wenn die Konvertierung erfolgreich war.

FALSE wird ausgegeben, wenn die Daten nicht vollständig konvertiert werden konnten oder "SizeOfBase64Buffer" und "SizeOfOutputBuffer" nicht zueinander passen.

Nur kompatible Daten werden konvertiert.

Beispiele

Die Funktion **IL_Base64ToAsciiStream** wird verwendet, um eine im **BASE64**-Format abgelegte Zeichenkette ins ASCII-Format zu konvertieren.

Der Rückgabewert vom Typ BOOL gibt Informationen zum Erfolg der Operation.

```
VAR
  bIL_Base64ToAsciiStream: BOOL;
  uiNoOfBase64Bytes : UINT :=4;
  arBase64Buffer : STRING(128) :='TWFu';
  uiNoOfConvertBytes: UINT;
  arOutputBuffer: STRING(96);
END_VAR

bIL_Base64ToAsciiStream := IL_Base64ToAsciiStream(
  NoOfBase64Bytes:= uiNoOfBase64Bytes,
  Base64Buffer:= ADR(arBase64Buffer),
  SizeOfBase64Buffer:= SIZEOF(arBase64Buffer),
  OutputBuffer:= ADR(arOutputBuffer),
  SizeOfOutputBuffer:= SIZEOF(arOutputBuffer),
  NoOfConvertBytes=> uiNoOfConvertBytes);
```

Das Ergebnis des Beispiels "IL_Base64ToAsciiStream" wird nachfolgend gezeigt:

◆ bIL_Base64ToAsciiStream	BOOL	TRUE
◆ uiNoOfBase64Bytes	UINT	4
◆ arBase64Buffer	STRING(128)	'TWFu'
◆ uiNoOfConvertBytes	UINT	3
◆ arOutputBuffer	STRING(96)	'Man'

Abb. 4-93: Beispiel "IL_Base64ToAsciiStream"

4.12.10 IL_BufferToString

Kurzbeschreibung

Die Funktion **IL_BufferToString** kopiert eine ASCII-Zeichenkette aus einem Byte-Array.

Zuweisung: Zielsystem/Bibliothek

Zielsystem	Bibliothek
MLC 15VRS	RIL_Uilities

Tab. 4-82: Referenztable des IL_BufferToString

Schnittstellenbeschreibung

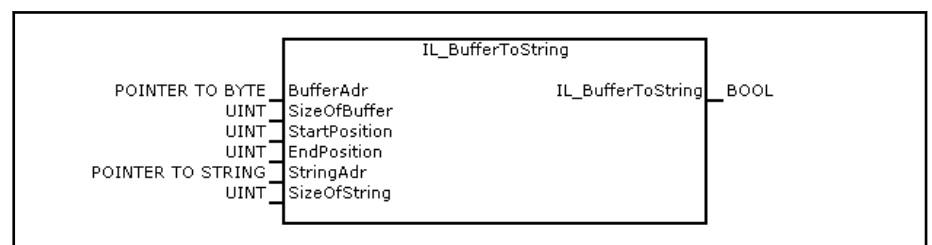


Abb. 4-94: IL_BufferToString

	Name	Typ	Kommentar
VAR_INPUT	BufferAdr	POINTER TO BYTE	(Anfangs-)Adresse des Byte-Array-Puffers Verwenden Sie den ADR ()-Operator
	SizeOfBuffer	UINT	Größe des Byte-Array-Puffer Verwenden Sie den SIZEOF ()-Operator
	StartPosition	UINT	Startposition der Zeichenkette im Byte-Array-Puffer
	EndPosition	UINT	Endposition der Zeichenkette im Byte-Array-Puffer
	StringAdr	POINTER TO STRING	Zeiger auf kopierte Zeichenkette Verwenden Sie den ADR()-Operator
	SizeOfString	UINT	Maximal zulässige Größe der Zeichenkette Verwenden Sie den SIZEOF ()-Operator SizeOF(StringAdr)
RETURN		BOOL	Konvertierung beendet

Tab. 4-83: Funktion IL_BufferToString

Funktionsbeschreibung

Die Funktion **IL_BufferToString** kopiert eine ASCII-Zeichenkette aus einem Byte-Array.

"BufferAdr" enthält die (Anfangs-)Adresse des bereitgestellten Byte-Arrays.

"StartPosition" enthält die Position, ab der die Zeichenkette innerhalb des Arrays beginnt.

"EndPosition" enthält die Position des letzten zu lesenden Zeichens im Array.

Die Funktion fragt also eine Zeichenkette aus einem Byte-Array, beginnend an der "StartPosition" und endend an der "EndPosition", ab.

Die ausgelesene Zeichenkette wird unter "StringAdr" gespeichert.

Die maximale Länge der auslesbaren Zeichenkette wird unter "SizeOfString" vorgegeben.

Der Rückgabewert der Funktion "IL_BufferToString" vom Typ BOOL bestätigt die Konvertierung.

TRUE wird ausgegeben, wenn das Kopieren erfolgreich war.

FALSE wird ausgegeben, wenn das Kopieren nicht erfolgreich war, z. B. wenn "SizeOfBuffer" und "SizeOfString" nicht zueinander passen.

Nur kompatible Daten werden kopiert.

Beispiele

Die Funktion **IL_BufferToString** kopiert eine ASCII-Zeichenkette aus einem Byte-Array.

```

VAR
  bIL_BufferToString : BOOL;
  strIL_BufferToString: STRING(300);
  arBufferAdr: ARRAY [0..300] OF BYTE:= [63,64,65,66,67,68,69,70,71,72,73];
END_VAR

bIL_BufferToString := IL_BufferToString(BufferAdr:= ADR(arBufferAdr),
  SizeOfBuffer:=SIZEOF(arBufferAdr), StartPosition:=2, EndPosition:=8,
  StringAdr:= ADR(strIL_BufferToString),
  SizeOfString:= SIZEOF(strIL_BufferToString));

```

Das Ergebnis des Beispiels "IL_BufferToString" wird nachfolgend gezeigt:

bIL_BufferToString	BOOL	TRUE
strIL_BufferToString	STRING(300)	'ABCDEFG'
arBufferAdr	ARRAY [0..300] OF BYTE	
arBufferAdr[0]	BYTE	63
arBufferAdr[1]	BYTE	64
arBufferAdr[2]	BYTE	65
arBufferAdr[3]	BYTE	66
arBufferAdr[4]	BYTE	67
arBufferAdr[5]	BYTE	68
arBufferAdr[6]	BYTE	69
arBufferAdr[7]	BYTE	70
arBufferAdr[8]	BYTE	71
arBufferAdr[9]	BYTE	72
arBufferAdr[10]	BYTE	73
arBufferAdr[11]	BYTE	0

Abb. 4-95: Beispiel "IL_BufferToString"

4.12.11 IL_StringToBuffer

Kurzbeschreibung

Die Funktion **IL_StringToBuffer** kopiert eine ASCII-Zeichenkette in ein Byte-Array.

Zuweisung: Zielsystem/Bibliothek

Zielsystem	Bibliothek
MLC 15VRS	RIL_Utilities

Tab. 4-84: Referenztable des IL_StringToBuffer

Schnittstellenbeschreibung

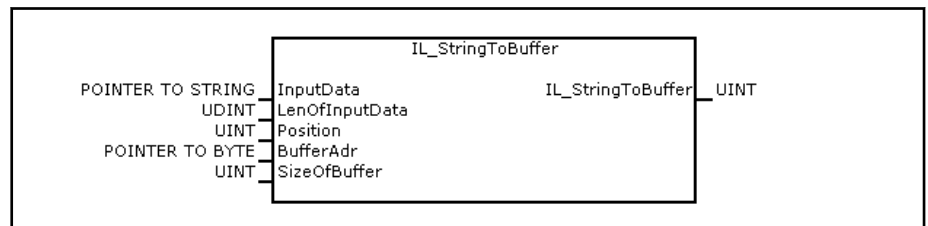


Abb. 4-96: Funktion IL_StringToBuffer

	Name	Typ	Kommentar
VAR_INPUT	InputData	POINTER TO STRING	Zeiger auf die zu konvertierende Zeichenkette Verwenden Sie den ADR ()-Operator
	LenOfInputData	UDINT	Länge der Eingangszeichenkette Verwenden Sie IL_Len (InputData)
	Position	UINT	Startposition der Zeichenkette im Byte-Array-Puffer
	BufferAdr	POINTER TO BYTE	Adresse des Byte-Array-Puffers Verwenden Sie den ADR ()-Operator
	SizeOfBuffer	UINT	Größe des Byte-Array-Puffers. Verwenden Sie den SIZEOF ()-Operator
RETURN		UINT	Endposition der Zeichenkette im Byte-Array-Puffer

Tab. 4-85: Funktion IL_StringToBuffer

Funktionsbeschreibung

Die Funktion **IL_StringToBuffer** kopiert eine ASCII-Zeichenkette in ein Byte-Array.

"InputData" enthält den Zeiger auf die zu kopierende Zeichenkette.

Die Anzahl der Zeichen ist in "LenOfInputData" vorgegeben. Sie ist mit Hilfe der "IL_Len"-Funktion zu ermitteln.

Die Funktion ruft also die Zeichenkette ab und legt sie im "BufferAdr"-Array ab.
"Position" ist die Startposition innerhalb des "BufferAdr"-Arrays, ab der die Zeichenkette abgelegt werden soll.
Das erste Zeichen im "BufferAdr"-Array hat die Positionsnummer 0.

Der Rückgabewert "IL_StringToBuffer" vom Typ UINT gibt die Position des letzten kopierten Zeichens zurück.
Im Fehlerfall, wenn z. B. "SizeOfBuffer" und "LenOfInputData" nicht zueinander passen, wird das Kopieren abgebrochen.
Nur kompatible Daten werden kopiert.

Beispiele Die Funktion **IL_StringToBuffer** kopiert eine ASCII-Zeichenkette in ein Byte-Array.

```
VAR
  strInputData: STRING(300) :='ABCDEFGH';
  uiIL_StringToBuffer: UINT;
  arBufferAdr: ARRAY [0..300] OF BYTE;
END_VAR

uiIL_StringToBuffer := IL_StringToBuffer(InputData:=ADR(strInputData),
  LenOfInputData := IL_LEN(ADR(strInputData)), Position:=4,
  BufferAdr:=ADR(arBufferAdr) , SizeOfBuffer:= SIZEOF(arBufferAdr));
```

Das Ergebnis des Beispiels "IL_StringToBuffer" wird nachfolgend gezeigt:

strInputData	STRING(300)	'ABCDEFGH'
uiIL_StringToBuffer	UINT	11
arBufferAdr	ARRAY [0..300]...	
arBufferAdr[0]	BYTE	0
arBufferAdr[1]	BYTE	0
arBufferAdr[2]	BYTE	0
arBufferAdr[3]	BYTE	0
arBufferAdr[4]	BYTE	65
arBufferAdr[5]	BYTE	66
arBufferAdr[6]	BYTE	67
arBufferAdr[7]	BYTE	68
arBufferAdr[8]	BYTE	69
arBufferAdr[9]	BYTE	70
arBufferAdr[10]	BYTE	71
arBufferAdr[11]	BYTE	72
arBufferAdr[12]	BYTE	0
arBufferAdr[13]	BYTE	0
arBufferAdr[14]	BYTE	0

Abb. 4-97: Beispiel "IL_StringToBuffer"

4.12.12 IL_StrCopy

Kurzbeschreibung

Die Funktion **IL_StrCopy** kopiert eine Zeichenfolge von der Quell- zur Zieladresse.
Rückgabewert: Adresse der Ziel-Zeichenfolge.

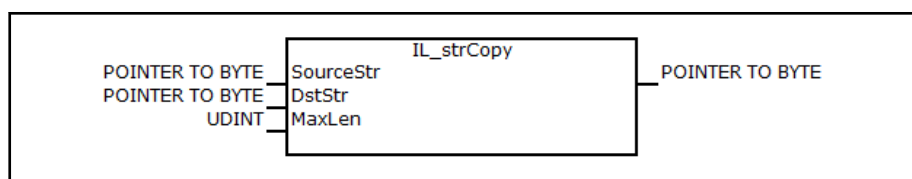
Schnittstellenbeschreibung

Abb. 4-98: FUN IL_StrCopy

Name	Typ	Kommentar
SourceStr	POINTER TO BYTE	Adresse der Quell-Zeichenfolge
DstStr	POINTER TO BYTE	Adresse der Ziel-Zeichenfolge
MaxLen	UDINT	Max. Länge der Ziel-Zeichenfolge (in Byte/Char)

Tab. 4-86: Schnittstelle der FUN IL_StrCopy

Beispiel

Str1: STRING := 'abcde';

Str2: STRING;

IL_StrCopy (ADR(Str1), ADR(Str2), 5);

Rückgabewert: ADR(Str2)

Str2: 'abcde'

4.12.13 IL_StrNCopy**Kurzbeschreibung**

Die Funktion **IL_StrNCopy** kopiert eine Teil-Zeichenfolge aus einer Zeichenfolge zur Zieladresse.

Die Länge der Ziel-Zeichenfolge ist mit "MaxLen" festgelegt.

Wenn die Quell-Zeichenfolge kürzer als die eingegebene Länge ist, wird die Ziel-Zeichenfolge mit Füllzeichen ergänzt.

Rückgabewert: Adresse der Ziel-Zeichenfolge.

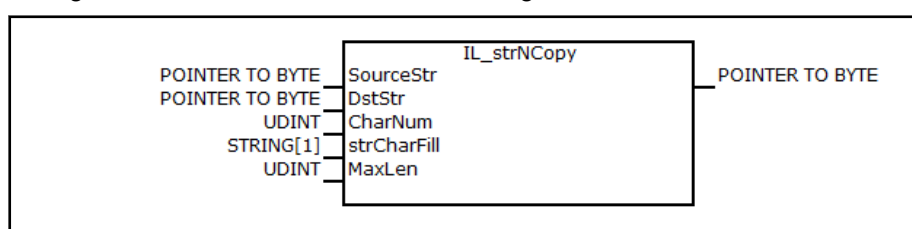
Schnittstellenbeschreibung

Abb. 4-99: FUN IL_StrNCopy

Name	Typ	Kommentar
SourceStr	POINTER TO BYTE	Adresse der Quell-Zeichenfolge
DstStr	POINTER TO BYTE	Adresse der Ziel-Zeichenfolge
CharNum	UDINT	Länge der Ziel-Zeichenfolge (in Byte/Char)
strCharFill	STRING[1]	Füllzeichen
MaxLen	UDINT	Max. Länge der Ziel-Zeichenfolge (in Byte/Char)

Tab. 4-87: Schnittstelle der FUN IL_StrNCopy

Beispiel 1

Str1: STRING := 'abcde';
Str2: STRING;
IL_StrNCopy (ADR(Str1), ADR(Str2), 7, 'F' , 10);

Rückgabewert: ADR(Str2)
Str2: 'abcdeFF'

Beispiel 2

Str1: STRING := abcde;
Str2: STRING;
IL_StrNCopy (ADR(Str1), ADR(Str2), 7, 'F', 6);

Rückgabewert: ADR(Str2)
Str2: 'abcdeF'

Beispiel 3

Str1: STRING := abcde;
Str2: STRING;
IL_StrNCopy (ADR(Str1), ADR(Str2), 3, 'F',10);

Rückgabewert: ADR(Str2)
Str2: 'abc'

4.12.14 IL_CONCAT

Kurzbeschreibung Die Funktion **IL_CONCAT** fügt Zeichenfolge 2 hinter Zeichenfolge 1 an.



Das Ergebnis kann **nicht** unter der Adresse von Zeichenfolge 2 gespeichert werden.

Rückgabewert: Adresse der Ziel-Zeichenfolge.

Schnittstellenbeschreibung

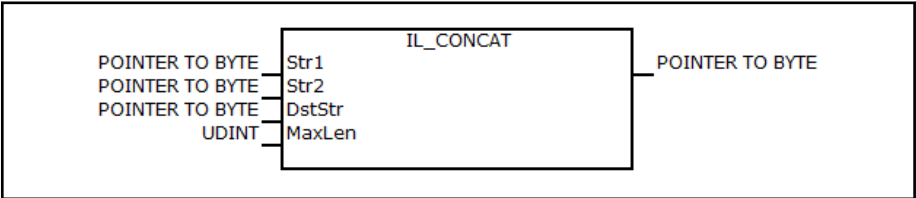


Abb. 4-100: FUN IL_CONCAT

Name	Typ	Kommentar
Str1	POINTER TO BYTE	Adresse der Zeichenfolge 1
Str2	POINTER TO BYTE	Adresse der Zeichenfolge 2
DstStr	POINTER TO BYTE	Adresse der Ziel-Zeichenfolge
MaxLen	UDINT	Max. Länge der Ziel-Zeichenfolge (in Byte/Char)

Tab. 4-88: Schnittstelle der FUN IL_CONCAT

Beispiel 1

```

Str1: STRING := 'ABCD';
Str2: STRING := 'EFGH';
Str3: STRING;
IL_CONCAT(ADR(Str1), ADR(Str2), ADR(Str3), 10);      Rückgabewert: ADR(Str3)
Str3: 'ABCDEFGH'

```

Beispiel 2

```

Str1: STRING := 'ABCD';
Str2: STRING := 'EFGH';
Str3: STRING;
IL_CONCAT(ADR(Str1), ADR(Str2), ADR(Str3), 2);      Rückgabewert: ADR(Str3)
Str3: 'AB'

```

Beispiel 3

```

Str1: STRING := 'ABCD';
Str2: STRING := 'EFGH';
IL_CONCAT(ADR(Str1), ADR(Str2), ADR(Str1), 10);      Rückgabewert: ADR(Str1)
Str1: 'ABCDEFGH'

```

4.12.15 IL_DELETE**Kurzbeschreibung**

Die Funktion **IL_DELETE** löscht eine Teil-Zeichenfolge aus einer Zeichenfolge ab einer vorgegebenen Position.

Das von links gezählte erste Zeichen der Zeichenfolge hat die Position 1.

Rückgabewert: Adresse der Ziel-Zeichenfolge.

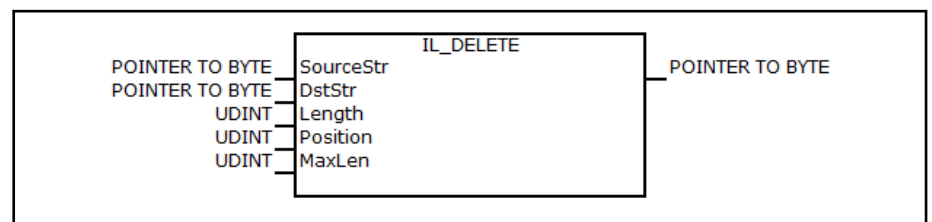
Schnittstellenbeschreibung

Abb. 4-101: FUN IL_DELETE

Name	Typ	Kommentar
SourceStr	POINTER TO BYTE	Adresse der Quell-Zeichenfolge
DstStr	POINTER TO BYTE	Adresse der Ziel-Zeichenfolge
Length	UDINT	Länge der zu entfernenden Teil-Zeichenfolge (in Byte/Char)
Position	UDINT	Die Position der Quell-Zeichenfolge, ab der die Teil-Zeichenfolge entfernt werden soll
MaxLen	UDINT	Max. Länge der Ziel-Zeichenfolge (in Byte/Char)

Tab. 4-89: Schnittstelle der FUN IL_DELETE

Beispiel 1

```
Str1: STRING := 'ABCDEFGH';
Str2: STRING;
IL_DELETE (ADR(Str1), ADR(Str2), 4, 3, 5);           Rückgabewert: ADR(Str2)
                                                    Str2: 'ABGH'
```

Beispiel 2

```
Str1: STRING := 'ABCDEFGH';
Str2: STRING;
IL_DELETE (ADR(Str1), ADR(Str2), 4, 3, 2);           Rückgabewert: ADR(Str2)
                                                    Str2: 'AB'
```

Beispiel 3

```
Str1: STRING := 'ABCDEFGH';
IL_DELETE (ADR(Str1), ADR(Str1), 4, 3, 5);           Rückgabewert: ADR(Str1)
                                                    Str1: 'ABGH'
```

4.12.16 IL_FIND

Kurzbeschreibung Die Funktion **IL_FIND** sucht die Anfangsposition der angegebenen Teil-Zeichenfolge in einer Zeichenfolge. Das von links gezählt erste Zeichen der Zeichenfolge hat die Position 1.

Die Position wird als Rückgabe der Funktion geliefert.

Schnittstellenbeschreibung

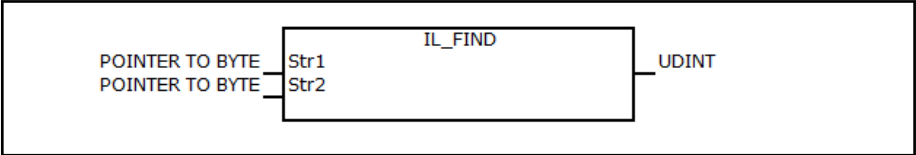


Abb. 4-102: FUN IL_FIND

Name	Typ	Kommentar
Str1	POINTER TO BYTE	Adresse der Quell-Zeichenfolge
Str2	POINTER TO BYTE	Adresse der zu suchenden Teil-Zeichenfolge

Tab. 4-90: Schnittstelle der FUN IL_FIND

Beispiel

```
Str1: STRING := 'ABCDEFGH';
Str2: STRING := 'CDEF';
IL_FIND (ADR(Str1), ADR(Str2));                     Rückgabewert: 3
```

4.12.17 IL_INSERT

Kurzbeschreibung Die Funktion **IL_INSERT** fügt eine Teil-Zeichenfolge ab einer vorgegebenen Position in die Zeichenfolge ein.

Das von links gezählt erste Zeichen der Zeichenfolge hat die Position 1.

Rückgabewert: Adresse des Ziel-Zeichenfolge.

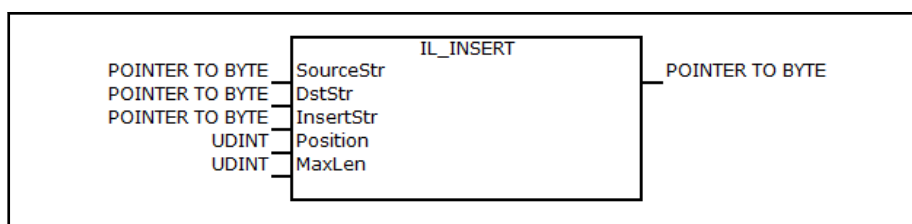
Schnittstellenbeschreibung

Abb. 4-103: FUN IL_INSERT

Name	Typ	Kommentar
SourceStr	POINTER TO BYTE	Adresse der Quell-Zeichenfolge
DstStr	POINTER TO BYTE	Adresse der Ziel-Zeichenfolge
InsertStr	POINTER TO BYTE	Adresse der einzufügenden Teil-Zeichenfolge
Position	UDINT	Die Position der Quell-Zeichenfolge, ab der die Teil-Zeichenfolge eingefügt werden soll
MaxLen	UDINT	Max. Länge der Ziel-Zeichenfolge (in Byte/Char)

Tab. 4-91: Schnittstelle der FUN IL_INSERT

Beispiel 1

Str1: STRING := 'ABCD';

Str2: STRING;

Str3: STRING := 'EFGH';

IL_INSERT(ADR(Str1), ADR(Str2), ADR(Str3), 3, 10);

Rückgabewert: ADR(Str2)

Str2: 'ABEFGHCD'

Beispiel 2

Str1: STRING := 'ABCD';

Str2: STRING;

Str3: STRING := 'EFGH';

IL_INSERT(ADR(Str1), ADR(Str2), ADR(Str3), 3, 2);

Rückgabewert: ADR(Str2)

Str2: 'AB'

Beispiel 3

Str1: STRING := 'ABCD';

Str3: STRING := 'EFGH';

IL_INSERT(ADR(Str1), ADR(Str1), ADR(Str3), 3, 10);

Rückgabewert: ADR(Str1)

Str1: 'ABEFGHCD'

4.12.18 IL_LEFT**Kurzbeschreibung**

Die Funktion **IL_LEFT** liefert die linke Teil-Zeichenfolge einer Zeichenfolge.
 Rückgabewert: Adresse der Ziel-Zeichenfolge.

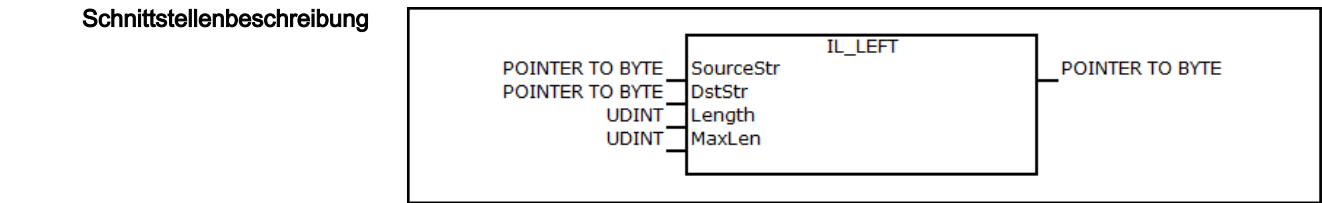


Abb. 4-104: IL_LEFT

Name	Typ	Kommentar
SourceStr	POINTER TO BYTE	Adresse der Quell-Zeichenfolge
DstStr	POINTER TO BYTE	Adresse der Ziel-Zeichenfolge
Length	UDINT	Länge der linken Teil-Zeichenfolge (in Byte/Char)
MaxLen	UDINT	Max. Länge der Ziel-Zeichenfolge (in Byte/Char)

Tab. 4-92: Schnittstelle der FUN IL_LEFT

Beispiel 1

Str1: STRING := 'ABCD';
Str2: STRING;
IL_LEFT(ADR(Str1), ADR(Str2), 3, 5);

Rückgabewert: ADR(Str2)
Str2: 'ABC'

Beispiel 2

Str1: STRING := 'ABCD';
Str2: STRING;
IL_LEFT(ADR(Str1), ADR(Str2), 3, 2);

Rückgabewert: ADR(Str2)
Str2: 'AB'

Beispiel 3

Str1: STRING := 'ABCD';
IL_LEFT(ADR(Str1), ADR(Str1), 3, 5);

Rückgabewert: ADR(Str1)
Str1: 'ABC'

4.12.19 IL_LEN

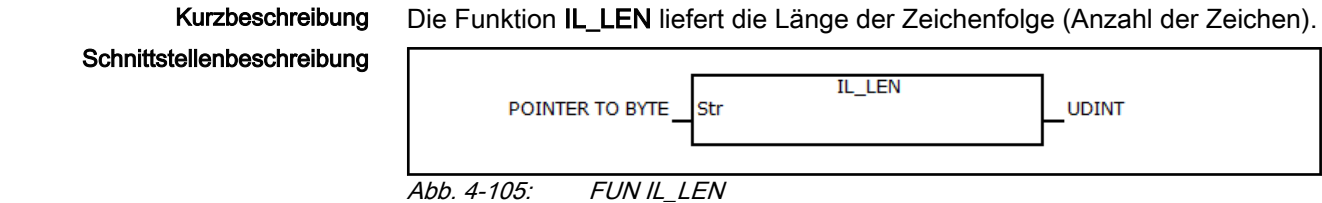


Abb. 4-105: FUN IL_LEN

Name	Typ	Kommentar
Str	POINTER TO BYTE	Adresse der Zeichenfolge

Tab. 4-93: Schnittstelle der FUN IL_LEN

Beispiel

Str1: STRING := 'ABCDEFGH';

IL_LEN(ADR(Str1));

Rückgabewert: 8

4.12.20 IL_MID**Kurzbeschreibung**

Die Funktion **IL_MID** liefert eine Teil-Zeichenfolge vorgegebener Länge ab vorgegebener Position aus einer Zeichenfolge.

Die Position entspricht der Zeichennummer. Das von links gezählte erste Zeichen der Zeichenfolge hat Position 1.

Rückgabewert: Adresse der Ziel-Zeichenfolge.

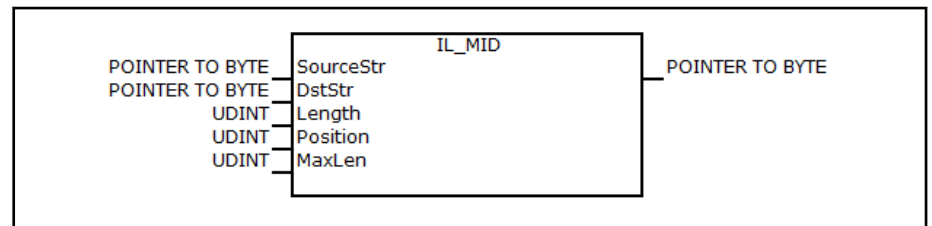
Schnittstellenbeschreibung

Abb. 4-106: FUN IL_MID

Name	Typ	Kommentar
SourceStr	POINTER TO BYTE	Adresse der Quell-Zeichenfolge
DstStr	POINTER TO BYTE	Adresse der Ziel-Zeichenfolge
Length	UDINT	Länge der Teil-Zeichenfolge (in Byte/Char)
Position	UDINT	Position der Quell-Zeichenfolge, ab der die Teil-Zeichenfolge abgerufen werden soll
MaxLen	UDINT	Max. Länge der Ziel-Zeichenfolge (in Byte/Char)

Tab. 4-94: Schnittstelle der FUN IL_MID

Beispiel 1

Str1: STRING := 'ABCDEFGH';

Str2: STRING;

IL_MID(ADR(Str1), ADR(Str2), 4, 3, 10);

Rückgabewert: ADR(Str2)

Str2: 'CDEF'

Beispiel 2

Str1: STRING := 'ABCDEFGH';

Str2: STRING;

IL_MID(ADR(Str1), ADR(Str2), 4, 3, 2);

Rückgabewert: ADR(Str2)

Str2: 'CD'

Beispiel 3

```
Str1: STRING := 'ABCDEFGH';  
IL_MID(ADR(Str1), ADR(Str1), 4, 3, 10);  
Rückgabewert: ADR(Str1)  
Str1: 'CDEF'
```

4.12.21 IL_REPLACE

Kurzbeschreibung Die Funktion **IL_REPLACE** ersetzt eine Teil-Zeichenfolge einer Zeichenfolge durch eine Ersatz-Zeichenfolge.
Rückgabewert: Adresse der Ziel-Zeichenfolge.

Schnittstellenbeschreibung

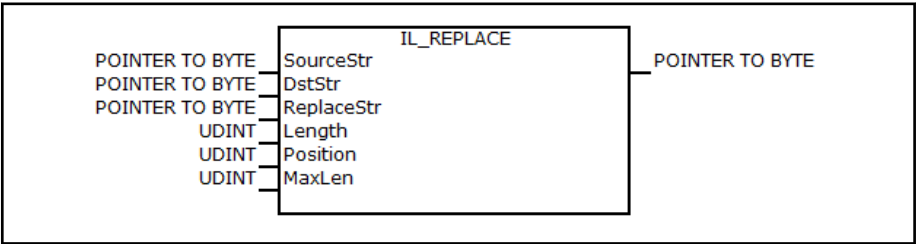


Abb. 4-107: FUN IL_REPLACE

Name	Typ	Kommentar
SourceStr	POINTER TO BYTE	Adresse der Quell-Zeichenfolge
DstStr	POINTER TO BYTE	Adresse der Ziel-Zeichenfolge
ReplaceStr	POINTER TO BYTE	Adresse der einzufügenden Teil-Zeichenfolge (Ersatz-Zeichenfolge)
Length	UDINT	Länge der zu entfernenden Teil-Zeichenfolge (in Byte/Char)
Position	UDINT	Position der Quell-Zeichenfolge, ab der die Teil-Zeichenfolge durch die Ersatz-Zeichenfolge ersetzt werden soll
MaxLen	UDINT	Max. Länge der Ziel-Zeichenfolge (in Byte/Char)

Tab. 4-95: Schnittstelle der FUN IL_REPLACE

Beispiel 1

```
Str1: STRING := 'ABCDE';  
Str2: STRING;  
Str3: STRING := 'XX';  
IL_REPLACE(ADR(Str1), ADR(Str2), ADR(Str3), 3, 2, 5);  
Rückgabewert: ADR(Str2)  
Str3: 'AXXE'
```

Beispiel 2

```
Str1: STRING := 'ABCDE';  
Str2: STRING;  
Str3: STRING := 'XX';  
IL_REPLACE(ADR(Str1), ADR(Str2), ADR(Str3), 3, 2, 2);  
Rückgabewert: ADR(Str2)  
Str3: 'AX'
```

Beispiel 3

Str1: STRING := 'ABCDE';

Str3: STRING := 'XX';

IL_REPLACE(ADR(Str1), ADR(Str1), ADR(Str3), 3, 2, 5); Rückgabewert: ADR(Str1)

Str1: 'AXXE'

4.12.22 IL_RIGHT**Kurzbeschreibung**Die Funktion **IL_RIGHT** liefert die rechte Teil-Zeichenfolge einer Zeichenfolge.

Rückgabewert: Adresse der Ziel-Zeichenfolge.

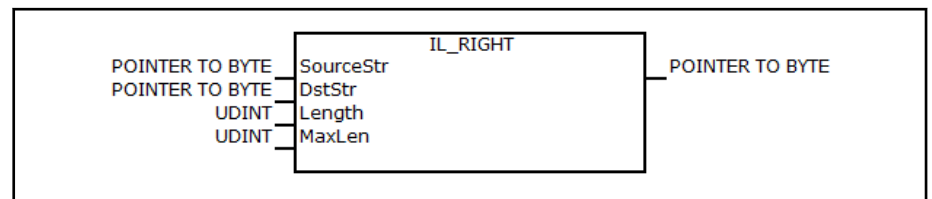
Schnittstellenbeschreibung

Abb. 4-108: FUN IL_RIGHT

Name	Typ	Kommentar
SourceStr	POINTER TO BYTE	Adresse der Quell-Zeichenfolge
DstStr	POINTER TO BYTE	Adresse der Ziel-Zeichenfolge
Length	UDINT	Länge der rechten Teil-Zeichenfolge (in Byte/Char)
MaxLen	UDINT	Max. Länge der Ziel-Zeichenfolge (in Byte/Char)

Tab. 4-96: Schnittstelle der FUN IL_RIGHT

Beispiel 1

Str1: STRING := 'ABCD';

Str2: STRING;

IL_RIGHT(ADR(Str1), ADR(Str2), 3, 5);

Rückgabewert: ADR(Str2)

Str2: 'BCD'

Beispiel 2

Str1: STRING := 'ABCD';

Str2: STRING;

IL_RIGHT(ADR(Str1), ADR(Str2), 3, 2);

Rückgabewert: ADR(Str2)

Str2: 'BC'

Beispiel 3

Str1: STRING := 'ABCD';

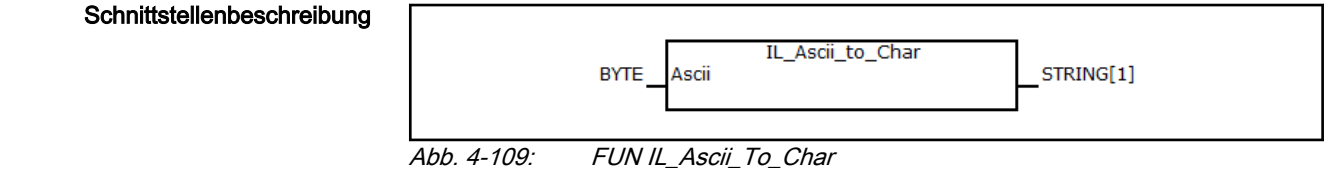
IL_RIGHT(ADR(Str1), ADR(Str1), 3, 5);

Rückgabewert: ADR(Str1)

Str1: 'BCD'

4.12.23 IL_Ascii_To_Char

Kurzbeschreibung Die Funktion **IL_Ascii_To_Char** konvertiert ASCII-Code in ein Zeichen.
Rückgabewert: Zeichen



Name	Typ	Kommentar
Ascii	BYTE	zu konvertierender ASCII-Code

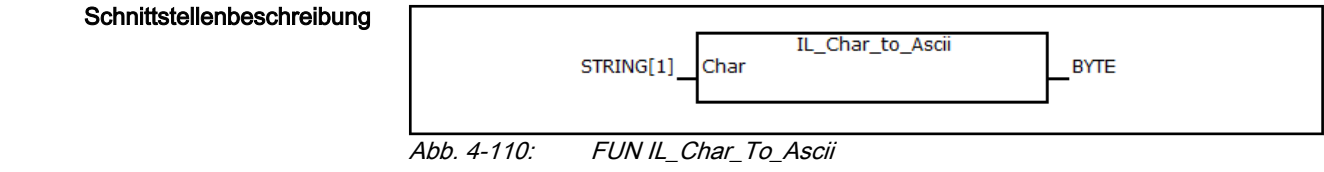
Tab. 4-97: Schnittstelle der FUN IL_Ascii_To_Char

Beispiel

IL_Ascii_to_Char (16#63);	Rückgabewert: 'c'
IL_Ascii_to_Char (99);	Rückgabewert: 'c'

4.12.24 IL_Char_To_Ascii

Kurzbeschreibung Die Funktion **IL_Char_To_Ascii** konvertiert ein Zeichen in ASCII-Code.
Rückgabewert: ASCII-Code.



Name	Typ	Kommentar
Char	STRING[1]	zu konvertierendes Zeichen

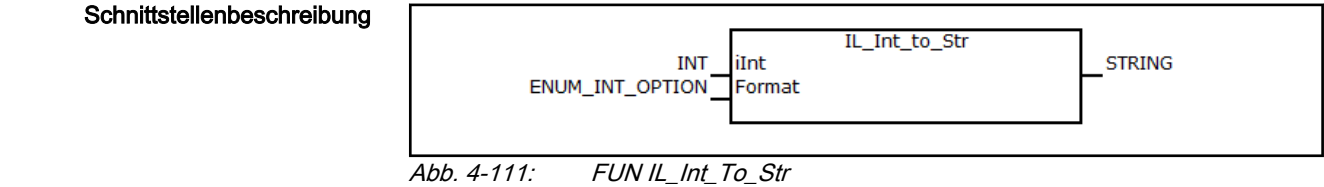
Tab. 4-98: Schnittstelle der FUN IL_Char_To_Ascii

Beispiel

IL_Char_To_Ascii ('c')	Rückgabewert: 16#63
IL_Char_To_Ascii ('c')	Rückgabewert: 99

4.12.25 IL_Int_To_Str

Kurzbeschreibung Die Funktion **IL_Int_To_Str** konvertierte eine Ganzzahl formatabhängig in eine Zeichenfolge.
Rückgabewert: Zeichenfolge.



Name	Typ	Kommentar
ilInt	INT	zu konvertierende Ganzzahl
Format	ENUM_INT_OPTION	Auswahl der Zahlendarstellung in der Zeichenfolge ENUM_INT_OPTION, Seite 165

Tab. 4-99: Schnittstelle der FUN IL_Int_To_Str

Beispiel 1

IL_Int_To_Str (10, EN_OPT_BIN); Rückgabewert: '1010'
Beispiel 2

IL_Int_To_Str (10, EN_OPT_DEZ); Rückgabewert: '10'
Beispiel 3

IL_Int_To_Str (10, EN_OPT_HEX); Rückgabewert: 'A'
4.12.26 ENUM_INT_OPTION

Der Aufzählungstyp **ENUM_INT_OPTION** gestattet die Auswahl der Darstellung der Ganzzahl in der Zeichenfolge.

Name	Wert	Beschreibung
EN_OPT_BIN	0	Dualzahl
EN_OPT_DEZ	1	Dezimalzahl
EN_OPT_HEX	2	Hexadezimalzahl

Tab. 4-100: Elemente von ENUM_INT_OPTION

4.12.27 IL_Real_To_Str**Kurzbeschreibung**

Die Funktion **IL_Real_To_Str** konvertierte eine REAL-Zahl (max. sieben signifikante Stellen) formatabhängig in eine Zeichenfolge.

Rückgabewert: Adresse der Ziel-Zeichenfolge.

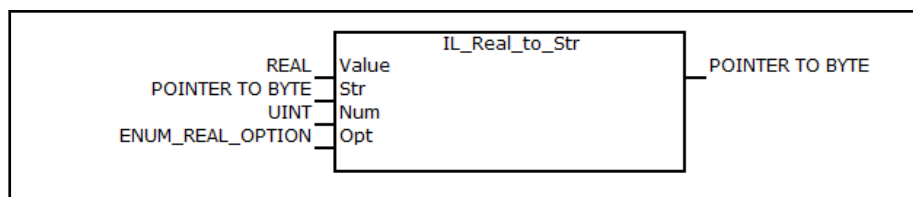
Schnittstellenbeschreibung

Abb. 4-112: FUN IL_Real_To_Str

Name	Typ	Kommentar
Value	REAL	zu konvertierende REAL-Zahl
Str	POINTER TO BYTE	Adresse der Ziel-Zeichenfolge
Num	UINT	Die Anzahl der anzuzeigenden Dezimalstellen
Opt	ENUM_REAL_OPTION	Auswahl der Zahlendarstellung in der Zeichenfolge ENUM_REAL_OPTION, Seite 166

Tab. 4-101: Schnittstelle der FUN IL_Real_To_Str

Beispiel 1

IL_Real_To_Str (263.355, 2, EN_OPT_FLOATING);	Rückgabewert: '263.36'

Beispiel 2

IL_Real_To_Str (263.355, 2, EN_OPT_EXP);	Rückgabewert: '2.63E+2'

Beispiel 3

IL_Real_To_Str (263.355, 0, EN_OPT_ENGINEERING);	Rückgabewert: '263*10^0'
IL_Real_To_Str (263.355, 2, EN_OPT_ENGINEERING);	Rückgabewert: '2.63*10^+3'

4.12.28 ENUM_REAL_OPTION

Der Aufzählungstyp **ENUM_REAL_OPTION** gestattet die Auswahl der Darstellung der REAL-Zahl in der Zeichenfolge.

Name	Wert	Beschreibung
EN_OPT_FLOATING	0	Floating-Point-Darstellung
EN_OPT_EXP	1	Exponential-Darstellung
EN_OPT_ENGINEERING	2	Ingenieur-Darstellung

Tab. 4-102: Elemente von ENUM_REAL_OPTION

4.12.29 IL_String_To_Unicode

Kurzbeschreibung

Die Funktion **IL_String_To_Unicode** konvertiert eine Zeichenfolge entsprechend der vorgegebenen Option (utf8 / utf16) in Unicode und speichert die Werte in einem DWORD-Array ab.



Pro String-Zeichen ist nur 1 Byte an Speicherplatz reserviert. Deshalb können nur die Zeichen, die ASCII Code und Windows 1252 entsprechen, erkannt werden.

Schnittstellenbeschreibung

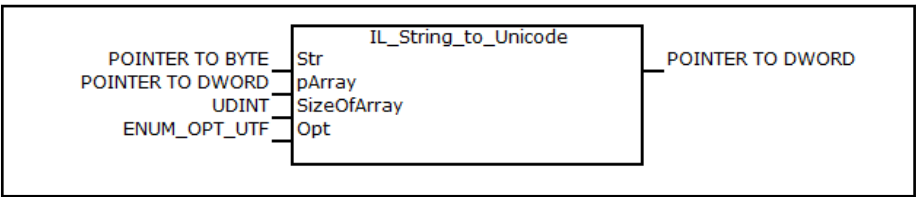


Abb. 4-113: FUN IL_String_To_Unicode

Name	Typ	Kommentar
Str	POINTED TO BY BYTE	Adresse der Quell-Zeichenfolge
pArray	POINTED TO DWORD	Adresse des Zielarrays
SizeOfArray	UDINT	Max. Größe des Arrays (in Byte)
Opt	ENUM_OPT_UTF	Unicode-Kodierungsformat ENUM_OPT_UTF, Seite 167

Tab. 4-103: Schnittstelle der FUN IL_String_To_Unicode

Beispiel 1

Str1: STRING := 'ABÝæ';

arArray: Array[0..3] OF DWORD;

IL_String_To_Unicode (ADR(Str1), ADR(arArray), 4, EN_UTF8);

Rückgabewert: ADR(arArray)

arArray[0]: 16#41

arArray[1]: 16#42

arArray[2]: 16#E6

arArray[3]: 16#DD

Beispiel 2

Str1: STRING := 'ABÝæ';

arArray: Array[0..3] OF DWORD;

IL_String_To_Unicode (ADR(Str1), ADR(arArray), 4, EN_UTF16);

Rückgabewert: ADR(arArray)

arArray[0]: 16#41

arArray[1]: 16#42

arArray[2]: 16#C39D

arArray[3]: 16#C3A6

4.12.30 ENUM_OPT_UTF

Der Aufzählungstyp **ENUM_OPT_UTF**, Typ INT, erlaubt die Auswahl des Unicode-Kodierungsformats.

Name	Wert	Beschreibung
EN_UTF16	0	UTF16-Code
EN_UTF8	1	UTF8-Code

Tab. 4-104: Elemente von *ENUM_OPT_UTF*

4.12.31 IL_Unicode_To_String**Kurzbeschreibung**

Die Funktion **IL_Unicode_To_String** konvertiert Unicode in STRING.

Der Unicode wird in einem DWORD-Array gespeichert. Ein Array-Element entspricht einem Unicode-Zeichen.



Pro String-Zeichen ist nur 1 Byte an Speicherplatz reserviert. Deshalb können nur die Zeichen, die ASCII-Code und Windows 1252 entsprechen, erkannt werden.

Rückgabewert: Adresse der Ziel-Zeichenfolge.

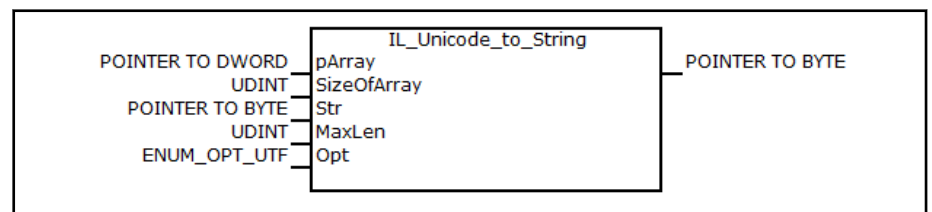
Schnittstellenbeschreibung

Abb. 4-114: *FUN IL_Unicode_To_String*

Name	Typ	Kommentar
pArray	POINTER TO DWORD	Adresse des Quellarrays
SizeOfArray	UDINT	Max. Größe des Arrays (in Byte)
Str	POINTER TO BYTE	Adresse der Ziel-Zeichenfolge
Opt	ENUM_OPT_UTF	Unicode-Kodierungsformat ENUM_OPT_UTF, Seite 167

Tab. 4-105: Schnittstelle der FUN IL_Unicode_To_String

Beispiel 1

```
arArray: Array[0..3] of DWORD := [16#41, 16#42, 16#E6, 16#DD];
Str1: STRING;
IL_Unicode_To_String (ADR(arArray), 4, ADR(Str1), 4, EN_UTF8);
                                                                    Rückgabewert: ADR(Str1)
                                                                    Str1: 'ABŸæ'
```

Beispiel 2

```
arArray: Array[0..3] of DWORD := [16#41, 16#42, 16#C39D, 16#C3A6];
Str1: STRING;
IL_String_To_Unicode (ADR(Str1), ADR(arArray), 4, EN_UTF16);
                                                                    Rückgabewert: ADR(Str1)
                                                                    Str1: 'ABŸæ'
```

4.12.32 IL_BufferGetElementCount

Kurzbeschreibung

Die Funktion **IL_BufferGetElementCount** bestimmt die Anzahl der durch ein "Separator"-Zeichen getrennten Objekte in einer Zeichenkette.

Zuweisung: Zielsystem/Bibliothek

Zielsystem	Bibliothek
MLC 15VRS	RIL_Uutilities

Tab. 4-106: Referenztabelle des IL_BufferGetElementCount

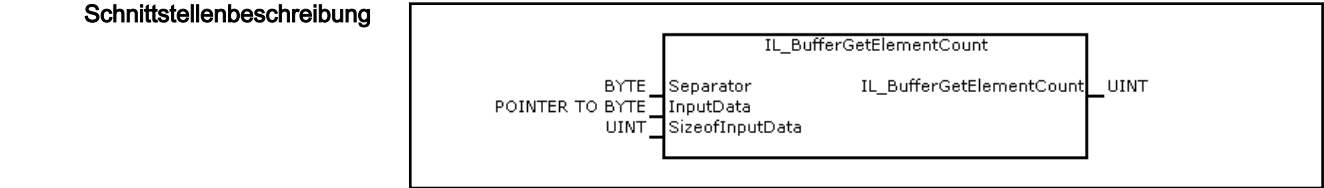


Abb. 4-115: Funktion IL_BufferGetElementCount

	Name	Typ	Kommentar
VAR_INPUT	Separator	BYTE	ASCII-Trennzeichen in der auszuwertenden Zeichenkette
	InputData	POINTER TO BYTE	Zeiger auf auszuwertende Zeichenkette Verwenden Sie den ADR()-Operator

	Name	Typ	Kommentar
	SizeOfInputData	UINT	Gesamtlänge der Zeichenkette Verwenden Sie die Funktion IL_Len() und den Operator ADR() für die zu analysierende Zeichenkette
RETURN		UINT	Anzahl der separierten Objekte

Tab. 4-107: Funktion IL_BufferGetElementCount

Funktionsbeschreibung

Die Funktion **IL_BufferGetElementCount** bestimmt die Anzahl der durch ein "Separator"-Zeichen getrennten Objekte in einer Zeichenkette.

Beispiel: Zeichenkette: '0/123/2/3', Separator: /, Rückgabewert: 4.

Gibt es nur ein Zeichen in der Zeichenkette "InputData", wird kein "Separator" benötigt.

Als Rückgabewert der Funktion wird 1 ausgegeben, z.B. ('X',/) = 1.

Ist "InputData" eine leere Zeichenkette, wird 0 als Rückgabewert der Funktion ausgegeben, z.B. ("",/) = 0.

Beispiele

Die Funktion **IL_BufferGetElementCount** bestimmt die Anzahl der durch ein "Separator"-Zeichen getrennten Objekte in einer Zeichenkette.

```
VAR
  strInputString:   STRING(300) ;
  uiNoOfCharacters: UINT;
END_VAR

strInputString := '1/2/333/4/a/b/C/D/!/@/</>;/,{/[';

uiNoOfCharacters := IL_BufferGetElementCount(
  Separator:= IL_AsciiToByte('/'),
  InputData:= ADR(strInputString),
  SizeOfInputData := UDINT_TO_UINT (IL_Len(ADR(strInputString))));
```

Das Ergebnis des Beispiels "IL_BufferGetElementCount" wird nachfolgend gezeigt:

strInputString	STRING(300)	'1/2/333/4/a/b/C/D/!/@/</>;/,{/['	
uiNoOfCharacters	UINT	15	

Abb. 4-116: Beispiel "IL_BufferGetElementCount"

4.12.33 IL_Code

Kurzbeschreibung

Die Funktion **IL_Code** ermittelt den ASCII-Code eines vorgegebenen Zeichens in einer Zeichenkette in dezimaler Darstellung.

Zuweisung: Zielsystem/Bibliothek

Zielsystem	Bibliothek
MLC 15VRS	RIL_Uilities

Tab. 4-108: Referenztable des IL_Code

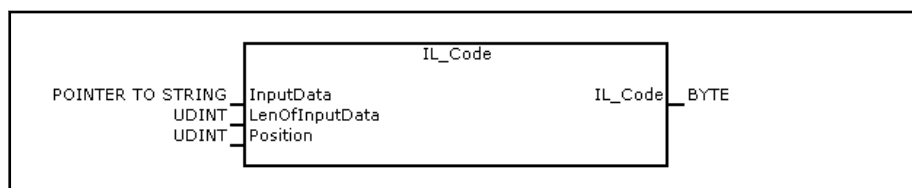
Schnittstellenbeschreibung

Abb. 4-117: Funktion IL_Code

	Name	Typ	Kommentar
VAR_INPUT	InputData	POINTER TO STRING	Zeiger auf eine ASCII-Zeichenkette Verwenden Sie den ADR()-Operator
	LenOfInputData	UDINT	Länge der Zeichenkette Verwenden Sie: IL_Len (InputData)
	Position	UDINT	Position, an der das Zeichen gelesen wird
RETURN		BYTE	Code des Zeichens an der "Position"

Tab. 4-109: Funktion IL_Code

Funktionsbeschreibung

Die Funktion **IL_Code** ermittelt den ASCII-Code eines vorgegebenen Zeichens in einer Zeichenkette in dezimaler Darstellung.

Die Lage des Zeichens wird durch den Wert "Position" vorgegeben. Das erste Zeichen der Zeichenkette hat die Position 1.

Wird **IL_Code** mit einer "Position" kleiner als 1 oder größer als die Länge der Zeichenkette "LenOfInputData" aufgerufen, wird der Wert 0 zurückgegeben.

Berechnen Sie "LenOfInputData" mit Hilfe der IL_Len-Funktion.

Beispiel:

IL_Code ('ABC 123', 4) ergibt 32 (das Leer-Zeichen ist mit dem Wert 32 kodiert). Der Rückgabewert vom Typ BYTE gibt das Zeichen als ASCII-Wert zurück.

Beispiele

Die Funktion **IL_Code** ermittelt den ASCII-Code in dezimaler Darstellung für das Zeichen an der Position "Position" einer Zeichenkette.

```
VAR
    strCode: STRING(1000) := 'ABCDEF';
    byIL_Code: BYTE;
END_VAR

byIL_Code := IL_Code(InputData:= ADR(strCode),
    LenOfInputData := IL_Len(ADR(strCode)), Position:=3 );
```

Das Ergebnis des Beispiels "IL_Code" wird nachfolgend gezeigt:

strCode	STRING(1000)	'ABCDEF'
byIL_Code	BYTE	67

Abb. 4-118: Beispiel "IL_Code"

4.12.34 IL_StrInverse

Kurzbeschreibung

Die Funktion **IL_StrInverse** invertiert eine Zeichenfolge.

Rückgabewert: Adresse der Ziel-Zeichenfolge.

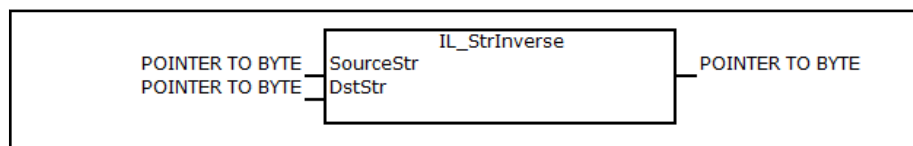
Schnittstellenbeschreibung

Abb. 4-119: FUN IL_StrInverse

Name	Typ	Kommentar
SourceStr	POINTER TO BYTE	Adresse der Quell-Zeichenfolge
DstStr	POINTER TO BYTE	Adresse der Ziel-Zeichenfolge

Tab. 4-110: Schnittstelle der FUN IL_StrInverse

Beispiel 1

Str1: STRING := 'abcde';

Str2: STRING;

IL_StrInverse (ADR(Str1), ADR(Str2));

Rückgabewert: ADR(Str2)

Str2: 'edcba'

Beispiel 2

Str1: STRING := 'abcde';

IL_StrInverse (ADR(Str1), ADR(Str1));

Rückgabewert: ADR(Str1)

Str1: 'edcba'

4.12.35 IL_StrSetEndOfString**Kurzbeschreibung**

Die Funktion **IL_StrSetEndOfString** setzt das Endekennzeichen einer Zeichenfolge und begrenzt so die Länge der Ziel-Zeichenfolge.

Rückgabewert: Adresse der Ziel-Zeichenfolge.

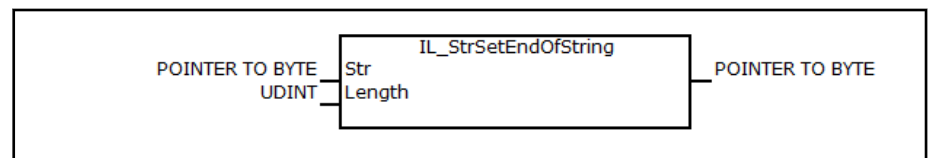
Schnittstellenbeschreibung

Abb. 4-120: FUN IL_StrSetEndOfString

Name	Typ	Kommentar
Str	POINTER TO BYTE	Adresse der Quell-Zeichenfolge
Length	UDINT	Länge der Ziel-Zeichenfolge

Tab. 4-111: Schnittstelle der FUN IL_StrSetEndOfString

Beispiel

Str1: STRING := 'abcde';

IL_StrSetEndOfString (ADR(Str1), 3);

Rückgabewert: ADR(Str1)

Str1: 'abc'

4.12.36 IL_StrGetElement**Kurzbeschreibung**

Die Quell-Zeichenfolge ist durch Trenn-Zeichenfolgen in Elemente aufgeteilt. Die Funktion **IL_StrGetElement** ruft das durch "ElementNum" spezifizierte Element ab.

Rückgabewert: Adresse des Elements.

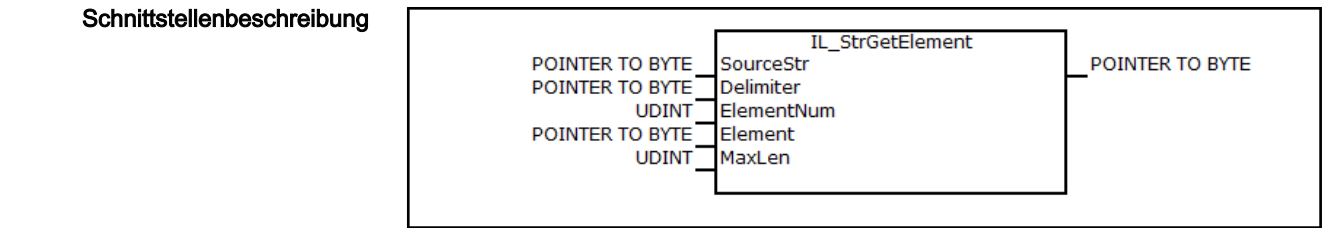


Abb. 4-121: FUN IL_StrGetElement

Name	Typ	Kommentar
SourceStr	POINTER TO BYTE	Adresse der Quell-Zeichenfolge
Delimiter	POINTER TO BYTE	Adresse der Trenn-Zeichenfolge
ElementNum	UDINT	Nummer des Elements, das abgerufen werden soll. Das erste Element hat die Nummer 1
Element	POINTER TO BYTE	Adresse, unter der das abgerufenen Element abgelegt werden soll
MaxLen	UDINT	Max. Länge des Elements (in Byte/Char)

Tab. 4-112: Schnittstelle der FUN IL_StrGetElement

Beispiel 1

Str1: STRING := 'ab;;cd;;e';
Str2: STRING := ';;' ;
Str3: STRING;
IL_StrGetElement (ADR(Str1), ADR(Str2), 2 , ADR(Str3), 5);

Rückgabewert: ADR(Str3)
Str3: 'cd'

Beispiel 2

Str1: STRING := 'ab;;cd;;e';
Str2: STRING := ';;' ;
Str3: STRING;
IL_StrGetElement (ADR(Str1), ADR(Str2), 2 , ADR(Str3), 1);

Rückgabewert: ADR(Str3)
Str3: 'cd'

Beispiel 3

Str1: STRING := 'ab;;cd;;e';
Str2: STRING := ';;' ;
IL_StrGetElement (ADR(Str1), ADR(Str2), 2 , ADR(Str1), 5);

Rückgabewert: ADR(Str1)
Str1: 'cd'

4.12.37 IL_StrLChr

Kurzbeschreibung

Die Funktion **IL_StrLChr** übernimmt die Teil-Zeichenfolge von Anfang der Quell-Zeichenfolge bis zum ersten Auftreten des eingegebenen Zeichens "Char".
Rückgabewert: Adresse der Ziel-Zeichenfolge.

Schnittstellenbeschreibung

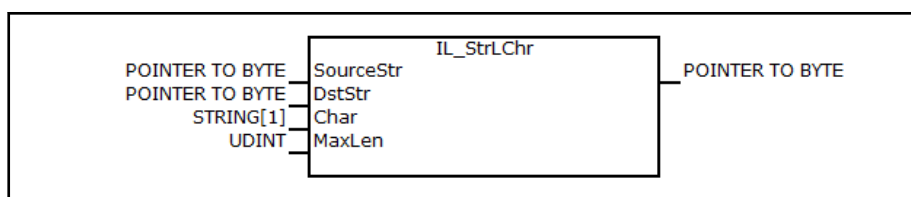


Abb. 4-122: FUN IL_StrLChr

Name	Typ	Kommentar
SourceStr	POINTER TO BYTE	Adresse der Quell-Zeichenfolge
DstStr	POINTER TO BYTE	Adresse der Ziel-Zeichenfolge
Char	STRING[1]	Begrenzer der Teil-Zeichenfolge
MaxLen	UDINT	Max. Länge des Elements (in Byte/Char)

Tab. 4-113: Schnittstelle der FUN IL_StrLChr

Beispiel 1

Str1: STRING := 'This is a string' ;

Str2: STRING;

IL_StrLChr (ADR(Str1), ADR(Str2), 'i', 100);

Rückgabewert: ADR(Str2)

Str2: 'Thi'

Beispiel 2

Str1: STRING := 'This is a string' ;

IL_StrLChr (ADR(Str1), ADR(Str1), 'i', 100);

Rückgabewert: ADR(Str1)

Str1: 'Thi'

4.12.38 IL_StrRChr

Kurzbeschreibung

Die Funktion **IL_StrRChr** übernimmt die Teil-Zeichenfolge vom letzten Auftreten des eingegebenen Zeichens "Char" bis zum Ende der Quell-Zeichenfolge.

Rückgabewert: Adresse des Zielstrings.

Schnittstellenbeschreibung

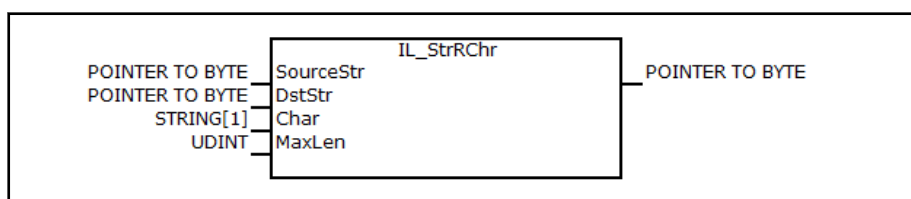


Abb. 4-123: FUN IL_StrRChr

Name	Typ	Kommentar
SourceStr	POINTER TO BYTE	Adresse der Quell-Zeichenfolge
DstStr	POINTER TO BYTE	Adresse der Ziel-Zeichenfolge
Char	STRING[1]	Begrenzer der Teil-Zeichenfolge
MaxLen	UDINT	Max. Länge der Ziel-Zeichenfolge (in Byte/Char)

Tab. 4-114: Schnittstelle der FUN IL_StrRChr

Beispiel 1

Str1: STRING := 'This is a string' ;
Str2: STRING;
IL_StrRChr (ADR(Str1), ADR(Str2), 'i', 100);

Rückgabewert: ADR(Str2)
Str2: 'ing'

Beispiel 2

Str1: STRING := 'This is a string' ;
IL_StrRChr (ADR(Str1), ADR(Str1), 'i', 100);

Rückgabewert: ADR(Str1)
Str1: 'ing'

4.12.39 IL_StrSetElement

Kurzbeschreibung

Die Quell-Zeichenfolge ist durch Trenn-Zeichenfolgen in Elemente aufgeteilt.
Die Funktion **IL_StrSetElement** fügt die Zeichenfolge von "Element" nebst Trenn-Zeichenfolge an der durch "ElementNum" spezifizierten Stelle ein.
Rückgabewert: Adresse der Ziel-Zeichenfolge.

Schnittstellenbeschreibung

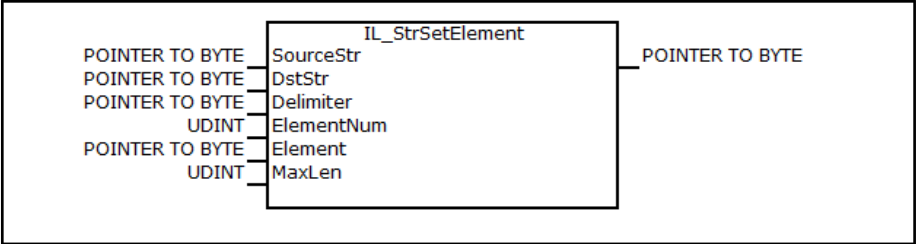


Abb. 4-124: FUN IL_StrSetElement

Name	Typ	Kommentar
SourceStr	POINTER TO BYTE	Adresse der Quell-Zeichenfolge
DstStr	POINTER TO BYTE	Adresse der Ziel-Zeichenfolge
Delimiter	POINTER TO BYTE	Adresse der Trenn-Zeichenfolge
ElementNum	UDINT	Nummer der Elementposition, an der das Element eingefügt werden soll. Das erste Element hat die Nummer 1
Element	POINTER TO BYTE	Adresse des Element, das eingefügt werden soll
MaxLen	UDINT	Max. Länge der Ziel-Zeichenfolge (in Byte/Char)

Tab. 4-115: Schnittstelle der FUN IL_StrSetElement

Beispiel 1

Str1: STRING := 'ab;;cd;;e';

Str2: STRING;

Str3: STRING := ';;' ;

Str4: STRING := 'fg';

IL_StrSetElement (ADR(Str1), ADR(Str2), ADR(Str3), 2 , ADR(Str4), 15);

Rückgabewert: ADR(Str2)

Str2: 'ab;;fg;;cd;;e'

Beispiel 2

Str1: STRING := 'ab;;cd;;e';

Str2: STRING;

Str3: STRING := ';;' ;

Str4: STRING := 'fg';

IL_StrSetElement (ADR(Str1), ADR(Str2), ADR(Str3), 2 , ADR(Str4), 2);

Rückgabewert: ADR(Str2)

Str2: 'ab'

Beispiel 3

Str1: STRING := 'ab;;cd;;e';

Str3: STRING := ';;' ;

Str4: STRING := 'fg';

IL_StrSetElement (ADR(Str1), ADR(Str1), ADR(Str3), 2 , ADR(Str4), 5);

Rückgabewert: ADR(Str1)

Str1: 'ab;;fg;;cd;;e'

4.12.40 IL_StrSplit**Kurzbeschreibung**

Die Quell-Zeichenfolge ist durch Trennzeichen in Teil-Zeichenfolgen aufgeteilt.

Die Adresse jeder Teil-Zeichenfolge wird in ein vorgegebenes Array gelegt.

Der Datentyp eines Arrayelements ist "POINTER TO STRING[255]"



Bei Aufruf der Funktion wird die Quell-Zeichenfolge geändert!

Schnittstellenbeschreibung

Rückgabewert: Anzahl der getrennten Teil-Zeichenfolgen.

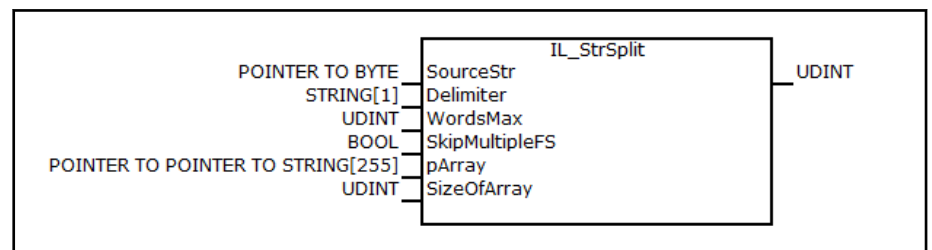


Abb. 4-125: FUN IL_StrSplit

Name	Typ	Kommentar
SourceStr	POINTER TO BYTE	Adresse der Quell-Zeichenfolge
Delimiter	STRING[1]	Trennzeichen

Name	Typ	Kommentar
MaxWord	UDINT	Maximale Anzahl der zu separierenden Teil-Zeichenfolgen
SkipMultipleFS	BOOL	TRUE: ignoriert mehrere unmittelbar aufeinander folgende Trennzeichen FALSE: falls zwei oder mehrere Trennzeichen unmittelbar aufeinander folgen, wird jeweils ein Leerstring 'NUL' als Teil-Zeichenfolgen generiert
pArray	POINTER TO POINTER TO STRING[255]	Adresse des Arrays, welches die Adressen der Teil-Zeichenfolgen enthält
SizeOfArray	UDINT	Länge des Arrays (in Byte)

Tab. 4-116: Schnittstelle der FUN IL_StrSplit

Beispiel 1

Str1: STRING := 'ab;;cd;e';

arArray: ARRAY[0..200] OF POINTER TO STRING[255];

IL_StrSplit(ADR(Str), ';', 4, FALSE, ADR(arArray), SIZEOF(arArray));

Rückgabewert: 4

arArray [1]: 'ab'

arArray [2]: ''

arArray [3]: 'cd'

arArray [4]: 'e'

Beispiel 2

Str1: STRING := 'ab;;cd;e';

arArray: ARRAY[0..200] OF POINTER TO STRING[255];

IL_StrSplit(ADR(Str), ';', 3, TRUE, ADR(arArray), SIZEOF(arArray));

Rückgabewert: 3

arArray [1]: 'ab'

arArray [2]: 'cd'

arArray [3]: 'e'

Beispiel 3

Str1: STRING := 'ab;;cd;e';

arArray: ARRAY[0..200] OF POINTER TO STRING[255];

IL_StrSplit(ADR(Str), ';', 2, TRUE, ADR(arArray), SIZEOF(arArray));

Rückgabewert: 2

arArray [1]: 'ab'

arArray [2]: 'cd;e'

4.12.41 IL_StrTrimAll**Kurzbeschreibung**

Die Funktion **IL_StrTrimAll** entfernt in einer Zeichenfolge alle "Char"-Zeichen.
Rückgabewert: Adresse der Ziel-Zeichenfolge

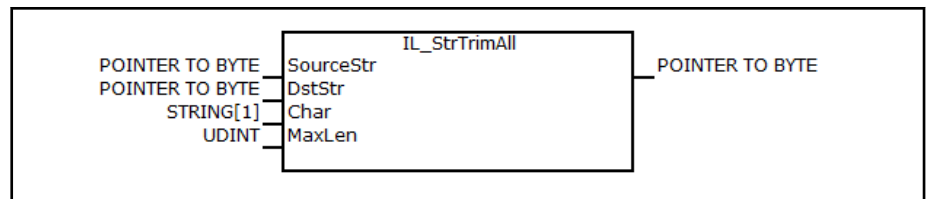
Schnittstellenbeschreibung

Abb. 4-126: FUN IL_StrTrimAll

Name	Typ	Kommentar
SourceStr	POINTER TO BYTE	Adresse der Quell-Zeichenfolge
DstStr	POINTER TO BYTE	Adresse der Ziel-Zeichenfolge
Char	STRING[1]	Zeichen, die in der Zeichenfolge entfernt werden sollen
MaxLen	UDINT	Max. Länge der Ziel-Zeichenfolge (in Byte/Char)

Tab. 4-117: Schnittstelle der FUN IL_StrTrimAll

Beispiel 1

Str1: STRING := ';;;;ab;;;;cde;;;;';

Str2: STRING;

IL_StrTrimAll (ADR(Str1), ADR(Str2), ',',10);

Rückgabewert: ADR(Str2)

Str2: 'abcde'

Beispiel 2

Str1: STRING := ';;;;ab;;;;cde;;;;';

Str2: STRING;

IL_StrTrimAll (ADR(Str1), ADR(Str2), ',',2);

Rückgabewert: ADR(Str2)

Str2: 'ab'

Beispiel 3

Str1: STRING := ';;;;ab;;;;cde;;;;';

IL_StrTrimAll (ADR(Str1), ADR(Str1), ',',10);

Rückgabewert: ADR(Str1)

Str1: 'abcde'

4.12.42 IL_StrTrimSpace**Kurzbeschreibung**

Die Funktion **IL_StrTrimSpace** entfernt je nach vorgegebener Option die Leerzeichen aus einer Zeichenfolge.

Rückgabewert: Adresse der Ziel-Zeichenfolge

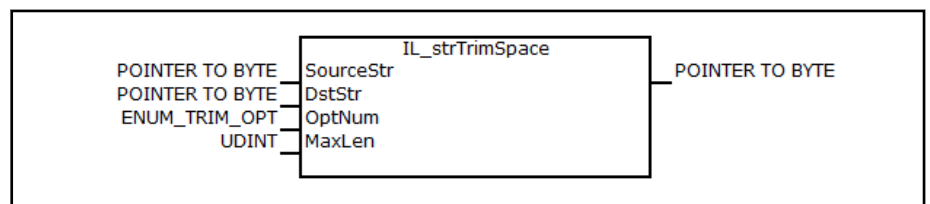
Schnittstellenbeschreibung

Abb. 4-127: FUN IL_StrTrimSpace

Name	Typ	Kommentar
SourceStr	POINTER TO BYTE	Adresse der Quell-Zeichenfolge
DstStr	POINTER TO BYTE	Adresse der Ziel-Zeichenfolge
OptNum	ENUM_TRIM_OPT	steuert das Entfernen der Leerzeichen in der Zeichenfolge ENUM_TRIM_OPT , Seite 178
MaxLen	UDINT	Max. Länge der Ziel-Zeichenfolge (in Byte/Char)

Tab. 4-118: Schnittstelle der FUN IL_StrTrimSpace

Beispiel 1

Str1: STRING := ' This is a string ';

Str2: STRING;

IL_StrTrimSpace (ADR(Str1), ADR(Str2), EN_TRIM_BOTH_ENDS, 100);

Rückgabewert: ADR(Str2)

Str2: 'This is a string'

Beispiel 2

Str1: STRING := ' This is a string ';

Str2: STRING;

IL_StrTrimSpace (ADR(Str1), ADR(Str2), EN_TRIM_BEGINNING, 100);

Rückgabewert: ADR(Str2)

Str2: 'This is a string '

Beispiel 3

Str1: STRING := ' This is a string ';

Str2: STRING;

IL_StrTrimSpace (ADR(Str1), ADR(Str2), EN_TRIM_END, 100);

Rückgabewert: ADR(Str2)

Str2: ' This is a string'

Beispiel 4

Str1: STRING := ' This is a string ';

Str2: STRING;

IL_StrTrimSpace (ADR(Str1), ADR(Str2), EN_TRIM_ALL, 100);

Rückgabewert: ADR(Str2)

Str2: 'This is a string'

4.12.43 ENUM_TRIM_OPT

Der Aufzählungstyp **ENUM_TRIM_OPT** beeinflusst die Art des Entferns der Leerzeichen in der Zeichenfolge.

Name	Wert	Beschreibung
EN_TRIM_BOTH_ENDS	0	entfernt alle Leerzeichen am Anfang und Ende der Zeichenfolge
EN_TRIM_BEGINNING	1	entfernt alle Leerzeichen am Anfang der Zeichenfolge

Name	Wert	Beschreibung
EN_TRIM_END	2	entfernt alle Leerzeichen am Ende der Zeichenfolge
EN_TRIM_ALL	3	entfernt alle Leerzeichen außer einem einzelnen Leerzeichen zwischen zwei Wörtern

Tab. 4-119: Elemente von ENUM_TRIM_OPT

4.13 Zeit- und Datumberechnung, Version 2 (Time)

4.13.1 Einführung und Übersicht



Zeitbasis für alle Zeitfunktionen und Funktionsbausteinen ist die Unixzeit (ab 1.1.1970). Wenn ein vorheriges Datum ausgegeben bzw. umgewandelt werden soll, so wird stets das Basisdatum (1.1.1970) ausgegeben.

Die folgenden Funktionen und Funktionsbausteine liefern die Systemzeit bzw. das Datum als Rückgabewert:

Anwendungsgebiete und Funktionen / Funktionsbausteine von RIL_Uilities

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine	Beschreibung
Zeit- und Datumberechnung (Time)	
IL_Date, Seite 179	liest das aktuelle Datum des Systems
IL_DateAndTime, Seite 180	liest das aktuelle Datum und die aktuelle Uhrzeit des Systems
IL_ExtSysTimeDate, Seite 180	liest das aktuelle Datum und die aktuelle Uhrzeit des Systems im erweiterten Systemformat
IL_LocalTime, Seite 181	rechnet steuerungsinterne UTC-Zeit auf lokale Zeit um
IL_SysTime64, Seite 183	liest das aktuelle Datum und die aktuelle Uhrzeit des Systems in Mikrosekunden seit 01.01.1970
IL_SysTime64ToSysTimeDate, Seite 184	dient der Formatumwandlung des Systemdatums und der Systemzeit
IL_SysTimeDate, Seite 184	liest das aktuelle Datum und die aktuelle Uhrzeit des Systems
IL_SysTimeDateToSysTime64, Seite 185	dient der Formatumwandlung des Systemdatums und der Systemzeit
IL_TimeOfDay, Seite 185	FUN liest die aktuelle Uhrzeit des Systems

Tab. 4-120: RIL_Uilities, Zeit- und Datumberechnung

4.13.2 IL_Date

Kurzbeschreibung

Die Funktion IL_Date liest das aktuelle Datum des Systems.

Schnittstellenbeschreibung

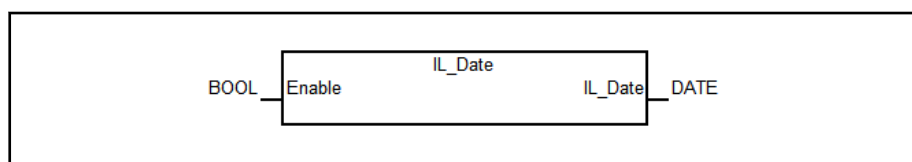


Abb. 4-128: Funktion IL_Date

	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe der Funktion (zyklisch, zustandsgesteuert)
RETURN	IL_Date	DATE	aktuelles Systemdatum gemäß der IEC61131-3

Tab. 4-121: Schnittstelle IL_Date

Funktionsbeschreibung

Die Funktion IL_Date liest nach der Bearbeitungsfreigabe der Funktion mittels "Enable" zyklisch das gemäß der IEC61131-3 formatierte Systemdatum aus.

```
CurrentDate = D#1996-01-01
```

Abb. 4-129: Formatierung IL_Date

Fehlerbehandlung

Für diese Funktion ist keine Fehlerbehandlung verfügbar.

4.13.3 IL_DateAndTime

Kurzbeschreibung

Die Funktion IL_DateAndTime liest das aktuelle Datum und die aktuelle Uhrzeit des Systems.

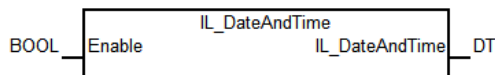
Schnittstellenbeschreibung

Abb. 4-130: Funktion IL_DateAndTime

	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe der Funktion (zyklisch, zustandsgesteuert)
RETURN	IL_DateAndTime	DT	aktuelles Systemdatum und Systemzeit gemäß der IEC61131-3

Tab. 4-122: Schnittstelle IL_DateAndTime

Funktionsbeschreibung

Die Funktion IL_DateAndTime liest nach der Bearbeitungsfreigabe der Funktion mittels "Enable" zyklisch das aktuelle Systemdatum sowie die aktuelle Systemzeit gemäß der IEC61131-3 formatiert aus.

```
CurrentDateAndTime = DT#1996-01-01-00:13:26
```

Abb. 4-131: Formatierung IL_DateAndTime

Fehlerbehandlung

Für diese Funktion ist keine Fehlerbehandlung verfügbar.

4.13.4 IL_ExtSysTimeDate

Kurzbeschreibung

Der Funktionsbaustein IL_ExtSysTimeDate liest das aktuelle Datum und die aktuelle Uhrzeit des Systems.

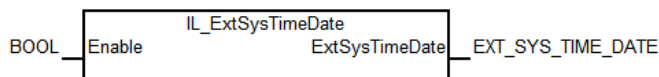
Schnittstellenbeschreibung

Abb. 4-132: Funktionsbaustein IL_ExtSysTimeDate

	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe der Funktion (zyklisch, zustandsgesteuert)
VAR_OUTPUT	ExtSysTimeDate	EXT_SYS_TIME_DATE	aktuelles Systemdatum und Systemzeit im erweiterten Systemformat

Tab. 4-123: Schnittstelle IL_ExtSysTimeDate

Funktionsbeschreibung

Der Funktionsbaustein IL_ExtSysTimeDate liest nach der Bearbeitungsfreigabe des Funktionsbausteins mittels "Enable" zyklisch das aktuelle Systemdatum sowie die aktuelle Systemzeit im erweiterten Systemformat aus.



Das erweiterte Systemformat setzt sich aus den Systemformaten SYS_TIME64 sowie SYS_TIME_DATE zusammen.

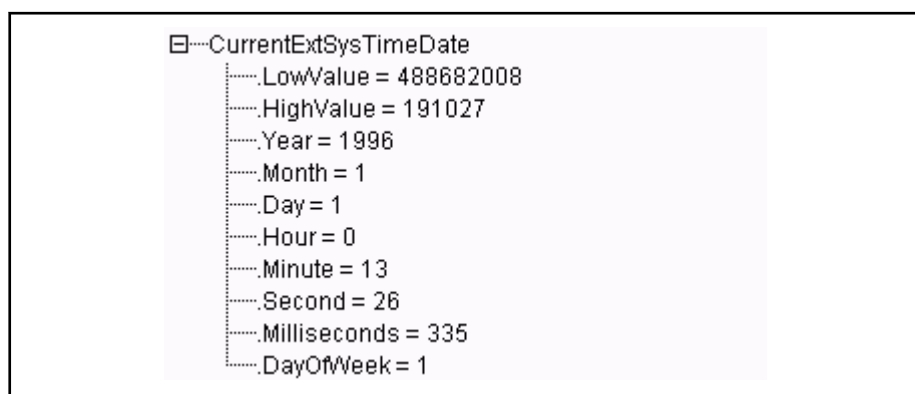


Abb. 4-133: Formatierung IL_ExtSysTimeDate

Fehlerbehandlung

Für diesen Funktionsbaustein ist keine Fehlerbehandlung verfügbar.

4.13.5 IL_LocalTime

Kurzbeschreibung

Die Funktionalität des Bausteines "IL_LocalTime" wird über den Eingang "CurrentUTCTime" gesteuert.

Wenn der Eingang "CurrentUTCTime" den Defaultwert enthält (Eingang ist nicht beschaltet), liest der Funktionsbaustein die Systemzeit der Steuerung aus und rechnet sie anhand der Zeitzoneinformation in die lokale Zeit um.

Wenn der Eingang "CurrentUTCTime" nicht den Defaultwert enthält, wird der Zeitwert über den Eingang "CurrentUTCTime" in die Lokalzeit umgerechnet und am Ausgang "LocalDateTime" ausgegeben. In diesem Fall wird die Zeit nicht aus der Steuerung gelesen.

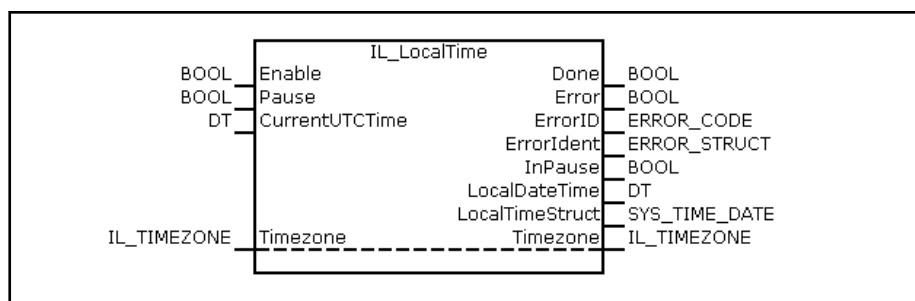
Schnittstellenbeschreibung

Abb. 4-134: Funktionsbaustein IL_LocalTime

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Startet bei steigender Flanke. Binäre Ausgänge werden bei fallender Flanke zurückgesetzt.
	Pause	BOOL	Unterbrechung der Bearbeitung
	CurrentUTC-Time	DT	Wird über den Eingang ein Zeitwert vorgegeben, so wird diese Zeit in die Lokalzeit umgerechnet und am Ausgang "LocalDate-Time" ausgegeben. Ist der Eingang unbeschaltet, wird die Systemzeit der Steuerung umgerechnet.
VAR_OUTPUT	Done	BOOL	Neue Zeit ausgegeben, Zeitausgänge sind gültig
	Error	BOOL	TRUE zeigt einen Fehler an.
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerinformation (siehe Fehlerbehandlung auf Seite 183)
	InPause	BOOL	Abarbeitungsunterbrechung aktiv
	LocalDateTime	DT	Lokale Zeit als DT-Datentyp
	LocalTimeStruct	SYS_TIME_DATE	Lokale Zeit als SYS_TIME_DATE
VAR_IN_OUT	Timezone	IL_TIMEZONE	Zeitzonenbeschreibung

Tab. 4-124: Interface IL_LocalTime

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
Pause	BOOL			FALSE	Steigende Flanke an Enable
CurrentUTC-Time	DT			DT#1970-1-1-0:0:0	Kontinuierlich wenn Enable = TRUE
Timezone	IL_TIMEZONE				Steigende Flanke an Enable

Tab. 4-125: Min-, Max- und Default-Werte der Eingänge des Funktionsbausteins IL_LocalTime

Funktionsbeschreibung

Wenn der Eingang CurrentUTCTime den Defaultwert enthält (Eingang ist nicht beschaltet), dann liest der FB bei steigender Flanke an "Enable" die Zeitzoneninformation vom Eingang "Timezone" ein. Danach wird die Systemzeit zyklisch gelesen, anhand der Zeitzoneninformation auf lokale Zeit umgerechnet und auf den Ausgängen "LocalDateTime" und "LocalTimeStruct" ausgegeben. Die Ausgabe wird einmal pro Sekunde aktualisiert. Bei jeder Aktualisierung wird der Ausgang "Done" für einen SPS-Zyklus gesetzt.

Wenn der Eingang CurrentUTCTime nicht den Defaultwert enthält, dann wird bei gesetztem "Enable" Eingang der Zeitwert über den Eingang CurrentUTC-Time eingelesen. Dieser Zeitwert wird in die Lokalzeit umgerechnet und am Ausgang "LocalDateTime" ausgegeben. In diesem Fall wird die Zeit nicht aus der Steuerung gelesen.

Die Abarbeitung kann mit dem Eingang "Pause" unterbrochen werden. Ist dieser Eingang gesetzt, so wird keine Zeit gelesen und die Ausgänge "LocalDateTime" und "LocalTimeStruct" werden eingefroren.

Tritt während der Abarbeitung ein Fehler auf, so wird der Ausgang "Error" gesetzt. "ErrorID" und "ErrorIdent" enthalten dann genauere Informationen zum Fehler.

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle F_RELATED_TABLE, 16#0170.

Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR	16#00002500	16#00000000	Winterzeiteinstellungen inkonsistent
INPUT_RANGE_ERROR	16#00002500	16#00000001	Zeit in WinterDiffUtc größer als 24 Stunden
INPUT_RANGE_ERROR	16#00002500	16#00000002	Wert in SwiWinterMonth größer als 12
INPUT_RANGE_ERROR	16#00002500	16#00000003	Wert in SwiWinterWeek größer als 5
INPUT_RANGE_ERROR	16#00002500	16#00000004	Wert in SwiWinterWeekday größer als 7
INPUT_RANGE_ERROR	16#00002500	16#00000005	Wert in SwiWinterDayYear größer als 365
INPUT_INVALID_ERROR	16#00002501	16#00000000	Sommerzeiteinstellungen inkonsistent
INPUT_RANGE_ERROR	16#00002501	16#00000001	Zeit in SummerDiffUtc größer als 24 Stunden
INPUT_RANGE_ERROR	16#00002501	16#00000002	Wert in SwiSummerMonth größer als 12
INPUT_RANGE_ERROR	16#00002501	16#00000003	Wert in SwiSummerWeek größer als 5
INPUT_RANGE_ERROR	16#00002501	16#00000004	Wert in SwiSummerWeekday größer als 7
INPUT_RANGE_ERROR	16#00002501	16#00000005	Wert in SwiSummerDayYear größer als 365
INPUT_INVALID_ERROR	16#00002502	16#00000000	Fehler bei der Umrechnung der Zeit aufgrund von falschen Zeitoneninformationen

Tab. 4-126: Fehlercodes, die von IL_LocalTime generiert werden

4.13.6 IL_SysTime64

Kurzbeschreibung Der Funktionsbaustein IL_SysTime64 liest das aktuelle Datum und die aktuelle Uhrzeit des Systems.

Schnittstellenbeschreibung

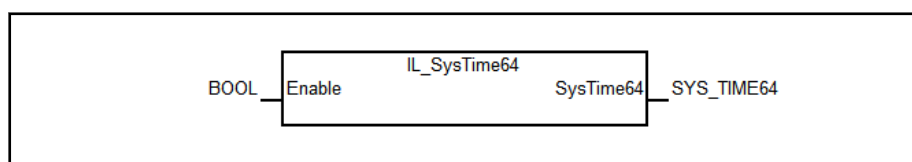


Abb. 4-135: Funktionsbaustein IL_SysTime64

	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe der Funktion (zyklisch, zustandsgesteuert)
VAR_OUTPUT	SysTime64	SYS_TIME64	aktuelles Systemdatum und Systemzeit in Mikrosekunden seit 01.01.1970

Tab. 4-127: Schnittstelle IL_SysTime64

Funktionsbeschreibung

Der Funktionsbaustein IL_SysTime64 liest nach der Bearbeitungsfreigabe des Funktionsbausteins mittels "Enable" zyklisch das aktuelle Systemdatum und die aktuelle Systemzeit in Mikrosekunden seit 01.01.1970 aus.

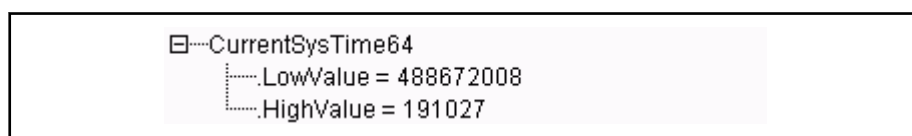


Abb. 4-136: Formatierung IL_SysTime64

Fehlerbehandlung Für diesen Funktionsbaustein ist keine Fehlerbehandlung verfügbar.

4.13.7 IL_SysTime64ToSysTimeDate

Kurzbeschreibung Der Funktionsbaustein IL_SysTime64ToSysTimeDate dient der Formatumwandlung des Systemdatums und der Systemzeit.

Schnittstellenbeschreibung

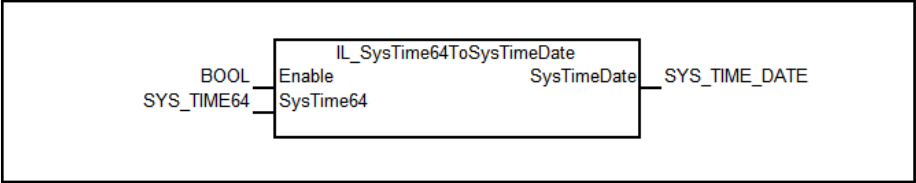


Abb. 4-137: Funktionsbaustein IL_SysTime64ToSysTimeDate

	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des FBs (zyklisch, zustandsgesteuert)
	SysTime64	SYS_TIME64	Systemdatum und Systemzeit in Mikrosekunden seit 01.01.1970
VAR_OUTPUT	SysTimeDate	SYS_TIME_DATE	Systemdatum und Systemzeit im Systemformat

Tab. 4-128: Schnittstelle IL_SysTime64ToSysTimeDate

Funktionsbeschreibung Der Funktionsbaustein IL_SysTime64ToSysTimeDate wandelt nach der Bearbeitungsfreigabe des Funktionsbausteins mittels "Enable" zyklisch das am Eingang "SysTime64" anliegende Systemdatum inkl. Systemzeit in Mikrosekunden (Basis: 01.01.1970) in das Systemdatum und die Systemzeit im Systemformat um.

Fehlerbehandlung Für diesen Funktionsbaustein ist keine Fehlerbehandlung verfügbar.

4.13.8 IL_SysTimeDate

Kurzbeschreibung Der Funktionsbaustein IL_SysTimeDate liest das aktuelle Datum und die aktuelle Uhrzeit des Systems.

Schnittstellenbeschreibung

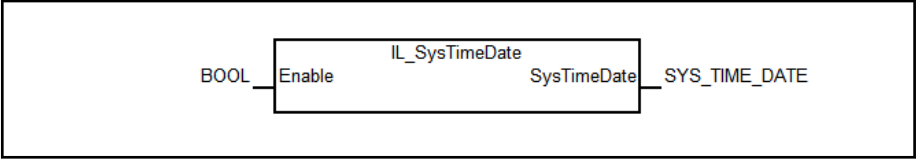


Abb. 4-138: Funktionsbaustein IL_SysTimeDate

	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe der Funktion (zyklisch, zustandsgesteuert)
VAR_OUTPUT	SysTimeDate	SYS_TIME_DATE	aktuelles Systemdatum und Systemzeit im Systemformat

Tab. 4-129: Schnittstelle IL_SysTimeDate

Funktionsbeschreibung Der Funktionsbaustein IL_SysTimeDate liest nach der Bearbeitungsfreigabe des Funktionsbausteins mittels "Enable" zyklisch das aktuelle Systemdatum und die aktuelle Systemzeit im Systemformat aus.

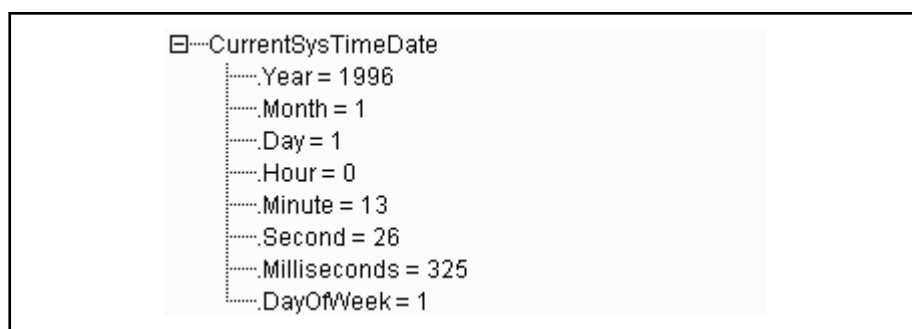


Abb. 4-139: Formatierung IL_SysTimeDate

Fehlerbehandlung Für diesen Funktionsbaustein ist keine Fehlerbehandlung verfügbar.

4.13.9 IL_SysTimeDateToSysTime64

Kurzbeschreibung Der Funktionsbaustein IL_SysTimeDateToSysTime64 dient der Formatumwandlung des Systemdatums und der Systemzeit.

Schnittstellenbeschreibung

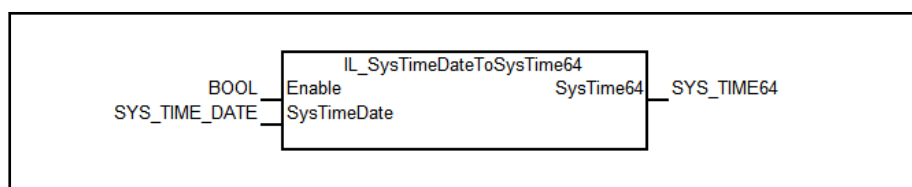


Abb. 4-140: Funktionsbaustein IL_SysTimeDateToSysTime64

	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des FBs (zyklisch, zustandsgesteuert)
	SysTimeDate	SYS_TIME_DATE	Systemdatum und Systemzeit im Systemformat
VAR_OUTPUT	SysTime64	SYS_TIME64	Systemdatum und Systemzeit in Mikrosekunden seit 01.01.1970

Tab. 4-130: Schnittstelle IL_SysTimeDateToSysTime64

Funktionsbeschreibung Der Funktionsbaustein IL_SysTimeDateToSysTime64 wandelt nach der Bearbeitungsfreigabe des Funktionsbausteins mittels "Enable" zyklisch das am Eingang "SysTimeDate" anliegende Systemdatum inkl. Systemzeit im Systemformat in das Systemdatum und die Systemzeit in Mikrosekunden (Basis: 01.01.1970) um.

Fehlerbehandlung Für diesen Funktionsbaustein ist keine Fehlerbehandlung verfügbar.

4.13.10 IL_TimeOfDay

Kurzbeschreibung Die Funktion IL_TimeOfDay liest die aktuelle Uhrzeit des Systems.

Schnittstellenbeschreibung

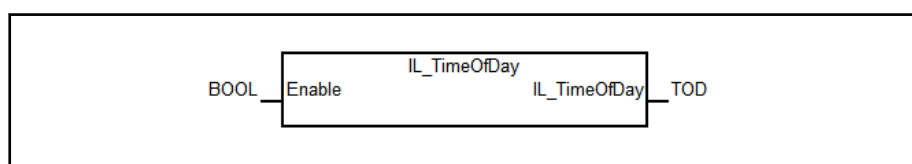


Abb. 4-141: Funktion IL_TimeOfDay

	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe der Funktion (zyklisch, zustandsgesteuert)
RETURN	IL_TimeOfDay	TOD	aktuelle Systemzeit gemäß der IEC61131-3

Tab. 4-131: Schnittstelle IL_TimeOfDay

Funktionsbeschreibung Die Funktion IL_TimeOfDay liest nach der Bearbeitungsfreigabe der Funktion mittels "Enable" zyklisch die gemäß der IEC61131-3 formatierte Systemzeit aus.

```
CurrentTimeOfDay = TOD#00:13:26.335
```

Abb. 4-142: Formatierung IL_TimeOfDay

Fehlerbehandlung Für diese Funktion ist keine Fehlerbehandlung verfügbar.

4.13.11 IL_TIMEZONE

Diese Datenstruktur beinhaltet Informationen über die Zeitzone.

Element	Datentyp	Min-Wert	Max-Wert	Default-Wert	Beschreibung
WinterDiffUtc	TIME			t#0s	Differenz lokale Winterzeit zu UTC
WinterLoc	BOOL			FALSE	TRUE: östlich von Greenwich FALSE: westlich von Greenwich
SwiWinterMonth	USINT	0	12	0	Monat der Umstellung auf Winterzeit 0: nicht verwendet 1..12: Monat
SwiWinterWeek	USINT	0	5	0	Woche innerhalb des Monats der Umstellung auf Winterzeit 0: nicht verwendet 1..4 Woche der Umstellung 5: letzte Woche im Monat
SwiWinterWeekday	USINT	0	7	0	Wochentag der Umstellung auf Winterzeit 0: nicht verwendet 1..7: Wochentag (1: Mo, 7: So)
SwiWinterDayYear	UINT	0	365	0	Tag im Jahr der Umstellung auf Winterzeit 0: nicht verwendet
SwiWinterTime	TOD	t#0s	t#23h59m59s	t#0s	Uhrzeit der Umstellung auf Winterzeit
SummerDiffUtc	TIME			t#0s	Differenz lokale Sommerzeit zu UTC
SummerLoc	BOOL			FALSE	TRUE: östlich von Greenwich FALSE: westlich von Greenwich
SwiSummerMonth	USINT	0	12	0	Monat der Umstellung auf Sommerzeit 0: nicht verwendet 1..12: Monat

Element	Datentyp	Min-Wert	Max-Wert	Default-Wert	Beschreibung
SwiSummerWeek	USINT	0	5	0	Woche innerhalb des Monats der Umstellung auf Sommerzeit 0: nicht verwendet 1..4 Woche der Umstellung 5: letzte Woche im Monat
SwiSummerWeekday	USINT	0	7	0	Wochentag der Umstellung auf Sommerzeit 0: nicht verwendet 1..7: Wochentag (1: Mo, 7: So)
SwiSummerDayYear	UINT	0	365	0	Tag im Jahr der Umstellung auf Sommerzeit 0: nicht verwendet
SwiSummerTime	TOD	t#0s	t#23h59m59s	t#0s	Uhrzeit der Umstellung auf Sommerzeit
South	BOOL			FALSE	Ort südlich des Äquators

Tab. 4-132: Datenstruktur IL_TIMEZONE

Vordefinierte Zeitzonen

Um dem Umgang mit den Zeitzonen zu erleichtern, sind bereits einige wichtige Zeitzonen definiert. Somit muss nur im Ausnahmefall die Struktur ausgefüllt werden.

Bei der Verwendung der vordefinierten Zeitzonen muss zwingend der Präfix "Timezone." verwendet werden.

Timezone.CEST für "Mitteleuropäischen Zeitzone"

Konstante	Zeitzone	Beschreibung
ACST	ACST	UTC+9:30 Australien
ACDT	ACST/ ACDT	UTC+9:30/UTC+10:30 Australien
AEST	AEST	UTC+10 Australien
AEDT	AEST/ AEDT	UTC+10/UTC+11 Australien
AMT	AMT, BOT	UTC-4 Brasilien, Bolivien
AMST	AMT/ AMST	UTC-4/UTC-3 Brasilien
ART	ART	UTC-3 Argentinien
ARST	ART/ARST	UTC-3/UTC-2 Argentinien

Konstante	Zeitzone	Beschreibung
AWST	AWST	UTC+8 Australien
BRT	BRT	UTC-3 Brasilien
BRST	BRT/BRST	UTC-3/UTC-2 Brasilien
BST	GMT/BST	UTC+0/UTC+1 Großbritannien
CAT	CAT	UTC+2 Botswana, Burundi, Kongo, Mosambique, Ruanda, Sambia, Simbabwe
CET	CET	UTC+1 Algerien, Tunesien
CEST	CET/CEST	UTC+1/UTC+2 Albanien, Andorra, Österreich, Belgien, Bosnien-Herzegovina, Kroatien, Czechei, Dänemark, Frankreich, Deutschland, Gibraltar, Vatican Stadt, Ungarn, Italien, Kosovo, Lichtenstein, Luxemburg, Mazedonien, Malta, Monakko, Montenegro, Niederlande, Norwegen, Polen, San Marino, Serbien, Slowakei, Slovenien, Spanien - außer Kanarische Inseln, Schweden, Schweiz
CCT	CCT, HKT	UTC+8 China, Makau, Taiwan, Hong Kong
CDT	CST/CDT	UTC-6/-5 Kanada, USA, Mexiko
CVT	CVT	UTC-1 Kap Verde
EAT	EAT	UTC+3 Djibouti, Eritrea, Äthiopien, Kenia, Madagaska, Somalia, Sudan, Tansania, Uganda
EDT	EST/EDT	UTC-5/UTC-4 Kanada, USA
EET	EET	UTC+2 Lybien
EEST	EET/EEST	UTC+2/UTC+3 Weißrussland, Bulgarien, Estland, Finnland, Griechenland, Letland, Litauen, Moldavien, Rumänien, Russland (Kalininograd), Türkei, Ukraine, Ägypten
FNT	FNT	UTC-2 Brasilien

Konstante	Zeitzone	Beschreibung
GMT	GMT	UTC+0 Island, Burkina Faso, Elfenbeinküste, Gambia, Ghana, Guinea-Bissau, Guinea, Liberia, Mali, Mauritien, Saint Helena, Senegal, Sierra Leone, Togo
ICT	ICT	UTC+7 Kambodscha, Laos, Thailand, Vietnam
IST	IST	UTC+5:30 Indien
JST	JST	UTC+9 Japan
MSK	MSK/MSD	UTC+3/UTC+4 Russland
MDT	MST/MDT	UTC-7/UTC-6 USA
PDT	PDT	UTC-8/UTC-7 Kanada, USA
SAST	SAST	UTC+2 Lesotho, Südafrika, Swasiland
WAT	WAT	UTC+1 Angola, Benin, Zentralafrikanische Republik, Tschad, Kongo, Äquatorial Guinea, Gabon, Niger, Nigerien
WAST	WAT/WAST	UTC+1/UTC+2 Namibia
WEST	WET/WEST	UTC+0/UTC+1 Färöer Inseln, Portugal, Kanarische Inseln, Madeira, Marokko

Tab. 4-133: Vordefinierte Zeitzonen

4.14 Doppelt-verkettete Listen (Utilities/ Logic)

4.14.1 Einführung und Übersicht

Der nachfolgende Baustein ist für Anwendungen gedacht, bei denen eine doppelt-verkettete Liste benötigt wird. Dabei können Elemente beliebig hinzugefügt und gelöscht werden.

Anwendungsgebiete und Funktionen / Funktionsbausteine von RIL_Uilities

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine	Beschreibung

Doppelt-verkettete Listen (Utilities/ Logic)	
IL_LinkedList, Seite 190,	verwaltet doppelt-verkettete Listen (verwendet für Ringpuffer, FIFOs und LIFOs) Der Baustein nutzt die Datentypen - IL_LINKED_LIST_MAINTENANCE, Seite 194 - IL_LINKED_LIST_MODE, Seite 194

Tab. 4-134: RIL_Uutilities, doppelt-verkettete Listen

4.14.2 IL_LinkedList

Kurzbeschreibung

Der Funktionsbaustein verwaltet eine doppelt-verkettete Liste. Der Listeninhalt kann vom Anwender selbst definiert werden.



Der Funktionsbaustein "IL_LinkedList (ML_TechBase-Bibliothek)" wurde mit Version 14VRS in die "RIL_Uutilities-Bibliothek" verlegt

Schnittstellenbeschreibung

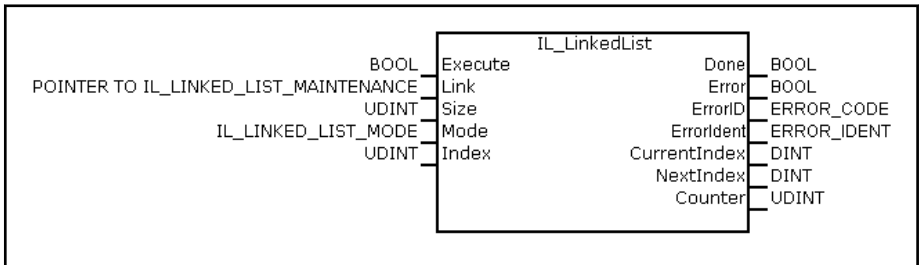


Abb. 4-143: Funktionsbaustein IL_LinkedList

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bei steigender Flanke beginnt der Baustein mit seiner Funktionalität
	Link	POINTER TO IL_LINKED_LIST_MAINTENANCE	Pointer zu einem Feld von IL_LINKED_LIST_MAINTENANCE-Strukturen, die für die Verwaltung der Liste benötigt wird IL_LINKED_LIST_MAINTENANCE, Seite 194
	Size	UDINT	Anzahl der Listenelemente, die zu verwalten sind
	Mode	IL_LINKED_LIST_MODE	Modus, den der Baustein ausführen soll, z.B. Element hinzufügen oder entfernen IL_LINKED_LIST_MODE, Seite 194
	Index	DINT	Index auf den die Operation angewendet werden soll

E/A-Typ	Name	Datentyp	Beschreibung
VAR_OUTPUT	Done	BOOL	Wenn dieser Ausgang TRUE wird, hat der Baustein seine Arbeit erfolgreich abgeschlossen
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist. Die Ausgangsdaten werden alle ungültig
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehlerklassen Fehlercodes, Seite 194
	CurrentIndex	DINT	gegenwärtiger Index
	NextIndex	DINT	nächster Index
	Counter	UDINT	Anzahl an Elementen, die sich in der Liste befinden

Tab. 4-135: Schnittstellenvariablen Funktionsbaustein IL_LinkedList

Min.- Max.- und Default-Werte der Eingänge

Die nachfolgenden Wertebereiche und Defaultwerte können an den Baustein angelegt werden:

Name	Typ	Min-Wert	Max-Wert	Default-Wert
Link	0	0	16#FFFFFFFF	0
Size	UDINT	0	16#FFFFFFFF	0
Mode	IL_LINKED_LIST_MODE	ModeInitClear	ModeCheckIndex	ModeInitClear
Index	UDINT	0	16#FFFFFFFF	0

Tab. 4-136: Min-/Max- und Defaultwerte Funktionsbaustein IL_LinkedList

Funktionsbeschreibung

Der Funktionsbaustein verwaltet eine doppelt verkettete Liste. Die Liste ist in einem Feld abgelegt, bei der lediglich die Indizes in diesem Feld verwaltet werden. Der Anwender muss für die Verwaltung der Liste ein Feld vom Typ IL_LINKED_LIST_MAINTENANCE anlegen (siehe [Verwendung der Struktur IL_LINKED_LIST_MAINTENANCE, Seite 191](#)), das genauso groß sein muss wie seine eigenen Daten.

Die Liste wird durch den Aufruf des Bausteins manipuliert. Die Art der Manipulation kann durch den "Mode"-Eingang festgelegt werden ([IL_LINKED_LIST_MODE, Seite 194](#)).

Verwendung der Struktur IL_LINKED_LIST_MAINTENANCE

Die Datenstruktur IL_LINKED_LIST_MAINTENANCE wird durch den Baustein verwendet und darf nicht vom Anwender überschrieben werden. Da der Anwender die Länge der verketteten Liste selbst bestimmen kann, muss er auch die Verwaltungsstrukturen für die Liste selbst anlegen.

Dies geschieht durch die Definition eines Feldes vom Typ [IL_LINKED_LIST_MAINTENANCE, Seite 194](#).

Die Struktur selbst kann für Debug-Zwecke verwendet werden.

"Prev" ist dabei ein Pointer auf das vorhergehende Listenelement

"Next" ist dabei ein Pointer auf das nachfolgende Listenelement.

Ist der Wert Null, ist dies das Ende des Feldes, der Anfang oder ein unbenutztes Element.

Die Eingänge "Link" und "Size" des Funktionsbausteins dürfen nach der Initialisierung durch den "Mode ModelInitClear" nicht mehr verändert werden. Dies gilt auch bei einem Online Change, wenn z.B. die Länge der Datenstruktur verändert wird.

Beispiele Beispiel für die Verwendung des Bausteins:

Definition von Variablen:

```
(* Maximum array size *)
uiMAX_ARRAY_SIZE: INT := 10;
(* My data structure array *)
arMyData: ARRAY [1..uiMAX_ARRAY_SIZE] OF MY_STRUCTURE;
(* List management array*)
arLinkedList: ARRAY [1..uiMAX_ARRAY_SIZE] OF IL_LINKED_LIST_MAINTANANCE;
(* Function block instance *)
fbMyList: IL_LinkedList;
(* Index variable*)
udiIndex: DINT;
```

Aufruf zum Initialisieren:

```
fbMyList(Execute:= TRUE,
  Link:= ADR(arLinkedList),
  Size:= uiMAX_ARRAY_SIZE,
  Mode:= ModeInitClear,
  Index:= ,
  Done=> ,
  Error=> ,
  ErrorId =>,
  ErrorIdent=> ,
  CurrentIndex=> ,
  NextIndex=>);
```

Kopf in Liste einfügen:

```
(* Reset "Execute" input first (value of mode is not important) *)
fbMyList(Execute:= FALSE,
  Link:= ADR(arLinkedList),
  Size:= uiMAX_ARRAY_SIZE,
  Mode:= ModeInitClear,
  Index:= ,
  Done=> ,
  Error=> ,
  ErrorId =>,
  ErrorIdent=> ,
  CurrentIndex=> ,
  NextIndex=>);

(* Add head *)
fbMyList(Execute:= TRUE,
  Link:= ADR(arLinkedList),
  Size:= uiMAX_ARRAY_SIZE,
  Mode:= ModeAddHead,
  Index:= ,
  Done=> ,
  Error=> ,
  ErrorId =>,
  ErrorIdent=> ,
  CurrentIndex=> udiIndex,
  NextIndex=>);

(* Return value in CurrentIndex contains index for list head or
-1 if an error occurred. *)
IF udiIndex <> -1 THEN
  arMyData[udiIndex].StructureMember := MyValue;
END_IF;
```

Durch eine Liste laufen:

```
(* Reset "Execute" input first (value of mode is not important) *)
fbMyList(Execute:= FALSE,
  Link:= ADR(arLinkedList),
  Size:= uiMAX_ARRAY_SIZE,
  Mode:= ModeGetHead,
  Index:= ,
  Done=> ,
```



```

        Error=> ,
        ErrorId =>,
        ErrorIdent=> ,
        CurrentIndex=> ,
        NextIndex=>);

(* Determine head *)
fbMyList(Execute:= TRUE,
        Link:= ADR(arLinkedList),
        Size:= uiMAX_ARRAY_SIZE,
        Mode:= ModeGetHead,
        Index:= ,
        Done=> ,
        Error=> ,
        ErrorId =>,
        ErrorIdent=> ,
        CurrentIndex=> udiIndex,
        NextIndex=>);

(* Check return value (-1 means error) *)
WHILE (udiIndex > 0) DO
    (* Return value in CurrentIndex contains index in list *)
    arMyData[udiIndex].StructureMember := MyValue;

    (* Reset "Execute" input again *)
    fbMyList(Execute:= FALSE,
            Link:= ADR(arLinkedList),
            Size:= uiMAX_ARRAY_SIZE,
            Mode:= ModeGetNext,
            Index:= ,
            Done=> ,
            Error=> ,
            ErrorId =>,
            ErrorIdent=> ,
            CurrentIndex=> ,
            NextIndex=>);

    (* Get next element*)
    fbMyList(Execute:= TRUE,
            Link:= ADR(arLinkedList),
            Size:= uiMAX_ARRAY_SIZE,
            Mode:= ModeGetNext,
            Index:= ,
            Done=> ,
            Error=> ,
            ErrorId =>,
            ErrorIdent=> ,
            CurrentIndex=> udiIndex,
            NextIndex=>);
END_WHILE

```

Belegung der Ausgänge in Abhängigkeit des "Mode"-Eingangs

Je nachdem wie der Eingang "Mode" belegt ist, werden der Eingang "Index" und die Ausgänge "CurrentIndex" und "NextIndex" belegt. Die nachfolgende Tabelle gibt die entsprechenden Werte an. Ein X steht dabei für einen Wert der unverändert gegenüber dem vorhergehenden Aufruf ist. Tritt ein Fehler auf, wird -1 für "CurrentIndex" zurückgegeben.

Mode	Index	CurrentIndex	NextIndex
ModeInitClear	X	-1	nächster freier Index
ModeAddTail	X	Index des Schwanzes	nächster freier Index
ModeAddHead	X	Index des Kopfes	nächster freier Index
ModeRemoveTail	X	Index des neuen Schwanzes	nächster freier Index
ModeRemoveHead	X	Index des neuen Kopfes	nächster freier Index
ModeRemoveIndex	Index zum Entfernen	Index des vorhergehenden Elementes	nächster freier Index
ModeGetHead	X	Index des Kopfes	X
ModeGetTail	X	Index des Schwanzes	X

Mode	Index	CurrentIndex	NextIndex
ModeGetNext	vorhergehender Index	Index des nachfolgenden Elementes	X
ModeGetPrev	nächster Index	Index des vorhergehenden Elementes	X
ModeGetEmptyIndex	X	X	nächster freier Index
ModeCheckIndex	Index für Prüfung	1: gefunden -1: nicht gefunden	-1

Tab. 4-137: Bausteinausgänge in Abhängigkeit von "Mode"

Fehlercodes Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR	16#1900	16#0	Eingang "Index" außerhalb gültiger Werte
INPUT_INVALID_ERROR	16#1901	16#0	Eingang "Link" ist ein Null Pointer
ACCESS_ERROR	16#1902	16#xxx	Ein Index ist immer noch in der Liste verlinkt, obwohl diese leer ist (interner Fehler)
ACCESS_ERROR	16#1903	16#0	Element in Liste soll freigegeben werden, aber Liste ist bereits leer
RESOURCE_ERROR	16#1904	16#0	Liste ist voll
STATE_MACHINE_ERROR	16#1905	16#0	Eingang "Mode" besitzt einen ungültigen Wert
RESOURCE_ERROR	16#1906	16#0	Eingang "Size" hat sich nach der Initialisierung noch einmal verändert (ggf. durch Online Change).

Tab. 4-138: Fehlercodes Funktionsbaustein IL_LinkedList

4.14.3 IL_LINKED_LIST_MAINTENANCE

Die Struktur wird durch den Funktionsbaustein [IL_LinkedList](#), Seite 190, verwendet werden.

"Prev" ist dabei ein Pointer auf das vorhergehende Listenelement, "Next" auf das nachfolgende. Ist der Wert Null, ist dies das Ende des Feldes, der Anfang oder ein unbenutztes Element.

Durch diese Struktur wird das Debuggen der Listen erleichtert.

Typdefinition

Programm:

```

TYPE IL_LINKED_LIST_MAINTENANCE :
STRUCT
    Prev: POINTER TO DINT;
    Next: POINTER TO DINT;
END_STRUCT
END_TYPE

```

4.14.4 IL_LINKED_LIST_MODE

Mit Hilfe des Aufzählungstyps bestimmt der Anwender, welche Aktion durchgeführt werden soll.

Die Aufzählung wird genutzt durch den Funktionsbaustein [IL_LinkedList](#), Seite 190.

Enumeration	Bedeutung
ModeInitClear	Die verkettete Liste wird initialisiert. Sämtliche Einträge werden gelöscht
ModeAddTail	Am Listenende wird ein Element eingefügt. Ist die Liste leer, wird das erste Element angelegt
ModeAddHead	Am Listenanfang wird ein Element eingefügt. Ist die Liste leer, wird das erste Element angelegt
ModeRemoveTail	Das letzte Element in der Liste wird entfernt
ModeRemoveHead	Das erste Element in der Liste wird entfernt
ModeRemoveIndex	Das Element an Index "Index" wird entfernt
ModeGetHead	Liefert den Index des Listenanfangs zurück
ModeGetTail	Liefert den Index des Listenendes zurück
ModeGetNext	Liefert das nachfolgende Element in der Liste zurück
ModeGetPrev	Liefert das vorhergehende Element in der Liste zurück
ModeGetEmptyIndex	Liefert den nächsten freien Index in der Liste zurück
ModeCheckIndex	Prüft, ob der angegebene Index in der verketteten Liste verwendet wird. Das Ergebnis wird in "CurrentIndex" zurückgegeben: 1: Index wird verwendet -1: Index wird nicht verwendet

Tab. 4-139: Enumerations und deren Bedeutung im Funktionsbaustein IL_LinkedList

4.15 Modulwert-Berechnung (Utilities/ Logic)

4.15.1 Einführung und Übersicht

Die nachfolgenden Funktionen berechnen einen Modulwert. Je nachdem welche Funktion gewählt wird, werden unterschiedliche Wertebereiche zugrunde gelegt.



Die genannten Funktionen haben beim Übergang zur Version 14 ihre Namen von MB_* nach IL_* geändert und sind von "ML_TechBase" nach "RIL_Uilities" übergegangen.

Anwendungsgebiete und Funktionen / Funktionsbausteine von RIL_Uilities

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine	Beschreibung
Modulwert-Berechnung (Utilities/ Logic)	
IL_GetModuloType01, Seite 196	berechnen einen Modulwert, Wertebereich liegt zwischen 0.0 und + Modulwert (REAL)
IL_GetModuloType02, Seite 197	berechnen einen Modulwert, Wertebereich liegt zwischen 0 und + Modulwert (DINT)
IL_GetModuloType03, Seite 198	berechnen einen Modulwert, Wertebereich liegt zwischen - Modulwert und + Modulwert (REAL)
IL_GetModuloType04, Seite 199	berechnen einen Modulwert., Wertebereich liegt zwischen - modulo / 2 und + modulo / 2 (REAL)

Tab. 4-140: RIL_Uilities, Modulwert-Berechnung

4.15.2 IL_GetModuloType01

Kurzbeschreibung Die Funktion liefert den Modulowert zurück. Der zurückgelieferte Wertebereich liegt zwischen 0.0 und + Modulowert.

 Die Funktion "MB_GetModuloType01 (ML_TechBase-Bibliothek)" wurde mit Version 14VRS in "IL_GetModuloType01" umbenannt.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek
MLC 15VRS	RIL_Uilities
MLD mit MPx18	MX_Technology07.lib

Tab. 4-141: Referenztabelle IL_GetModuloType01

Schnittstellenbeschreibung

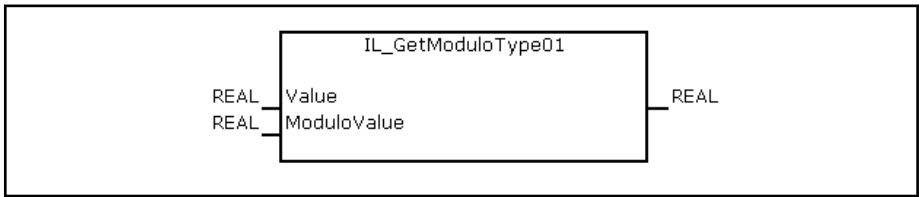


Abb. 4-144: Funktion IL_GetModuloType01

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Value	REAL	Wert an dem die Modulofunktion angewendet werden soll
	ModuloValue	REAL	Modulowert der für die Modulooperation verwendet werden soll
RETURN		REAL	Modulowert: der zurückgelieferte Wertebereich liegt zwischen 0.0 und + Modulowert

Tab. 4-142: Schnittstellenvariablen Funktion IL_GetModuloType01

Min.- Max.- und Default-Werte In der folgenden Tabelle sind die Min-/Max- und Default-Werte der Bausteineingänge aufgelistet.

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Value	REAL	- 1,175494e-38	3,402823e+38	0.0	mit jedem Aufruf
ModuloValue	REAL	- 1,175494e-38	3,402823e+38	360.0	mit jedem Aufruf

Tab. 4-143: Min-/Max- und Default-Werte Funktion IL_GetModuloType01

Funktionsbeschreibung Die Funktion liefert den Modulowert zurück. Der zurückgelieferte Wertebereich liegt zwischen 0.0 und + Modulowert. Positiver und negativer Grenzwert ist nicht Teil des Wertebereichs (siehe auch nachfolgende Grafik, in der auf der Abszisse der an die Funktion übergebene Wert und auf der Ordinate der von der Funktion zurückgelieferte Wert dargestellt sind).
Wird für "ModuloValue" ein Wert 0.0 übergeben, liefert die Funktion den Wert "Value" zurück.

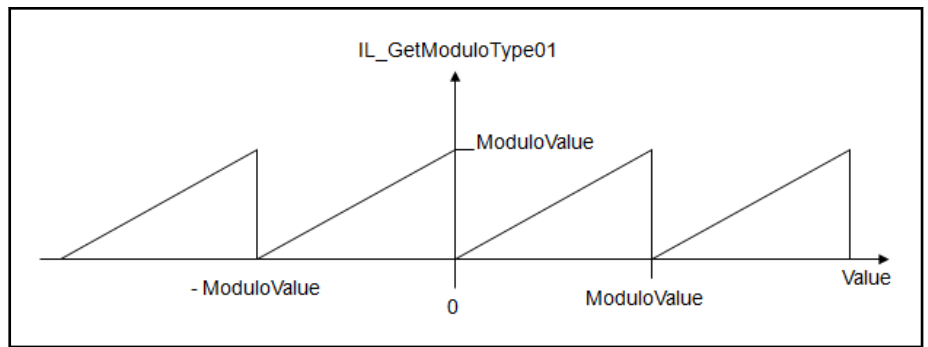


Abb. 4-145: Grafische Darstellung der Modulo-Funktion

Fehlerbehandlung Die Funktion generiert keine Fehler.

4.15.3 IL_GetModuloType02

Kurzbeschreibung Die Funktion liefert den Modulowert zurück. Der zurückgelieferte Wertebereich liegt zwischen 0 und + Modulowert.



Die Funktion "MB_GetModuloType02 (ML_TechBase-Bibliothek)" wurde mit Version 14VRS in "IL_GetModuloType02" umbenannt.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek
MLC 15VRS	RIL_Uilities
MLD mit MPx07 ohne MA Funktionspaket	MX_Technology_07.lib

Tab. 4-144: Referenztable IL_GetModuloType02

Schnittstellenbeschreibung

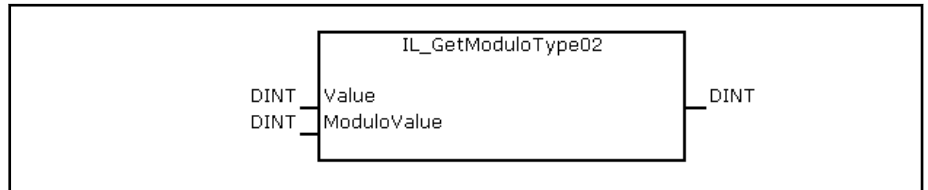


Abb. 4-146: Funktion IL_GetModuloType02

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Value	DINT	Wert, an dem die Modulofunktion angewendet werden soll
	ModuloValue	DINT	Modulowert, der für die Modulooperation verwendet werden soll
RETURN		DINT	Modulowert: der zurückgelieferte Wertebereich liegt zwischen 0 und + Modulowert

Tab. 4-145: Schnittstellenvariablen Funktion IL_GetModuloType02

Min.- Max.- und Default-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte der Bausteineingänge aufgelistet.

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Value	DINT	- 2147483648	2147483647	0	mit jedem Aufruf
ModuloValue	DINT	- 2147483648	2147483647	360	mit jedem Aufruf

Tab. 4-146: Min-/Max- und Default-Werte Funktion IL_GetModuloType02

Funktionsbeschreibung

Die Funktion liefert den Modulwert zurück. Der zurückgelieferte Wertebereich liegt zwischen 0 und + Modulwert. Positiver und negativer Grenzwert ist nicht Teil des Wertebereichs (siehe auch nachfolgende Grafik, in der auf der Abszisse der an die Funktion übergebene Wert und auf der Ordinate der von der Funktion zurückgelieferte Wert dargestellt sind).
Wird für "ModuloValue" ein Wert 0 übergeben, liefert die Funktion den Wert "Value" zurück.

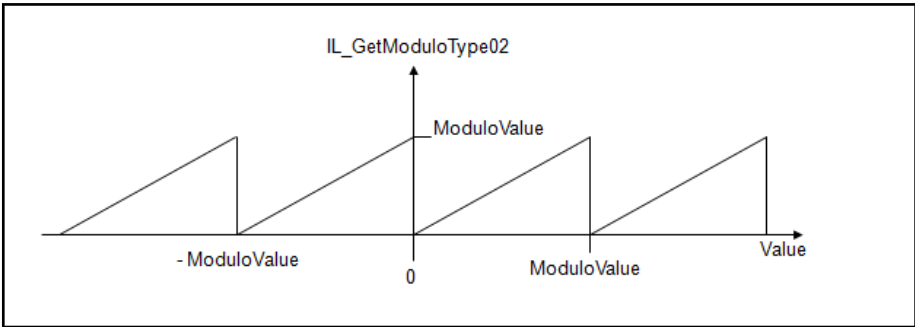


Abb. 4-147: Grafische Darstellung der Modulo-Funktion

Fehlerbehandlung

Die Funktion generiert keine Fehler.

4.15.4 IL_GetModuloType03

Kurzbeschreibung

Die Funktion liefert den Modulwert zurück. Der zurückgelieferte Wertebereich liegt zwischen - Modulwert und + Modulwert.

Die Funktion "MB_GetModuloType03 (ML_TechBase-Bibliothek)" wurde mit Version 14VRS in "IL_GetModuloType03" umbenannt.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek
MLC 15VRS	RIL_Uilities
MLD mit MPx18	MX_Technology07.lib

Tab. 4-147: Referenztable IL_GetModuloType03

Schnittstellenbeschreibung

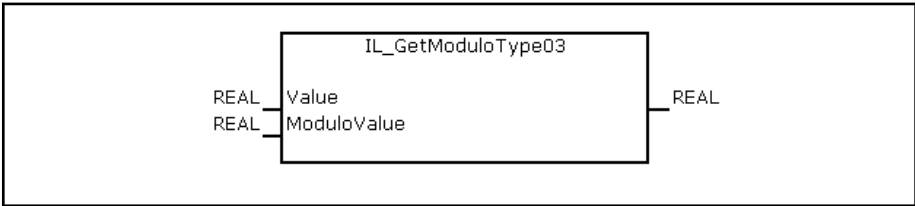


Abb. 4-148: Funktion IL_GetModuloType03

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Value	REAL	Wert an dem die Modulofunktion angewendet werden soll
	ModuloValue	REAL	Modulwert der für die Moduloperation verwendet werden soll
RETURN		REAL	Modulwert: der zurückgelieferte Wertebereich liegt zwischen - Modulwert und + Modulwert

Tab. 4-148: Schnittstellenvariablen Funktion IL_GetModuloType03

Min.- Max.- und Default-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte der Bausteineingänge aufgelistet.

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Value	REAL	- 1,175494e-38	3,402823e+38	0.0	mit jedem Aufruf
ModuloValue	REAL	- 1,175494e-38	3,402823e+38	360.0	mit jedem Aufruf

Tab. 4-149: Min-/Max- und Default-Werte Funktion IL_GetModuloType03

Funktionsbeschreibung

Die Funktion liefert den Modulowert zurück. Der zurückgelieferte Wertebereich liegt zwischen - Modulowert und + Modulowert. Positiver und negativer Grenzwert ist nicht Teil des Wertebereichs (siehe auch nachfolgende Grafik in der auf der Abszisse der an die Funktion übergebene Wert und auf der Ordinate der von der Funktion zurückgelieferte Wert dargestellt sind).

Wird für "ModuloValue" ein Wert 0.0 übergeben, liefert die Funktion den Wert "Value" zurück.

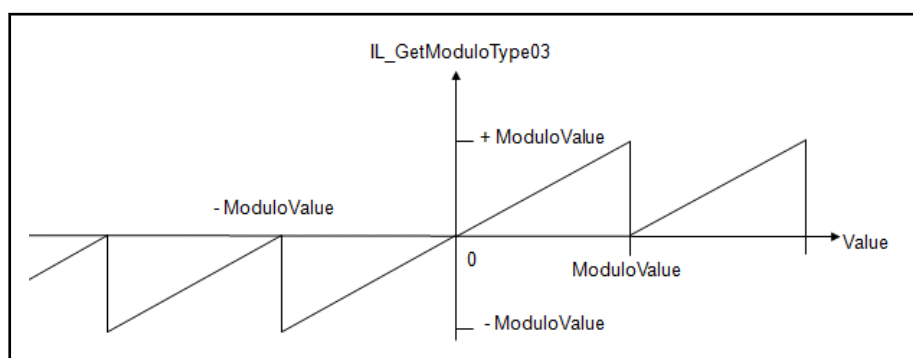


Abb. 4-149: Grafische Darstellung der Modulo-Funktion

Fehlerbehandlung

Die Funktion generiert keine Fehler.

4.15.5 IL_GetModuloType04

Kurzbeschreibung

Die Funktion liefert den Modulowert zurück. Der zurückgelieferte Wertebereich liegt zwischen - modulo / 2 und + modulo / 2.



Die Funktion "MB_GetModuloType04 (ML_TechBase-Bibliothek)" wurde mit Version 14VRS in "IL_GetModuloType04" umbenannt.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek
MLC 15VRS	RIL_Uilities
MLD mit MPx18	MX_Technology07.lib

Tab. 4-150: Referenztable IL_GetModuloType04

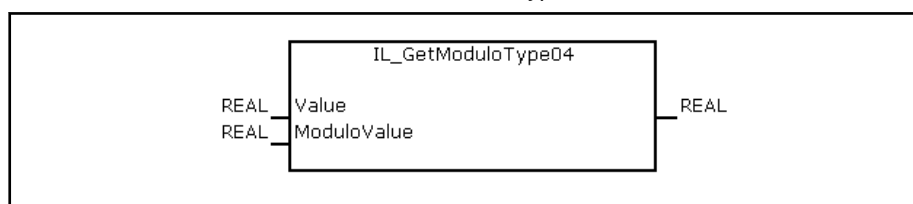
Schnittstellenbeschreibung

Abb. 4-150: Funktion IL_GetModuloType04

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Value	REAL	Wert an dem die Modulofunktion angewendet werden soll
	ModuloValue	REAL	Modulowert, der für die Modulooperation verwendet werden soll
RETURN		REAL	Modulowert: der zurückgelieferte Wertebereich liegt zwischen $- \text{modulo} / 2$ und $+ \text{modulo} / 2$

Tab. 4-151: Schnittstellenvariablen Funktion IL_GetModuloType04

Min.- Max.- und Default-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte der Bausteineingänge aufgelistet.

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Value	REAL	- 1,175494e-38	3,402823e+38	0.0	mit jedem Aufruf
ModuloValue	REAL	- 1,175494e-38	3,402823e+38	360.0	mit jedem Aufruf

Tab. 4-152: Min-/Max- und Default-Werte Funktion IL_GetModuloType04

Funktionsbeschreibung

Die Funktion liefert den Modulowert zurück. Der zurückgelieferte Wertebereich liegt zwischen $- 0.5 * \text{Modulowert}$ und $+ 0.5 * \text{Modulowert}$. Positiver und negativer Grenzwert sind Teil des Wertebereichs (siehe auch nachfolgende Grafik in der auf der Abszisse der an die Funktion übergebene Wert und auf der Ordinate der von der Funktion zurückgelieferte Wert dargestellt sind).

Wird für "ModuloValue" ein Wert 0.0 übergeben, liefert die Funktion den Wert "Value" zurück.

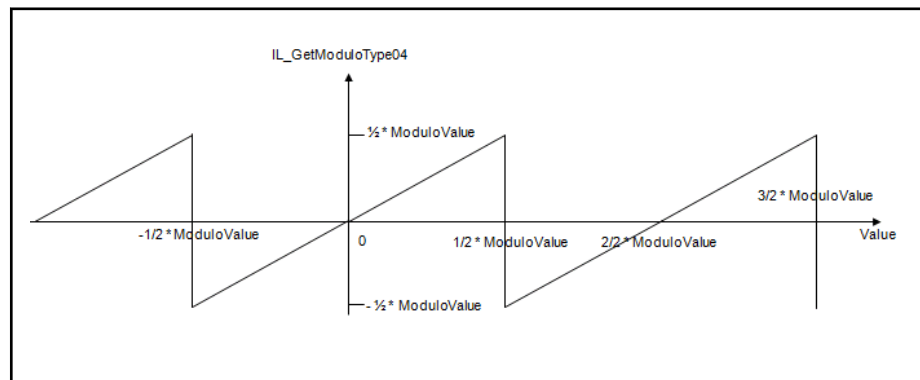


Abb. 4-151: Graphische Darstellung der Modulo-Funktion

Fehlerbehandlung

Die Funktion generiert keine Fehler.

5 RIL_ModbusTCP.library

5.1 Einführung und Übersicht



Für die Bibliothek ist das Basis-Funktionspaket PLC des Systems MLC 15VRS ausreichend.

Modbus TCP dient dem Datenaustausch zwischen Ethernet-Geräten, die das Modbus-Protokoll unterstützen. Hier sind insbesondere HMI-Geräte interessant. Die Steuerung bildet hierbei den Modbus-Slave (Server), auf den sich die Modbus-Master (Clients) verbinden. Ein Modbus TCP-Server ist jeweils in den Funktionsbausteinen [IL_ModbusTCPServer](#), Seite 216, und [IL_ModbusTCPServerType02](#), Seite 219, implementiert, die in der RIL_ModbusTCP.library enthalten sind.

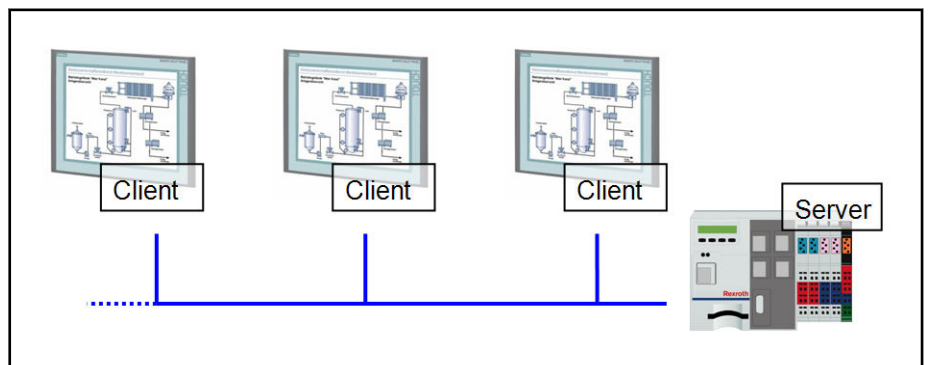


Abb. 5-1: Verbindung mehrerer Modbus-Clients an einen Modbus-Server



Die beiden Funktionsbausteine [IL_ModBusTCPServer](#) und [IL_ModBusTCPServerType02](#) unterscheiden sich wie folgt:

Entgegen der Modbus-Spezifikation sind beim Funktionsbaustein "IL_ModBusTCPServer" die Datenbereiche für "Coil Data" und "Discrete Input Data" unter "BitData", sowie "Holding Register" und "Input Register" unter "RegisterData" zusammengefasst.

Es wird empfohlen, den Funktionsbaustein [IL_ModbusTCPServerType02](#), Seite 219, zu verwenden.

Zielsystem

Die Bibliothek RIL_ModBusTCP wird vom System MLC ab Version 10VRS unterstützt.

ModbusTCP, Begriffe

- **Coils:** Coils bilden ein Bit-Feld mit einer maximalen Länge von 65536 Bits. Die Bits können sowohl gelesen, als auch geschrieben werden.
- **Discrete Input:** Discrete Inputs bilden ein Bit-Feld mit einer maximalen Länge von 65536 Bits. Die Bits können nur gelesen werden.
- **Holding Register:** Die Register bilden ein Feld aus Worten (16 Bit-Werten) mit einer maximalen Länge von 65536 Worten. Die Register können sowohl gelesen, als auch geschrieben werden.
- **Input Register:** Die Register bilden ein Feld aus Worten (16 Bit-Werten) mit einer maximalen Länge von 65536 Worten. Die Register können nur gelesen werden.

5.2 ModbusTCP Server

5.2.1 ModbusTCP Server, Anwendungen

Allgemein

Bei der Projektierung kann es sinnvoll sein, erste Tests mit Modbus-Test-Clients zu realisieren. Beispiele für frei im Internet erhältliche Test-Clients sind (Stand 09.2008):

- IPC@CHIP Modbus Client V1.4
- Modbus TCP/IP Request Tool Pro

In den folgenden Kapiteln wird auf die Details der Projektierung einer Modbus TCP Anbindung mit

- **Siemens WinCC-flexible 2007** ([Projektierung mit Siemens WinCC-flexible 2007 auf Seite 204](#)) und
- **Pro-face GP-Pro EX2.2** ([Projektierung mit Pro-face GP-Pro EX 2.2 auf Seite 208](#)) eingegangen.



Die Schnittstellen der Komponenten anderer Hersteller können Modifikationen unterworfen sein. Die Anwendungsbeispiele dienen daher primär dem Wissenserwerb. Bosch Rexroth übernimmt keine Garantie für die Korrektheit der Angaben.

Ablageort Beispielprojekte

Die Beispiele für die Visualisierungssysteme Siemens WinCC-flexible und Pro-face GP-Pro EX2.2 befinden sich im AddOn-Verzeichnis der IndraWorks Installations-DVD.

Adressierung der Daten aus dem SPS-Projekt

Im SPS-Projekt werden die Datenbereiche für Register-Daten und Bit-Daten mittels Pointer an die Instanz des Funktionsbausteins IL_ModbusTCPsServer übergeben.



Der Zugriff auf die Datenbereiche des Modbus TCP Server erfolgt über folgende Adressierung durch die Modbus-Clients

Register-Daten-Bereich:

Adressierung der "Holding Register"-Daten mittels **%MW** (lesend und schreibend) oder der "Input Register"-Daten mittels **%IW** (nur lesend).

Bit-Daten-Bereich:

Adressierung der "Coil"-Daten mittels **%M** (lesend und schreibend) oder der "Discret Input"-Daten mittels **%I** (nur lesend).

Bit-Daten

Der Bit-Daten-Bereich wurde im Testprojekt als Array of Byte angelegt, aus dem fortlaufend jedes einzelne Bit adressiert werden kann.



Für den Bit-Datenbereich kann kein ARRAY OF BOOL verwendet werden, da in IndraLogic eine Variable vom Typ BOOL 8 Bit Speicher belegt.

Die einzelnen Bits aus einem Byte können aber auf Variablen vom Typ BOOL kopiert werden (und umgekehrt)

Die Adresse einer Variablen aus dem Bit-Daten-Bereich entspricht der Bit-Nummer aus dem Byte-Array, das für die Bit-Daten verwendet wird.

Beispiel:

Bit-Daten (%M)

Adresse **%M0** entspricht der Variablen **BitData[0].0**

Adresse **%M7** entspricht der Variablen **BitData[0].7**

Adresse **%M8** entspricht der Variablen **BitData[1].0**

Adresse **%M12** entspricht der Variablen **BitData[1].4**

Register-Daten

Der Register-Daten-Bereich kann bis zu 65536 Worte groß sein und durch den Anwender beliebig organisiert werden.

Um eine bessere Übersicht über die Register-Daten zu erhalten, wurde im Testprojekt für den Register-Daten-Bereich eine Struktur verwendet, welche Arrays der verschiedenen SPS-Datentypen beinhaltet. Diese Struktur ist so aufgebaut, dass die einzelnen Datentypen-Bereiche immer 1000 Worte lang sind und somit an einer definierten Wort-Adresse beginnen.

Beispiel einer Datenstruktur für Register_Data

```

TYPE REGISTER_DATA :
STRUCT
  arWord:  ARRAY[0..999]OF WORD;      // %MW0000 - %MW0999
  arInt:   ARRAY[0..999]OF INT;       // %MW1000 - %MW1999
  arUInt:  ARRAY[0..999]OF UINT;      // %MW2000 - %MW2999
  arDint:  ARRAY[0..499]OF DINT;      // %MW3000 - %MW3999
  arUdint: ARRAY[0..499]OF UDINT;     // %MW4000 - %MW4999
  arReal:  ARRAY[0..499]OF REAL;      // %MW5000 - %MW5999
  arString: ARRAY[0..99]OF STRING(19); // %MW6000 - %MW6999
  // String(19): because 19 byte string + 1 byte end sign
  // together 7000 WORDs
END_STRUCT
END_TYPE

```

Text-String

Über den Register-Daten-Bereich können auch Texte (STRING) übertragen werden.



In IndraLogic beträgt die Länge eines STRING (in Byte) immer die angegebene Länge + 1 Byte Ende-Kennung

Um die STRING mit ganzer Wortlänge zu definieren, muss die Länge also immer mit einer ungeraden Byte-Anzahl angegeben werden

(z.B. String(19) => 19 Byte String + 1Byte Endekennung = 20 Byte = 10 Worte)

Die Adresse einer Variablen aus dem Register-Daten-Bereich ergibt sich aus dem Aufbau der Struktur, die für die Register-Daten verwendet wird.

Legt man die zuvor gezeigte Datenstruktur für **RegisterData** zu Grunde, ergeben sich folgende Adressen:

Beispiel:

Register-Daten (%MW)

Adresse **%MW0** entspricht der Variablen **RegisterData.arWord[0]**

Adresse **%MW1000** entspricht der Variablen **RegisterData.arInt[0]**

Adresse **%MW1001** entspricht der Variablen **RegisterData.arInt[1]**

Adresse **%MW5000** entspricht der Variablen **RegisterData.arReal[0]**

Adresse **%MW5002** entspricht der Variablen **RegisterData.arReal[1]**

Adresse **%MW6000** entspricht der Variablen **RegisterData.arString[0]**

Adresse **%MW6010** entspricht der Variablen **RegisterData.arString[1]**

SPS-Programm

Deklarationsteil

```

PROGRAM ModbusCom
// Run in a low priority task
VAR
fb_IL_ModbusTCPServerType02: IL_ModbusTCPServerType02; // Instance of FB_IL_ModbusTCPServerType02
bEnableModbusSrv:      BOOL := TRUE;           // Enable FB_IL_ModbusTCPServerType02
wPortNumber:           WORD := 502;           // Modbus server port - 502 is Modbus default
arbyBitData:           ARRAY[0..8191] OF BYTE; // Coil data area for FC2, FC5 and FC15
arbyDiscreteInputs:    ARRAY[0..8191] OF BYTE; // Discrete Input data area for FC1 (read only)
RegisterData:          REGISTER_DATA;         // Holding Register data area for FC3, FC6, FC16 and FC23
InputRegisters:        REGISTER_DATA;         // Input Register data area for FC4 (read only)
bInOperation:          BOOL;                  // FB_IL_ModbusTCPServer is in operation
bError:                BOOL;                  // FB_IL_ModbusTCPServer quits operation with error
ErrorID:               ERROR_CODE;            // general information about the error
ErrorIdent:            ERROR_STRUCT;          // detailed information about the error
Stats:                 IL_MODBUS_STATS;       // connection statistics
END_VAR

```

Implementationsteil

```

fb_IL_ModbusTCPServerType02(
    Enable := bEnableModbusSrv,
    Port := wPortNumber,
    CoilData:= ADR(arbyBitData),
    SizeOfCoilData:= SIZEOF(arbyBitData),
    InputData:= ADR(arbyDiscreteInputs),
    SizeOfInputData:= SIZEOF(arbyDiscreteInputs),
    HoldingRegisterData:= ADR(RegisterData),
    SizeOfHoldingRegisterData:= SIZEOF(RegisterData),
    InputRegisterData:= ADR(InputRegisters),
    SizeOfInputRegisterData:= SIZEOF(InputRegisters),
    InOperation => bInOperation,
    Error => bError,
    ErrorID => ErrorID,
    ErrorIdent => ErrorIdent,
    Stats => Stats);

```

Projektiertung mit Siemens WinCC-flexible 2007

Kommunikationseinstellungen

Zur Konfiguration der Verbindung in WinCC-flexible sind folgende Einstellungen notwendig:

1. Im Projektbaum des Bediengeräts das Menü **Communication** ► **Connections** öffnen
2. Name der Verbindung (und optional Kommentar) eingeben,
Communication driver "**Modicon MODBUS TCP/IP**" auswählen
3. Im darauf erscheinenden Dialog müssen nun die Kommunikationsparameter konfiguriert werden

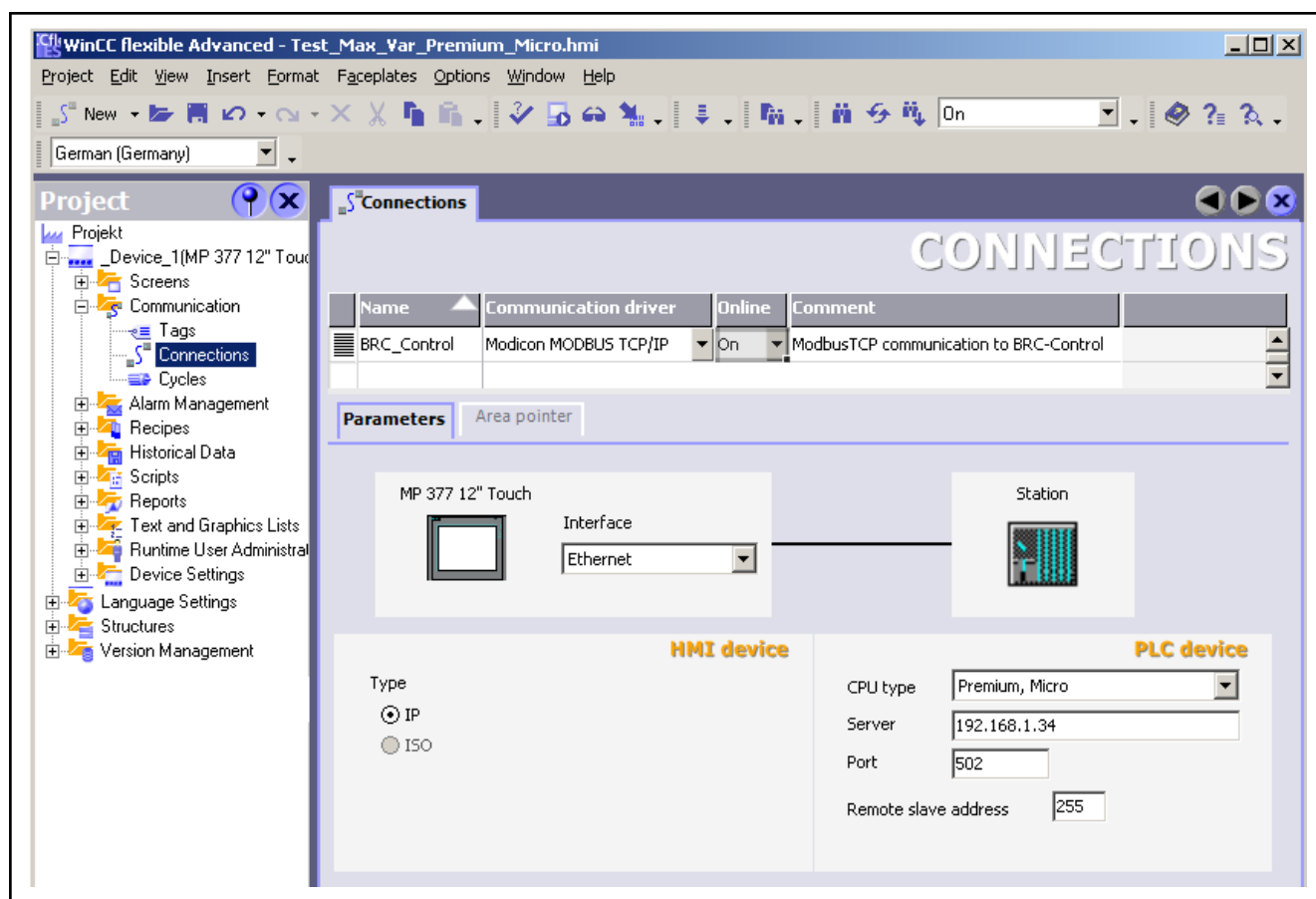


Abb. 5-2: WinCC-flexible Modbus TCP Kommunikationsparameter

Sektion	Name	Wert	Beschreibung
HMI device	Interface	Ethernet	Auswahl der Kommunikationsschnittstelle
	Type	IP	Nur IP unterstützt
PLC device	CPU type	Premium, Micro	Schneider Modicon Steuerungstyp der ohne Adressoffset und mit üblicher Bit-Zählweise arbeitet (siehe Siemens Benutzerhandbuch "WinCC-flexible 2007 Kommunikation Teil 2 - Kapitel 6")
	Server	"192.168.1.34"	IP - Adresse der Steuerung
	Port	"502"	Port der Kommunikationsverbindung (Port der Instanz des Funktionsbausteins IL_ModbusTCPsServer)
	Remote slave address	"255"	Standardeinstellung beibehalten (255) ! für Modbus TCP nicht relevant - siehe "MODBUS Messaging on TCP/IP Implementation Guide V1.0b" unter www.Modbus.org

Tab. 5-1: WinCC-flexible Modbus TCP Kommunikationsparameter

Variablendeklaration

Nachdem die Verbindung konfiguriert wurde, können nun die Variablen deklariert werden.

Hierzu muss im Projektbaum des Bediengeräts das Menü **Communication / Tags** geöffnet werden. Im darauf erscheinenden Dialog können nun die benötigten Variablen angelegt werden.

Beispiel einer Variablendeklaration in WinCC-flexible 2007:

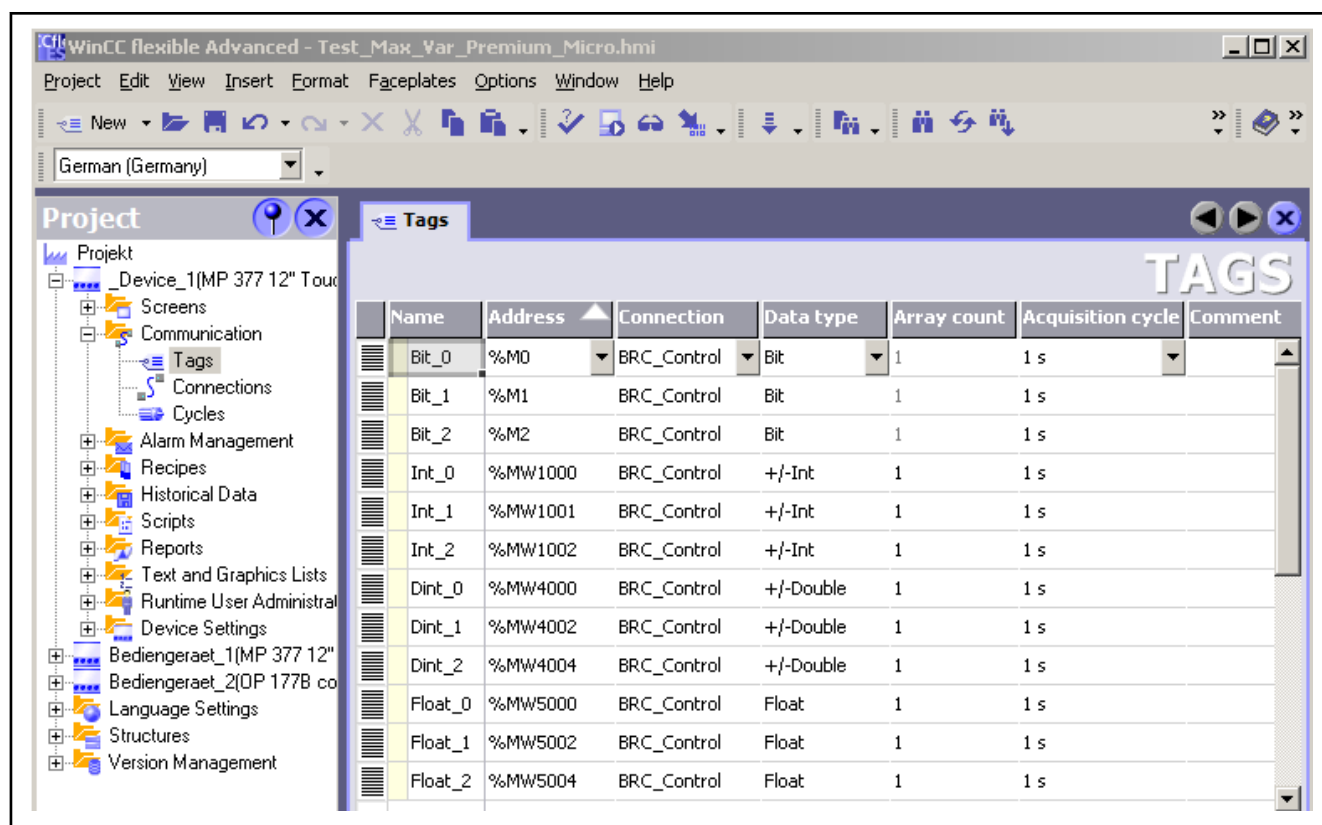


Abb. 5-3: WinCC-flexible Variablendeklaration - Beispiel

WinCC - flexible Variablendeklaration

Eingabefeld	Beispiel	Beschreibung
Name	"Bit_0"	Name der Variable
Address	"%M0"	Modbus Adresse der Variable ("Adressierung der Daten aus dem SPS-Projekt" auf Seite 202)
Connection	"BRC_Control"	Name der Verbindung
Data type	"Bit"	WinCC-flexible Datentyp der Variable (Tab. 5-3 "WinCC-flexible Datentypen / IndraLogic Datentypen" auf Seite 207)
Array count	"1"	Anzahl der Array-Elemente falls die Variable als Array angelegt werden soll
Acquisition cycle	"1s"	Aktualisierungsrate in der der Wert der Variablen aus der Steuerung gelesen werden soll

Tab. 5-2: WinCC - flexible Variablendeklaration

In WinCC-flexible 2007 stehen verschiedene Datentypen zur Verfügung. Die entsprechenden IndraLogic-Datentypen sind nachfolgend aufgelistet.

WinCC - flexible Datentypen

WinCC - flexible	IndraLogic	Beschreibung
Int	UINT	Unsigned Integer Länge 16 Bit Wertebereich: 0 bis 65535
+/-Int	INT	Signed Integer Länge 16 Bit Wertebereich: -32768 bis 32767
Double	UDINT	Unsigned Double Integer Länge 32 Bit Wertebereich: 0 bis 4294967295
+/-Double	DINT	Signed Double Integer Länge 32 Bit Wertebereich: -2147483648 bis 2147483647
Float	REAL	Floating Point Länge 32 Bit Wertebereich: 1.175494351e-38F bis 3.402823466e+38F
Bit	einzelnes Bit eines Byte	Single Bit Länge 1 Bit ("Bit-Daten" auf Seite 202)
16 Bit Group	WORD	WORD Länge 16 Bit Achtung: Bei 16-Bit-Gruppen ist die Bitzählweise in WinCC-flexible umgekehrt zur Zählweise in IndraLogic
ASCII	STRING	Die Länge des Strings (in Byte) muss in WinCC-flexible über die Tag Properties konfiguriert werden. ("Text-String" auf Seite 203)

Tab. 5-3: WinCC-flexible Datentypen / IndraLogic Datentypen

Die deklarierten Variablen können nun für die verschiedenen Funktionalitäten von WinCC-flexible verwendet werden.

Beispiel:

Anzeige und Eingabe von verschiedenen Variablen

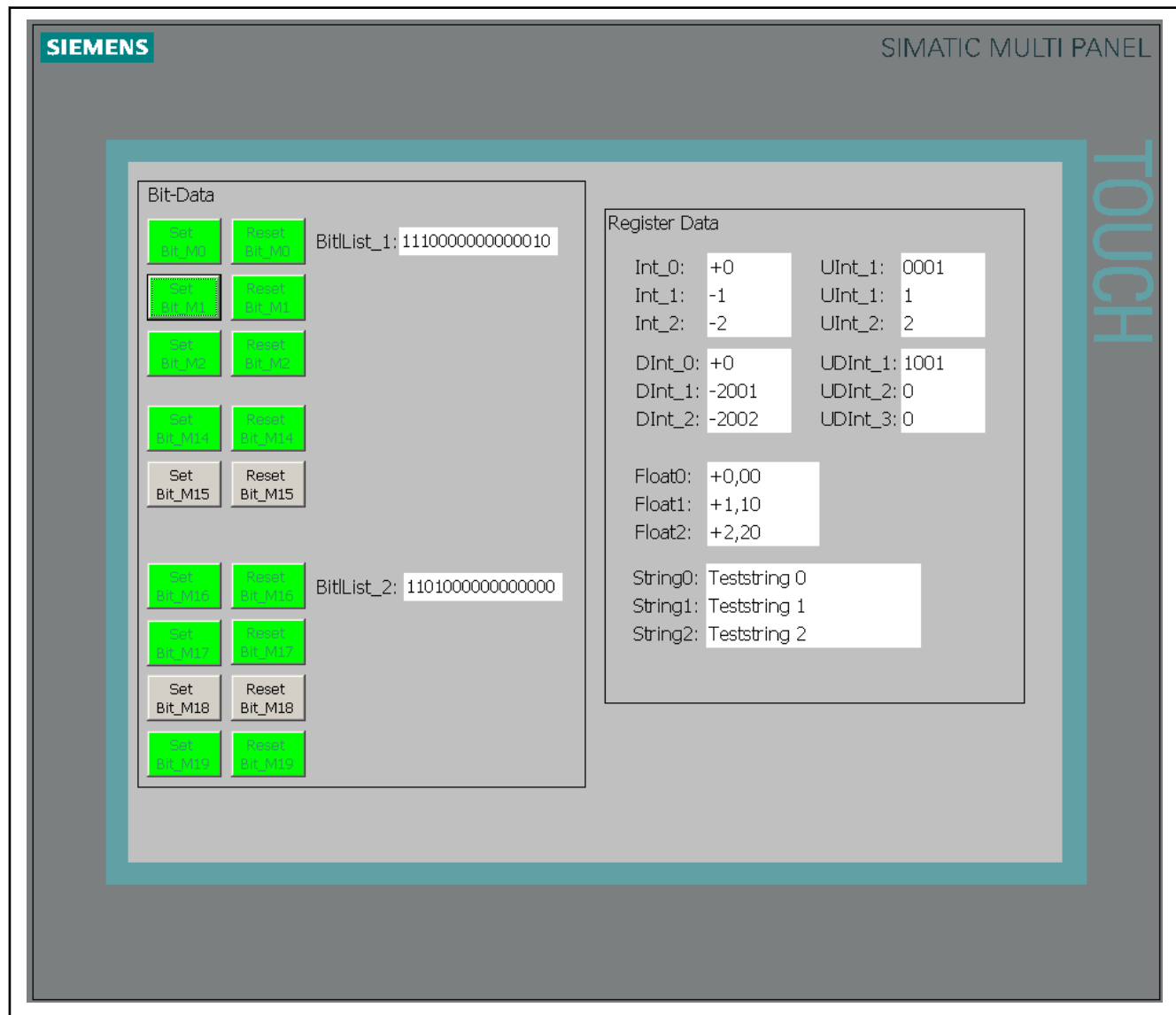


Abb. 5-4: WinCC-flexible - Visualisierungsbeispiel

Projektierung mit Pro-face GP-Pro EX 2.2

Kommunikationseinstellungen

Zur Konfiguration der Verbindung in Pro-face GP-Pro EX 2.2 sind folgende Einstellungen notwendig:

1. Im Menü **System Settings** den Eintrag **"Device/PLC"** wählen
2. Im Fenster des Arbeitsbereichs den Link **"Add Device/PLC"** ausführen
3. Im darauf erscheinenden Dialog **"Add Device/PLC"** die folgenden Einstellungen vornehmen:

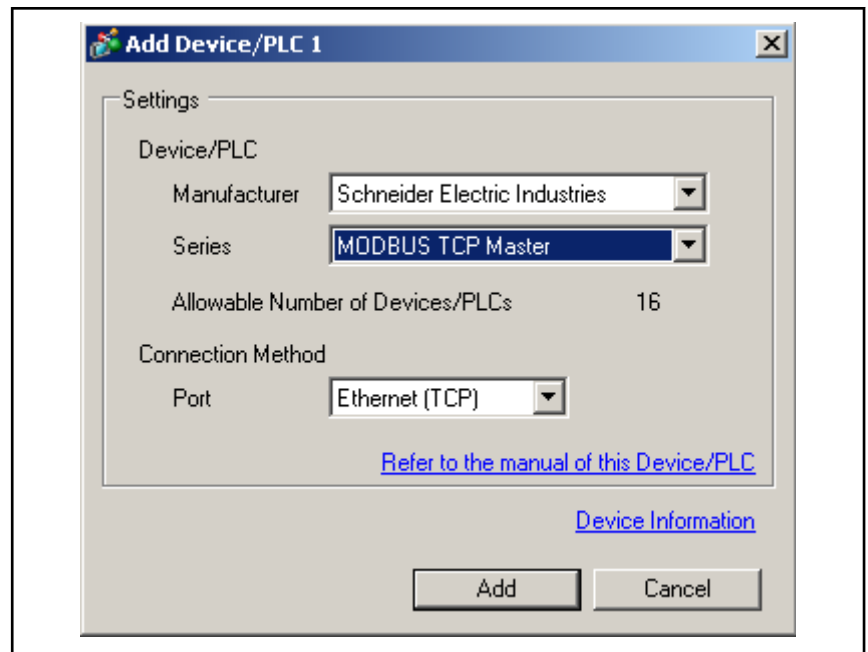


Abb. 5-5: GP-Pro EX Kommunikationseinstellungen - Geräteauswahl

Auswahlfeld	Auswahl
Manufacturer	Schneider Electric Industries
Series	MODBUS TCP Master
Port	Ethernet (TCP)

Tab. 5-4: Kommunikationseinstellungen - Geräteauswahl

Auswahl mit <Add> bestätigen

4. Im nun erweiterten Fenster des Arbeitsbereichs unter "Device Name" den Namen für die Verbindung eintragen und den Button  "Settings" betätigen

Device/PLC

[Add Device/PLC](#) [Delete Device/PLC](#)

Device/PLC 1

Summary [Change Device/PLC](#)

Manufacturer Series Port

Text Data Mode [Change](#)

Communication Settings

Timeout (sec)

Retry

Wait To Send (ms)

Device-Specific Settings

Allowable Number of Devices/PLCs 16

Number	Device Name	Settings
1 1	BRC_Control	Settings IP Address=000.000.000.000,Port No.=502,Unit ID=255,Rest of the bits in this

Abb. 5-6: GP-Pro EX Kommunikationseinstellungen - Arbeitsbereich

- Im anschließend erscheinenden Dialog "Individual Device Settings" im Register "Equipment Configuration" die folgenden Einstellungen vornehmen:

Abb. 5-7: GP-Pro EX Kommunikationseinstellungen - Geräteparameter

Sektion	Name	Wert	Beschreibung
Equipment Address	IP Address	"192.168.1.34"	IP - Adresse der Steuerung
	Type	"502"	Port der Kommunikationsverbindung (Port der Instanz des Funktionsbausteins IL_ModbusTCPsServer)
	Unit ID	"255"	Standardeinstellung beibehalten (255) ! für Modbus TCP nicht relevant - siehe "MODBUS Messaging on TCP/IP Implementation Guide V1.0b" unter www.Modbus-IDA.org
Bit manipulation to Holding Register	Reset of the bits in this word	"Do not clear"	Standardeinstellung beibehalten (Do not clear)

Sektion	Name	Wert	Beschreibung
IEC61131 Syntax	Address Mode	"0-based (Default)"	IEC61131 Syntax aktivieren und für Address Mode die Standardeinstellung beibehalten (0-based)
Variables	Double Word word order	"Low word first (L/H)"	Anordnung der Worte innerhalb eines Doppelwortes

Tab. 5-5: GP-Pro EX Kommunikationseinstellungen - Geräteparameter

Im Register "MaxQuery" kann die maximale Anzahl der Daten, die in eine Modbus-Anfrage gepackt werden, definiert werden. Hier können die Standardeinstellungen beibehalten werden.

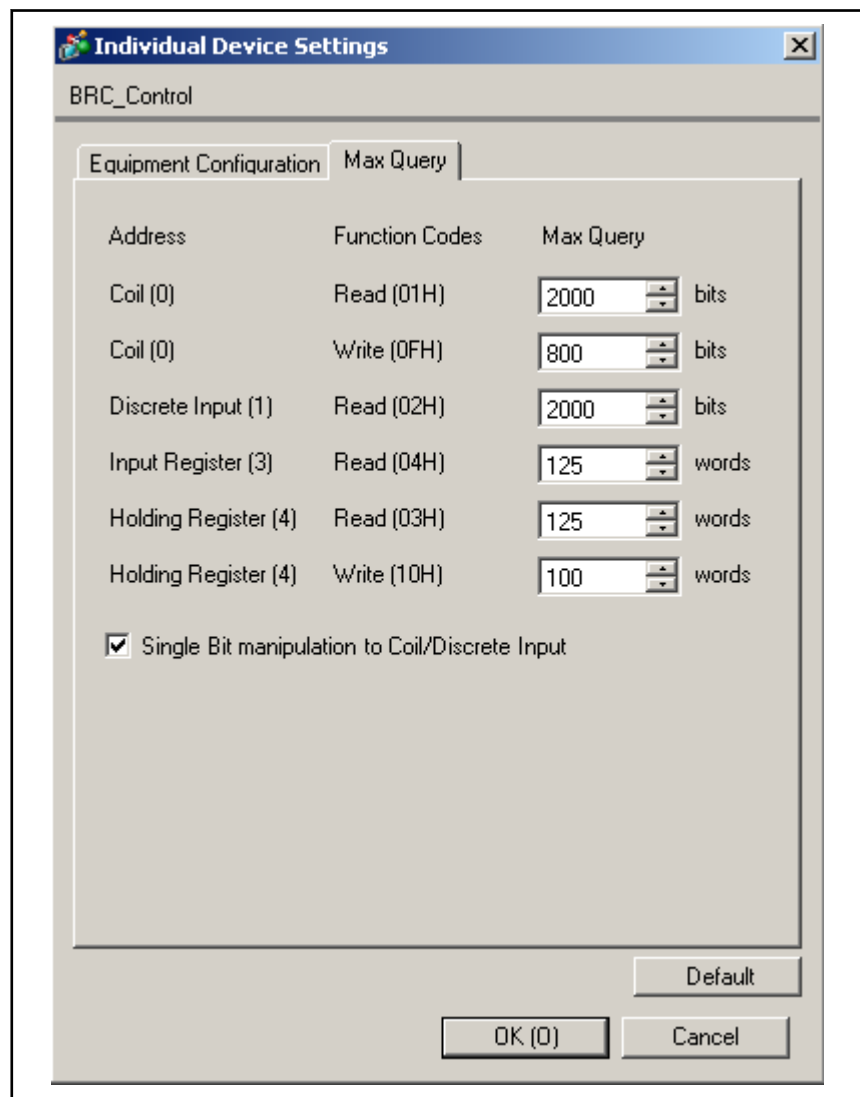


Abb. 5-8: GP-Pro EX Kommunikationseinstellungen - Geräteparameter
Auswahl mit <OK> bestätigen.

- Nun muss noch der "Text Data Mode" zur Interpretation von Texten angepasst werden. Hierzu im Fenster des Arbeitsbereichs den Link "Change" betätigen.

Device/PLC

[Add Device/PLC](#) [Delete Device/PLC](#)

Device/PLC 1

Summary [Change Device/PLC](#)

Manufacturer: Schneider Electric Industries Series: MODBUS TCP Master Port: Ethernet (TCP)

Text Data Mode: 2 [Change](#)

Communication Settings

Timeout: 3 (sec)

Retry: 0

Wait To Send: 0 (ms) [Default](#)

Device-Specific Settings

Allowable Number of Devices/PLCs: 16

Number	Device Name	Settings
1	BRC_Control	IP Address=000.000.000.000,Port No.=502,Unit ID=255,Rest of the bits in this

Abb. 5-9: GP-Pro EX Kommunikationseinstellungen - Arbeitsbereich

Im darauf erscheinenden Dialog "Change Text Data Mode" die Auswahl 2 selektieren und mit dem Button <Change> bestätigen.

Change Text Data Mode

Select a text data mode from the following list.

Data in Device Addresses	Bytes LH/HL storage	Double-word LH/HL storage	Text Data Mode	Select
Store from Top Data	LH Order	LH Order	4	☐
		HL Order	2	☒
	HL Order	LH Order	5	☐
		HL Order	1	☐
Store from Last Data	LH Order	LH Order	6	☐
		HL Order	7	☐
	HL Order	LH Order	8	☐
		HL Order	9	☐

[Change](#) [Cancel](#)

Abb. 5-10: GP-Pro EX Kommunikationseinstellungen - Text Data Mode

Verwendung der Adressen

Nachdem die Verbindung konfiguriert wurde, können nun in den verschiedenen Funktionalitäten von GP-Pro EX die Adressen direkt verwendet werden, siehe ["Adressierung der Daten aus dem SPS-Projekt"](#) auf Seite 202.

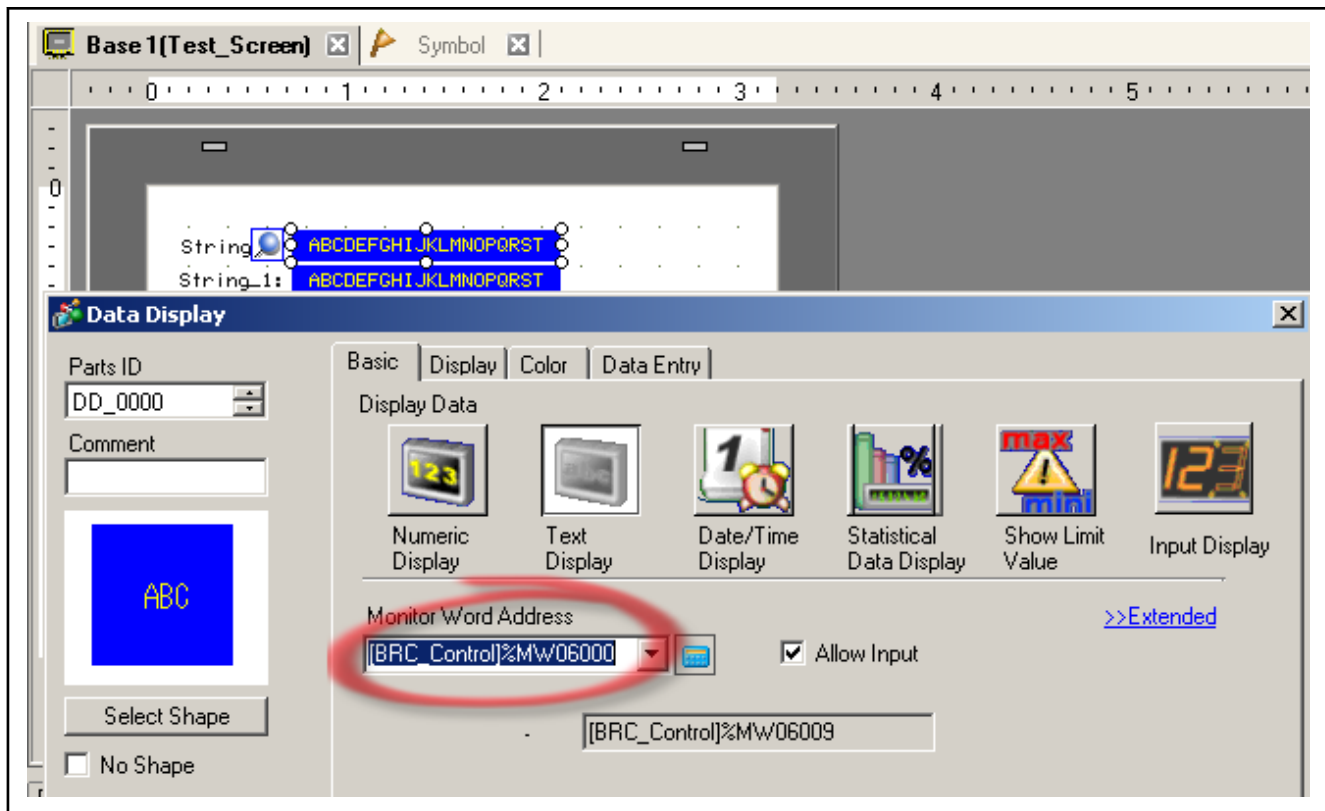


Abb. 5-11: GP-Pro EX - direkte Verwendung der Adressen

Variablendeklaration

Alternativ besteht auch die Möglichkeit symbolische Variablen vom Typ Bitaddress und Wordaddress zu deklarieren, siehe ["Adressierung der Daten aus dem SPS-Projekt" auf Seite 202](#). Über diese symbolischen Variablen können dann die Inhalte der Adressen in den verschiedenen Funktionalitäten von GP-Pro EX verwendet werden.

Über das Menü **Common Settings** ► **Symbol Variables** wird im Arbeitsbereich das Fenster zur Deklaration von Symbol Variablen geöffnet.

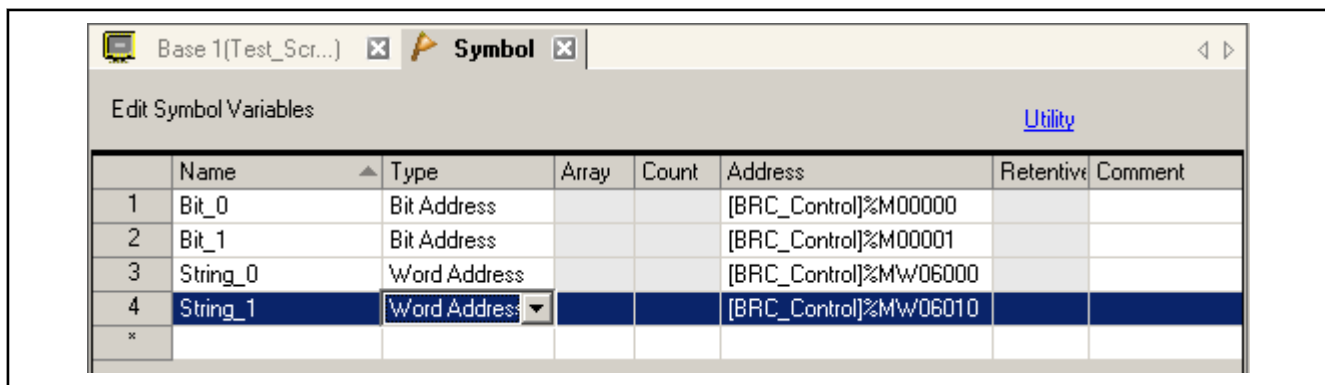


Abb. 5-12: GP-Pro EX - Symbol Variable anlegen

Nachdem die Symbol-Variablen angelegt wurden kann diese als "Platzhalter" für die Adresse verwendet werden.



Abb. 5-13: GP-Pro EX - Verwendung der Symbol Variablen



Die Anzahl der zu interpretierenden Register wird über den Datentyp bestimmt, der in den einzelnen Anzeigefeldern verwendet wird.

Bei Anzeigefeldern vom Typ "Text Display" wird die Anzahl der zu interpretierenden Register über die Anzahl darzustellender Zeichen bestimmt.

Beispiel:

Anzeige und Eingabe von verschiedenen Variablen

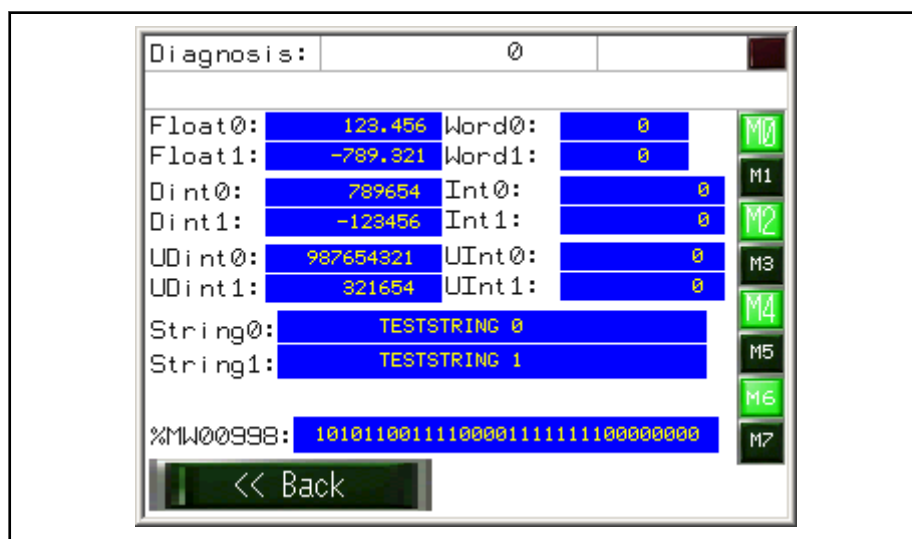


Abb. 5-14:

5.2.2 IL_ModbusTCPServer

Kurzbeschreibung Der Funktionsbaustein IL_ModbusTCPServer implementiert einen Modbus TCP-Slave. Es werden Discrete Inputs, Coils, Input Register und Holding-Register lesend und schreibend unterstützt. Der Datenaustausch erfolgt über Arrays.

Zielsystem	Bibliothek / Ordner
MLC 15VRS Basis-Funktionspaket PLC	RIL_ModbusTcp / Server

Tab. 5-6: Bibliothekszuordnung

Schnittstellenbeschreibung

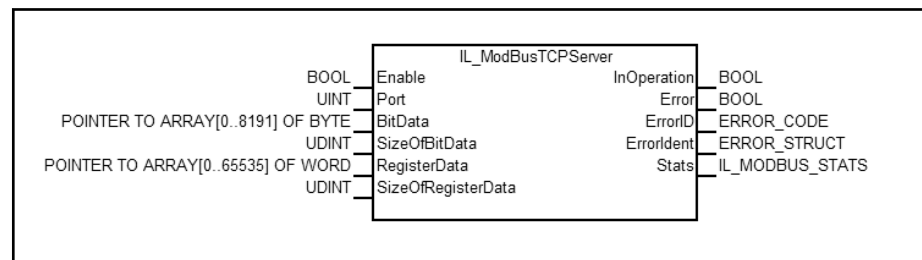


Abb. 5-15: Funktionsbaustein IL_ModbusTCPServer

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (pegelgesteuert).
	Port	UINT	TCP-Port, auf dem auf eingehende Verbindungen gewartet wird.
	BitData	POINTER TO ARRAY[0..8191] OF BYTE	Zeiger auf die Bit-Daten.
	SizeOfBitData	UDINT	Anzahl der Bytes in BitData (Bitanzahl / 8).
	RegisterData	POINTER TO ARRAY[0..65535] OF WORD	Zeiger auf Register-Daten in Wortbreite.
	SizeOfRegisterData	UDINT	Anzahl der Bytes in den Register-Daten (Registeranzahl * 2).
VAR OUTPUT	InOperation	BOOL	Der Modbus-Server läuft und nimmt eingehende Verbindungen an.
	Error	BOOL	Bei der Bearbeitung des Bausteins ist ein Fehler aufgetreten.
	ErrorID	ERROR_CODE	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang eine grobe Klassifizierung des Fehlers.
	ErrorIdent	ERROR_STRUCT	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang detaillierte Informationen über den Fehler. Siehe auch "Fehlerbehandlung" auf Seite 219 .
	Stats		Enthält Zähler für Datenpakete und Fehler. IL_MODBUS_STATS, Seite 225

Tab. 5-7: Schnittstelle IL_ModbusTCPServer

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Port	UINT	0	65535	502	Steigende Flanke an "Enable"
BitData	POINTER	0	n.def.	0	Steigende Flanke an "Enable"
SizeOfBitData	UDINT	0	8191	0	Steigende Flanke an "Enable"
RegisterData	POINTER	0	n.def.	0	Steigende Flanke an "Enable"
SizeOfRegisterData	UDINT	0	131070	0	Steigende Flanke an "Enable"

Tab. 5-8: Eingangsverhalten IL_ModbusTCPServer

Funktionsbeschreibung

Der Funktionsbaustein IL_ModbusTCPServer implementiert einen Modbus-Slave, auf den sich bis zu 16 Modbus-Master verbinden können. Der Baustein verhält sich im Datenaustausch passiv, d.h. der Modbus-Master bestimmt, welche Daten gelesen oder geschrieben werden. Es werden Discrete Inputs, Coils, Input Register und Holding-Register unterstützt.



Entgegen der Modbus-Spezifikation werden jeweils Discrete-Inputs und Coils sowie Input-Register und Holding-Register auf denselben Datenbereich abgebildet.

Versuchen mehr als 16 Teilnehmer gleichzeitig zu verbinden, so wird entsprechend der Modbus TCP-Spezifikation die Verbindung, deren Datenaustausch am längsten zurück liegt, getrennt, bevor eine neue Verbindung angenommen wird. Dieser Zustand sollte in der Praxis vermieden werden. In diesem Fall ist es sinnvoll, einen zweiten Modbus TCP-Server auf einem anderen Port zu öffnen.

Enable, InOperation

Der "Enable" ist die Bearbeitungsfreigabe des Bausteins. Solange "Enable" TRUE ist, ist der Baustein aktiv. Bei steigender Flanke an "Enable" werden alle Eingänge übernommen und der Baustein wird initialisiert. Nach der Initialisierung wird der Ausgang "InOperation" gesetzt, um anzuzeigen, dass eingehende Modbus TCP-Verbindungen angenommen und bearbeitet werden können. Bei fallender Flanke an "Enable" werden alle bestehenden Verbindungen geschlossen und der Port freigegeben.

Error, ErrorID, ErrorIdent

Tritt während der Abarbeitung des Funktionsbausteins ein Fehler auf, so wird der Ausgang "Error" gesetzt. "ErrorID" und "ErrorIdent" enthalten genauere Informationen über den aufgetretenen Fehler. Die Bearbeitung des Bausteins wird angehalten. Der Ausgang "Error" wird bei fallender Flanke an "Enable" zurückgesetzt.

Die Fehlerausgänge werden nur bei Fehlern bei der Abarbeitung des Funktionsbausteins gesetzt. Fehler, die vom Modbus TCP-Protokoll verursacht werden, erzeugen keine Ausgabe an den Fehlerausgängen. Hierfür wird das Feld "NumExceptions" im "Stats"-Ausgang verwendet.

Port

Modbus TCP verwendet normalerweise den Port 502. Es ist jedoch möglich, einen Modbus TCP-Server auf einem anderen Port zu öffnen. Hierfür muss vor dem Aktivieren des Funktionsbausteins die zu verwendende Portnummer auf den Eingang "Port" geschrieben werden.

BitData, SizeOfBitData

Der Eingang "BitData" bildet die in Modbus definierten Discrete Inputs und Coils ab. Coils bilden ein Bit-Feld mit einer maximalen Länge von 65536 Bits. Die Bits können sowohl gelesen, als auch geschrieben werden. Der Datenaustausch erfolgt über das am Eingang "BitData" angelegte Array. Jedes Byte des Arrays repräsentiert 8 Bits.

Die Größe des Arrays in Byte muss am Eingang "SizeOfBitData" angelegt werden. Hierfür kann der SIZEOF()-Operator verwendet werden. Der Funktionsbaustein überprüft anhand von "SizeOfBitData", ob eingehende Anfragen innerhalb des angelegten Feldes liegen.



Es ist sinnvoll, eine eigene Datenstruktur für den Zugriff auf die Bits zu definieren. Das Pragma `{bitaccess}` innerhalb IndraLogic 1.x, oder `{bit}` innerhalb IndraLogic 2G ermöglicht den Zugriff auf einzelne Bits. Für Details siehe Anwendungsbeschreibung zur Projektkonvertierung [DOK-MLC***-PRO-CONV*V15-APRS](#), Kapitel **Bit-Zugriff**.

RegisterData, SizeOfRegisterData

Der Eingang "RegisterData" bildet die in Modbus definierten Input-Register und Holding-Register ab. Die Register bilden ein Feld aus Wörtern (16 Bit-Werten) mit einer maximalen Länge von 65536 Wörtern. Der Datenaustausch erfolgt über das am Eingang "RegisterData" angelegte Array.

Die Größe des Arrays in Byte muss am Eingang "SizeOfRegisterData" angelegt werden. Hierfür kann der SIZEOF()-Operator verwendet werden. Der Funktionsbaustein überprüft anhand von "SizeOfRegisterData", ob eingehende Anfragen innerhalb des angelegten Feldes liegen.



Es ist sinnvoll eine eigene Datenstruktur für den Zugriff auf die Register zu definieren. Hierbei ist die Ausrichtung auf Wortgrenzen (16 Bit) zu achten. Siehe [Kap. 5.2.1 "ModbusTCP Server, Anwendungen" auf Seite 202](#).

Stats

Der Ausgang "Stats" beinhaltet eine Datenstruktur, in der Status- und Statistikeninformationen über die laufende Kommunikation und die IP Adressen der verbundenen Clients angezeigt werden. Die Lese- und Schreibzugriffe werden in den entsprechenden Feldern gezählt. Protokollfehler werden in Feld "NumExceptions" gezählt. Läuft ein Zähler über, so wird er wieder bei 0 gestartet. Der Maximalwert eines Zählers beträgt $2^{32}-1$.

Unterstützte Modbus Function Codes

Function Codes werden vom Modbus-Master verwendet um auf den Modbus-Slave zuzugreifen. Die Kenntnis der unterstützten Codes dient der Fehlersuche und wird für den normalen Betrieb nicht benötigt. Die folgenden Function Codes werden unterstützt:

Function Code	Name	Beschreibung
1	Read Coils	Liest eine Anzahl Bits aus "BitData" ab einer Startadresse
2	Read Discrete Inputs	Liest eine Anzahl Bits aus "BitData" ab einer Startadresse
3	Read Holding Registers	Liest eine Anzahl Register aus "RegisterData" ab einer Startadresse
4	Read Input Registers	Liest eine Anzahl Register aus "RegisterData" ab einer Startadresse
5	Write Single Coil	Schreibt ein Bit in "BitData" ab einer Startadresse
6	Write Single Register	Schreibt ein Register in "RegisterData" ab einer Startadresse
15	Write Multiple Coils	Schreibt eine Anzahl Bits in "BitData" ab einer Startadresse

Function Code	Name	Beschreibung
16	Write Multiple Registers	Schreibt eine Anzahl Register in "RegisterData" ab einer Startadresse
23	Read/Write Multiple Registers	Führt eine kombinierte Lese/Schreib-Operation auf "RegisterData" aus
43/14	Read Device Identification	Liest die Standard-Geräteinformationen des ModbusTCP Server aus: - VendorName: "Bosch Rexroth" - ProductCode: "IL_ModbusTCPServer" - MajorMinorRevision: "01Vxx"

Tab. 5-9: Unterstützte Modbus Function Codes



Die Funktionscodes 1, 2, 5, 15 greifen auf den Speicherbereich zu, der am Eingang "BitData" angelegt wird.



Die Funktionscodes 3, 4, 6, 16 greifen auf den Speicherbereich zu, der am Eingang "RegisterData" angelegt wird.

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle F_RELATED_TABLE, 16#170x. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#1800	16#0000	Der Eingang "BitData" ist ein Null-Zeiger.
INPUT_INVALID_ERROR (16#0001)	16#1801	16#0000	Der Eingang "RegisterData" ist ein Null-Zeiger.
INPUT_INVALID_ERROR (16#0001)	16#1802	16#0000	Der Eingang "SizeOfBitData" ist 0.
INPUT_INVALID_ERROR (16#0001)	16#1803	16#0000	Der Eingang "SizeOfRegisterData" ist 0.
INPUT_INVALID_ERROR (16#0001)	16#1804	16#0000	Der Eingang "SizeOfRegisterData" ist ungerade.
RESOURCE_ERROR (16#0003)	16#1805	16#0000	Konnte Socket nicht öffnen.
RESOURCE_ERROR (16#0003)	16#1806	16#0000	Konnte die eingehende Verbindung nicht annehmen.
RESOURCE_ERROR (16#0003)	16#1807	16#0000	Der Port ist belegt.
RESOURCE_ERROR (16#0003)	16#1808	16#0000	Port Listen ist fehlgeschlagen.

Tab. 5-10: IL_ModbusTCPServer Fehlercodes

5.2.3 IL_ModbusTCPServerType02

Kurzbeschreibung

Der Funktionsbaustein IL_ModbusTCPServerType02 implementiert einen Modbus TCP-Slave. Es werden Discrete Inputs, Coils, Input Register und Holding-Register lesend und schreibend unterstützt. Der Datenaustausch erfolgt über 4 separate Arrays für die einzelnen Datenbereiche.

Zielsystem	Bibliothek / Ordner
MLC 15VRS Basis-Funktionspaket PLC	RIL_ModbusTcp / Server

Tab. 5-11: Bibliothekszuordnung

Schnittstellenbeschreibung

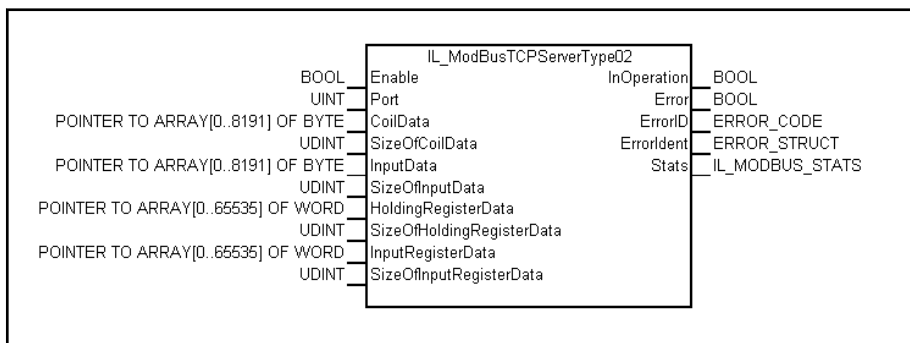


Abb. 5-16: Funktionsbaustein IL_ModbusTCPServerType02

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (pegel-gesteuert).
	Port	UINT	TCP-Port, auf dem auf eingehende Verbindungen gewartet wird.
	CoilData	POINTER TO AR- RAY[0..8191] OF BYTE	Zeiger auf die Coil-Daten (Bit-Datenbereich, auf den lesend und schreibend zugegriffen werden kann).
	SizeOfCoilData	UDINT	Anzahl der Bytes in CoilData (Bitanzahl / 8).
	InputData	POINTER TO AR- RAY[0..8191] OF BYTE	Zeiger auf die Discrete-Input-Daten (Bit-Datenbe- reich, auf den nur lesend zugegriffen werden kann).
	SizeOfInputData	UDINT	Anzahl der Bytes in InputData (Bitanzahl / 8).
	HoldingRegisterData	POINTER TO AR- RAY[0..65535] OF WORD	Zeiger auf Holding-Register-Daten in Wortbreite (Re- gister-Datenbereich, auf den lesend und schreibend zugegriffen werden kann).
	SizeOfHoldingRegister- Data	UDINT	Anzahl der Bytes in den Holding-Register-Daten (Re- gisteranzahl * 2).
	InputRegisterData	POINTER TO AR- RAY[0..65535] OF WORD	Zeiger auf Input-Register-Daten in Wortbreite (Regis- ter-Datenbereich, auf den nur lesend zugegriffen werden kann).
	SizeOfInputRegister- Data	UDINT	Anzahl der Bytes in den Input-Register-Daten (Regis- teranzahl * 2).

	Name	Typ	Beschreibung
VAR OUTPUT	InOperation	BOOL	Der Modbus-Server läuft und nimmt eingehende Verbindungen an.
	Error	BOOL	Bei der Bearbeitung des Bausteins ist ein Fehler aufgetreten.
	ErrorID	ERROR_CODE	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang eine grobe Klassifizierung des Fehlers.
	ErrorIdent	ERROR_STRUCT	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang detaillierte Informationen über den Fehler. Siehe auch "Fehlerbehandlung" auf Seite 224 .
	Stats		Enthält Zähler für Datenpakete und Fehler. IL_MODBUS_STATS, Seite 225

Tab. 5-12: Schnittstelle IL_ModbusTCPServerType02

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Port	UINT	0	65535	502	Steigende Flanke an "Enable"
CoilData	POINTER	0	n.def.	0	Steigende Flanke an "Enable"
SizeOfCoilData	UDINT	0	8191	0	Steigende Flanke an "Enable"
InputData	POINTER	0	n.def.	0	Steigende Flanke an "Enable"
SizeOfInputData	UDINT	0	8191	0	Steigende Flanke an "Enable"
HoldingRegisterData	POINTER	0	n.def.	0	Steigende Flanke an "Enable"
SizeOfHoldingRegisterData	UDINT	0	131070	0	Steigende Flanke an "Enable"
InputRegisterData	POINTER	0	n.def.	0	Steigende Flanke an "Enable"
SizeOfInputRegisterData	UDINT	0	131070	0	Steigende Flanke an "Enable"

Tab. 5-13: Eingangsverhalten IL_ModbusTCPServerType02

Name	Typ	Default-Wert	Beschreibung
NumReadBits	UDINT	0	Anzahl der Lesezugriffe auf Bits
NumReadRegisters	UDINT	0	Anzahl der Lesezugriffe auf Register
NumWriteBit	UDINT	0	Anzahl der Einzelschreibzugriffe auf Bits
NumWriteRegister	UDINT	0	Anzahl der Einzelschreibzugriffe auf Register
NumWriteBits	UDINT	0	Anzahl der Mehrfachschreibzugriffe auf Bits

Name	Typ	Default-Wert	Beschreibung
NumWriteRegisters	UDINT	0	Anzahl der Mehrfachschreibzugriffe auf Register
NumReadWriteRegs	UDINT	0	Anzahl der kombinierten Lese-/Schreibzugriffe auf Register
NumExceptions	UDINT	0	Anzahl der Modbus-Fehler
ConnectedClients	ARRAY[0..15] OF String(16)		IP-Adressen der verbundenen Clients

Tab. 5-14: Datenstruktur IL_MODBUS_STATS

Funktionsbeschreibung

Der Funktionsbaustein IL_ModbusTCPServerType02 implementiert einen Modbus-Slave, auf den sich bis zu 16 Modbus-Master verbinden können. Der Baustein verhält sich im Datenaustausch passiv, d.h. der Modbus-Master (Client) bestimmt, welche Daten gelesen oder geschrieben werden. Es werden Discrete Inputs, Coils, Input Register und Holding-Register unterstützt. Entsprechend der Modbus-Spezifikation werden Discrete-Inputs, Coils, Input-Register und Holding-Register auf separate Datenbereiche abgebildet.

Versuchen mehr als 16 Teilnehmer gleichzeitig zu verbinden, so wird entsprechend der Modbus TCP-Spezifikation die Verbindung, deren Datenaustausch am längsten zurück liegt, getrennt, bevor eine neue Verbindung angenommen wird. Dieser Zustand sollte in der Praxis vermieden werden. In diesem Fall ist es sinnvoll, einen zweiten Modbus TCP-Server auf einem anderen Port zu öffnen.

Enable, InOperation

Der "Enable" ist die Bearbeitungsfreigabe des Bausteins. Solange "Enable" TRUE ist, ist der Baustein aktiv. Bei steigender Flanke an "Enable" werden alle Eingänge übernommen und der Baustein wird initialisiert. Nach der Initialisierung wird der Ausgang "InOperation" gesetzt, um anzuzeigen, dass eingehende Modbus TCP-Verbindungen angenommen und bearbeitet werden können. Bei fallender Flanke an "Enable" werden alle bestehenden Verbindungen geschlossen und der Port freigegeben.

Error, ErrorID, ErrorIdent

Tritt während der Abarbeitung des Funktionsbausteins ein Fehler auf, so wird der Ausgang "Error" gesetzt. "ErrorID" und "ErrorIdent" enthalten genauere Informationen über den aufgetretenen Fehler. Die Bearbeitung des Bausteins wird angehalten. Der Ausgang "Error" wird bei fallender Flanke an "Enable" zurückgesetzt.

Die Fehlerausgänge werden nur bei Fehlern bei der Abarbeitung des Funktionsbausteins gesetzt. Fehler, die vom Modbus TCP-Protokoll verursacht werden, erzeugen keine Ausgabe an den Fehlerausgängen. Hierfür wird das Feld "NumExceptions" im "Stats"-Ausgang verwendet.

Port

Modbus TCP verwendet normalerweise den Port 502. Es ist jedoch möglich, einen Modbus TCP-Server auf einem anderen Port zu öffnen. Hierfür muss vor dem Aktivieren des Funktionsbausteins die zu verwendende Portnummer auf den Eingang "Port" geschrieben werden.

CoilData, SizeOfCoilData

Der Eingang "CoilData" bildet die in Modbus definierten Coils ab. Coils bilden ein Bit-Feld mit einer maximalen Länge von 65536 Bits. Die Bits können sowohl gelesen, als auch geschrieben werden. Der Datenaustausch erfolgt über das am Eingang "CoilData" angelegte Array. Jedes Byte des Arrays repräsentiert 8 Bits.

Die Größe des Arrays in Byte muss am Eingang "SizeOfCoilData" angelegt werden. Hierfür kann der SIZEOF()-Operator verwendet werden. Der Funktionsbaustein überprüft anhand von "SizeOfCoilData", ob eingehende Anfragen innerhalb des angelegten Feldes liegen.

	 Es ist sinnvoll, eine eigene Datenstruktur für den Zugriff auf die Bits zu definieren. Das Pragma {bitaccess } ermöglicht den Zugriff auf einzelne Bits.
InputData, SizeOfInputData	Der Eingang "InputData" bildet die in Modbus definierten Discrete Inputs ab. Discrete Inputs bilden ein Bit-Feld mit einer maximalen Länge von 65536 Bits. Die Bits können nur gelesen werden. Der Datenaustausch erfolgt über das am Eingang "InputData" angelegte Array. Jedes Byte des Arrays repräsentiert 8 Bits. Die Größe des Arrays in Byte muss am Eingang "SizeOfInputData" angelegt werden. Hierfür kann der SIZEOF()-Operator verwendet werden. Der Funktionsbaustein überprüft anhand von "SizeOfInputData", ob eingehende Anfragen innerhalb des angelegten Feldes liegen.
	 Es ist sinnvoll, eine eigene Datenstruktur für den Zugriff auf die Bits zu definieren. Das Pragma {bitaccess } ermöglicht den Zugriff auf einzelne Bits.
HoldingRegisterData, SizeOfHoldingRegisterData	Der Eingang "HoldingRegisterData" bildet die in Modbus definierten Holding-Register ab. Die Register bilden ein Feld aus Wörtern (16 Bit-Werten) mit einer maximalen Länge von 65536 Wörtern. Die Register können sowohl gelesen, als auch geschrieben werden. Der Datenaustausch erfolgt über das am Eingang "HoldingRegisterData" angelegte Array. Die Größe des Arrays in Byte muss am Eingang "SizeOfHoldingRegisterData" angelegt werden. Hierfür kann der SIZEOF()-Operator verwendet werden. Der Funktionsbaustein überprüft anhand von "SizeOfHoldingRegisterData", ob eingehende Anfragen innerhalb des angelegten Feldes liegen.
	 Es ist sinnvoll, eine eigene Datenstruktur für den Zugriff auf die Register zu definieren. Hierbei ist die Ausrichtung auf Wortgrenzen (16 Bit) zu achten. Siehe "Beispiel einer Datenstruktur für Register_Data" auf Seite 203.
InputRegisterData, SizeOfInputRegisterData	Der Eingang "InputRegisterData" bildet die in Modbus definierten Input-Register ab. Die Register bilden ein Feld aus Wörtern (16 Bit-Werten) mit einer maximalen Länge von 65536 Wörtern. Die Register können nur gelesen werden. Der Datenaustausch erfolgt über das am Eingang "InputRegisterData" angelegte Array. Die Größe des Arrays in Byte muss am Eingang "SizeOfInputRegisterData" angelegt werden. Hierfür kann der SIZEOF()-Operator verwendet werden. Der Funktionsbaustein überprüft anhand von "SizeOfInputRegisterData", ob eingehende Anfragen innerhalb des angelegten Feldes liegen.
	 Es ist sinnvoll, eine eigene Datenstruktur für den Zugriff auf die Register zu definieren. Hierbei ist die Ausrichtung auf Wortgrenzen (16 Bit) zu achten. Siehe "Beispiel einer Datenstruktur für Register_Data" auf Seite 203.
Stats	Der Ausgang "Stats" beinhaltet eine Datenstruktur, in der Status- und Statistikinformationen über die laufende Kommunikation und die IP Adressen der verbundenen Clients angezeigt werden. Die Lese- und Schreibzugriffe werden in den entsprechenden Feldern gezählt. Protokollfehler werden in Feld "NumExceptions" gezählt. Läuft ein Zähler über, so wird er wieder bei 0 gestartet. Der Maximalwert eines Zählers beträgt $2^{32}-1$.
Unterstützte Modbus Function Codes	Function Codes werden vom Modbus-Master (Client) verwendet um auf den Modbus-Slave (Server) zuzugreifen. Die Kenntnis der unterstützten Codes

dient der Fehlersuche und wird für den normalen Betrieb nicht benötigt. Die folgenden Function Codes werden unterstützt:

Function Code	Name	Beschreibung
1	Read Coils	Liest eine Anzahl Bits aus "CoilData" ab einer Startadresse
2	Read Discrete Inputs	Liest eine Anzahl Bits aus "InputData" ab einer Startadresse
3	Read Holding Registers	Liest eine Anzahl Register aus "HoldingRegisterData" ab einer Startadresse
4	Read Input Registers	Liest eine Anzahl Register aus "InputRegisterData" ab einer Startadresse
5	Write Single Coil	Schreibt ein Bit in "CoilData" ab einer Startadresse
6	Write Single Register	Schreibt ein Register in "HoldingRegisterData" ab einer Startadresse
15	Write Multiple Coils	Schreibt eine Anzahl Bits in "CoilData" ab einer Startadresse
16	Write Multiple Registers	Schreibt eine Anzahl Register in "HoldingRegisterData" ab einer Startadresse
23	Read/Write Multiple Registers	Führt eine kombinierte Lese/Schreib-Operation auf "HoldingRegisterData" aus
43/14	Read Device Identification	Liest die Standard Geräte-Informationen des Modbus TCP Server aus: • VendorName: "Bosch Rexroth" • ProductCode: "IL_ModbusTCPServerType02" • MajorMinorRevision: "01Vxx"

Tab. 5-15: Unterstützte Modbus Function Codes



Die Funktionscodes 1, 5, 15 greifen auf den Speicherbereich zu, der am Eingang "CoilData" angelegt wird.



Der Funktionscode 2 greift auf den Speicherbereich zu, der am Eingang "InputData" angelegt wird.



Die Funktionscodes 3, 6, 16 greifen auf den Speicherbereich zu, der am Eingang "HoldingRegisterData" angelegt wird.



Der Funktionscode 4 greift auf den Speicherbereich zu, der am Eingang "InputRegisterData" angelegt wird.

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle F_RELATED_TABLE, 16#170x. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00001800	16#00000000	Der Eingang "CoilData" ist ein Null-Zeiger.
		16#00000001	Der Eingang "InputData" ist ein Null-Zeiger.

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00001801	16#00000000	Der Eingang "HoldingRegisterData" ist ein Null-Zeiger.
		16#00000001	Der Eingang "InputRegisterData" ist ein Null-Zeiger.
INPUT_INVALID_ERROR (16#0001)	16#00001802	16#00000000	Der Eingang "SizeOfCoilData" ist 0.
		16#00000001	Der Eingang "SizeOfInputData" ist 0.
INPUT_INVALID_ERROR (16#0001)	16#00001803	16#00000000	Der Eingang "SizeOfHoldingRegisterData" ist 0.
		16#00000001	Der Eingang "SizeOfInputRegisterData" ist 0.
INPUT_INVALID_ERROR (16#0001)	16#00001804	16#00000000	Der Eingang "SizeOfHoldingRegisterData" ist ungerade.
		16#00000001	Der Eingang "SizeOfInputRegisterData" ist ungerade.
RESOURCE_ERROR (16#0003)	16#00001805	16#00000000	Konnte Socket nicht öffnen.
RESOURCE_ERROR (16#0003)	16#00001806	16#00000000	Konnte die eingehende Verbindung nicht annehmen.
RESOURCE_ERROR (16#0003)	16#00001807	16#00000000	Der Port ist belegt.
RESOURCE_ERROR (16#0003)	16#00001808	16#00000000	Port Listen ist fehlgeschlagen.

Tab. 5-16: IL_ModbusTCPTyp02 Fehlercodes

5.2.4 IL_MODBUS_STATS

Die Struktur IL_MODBUS_STATS enthält Status- und Statistikinformationen über die laufende Kommunikation und die IP Adressen der verbundenen Clients. Die Lese- und Schreibzugriffe werden in den entsprechenden Feldern gezählt. Protokollfehler werden in Feld "NumExceptions" gezählt. Läuft ein Zähler über, so wird er wieder bei 0 gestartet. Der Maximalwert eines Zählers beträgt $2^{32}-1$.

Name	Typ	Default-Wert	Beschreibung
NumReadBits	UDINT	0	Anzahl der Lesezugriffe auf Bits
NumReadRegisters	UDINT	0	Anzahl der Lesezugriffe auf Register
NumWriteBit	UDINT	0	Anzahl der Einzelschreibzugriffe auf Bits
NumWriteRegister	UDINT	0	Anzahl der Einzelschreibzugriffe auf Register
NumWriteBits	UDINT	0	Anzahl der Mehrfachschreibzugriffe auf Bits
NumWriteRegisters	UDINT	0	Anzahl der Mehrfachschreibzugriffe auf Register
NumReadWriteRegs	UDINT	0	Anzahl der kombinierten Lese-/Schreibzugriffe auf Register
NumExceptions	UDINT	0	Anzahl der Modbus-Fehler
ConnectedClients	ARRAY[0..15] OF String(16)		IP-Adressen der verbundenen Clients

Tab. 5-17: Datenstruktur IL_MODBUS_STATS

5.3 ModbusTCP Client

5.3.1 Einführung und Übersicht

Die Funktionsbausteine zum ModbusTCP-Client dienen dem Datenaustausch mit einem ModbusTCP-Server. Der Kommunikationspartner kann ein I/O-Gerät oder eine andere Steuerung sein.

Im nachfolgenden Beispiel sollen zyklisch Daten mit einer anderen Steuerung ausgetauscht werden.

Hierfür werden zwei Datenbereiche (Array of Word) definiert.

Zusätzlich wird noch je eine Instanz der Funktionsbausteine IL_ModbusTCPClient und IL_ModbusTCPReadWriteMultipleRegisters benötigt.

Programm:

```
// IP address of ModbusTCP server
strServerIP: STRING := '192.168.42.10';

// Data arrays
arOutData: ARRAY[100..199] OF WORD;
arInData: ARRAY[500..599] OF WORD;

// Function blocks
fbModBusTCPClient: IL_ModBusTCPClient;
fbModBusTCPReadWriteRegisters: IL_ModBusTCPReadWriteMultipleRegisters;
```

Der Funktionsbaustein IL_ModbusTCPClient dient dem Verbindungsaufbau zum ModbusTCP-Server.

Er tauscht noch keine Daten aus. Dies wird von den anderen Client-Bausteinen erledigt, die mit einer Instanz des Funktionsbausteins IL_ModbusTCPClient über den "ModbusLink" in Verbindung stehen.

Programm:

```
fbModBusTCPClient (Enable:=TRUE, Address:=strServerIP);

fbModBusTCPReadWriteRegisters (
  Enable := fbModBusTCPClient.InOperation,
  FirstReadRegister := 100,
  ReadQuantity := 100,
  ReadDataAddr := ADR(arInData),
  SizeOfReadData := SIZEOF(arInData),
  FirstWriteRegister := 500,
  WriteQuantity := 100,
  WriteDataAddr := ADR(arOutData),
  SizeOfWriteData := SIZEOF(arOutData)
);
```

Dieses stark vereinfachte Beispielprogramm baut eine ModbusTCP-Verbindung zu einem in strServerIP angegebenen ModbusTCP-Server auf.

Anschließend werden zyklisch die Holding-Register 100 bis 199 gelesen und die Holding-Register 500 bis 599 geschrieben. Dieses Beispiel enthält noch keine Fehlerbehandlung.

5.3.2 IL_ModbusTCPClient

Kurzbeschreibung Der Funktionsbaustein IL_ModbusTCPClient implementiert einen Modbus TCP-Master.

Er dient dem Aufbau einer Verbindung zu einem ModbusTCP-Server.

Die eigentlichen ModbusTCP-Anfragen werden von weiteren Funktionsbausteinen implementiert.

Zielsystem	Bibliothek / Ordner
MLC 15VRS Basis-Funktionspaket PLC	RIL_ModbusTcp / Client

Tab. 5-18: Bibliothekszuordnung

Schnittstellenbeschreibung

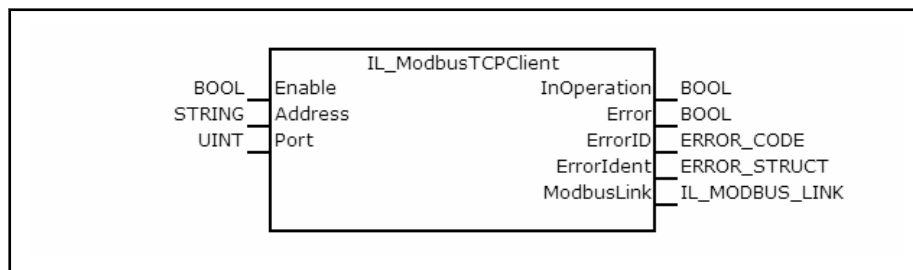


Abb. 5-17: Funktionsbaustein IL_ModbusTCPClient

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (pegelgesteuert).
	Address	STRING	IP-Adresse zu der verbunden werden soll.
	Port	UINT	TCP-Port des ModbusTCP-Servers.
VAR OUTPUT	InOperation	BOOL	Der Modbus-Server läuft und nimmt eingehende Verbindungen an.
	Error	BOOL	Bei der Bearbeitung des Bausteins ist ein Fehler aufgetreten.
	ErrorID	ERROR_CODE	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang eine grobe Klassifizierung des Fehlers.
	ErrorIdent	ERROR_STRUCT	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang detaillierte Informationen über den Fehler. Siehe auch "Fehlerbehandlung" auf Seite 228 .
	ModbusLink	IL_MODBUS_LINK	Handle auf die bestehende Modbus-Verbindung, wird von den Client-Funktionsbausteinen benötigt

Tab. 5-19: Schnittstelle IL_ModbusTCPClient

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Address	STRING			"	Steigende Flanke an "Enable"
Port	UINT	0	65535	502	Steigende Flanke an "Enable"
UnitID	USINT	0	255	255	Steigende Flanke an "Enable"

Tab. 5-20: Eingangsverhalten IL_ModbusTCPClient

Funktionsbeschreibung

Der Funktionsbaustein IL_ModbusTCPClient implementiert einen Modbus TCP-Master.

Er dient dem Aufbau einer Verbindung zu einem ModbusTCP-Server.

Die eigentlichen ModbusTCP-Anfragen werden von weiteren Funktionsbausteinen implementiert.

Enable, InOperation Bei steigender Flanke an "Enable" baut der Funktionsbaustein eine Verbindung zum in "Address" und "Port" angegebenen ModbusTCP-Server auf. Nach erfolgreichem Verbindungsaufbau wird der Ausgang "InOperation" gesetzt. Der Ausgang ModbusLink kann dann als Eingang für die ModbusTCP-Client Funktionsbausteine verwendet werden. Bei einer fallenden Flanke an "Enable" wird die Verbindung zum ModbusTCP-Server geschlossen.

Error, ErrorID, ErrorIdent Tritt während der Abarbeitung des Funktionsbausteins ein Fehler auf, so wird der Ausgang "Error" gesetzt. "ErrorID" und "ErrorIdent" enthalten genauere Informationen über den aufgetretenen Fehler. Die Bearbeitung des Bausteins wird angehalten. Der Ausgang "Error" wird bei fallender Flanke an "Enable" zurückgesetzt.

Die Fehlerausgänge werden nur bei Fehlern bei der Abarbeitung des Funktionsbausteins gesetzt. Fehler, die vom Modbus TCP-Protokoll verursacht werden, erzeugen keine Ausgabe an den Fehlerausgängen. Diese werden von den ModbusTCP-Client Funktionsbausteinen gemeldet.

Port Modbus TCP verwendet normalerweise den Port 502. Sollte der ModbusTCP-Server auf einem anderen TCP-Port laufen, kann die zu verwendende Portnummer auf den Eingang "Port" geschrieben werden.

Unterstützte Modbus-Function Codes Function Codes werden vom Modbus-Master verwendet um auf den Modbus-Slave zuzugreifen.

Die Kenntnis der unterstützten Codes dient der Fehlersuche und wird für den normalen Betrieb nicht benötigt.

Die folgenden Function Codes werden durch die Modbus-TCP-Client-Funktionsbausteine unterstützt:

Function Code	Funktionsbaustein	Beschreibung
1	IL_ModbusTCPReadCoils, Seite 229	Liest eine Anzahl Bits aus "BitData" ab Startadresse
2	IL_ModbusTCPReadDiscreteInputs, Seite 231	Liest eine Anzahl Bits aus "BitData" ab Startadresse
3	IL_ModbusTCPReadHoldingRegisters, Seite 234	Liest eine Anzahl Register aus "RegisterData" ab Startadresse
4	IL_ModbusTCPReadInputRegisters, Seite 236	Liest eine Anzahl Register aus "RegisterData" ab Startadresse
5	IL_ModbusTCPWriteSingleCoil, Seite 239	Schreibt ein Bit in "BitData" ab Startadresse
6	IL_ModbusTCPWriteSingleRegister, Seite 241	Schreibt ein Register in "RegisterData" ab Startadresse
15	IL_ModbusTCPWriteMultipleCoils, Seite 243	Schreibt eine Anzahl Bits in "BitData" ab Startadresse
16	IL_ModbusTCPWriteMultipleRegisters, Seite 245	Schreibt eine Anzahl Register in "RegisterData" ab Startadresse
23	IL_ModbusTCPReadWriteMultipleRegisters, Seite 248	Führt eine kombinierte Lese/Schreib-Operation auf "RegisterData" aus
43/14	IL_ModbusTCPReadDeviceIdentification, Seite 251	Liest die Standard-Geräteinformationen des ModbusTCP-Servers aus

Tab. 5-21: Unterstützte Modbus Function Codes

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR	16#0208	16#0000	Der Eingang "Address" ist ungültig.
INPUT_INVALID_ERROR	16#0204	16#0000	Der Eingang "Port" ist ungültig.
OTHER_ERROR	16#0040	16#0000	Konnte Socket nicht öffnen
OTHER_ERROR	16#0041	16#0000	Konnte I/O-Modus der Socket nicht setzen
OTHER_ERROR	16#0045	16#0209	Der Verbindungsaufbau wurde von der Gegenstelle abgelehnt - Es läuft dort kein ModBusTCP-Server
OTHER_ERROR	16#0045	16#020A	Die Gegenstelle antwortet nicht.
OTHER_ERROR	16#0045	16#020B	Die Gegenstelle wurde nicht gefunden.
OTHER_ERROR	16#0045	16#020C	Keine Route zur Gegenstelle.
COMMUNICATION_ERROR	16#020E	16#0000	Die Verbindung wurde geschlossen.
COMMUNICATION_ERROR	16#0211	16#0000	Die Gegenstelle hat die Verbindung beendet.

Tab. 5-22: IL_ModbusTCPClient Fehlercodes

5.3.3 IL_ModbusTCPReadCoils

Kurzbeschreibung Der Funktionsbaustein IL_ModbusTCPReadCoils liest zyklisch mehrere Coils von einem ModbusTCP-Server.

Hierfür wird eine durch den Funktionsbaustein IL_ModbusTCPClient verwaltete Verbindung benötigt.

Zielsystem	Bibliothek / Ordner
MLC 15VRS Basis-Funktionspaket PLC	RIL_ModbusTcp / Client

Tab. 5-23: Bibliothekszuordnung

Schnittstellenbeschreibung

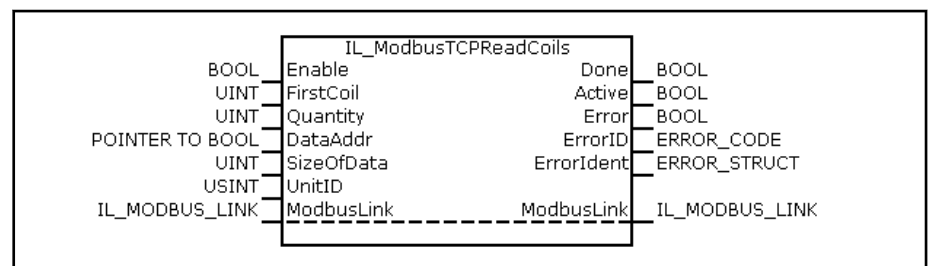


Abb. 5-18: Funktionsbaustein IL_ModbusTCPReadCoils

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (pegelgesteuert).
	FirstCoil	UINT	Erste Coil ab der gelesen werden soll
	Quantity	UINT	Anzahl der zu lesenden Coils
	DataAddr	POINTER TO BOOL	Adresse des Datenbereichs in den die Coils kopiert werden sollen
	SizeOfData	UINT	Größe des in DataAddr angelegten Datenbereichs in Byte
	UnitID	USINT	Unit-ID bzw. Slave-ID des Clients

	Name	Typ	Beschreibung
VAR_OUTPUT	Done	BOOL	Ein Übertragungszyklus ist abgeschlossen. Die empfangenen Daten wurden in den in DataAddr angelegten Bereich übertragen
	Active	BOOL	Übertragung läuft
	Error	BOOL	Bei der Bearbeitung des Bausteins ist ein Fehler aufgetreten.
	ErrorID	ERROR_CODE	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang eine grobe Klassifizierung des Fehlers.
	ErrorIdent	ERROR_STRUCT	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang detaillierte Informationen über den Fehler. Siehe auch " Fehlerbehandlung " auf Seite 231.
VAR_IN_OUT	ModbusLink	IL_MODBUS_LINK	Link auf die bestehende Modbus-Verbindung, wird von Funktionsbaustein IL_ModbusTCPClient bereitgestellt

Tab. 5-24: Schnittstelle IL_ModbusTCPReadCoils

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
FirstCoil	UINT	0	65535	0	Steigende Flanke an "Enable"
Quantity	UINT	1	2000	0	Steigende Flanke an "Enable"
DataAddr	POINTER TO BOOL	0		0	Steigende Flanke an "Enable"
SizeOfData	UINT	1	2000	0	Steigende Flanke an "Enable"
UnitID	USINT	0	255	255	Steigende Flanke an "Enable"

Tab. 5-25: Eingangsverhalten IL_ModbusTCPClient

Funktionsbeschreibung

Der Funktionsbaustein IL_ModbusTCPReadCoils setzt den Modbus Function Code 1 um.

Er liest zyklisch mehrere Coils von einem ModbusTCP-Server.

Hierfür wird eine durch den Funktionsbaustein IL_ModbusTCPClient verwaltete Verbindung benötigt.

Enable, Done

Bei steigender Flanke an "Enable" prüft der Funktionsbaustein seine Eingänge. Danach sendet er zyklisch eine Anfrage über den an ModbusLink angegebenen Funktionsbaustein IL_ModbusTCPClient an den Modbus-Server und wartet auf Antwort. Nach jedem erfolgreichem Übertragungszyklus ist "Done" für einen SPS-Zyklus gesetzt. Bei einer fallenden Flanke an "Enable" wird der Funktionsbaustein deaktiviert. Eine eventuell noch laufende Übertragung wird durch den IL_ModbusTCPClient-Funktionsbaustein weiter bearbeitet und das Ergebnis verworfen.

Error, ErrorID, ErrorIdent

Tritt während der Abarbeitung des Funktionsbausteins ein Fehler auf, so wird der Ausgang "Error" gesetzt. "ErrorID" und "ErrorIdent" enthalten genauere Informationen über den aufgetretenen Fehler. Die Bearbeitung des Bausteins

wird angehalten. Der Ausgang "Error" wird bei fallender Flanke an "Enable" zurückgesetzt.

Die Fehlerausgänge werden bei Fehlern an den Eingängen des Funktionsbausteins und bei ModbusTCP-Protokollfehlern gesetzt.

FirstCoil, Quantity Modbus stellt 65536 lesbare und schreibbare Coils bereit. Jede Coil kann entweder gesetzt oder gelöscht sein. Der Eingang "FirstCoil" gibt die erste zu lesende Coil an. Ab dieser Coil werden "Quantity" Coils gelesen. Es können maximal 2000 Coils mit einem Aufruf ausgelesen werden.

DataAddr, SizeOfData Die empfangenen Daten werden in den in "DataAddr" angegebenen Datenbereich abgelegt. Als Empfangspuffer wird ein Array of Bool erwartet. Die in "SizeOfData" angegebene Array-Größe muss mindestens so groß sein, wie die Anzahl der zu lesenden Coils. Es wird empfohlen den SIZEOF()-Operator zu verwenden, um die Größe des Arrays zu bestimmen.

UnitID Die Unit-ID des ModbusTCP-Protokolls entspricht der Slave-ID von Modbus-RTU. Sie wird verwendet, um verschiedene Clients auf dem Bus zu unterscheiden, bzw. um auf verschiedene Adressbereiche zugreifen zu können.

ModBusLink Der Eingang "ModbusLink" referenziert die eigentliche ModbusTCP-Verbindung. Er wird vom Funktionsbaustein IL_ModbusTCPClient ausgegeben und ist gültig, solange die Verbindung besteht.

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle F_RELATED_TABLE, 16#00000170. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
RESOURCE_ERROR	16#00001810	16#00000000	Der am Eingang ModbusLink angegebene Link ist nicht gültig.
INPUT_INVALID_ERROR	16#00001811	16#00000000	Die am Eingang "DataAddr" angegebene Adresse ist 0.
INPUT_INVALID_ERROR	16#00001812	16#00000000	Der Eingang "SizeOfData" ist 0
INPUT_INVALID_ERROR	16#00001813	16#00000000	Der Eingang "Quantity" ist 0
INPUT_RANGE_ERROR	16#00001814	16#00000000	Eingang "Quantity" ist größer als 2000
INPUT_INVALID_ERROR	16#00001815	16#00000000	Der Wert am Eingang "Quantity" ist größer als "SizeOfData"
RESOURCE_ERROR	16#00001817	Modbus-Fehlercode	Die Modbus-Operation lieferte einen Fehlercode zurück

Tab. 5-26: IL_ModbusTCPReadCoils Fehlercodes

5.3.4 IL_ModbusTCPReadDiscreteInputs

Kurzbeschreibung Der Funktionsbaustein IL_ModbusTCPReadDiscreteInputs liest zyklisch mehrere diskrete Eingänge von einem ModbusTCP-Server.

Hierfür wird eine durch den Funktionsbaustein IL_ModbusTCPClient verwaltete Verbindung benötigt.

Zielsystem	Bibliothek / Ordner
MLC 15VRS Basis-Funktionspaket PLC	RIL_ModbusTcp / Client

Tab. 5-27: Bibliothekszuordnung

Schnittstellenbeschreibung

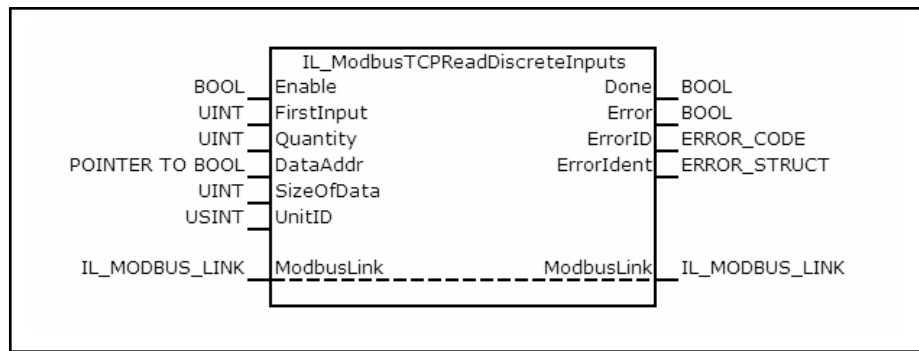


Abb. 5-19: Funktionsbaustein IL_ModbusTCPReadDiscreteInputs

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (pegelgesteuert).
	FirstCoil	UINT	Erster diskreter Eingang ab dem gelesen werden soll
	Quantity	UINT	Anzahl der zu lesenden Eingänge
	DataAddr	POINTER TO BOOL	Adresse des Datenbereichs in den die Eingänge kopiert werden sollen
	SizeOfData	UINT	Größe des in DataAddr angelegten Datenbereichs in Byte
	UnitID	USINT	Unit-ID bzw. Slave-ID des Clients
VAR_OUTPUT	Done	BOOL	Ein Übertragungszyklus ist abgeschlossen. Die empfangenen Daten wurden in den in DataAddr angelegten Bereich übertragen
	Error	BOOL	Bei der Bearbeitung des Bausteins ist ein Fehler aufgetreten.
	ErrorID	ERROR_CODE	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang eine grobe Klassifizierung des Fehlers.
	ErrorIdent	ERROR_STRUCT	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang detaillierte Informationen über den Fehler. Siehe auch "Fehlerbehandlung" auf Seite 233 .
VAR_IN_OUT	ModbusLink	IL_MODBUS_LINK	Link auf die bestehende Modbus-Verbindung, wird von Funktionsbaustein IL_ModbusTCPClient bereitgestellt

Tab. 5-28: Schnittstelle IL_ModbusTCPReadDiscreteInputs

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
FirstInput	UINT	0	65535	0	Steigende Flanke an "Enable"
Quantity	UINT	1	2000	0	Steigende Flanke an "Enable"
DataAddr	POINTER TO BOOL	0		0	Steigende Flanke an "Enable"

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
SizeOfData	UINT	1	2000	0	Steigende Flanke an "Enable"
UnitID	USINT	0	255	255	Steigende Flanke an "Enable"

Tab. 5-29: *Eingangsverhalten IL_ModbusTCPReadDiscreteInputs*

Funktionsbeschreibung	Der Funktionsbaustein IL_ModbusTCPReadDiscreteInputs setzt den Modbus Function Code 2 um. Er liest zyklisch mehrere diskrete Eingänge von einem ModbusTCP-Server. Hierfür wird eine durch den Funktionsbaustein IL_ModbusTCPClient verwaltete Verbindung benötigt.
Enable, Done	Bei steigender Flanke an "Enable" prüft der Funktionsbaustein seine Eingänge. Danach sendet er zyklisch eine Anfrage über den an ModbusLink angegebenen Funktionsbaustein IL_ModbusTCPClient an den Modbus-Server und wartet auf Antwort. Nach jedem erfolgreichem Übertragungszyklus ist "Done" für einen SPS-Zyklus gesetzt. Bei einer fallenden Flanke an "Enable" wird der Funktionsbaustein deaktiviert. Eine eventuell noch laufende Übertragung wird durch den IL_ModbusTCPClient-Funktionsbaustein weiter bearbeitet und das Ergebnis verworfen.
Error, ErrorID, ErrorIdent	Tritt während der Abarbeitung des Funktionsbausteins ein Fehler auf, so wird der Ausgang "Error" gesetzt. "ErrorID" und "ErrorIdent" enthalten genauere Informationen über den aufgetretenen Fehler. Die Bearbeitung des Bausteins wird angehalten. Der Ausgang "Error" wird bei fallender Flanke an "Enable" zurückgesetzt. Die Fehlerausgänge werden bei Fehlern an den Eingängen des Funktionsbausteins und bei ModbusTCP-Protokollfehlern gesetzt.
FirstInput, Quantity	Modbus stellt 65536 nur lesbare diskrete Eingänge bereit. Jeder Eingang kann entweder gesetzt oder gelöscht sein. Der Eingang "FirstInput" gibt den ersten zu lesenden Eingang an. Ab diesem Eingang werden "Quantity" Eingänge gelesen. Es können maximal 2000 Eingänge mit einem Aufruf ausgelesen werden.
DataAddr, SizeOfData	Die empfangenen Daten werden in den in "DataAddr" angegebenen Datenbereich abgelegt. Als Empfangspuffer wird ein Array of Bool erwartet. Die in "SizeOfData" angegebene Array-Größe muss mindestens so groß sein, wie die Anzahl der zu lesenden Eingänge. Es wird empfohlen den SIZEOF()-Operator zu verwenden, um die Größe des Arrays zu bestimmen.
UnitID	Die Unit-ID des ModbusTCP-Protokolls entspricht der Slave-ID von Modbus-RTU. Sie wird verwendet, um verschiedene Clients auf dem Bus zu unterscheiden, bzw. um auf verschiedene Adressbereiche zugreifen zu können.
ModBusLink	Der Eingang "ModbusLink" referenziert die eigentliche ModbusTCP-Verbindung. Er wird vom Funktionsbaustein IL_ModbusTCPClient ausgegeben und ist gültig, solange die Verbindung besteht.
Fehlerbehandlung	Der Funktionsbaustein nutzt die Fehlertabelle F_RELATED_TABLE, 16#00000170. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
RESOURCE_ERROR	16#00001810	16#00000000	Der am Eingang ModbusLink angegebene Link ist nicht gültig.
INPUT_INVALID_ERROR	16#00001811	16#00000000	Die am Eingang "DataAddr" angegebene Adresse ist 0.

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR	16#00001812	16#00000000	Der Eingang "SizeOfData" ist 0
INPUT_INVALID_ERROR	16#00001813	16#00000000	Der Eingang "Quantity" ist 0
INPUT_RANGE_ERROR	16#00001814	16#00000000	Eingang "Quantity" ist größer als 2000
INPUT_INVALID_ERROR	16#00001815	16#00000000	Der Wert am Eingang "Quantity" ist größer als "SizeOfData"
RESOURCE_ERROR	16#00001817	Modbus-Fehlercode	Die Modbus-Operation lieferte einen Fehlercode zurück

Tab. 5-30: IL_ModbusTCPReadDiscreteInputs Fehlercodes

5.3.5 IL_ModbusTCPReadHoldingRegisters

Kurzbeschreibung Der Funktionsbaustein IL_ModbusTCPReadHoldingRegisters liest zyklisch mehrere Holding-Register von einem ModbusTCP-Server.

Hierfür wird eine durch den Funktionsbaustein IL_ModbusTCPClient verwaltete Verbindung benötigt.

Zielsystem	Bibliothek / Ordner
MLC 15VRS	RIL_ModbusTCP / Client
Basis-Funktionspaket PLC	

Tab. 5-31: Bibliothekszuordnung

Schnittstellenbeschreibung

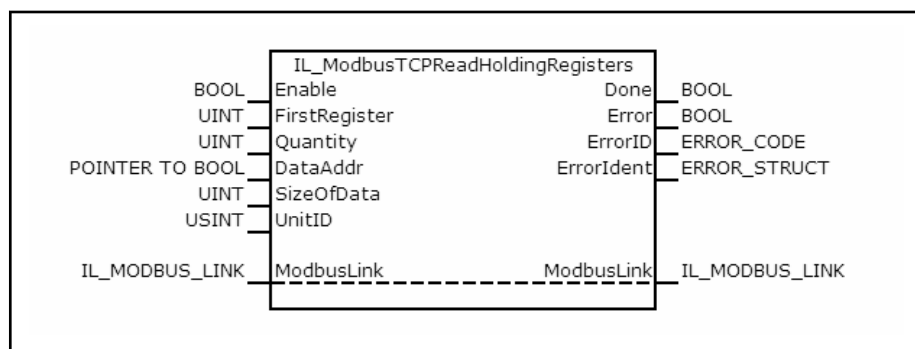


Abb. 5-20: Funktionsbaustein IL_ModbusTCPReadHoldingRegisters

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (pegelgesteuert).
	FirstRegister	UINT	Erstes Register ab dem gelesen werden soll
	Quantity	UINT	Anzahl der zu lesenden Register
	DataAddr	POINTER TO WORD	Adresse des Datenbereichs in den die Register kopiert werden sollen
	SizeOfData	UINT	Größe des in DataAddr angelegten Datenbereichs in Byte
	UnitID	USINT	Unit-ID bzw. Slave-ID des Clients

	Name	Typ	Beschreibung
VAR_OUTPUT	Done	BOOL	Ein Übertragungszyklus ist abgeschlossen. Die empfangenen Daten wurden in den in DataAddr angelegten Bereich übertragen
	Error	BOOL	Bei der Bearbeitung des Bausteins ist ein Fehler aufgetreten.
	ErrorID	ERROR_CODE	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang eine grobe Klassifizierung des Fehlers.
	ErrorIdent	ERROR_STRUCT	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang detaillierte Informationen über den Fehler. Siehe auch " Fehlerbehandlung " auf Seite 236.
VAR_IN_OUT	ModbusLink	IL_MODBUS_LINK	Link auf die bestehende Modbus-Verbindung, wird von Funktionsbaustein IL_ModbusTCPClient bereitgestellt

Tab. 5-32: Schnittstelle IL_ModbusTCPReadHoldingRegisters

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
FirstRegister	UINT	0	65535	0	Steigende Flanke an "Enable"
Quantity	UINT	1	125	0	Steigende Flanke an "Enable"
DataAddr	POINTER TO WORD	0		0	Steigende Flanke an "Enable"
SizeOfData	UINT	2	250	0	Steigende Flanke an "Enable"
UnitID	USINT	0	255	255	Steigende Flanke an "Enable"

Tab. 5-33: Eingangsverhalten IL_ModbusTCPReadHoldingRegisters

Funktionsbeschreibung	Der Funktionsbaustein IL_ModbusTCPReadHoldingRegisters setzt den Modbus Function Code 3 um. Er liest zyklisch mehrere Holding-Register von einem ModbusTCP-Server. Hierfür wird eine durch den Funktionsbaustein IL_ModbusTCPClient verwaltete Verbindung benötigt.
Enable, Done	Bei steigender Flanke an "Enable" prüft der Funktionsbaustein seine Eingänge. Danach sendet er zyklisch eine Anfrage über den an ModbusLink angegebenen Funktionsbaustein IL_ModbusTCPClient an den Modbus-Server und wartet auf Antwort. Nach jedem erfolgreichem Übertragungszyklus ist "Done" für einen SPS-Zyklus gesetzt. Bei einer fallenden Flanke an "Enable" wird der Funktionsbaustein deaktiviert. Eine eventuell noch laufende Übertragung wird durch den IL_ModbusTCPClient-Funktionsbaustein weiter bearbeitet und das Ergebnis verworfen.
Error, ErrorID, ErrorIdent	Tritt während der Abarbeitung des Funktionsbausteins ein Fehler auf, so wird der Ausgang "Error" gesetzt. "ErrorID" und "ErrorIdent" enthalten genauere Informationen über den aufgetretenen Fehler. Die Bearbeitung des Bausteins wird angehalten. Der Ausgang "Error" wird bei fallender Flanke an "Enable" zurückgesetzt. Die Fehlerausgänge werden bei Fehlern an den Eingängen des Funktionsbausteins und bei ModbusTCP-Protokollfehlern gesetzt.

FirstRegister, Quantity	Modbus stellt 65536 lesbare und schreibbare Holding-Register bereit. Jedes Register ist ein Wort (2 Byte) breit. Es kann einen Wert zwischen 0 und 65535 annehmen. Der Eingang "FirstRegister" gibt das erste zu lesende Register an. Ab diesem Register werden "Quantity" Register gelesen. Es können maximal 125 Register mit einem Aufruf ausgelesen werden.
DataAddr, SizeOfData	Die empfangenen Daten werden in den in "DataAddr" angegebenen Datenbereich abgelegt. Als Empfangspuffer wird ein Array of Word erwartet. Die in "SizeOfData" angegebene Array-Größe muss mindestens doppelt so groß sein, wie die Anzahl der zu lesenden Register. Es wird empfohlen den SIZE-OF()-Operator zu verwenden, um die Größe des Arrays zu bestimmen.
UnitID	Die Unit-ID des ModbusTCP-Protokolls entspricht der Slave-ID von Modbus-RTU. Sie wird verwendet, um verschiedene Clients auf dem Bus zu unterscheiden, bzw. um auf verschiedene Adressbereiche zugreifen zu können.
ModBusLink	Der Eingang "ModbusLink" referenziert die eigentliche ModbusTCP-Verbindung. Er wird vom Funktionsbaustein IL_ModbusTCPClient ausgegeben und ist gültig, solange die Verbindung besteht.
Fehlerbehandlung	Der Funktionsbaustein nutzt die Fehlertabelle F_RELATED_TABLE, 16#00000170.

Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
RESOURCE_ERROR	16#00001810	16#00000000	Der am Eingang ModbusLink angegebene Link ist nicht gültig.
INPUT_INVALID_ERROR	16#00001811	16#00000000	Die am Eingang "DataAddr" angegebene Adresse ist 0.
INPUT_INVALID_ERROR	16#00001812	16#00000000	Der Eingang "SizeOfData" ist 0
INPUT_INVALID_ERROR	16#00001813	16#00000000	Der Eingang "Quantity" ist 0
INPUT_RANGE_ERROR	16#00001814	16#00000001	Eingang "Quantity" ist größer als 125
INPUT_INVALID_ERROR	16#00001816	16#00000000	Der Wert am Eingang "Quantity"*2 ist größer als "SizeOfData"
RESOURCE_ERROR	16#00001817	Modbus-Fehlercode	Die Modbus-Operation lieferte einen Fehlercode zurück

Tab. 5-34: IL_ModbusTCPReadHoldingRegisters Fehlercodes

5.3.6 IL_ModbusTCPReadInputRegisters

Kurzbeschreibung	Der Funktionsbaustein IL_ModbusTCPReadInputRegisters liest zyklisch mehrere Input-Register von einem ModbusTCP-Server. Hierfür wird eine durch den Funktionsbaustein IL_ModbusTCPClient verwaltete Verbindung benötigt.

Zielsystem	Bibliothek / Ordner
MLC 15VRS Basis-Funktionspaket PLC	RIL_ModbusTcp / Client

Tab. 5-35: Bibliotheksuzuordnung

Schnittstellenbeschreibung

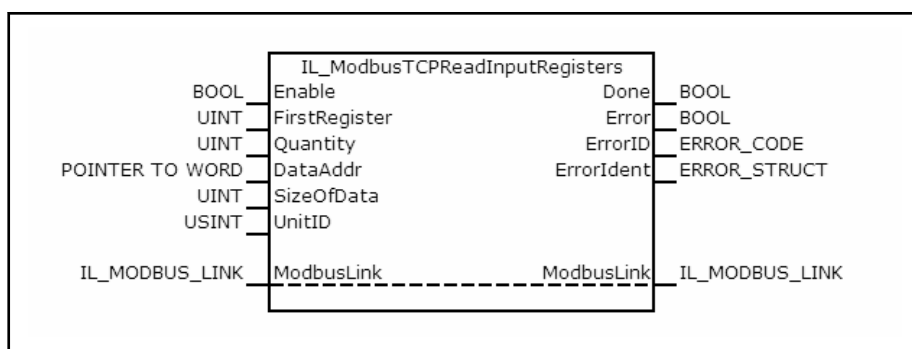


Abb. 5-21: Funktionsbaustein IL_ModbusTCPReadInputRegisters

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (pegelgesteuert).
	FirstRegister	UINT	Erstes Register ab dem gelesen werden soll
	Quantity	UINT	Anzahl der zu lesenden Register
	DataAddr	POINTER TO WORD	Adresse des Datenbereichs in den die Register kopiert werden sollen
	SizeOfData	UINT	Größe des in DataAddr angelegten Datenbereichs in Byte
	UnitID	USINT	Unit-ID bzw. Slave-ID des Clients
VAR_OUTPUT	Done	BOOL	Ein Übertragungszyklus ist abgeschlossen. Die empfangenen Daten wurden in den in DataAddr angelegten Bereich übertragen
	Error	BOOL	Bei der Bearbeitung des Bausteins ist ein Fehler aufgetreten.
	ErrorID	ERROR_CODE	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang eine grobe Klassifizierung des Fehlers.
	ErrorIdent	ERROR_STRUCT	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang detaillierte Informationen über den Fehler. Siehe auch "Fehlerbehandlung" auf Seite 238 .
VAR_IN_OUT	ModbusLink	IL_MODBUS_LINK	Link auf die bestehende Modbus-Verbindung, wird von Funktionsbaustein IL_ModbusTCPClient bereitgestellt

Tab. 5-36: Schnittstelle IL_ModbusTCPReadInputRegisters

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
FirstRegister	UINT	0	65535	0	Steigende Flanke an "Enable"
Quantity	UINT	1	125	0	Steigende Flanke an "Enable"
DataAddr	POINTER TO WORD	0		0	Steigende Flanke an "Enable"

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
SizeOfData	UINT	2	250	0	Steigende Flanke an "Enable"
UnitID	USINT	0	255	255	Steigende Flanke an "Enable"

Tab. 5-37: Eingangsverhalten IL_ModbusTCPReadInputRegisters

Funktionsbeschreibung Der Funktionsbaustein IL_ModbusTCPReadInputRegisters setzt den Modbus Function Code 4 um. Er liest zyklisch mehrere Input-Register von einem ModbusTCP-Server. Hierfür wird eine durch den Funktionsbaustein IL_ModbusTCPClient verwaltete Verbindung benötigt.

Enable, Done Bei steigender Flanke an "Enable" prüft der Funktionsbaustein seine Eingänge. Danach sendet er zyklisch eine Anfrage über den an ModbusLink angegebenen Funktionsbaustein IL_ModbusTCPClient an den Modbus-Server und wartet auf Antwort. Nach jedem erfolgreichem Übertragungszyklus ist "Done" für einen SPS-Zyklus gesetzt. Bei einer fallenden Flanke an "Enable" wird der Funktionsbaustein deaktiviert. Eine eventuell noch laufende Übertragung wird durch den IL_ModbusTCPClient-Funktionsbaustein weiter bearbeitet und das Ergebnis verworfen.

Error, ErrorID, ErrorIdent Tritt während der Abarbeitung des Funktionsbausteins ein Fehler auf, so wird der Ausgang "Error" gesetzt. "ErrorID" und "ErrorIdent" enthalten genauere Informationen über den aufgetretenen Fehler. Die Bearbeitung des Bausteins wird angehalten. Der Ausgang "Error" wird bei fallender Flanke an "Enable" zurückgesetzt.

Die Fehlerausgänge werden bei Fehlern an den Eingängen des Funktionsbausteins und bei ModbusTCP-Protokollfehlern gesetzt.

FirstRegister, Quantity Modbus stellt 65536 nur lesbare Input-Register bereit. Jedes Register ist ein Wort (2 Byte) breit. Es kann einen Wert zwischen 0 und 65535 annehmen. Der Eingang "FirstRegister" gibt das erste zu lesende Register an. Ab diesem Register werden "Quantity" Register gelesen. Es können maximal 125 Register mit einem Aufruf ausgelesen werden.

DataAddr, SizeOfData Die empfangenen Daten werden in den in "DataAddr" angegebenen Datenbereich abgelegt. Als Empfangspuffer wird ein Array of Word erwartet. Die in "SizeOfData" angegebene Array-Größe muss mindestens doppelt so groß sein, wie die Anzahl der zu lesenden Register. Es wird empfohlen den SIZE-OF()-Operator zu verwenden, um die Größe des Arrays zu bestimmen.

UnitID Die Unit-ID des ModbusTCP-Protokolls entspricht der Slave-ID von Modbus-RTU. Sie wird verwendet, um verschiedene Clients auf dem Bus zu unterscheiden, bzw. um auf verschiedene Adressbereiche zugreifen zu können.

ModBusLink Der Eingang "ModbusLink" referenziert die eigentliche ModbusTCP-Verbindung. Er wird vom Funktionsbaustein IL_ModbusTCPClient ausgegeben und ist gültig, solange die Verbindung besteht.

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle F_RELATED_TABLE, 16#00000170. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
RESOURCE_ERROR	16#00001810	16#00000000	Der am Eingang ModbusLink angegebene Link ist nicht gültig.
INPUT_INVALID_ERROR	16#00001811	16#00000000	Die am Eingang "DataAddr" angegebene Adresse ist 0.
INPUT_INVALID_ERROR	16#00001812	16#00000000	Der Eingang "SizeOfData" ist 0

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR	16#00001813	16#00000000	Der Eingang "Quantity" ist 0
INPUT_RANGE_ERROR	16#00001814	16#00000001	Eingang "Quantity" ist größer als 125
INPUT_INVALID_ERROR	16#00001816	16#00000000	Der Wert am Eingang "Quantity"*2 ist größer als "SizeOf-Data"
RESOURCE_ERROR	16#00001817	Modbus-Fehlercode	Die Modbus-Operation lieferte einen Fehlercode zurück

Tab. 5-38: IL_ModbusTCPReadInputRegisters Fehlercodes

5.3.7 IL_ModBusTCPWriteSingleCoil

Kurzbeschreibung Der Funktionsbaustein IL_ModBusTCPWriteSingleCoil schreibt einmalig eine Coil auf einen ModbusTCP-Server.
Hierfür wird eine durch den Funktionsbaustein IL_ModbusTCPClient verwaltete Verbindung benötigt.

Zielsystem	Bibliothek / Ordner
MLC 15VRS Basis-Funktionspaket PLC	RIL_ModbusTcp / Client

Tab. 5-39: Bibliothekszuordnung

Schnittstellenbeschreibung

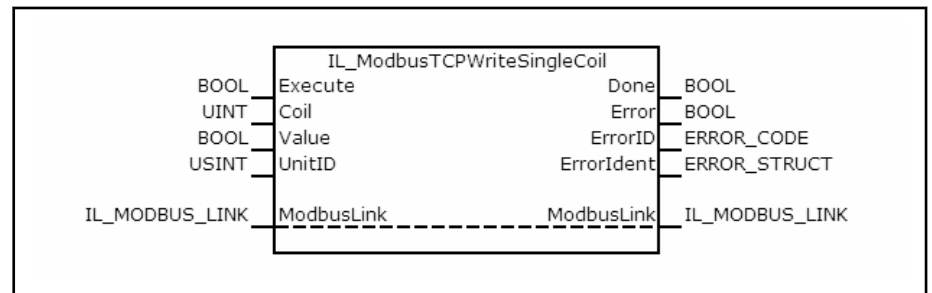


Abb. 5-22: Funktionsbaustein IL_ModBusTCPWriteSingleCoil

	Name	Typ	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (flankengesteuert).
	Coil	UINT	zu schreibende Coil
	Value	BOOL	zu schreibender Wert
	UnitID	USINT	Unit-ID bzw. Slave-ID des Clients

	Name	Typ	Beschreibung
VAR_OUTPUT	Done	BOOL	Ein Übertragungszyklus ist abgeschlossen.
	Error	BOOL	Bei der Bearbeitung des Bausteins ist ein Fehler aufgetreten.
	ErrorID	ERROR_CODE	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang eine grobe Klassifizierung des Fehlers.
	ErrorIdent	ERROR_STRUCT	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang detaillierte Informationen über den Fehler. Siehe auch " Fehlerbehandlung " auf Seite 241.
VAR_IN_OUT	ModbusLink	IL_MODBUS_LINK	Link auf die bestehende Modbus-Verbindung, wird von Funktionsbaustein IL_ModbusTCPClient bereitgestellt

Tab. 5-40: Schnittstelle IL_ModBusTCPWriteSingleCoil

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	kontinuierlich
Coil	UINT	0	65535	0	Steigende Flanke an "Execute"
Value	BOOL			FALSE	Steigende Flanke an "Execute"
UnitID	USINT	0	255	255	Steigende Flanke an "Execute"

Tab. 5-41: Eingangsverhalten IL_ModBusTCPWriteSingleCoil

Funktionsbeschreibung

Der Funktionsbaustein IL_ModBusTCPWriteSingleCoil setzt den Modbus Function Code 5 um. Er schreibt einmalig eine Coil auf einen ModbusTCP-Server. Hierfür wird eine durch den Funktionsbaustein IL_ModbusTCPClient verwaltete Verbindung benötigt.

Execute, Done

Bei steigender Flanke an "Execute" prüft der Funktionsbaustein seine Eingänge. Danach sendet er einmalig eine Anfrage über den an ModbusLink angegebenen Funktionsbaustein IL_ModbusTCPClient an den Modbus-Server und wartet auf Antwort. Nach erfolgreichem Übertragungszyklus ist "Done" gesetzt. Bei einer fallenden Flanke an "Execute" läuft der Funktionsbaustein weiter, bis die noch laufende Übertragung abgeschlossen ist. Dann wird für einen SPS-Zyklus "Done" gesetzt.

Error, ErrorID, ErrorIdent

Tritt während der Abarbeitung des Funktionsbausteins ein Fehler auf, so wird der Ausgang "Error" gesetzt. "ErrorID" und "ErrorIdent" enthalten genauere Informationen über den aufgetretenen Fehler. Die Bearbeitung des Bausteins wird angehalten. Der Ausgang "Error" wird bei fallender Flanke an "Enable" zurückgesetzt.

Die Fehlerausgänge werden bei Fehlern an den Eingängen des Funktionsbausteins und bei ModbusTCP-Protokollfehlern gesetzt.

Coil, Value

Modbus stellt 65536 lesbare und schreibbare Coils bereit. Jede Coil kann entweder gesetzt oder gelöscht sein. Der Eingang "Coil" gibt die zu schreibende Coil an. Der Wert in "Value" wird in diese Coil geschrieben.

UnitID

Die Unit-ID des ModbusTCP-Protokolls entspricht der Slave-ID von Modbus-RTU. Sie wird verwendet, um verschiedene Clients auf dem Bus zu unterscheiden, bzw. um auf verschiedene Adressbereiche zugreifen zu können.

ModBusLink Der Eingang "ModbusLink" referenziert die eigentliche ModbusTCP-Verbindung. Er wird vom Funktionsbaustein IL_ModbusTCPClient ausgegeben und ist gültig, solange die Verbindung besteht.

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle F_RELATED_TABLE, 16#00000170. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
RESOURCE_ERROR	16#00001810	16#00000000	Der am Eingang ModbusLink angegebene Link ist nicht gültig.
RESOURCE_ERROR	16#00001817	Modbus-Fehlercode	Die Modbus-Operation lieferte einen Fehlercode zurück

Tab. 5-42: IL_ModbusTCPWriteSingleCoil Fehlercodes

5.3.8 IL_ModbusTCPWriteSingleRegister

Kurzbeschreibung Der Funktionsbaustein IL_ModbusTCPWriteSingleRegister schreibt einmalig ein Holding-Register auf einen ModbusTCP-Server.

Hierfür wird eine durch den Funktionsbaustein IL_ModbusTCPClient verwaltete Verbindung benötigt.

Zielsystem	Bibliothek / Ordner
MLC 15VRS Basis-Funktionspaket PLC	RIL_ModbusTcp / Client

Tab. 5-43: Bibliothekszuordnung

Schnittstellenbeschreibung

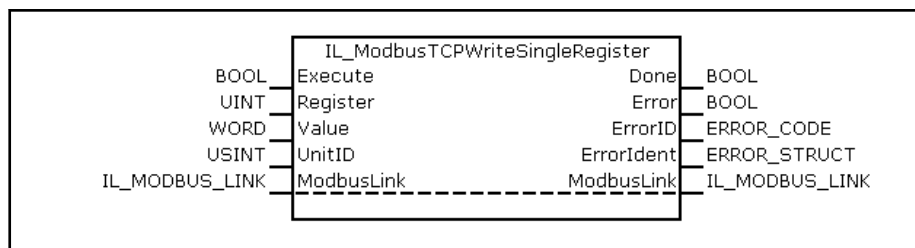


Abb. 5-23: Funktionsbaustein IL_ModbusTCPWriteSingleRegister

	Name	Typ	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (flankengesteuert).
	Register	UINT	zu schreibendes Register
	Value	WORD	zu schreibender Wert
	UnitID	USINT	Unit-ID bzw. Slave-ID des Clients

	Name	Typ	Beschreibung
VAR_OUTPUT	Done	BOOL	Ein Übertragungszyklus ist abgeschlossen.
	Error	BOOL	Bei der Bearbeitung des Bausteins ist ein Fehler aufgetreten.
	ErrorID	ERROR_CODE	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang eine grobe Klassifizierung des Fehlers.
	ErrorIdent	ERROR_STRUCT	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang detaillierte Informationen über den Fehler. Siehe auch " Fehlerbehandlung " auf Seite 243.
VAR_IN_OUT	ModbusLink	IL_MODBUS_LINK	Link auf die bestehende Modbus-Verbindung, wird von Funktionsbaustein IL_ModbusTCPClient bereitgestellt

Tab. 5-44: Schnittstelle IL_ModBusTCPWriteSingleRegister

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	kontinuierlich
HoldingRegister	UINT	0	65535	0	Steigende Flanke an "Execute"
Value	WORD	0	65535	0	Steigende Flanke an "Execute"
UnitID	USINT	0	255	255	Steigende Flanke an "Execute"

Tab. 5-45: Eingangsverhalten IL_ModBusTCPWriteSingleRegister

Funktionsbeschreibung

Der Funktionsbaustein IL_ModBusTCPWriteSingleRegister setzt den Modbus Function Code 6 um. Er liest schreibt ein Holding-Register auf einen ModbusTCP-Server. Hierfür wird eine durch den Funktionsbaustein IL_ModbusTCPClient verwaltete Verbindung benötigt.

Execute, Done

Bei steigender Flanke an "Execute" prüft der Funktionsbaustein seine Eingänge. Danach sendet er einmalig eine Anfrage über den an ModbusLink angegebenen Funktionsbaustein IL_ModbusTCPClient an den Modbus-Server und wartet auf Antwort. Nach erfolgreichem Übertragungszyklus ist "Done" gesetzt. Bei einer fallenden Flanke an "Execute" läuft der Funktionsbaustein weiter, bis die noch laufende Übertragung abgeschlossen ist. Dann wird für einen SPS-Zyklus "Done" gesetzt.

Error, ErrorID, ErrorIdent

Tritt während der Abarbeitung des Funktionsbausteins ein Fehler auf, so wird der Ausgang "Error" gesetzt. "ErrorID" und "ErrorIdent" enthalten genauere Informationen über den aufgetretenen Fehler. Die Bearbeitung des Bausteins wird angehalten. Der Ausgang "Error" wird bei fallender Flanke an "Enable" zurückgesetzt.

Die Fehlerausgänge werden bei Fehlern an den Eingängen des Funktionsbausteins und bei ModbusTCP-Protokollfehlern gesetzt.

Register, Value

Modbus stellt 65536 lesbare und schreibbare Holding-Register bereit. Jedes Register ist ein Wort (2 Byte) breit. Es kann einen Wert zwischen 0 und 65535 annehmen. Der Eingang "Register" gibt das zu schreibende Register an. In dieses Register wird der Wert von Eingang "Value" geschrieben.

UnitID

Die Unit-ID des ModbusTCP-Protokolls entspricht der Slave-ID von Modbus-RTU. Sie wird verwendet, um verschiedene Clients auf dem Bus zu unterscheiden, bzw. um auf verschiedene Adressbereiche zugreifen zu können.

ModBusLink Der Eingang "ModbusLink" referenziert die eigentliche ModbusTCP-Verbindung. Er wird vom Funktionsbaustein IL_ModbusTCPClient ausgegeben und ist gültig, solange die Verbindung besteht.

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle F_RELATED_TABLE, 16#00000170. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
RESOURCE_ERROR	16#00001810	16#00000000	Der am Eingang ModbusLink angegebene Link ist nicht gültig.
RESOURCE_ERROR	16#00001817	Modbus-Fehlercode	Die Modbus-Operation lieferte einen Fehlercode zurück

Tab. 5-46: IL_ModbusTCPWriteSingleRegister Fehlercodes

5.3.9 IL_ModbusTCPWriteMultipleCoils

Kurzbeschreibung Der Funktionsbaustein IL_ModbusTCPWriteMultipleCoils schreibt einmalig mehrere Coils auf einen ModbusTCP-Server.

Hierfür wird eine durch den Funktionsbaustein IL_ModbusTCPClient verwaltete Verbindung benötigt.

Zielsystem	Bibliothek / Ordner
MLC 15VRS Basis-Funktionspaket PLC	RIL_ModbusTcp / Client

Tab. 5-47: Bibliothekszuordnung

Schnittstellenbeschreibung

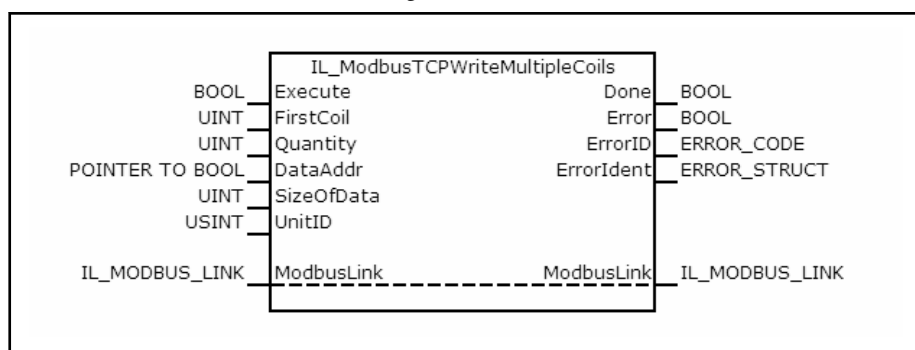


Abb. 5-24: Funktionsbaustein IL_ModbusTCPWriteMultipleCoils

	Name	Typ	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (flankengesteuert).
	FirstCoil	UINT	Erste Coil ab der geschrieben werden soll
	Quantity	UINT	Anzahl der zu schreibenden Coils
	DataAddr	POINTER TO BOOL	Adresse des Datenbereichs aus den die Coils gelesen werden sollen
	SizeOfData	UINT	Größe des in DataAddr angelegten Datenbereichs in Byte
	UnitID	USINT	Unit-ID bzw. Slave-ID des Clients

	Name	Typ	Beschreibung
VAR_OUTPUT	Done	BOOL	Ein Übertragungszyklus ist abgeschlossen.
	Error	BOOL	Bei der Bearbeitung des Bausteins ist ein Fehler aufgetreten.
	ErrorID	ERROR_CODE	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang eine grobe Klassifizierung des Fehlers.
	ErrorIdent	ERROR_STRUCT	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang detaillierte Informationen über den Fehler. Siehe auch " Fehlerbehandlung " auf Seite 245.
VAR_IN_OUT	ModbusLink	IL_MODBUS_LINK	Link auf die bestehende Modbus-Verbindung, wird von Funktionsbaustein IL_ModbusTCPClient bereitgestellt

Tab. 5-48: Schnittstelle IL_ModbusTCPWriteMultipleCoils

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	kontinuierlich
FirstCoil	UINT	0	65535	0	Steigende Flanke an "Execute"
Quantity	UINT	1	2000	0	Steigende Flanke an "Execute"
DataAddr	POINTER TO BOOL	0		0	Steigende Flanke an "Execute"
SizeOfData	UINT	1	2000	0	Steigende Flanke an "Execute"
UnitID	USINT	0	255	255	Steigende Flanke an "Execute"

Tab. 5-49: Eingangsverhalten IL_ModbusTCPWriteMultipleCoils

Funktionsbeschreibung

Der Funktionsbaustein IL_ModbusTCPWriteMultipleCoils setzt den Modbus Function Code 15 um.

Er schreibt einmalig mehrere Coils auf einen ModbusTCP-Server.

Hierfür wird eine durch den Funktionsbaustein IL_ModbusTCPClient verwaltete Verbindung benötigt.

Execute, Done

Bei steigender Flanke an "Execute" prüft der Funktionsbaustein seine Eingänge. Danach sendet er einmalig eine Anfrage über den an ModbusLink angegebenen Funktionsbaustein IL_ModbusTCPClient an den Modbus-Server und wartet auf Antwort. Nach erfolgreichem Übertragungszyklus ist "Done" gesetzt. Bei einer fallenden Flanke an "Execute" läuft der Funktionsbaustein weiter, bis die noch laufende Übertragung abgeschlossen ist. Dann wird für einen SPS-Zyklus "Done" gesetzt.

Error, ErrorID, ErrorIdent

Tritt während der Abarbeitung des Funktionsbausteins ein Fehler auf, so wird der Ausgang "Error" gesetzt. "ErrorID" und "ErrorIdent" enthalten genauere Informationen über den aufgetretenen Fehler. Die Bearbeitung des Bausteins wird angehalten. Der Ausgang "Error" wird bei fallender Flanke an "Enable" zurückgesetzt.

Die Fehlerausgänge werden bei Fehlern an den Eingängen des Funktionsbausteins und bei ModbusTCP-Protokollfehlern gesetzt.

FirstCoil, Quantity	Modbus stellt 65536 lesbare und schreibbare Coils bereit. Jede Coil kann entweder gesetzt oder gelöscht sein. Der Eingang "FirstCoil" gibt die erste zu schreibende Coil an. Ab dieser Coil werden "Quantity" Coils geschrieben. Es können maximal 2000 Coils mit einem Aufruf geschrieben werden.
DataAddr, SizeOfData	Die zu übertragenden Daten werden aus dem in "DataAddr" angegebenen Datenbereich gelesen. Als Sendepuffer wird ein Array of Bool erwartet. Die in "SizeOfData" angegebene Array-Größe muss mindestens so groß sein, wie die Anzahl der zu schreibenden Coils. Es wird empfohlen den SIZEOF()-Operator zu verwenden, um die Größe des Arrays zu bestimmen.
UnitID	Die Unit-ID des ModbusTCP-Protokolls entspricht der Slave-ID von Modbus-RTU. Sie wird verwendet, um verschiedene Clients auf dem Bus zu unterscheiden, bzw. um auf verschiedene Adressbereiche zugreifen zu können.
ModBusLink	Der Eingang "ModbusLink" referenziert die eigentliche ModbusTCP-Verbindung. Er wird vom Funktionsbaustein IL_ModbusTCPClient ausgegeben und ist gültig, solange die Verbindung besteht.
Fehlerbehandlung	Der Funktionsbaustein nutzt die Fehlertabelle F_RELATED_TABLE, 16#00000170. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
RESOURCE_ERROR	16#00001810	16#00000000	Der am Eingang ModbusLink angegebene Link ist nicht gültig.
INPUT_INVALID_ERROR	16#00001811	16#00000000	Die am Eingang "DataAddr" angegebene Adresse ist 0.
INPUT_INVALID_ERROR	16#00001812	16#00000000	Der Eingang "SizeOfData" ist 0
INPUT_INVALID_ERROR	16#00001813	16#00000000	Der Eingang "Quantity" ist 0
INPUT_RANGE_ERROR	16#00001814	16#00000000	Eingang "Quantity" ist größer als 2000
INPUT_INVALID_ERROR	16#00001815	16#00000000	Der Wert am Eingang "Quantity" ist größer als "SizeOfData"
RESOURCE_ERROR	16#00001817	Modbus-Fehlercode	Die Modbus-Operation lieferte einen Fehlercode zurück

Tab. 5-50: IL_ModBusTCPWriteMultipleCoils Fehlercodes

5.3.10 IL_ModBusTCPWriteMultipleRegisters

Kurzbeschreibung	Der Funktionsbaustein IL_ModBusTCPWriteMultipleRegisters schreibt einmalig mehrere Holding-Register auf einen ModbusTCP-Server. Hierfür wird eine durch den Funktionsbaustein IL_ModbusTCPClient verwaltete Verbindung benötigt.

Zielsystem	Bibliothek / Ordner
MLC 15VRS Basis-Funktionspaket PLC	RIL_ModbusTcp / Client

Tab. 5-51: Bibliothekszuordnung

Schnittstellenbeschreibung

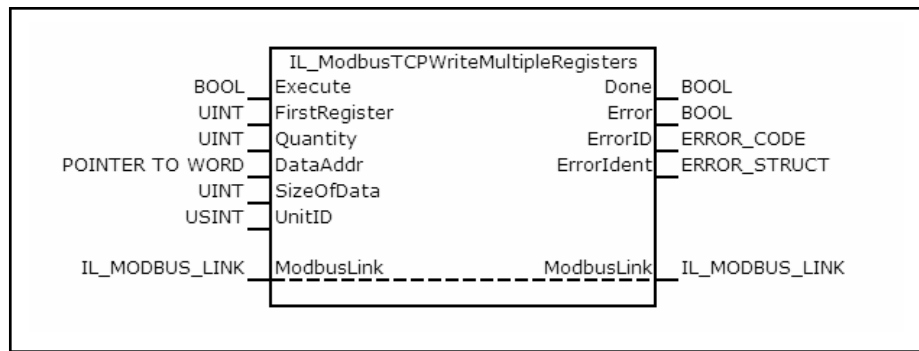


Abb. 5-25: Funktionsbaustein IL_ModBusTCPWriteMultipleRegisters

	Name	Typ	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (flankengesteuert).
	FirstRegister	UINT	Erstes Register ab dem geschrieben werden soll
	Quantity	UINT	Anzahl der zu schreibenden Register
	DataAddr	POINTER TO WORD	Adresse des Datenbereichs aus den die Register kopiert werden sollen
	SizeOfData	UINT	Größe des in DataAddr angelegten Datenbereichs in Byte
	UnitID	USINT	Unit-ID bzw. Slave-ID des Clients
VAR_OUTPUT	Done	BOOL	Ein Übertragungszyklus ist abgeschlossen.
	Error	BOOL	Bei der Bearbeitung des Bausteins ist ein Fehler aufgetreten.
	ErrorID	ERROR_CODE	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang eine grobe Klassifizierung des Fehlers.
	ErrorIdent	ERROR_STRUCT	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang detaillierte Informationen über den Fehler. Siehe auch " Fehlerbehandlung " auf Seite 247.
VAR_IN_OUT	ModbusLink	IL_MODBUS_LINK	Link auf die bestehende Modbus-Verbindung, wird von Funktionsbaustein IL_ModbusTCPClient bereitgestellt

Tab. 5-52: Schnittstelle IL_ModBusTCPWriteMultipleRegisters

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	kontinuierlich
FirstRegister	UINT	0	65535	0	Steigende Flanke an "Execute"
Quantity	UINT	1	125	0	Steigende Flanke an "Execute"
DataAddr	POINTER TO WORD	0		0	Steigende Flanke an "Execute"

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
SizeOfData	UINT	2	250	0	Steigende Flanke an "Execute"
UnitID	USINT	0	255	255	Steigende Flanke an "Execute"

Tab. 5-53: Eingangsverhalten IL_ModBusTCPWriteMultipleRegisters

Funktionsbeschreibung	Der Funktionsbaustein IL_ModBusTCPWriteMultipleRegisters setzt den Modbus Function Code 16 um. Er liest schreibt mehrere Holding-Register auf einen ModbusTCP-Server. Hierfür wird eine durch den Funktionsbaustein IL_ModbusTCPClient verwaltete Verbindung benötigt.
Execute, Done	Bei steigender Flanke an "Execute" prüft der Funktionsbaustein seine Eingänge. Danach sendet er einmalig eine Anfrage über den an ModbusLink angegebenen Funktionsbaustein IL_ModbusTCPClient an den Modbus-Server und wartet auf Antwort. Nach erfolgreichem Übertragungszyklus ist "Done" gesetzt. Bei einer fallenden Flanke an "Execute" läuft der Funktionsbaustein weiter, bis die noch laufende Übertragung abgeschlossen ist. Dann wird für einen SPS-Zyklus "Done" gesetzt.
Error, ErrorID, ErrorIdent	Tritt während der Abarbeitung des Funktionsbausteins ein Fehler auf, so wird der Ausgang "Error" gesetzt. "ErrorID" und "ErrorIdent" enthalten genauere Informationen über den aufgetretenen Fehler. Die Bearbeitung des Bausteins wird angehalten. Der Ausgang "Error" wird bei fallender Flanke an "Enable" zurückgesetzt. Die Fehlerausgänge werden bei Fehlern an den Eingängen des Funktionsbausteins und bei ModbusTCP-Protokollfehlern gesetzt.
FirstRegister, Quantity	Modbus stellt 65536 lesbare und schreibbare Holding-Register bereit. Jedes Register ist ein Wort (2 Byte) breit. Es kann einen Wert zwischen 0 und 65535 annehmen. Der Eingang "FirstRegister" gibt das erste zu schreibende Register an. Ab diesem Register werden "Quantity" Register geschrieben. Es können maximal 125 Register mit einem Aufruf geschrieben werden.
DataAddr, SizeOfData	Die zu sendenden Daten werden aus dem in "DataAddr" angegebenen Datenbereich gelesen. Als Sendepuffer wird ein Array of Word erwartet. Die in "SizeOfData" angegebene Array-Größe muss mindestens doppelt so groß sein, wie die Anzahl der zu schreibenden Register. Es wird empfohlen den SIZEOF()-Operator zu verwenden, um die Größe des Arrays zu bestimmen.
UnitID	Die Unit-ID des ModbusTCP-Protokolls entspricht der Slave-ID von Modbus-RTU. Sie wird verwendet, um verschiedene Clients auf dem Bus zu unterscheiden, bzw. um auf verschiedene Adressbereiche zugreifen zu können.
ModBusLink	Der Eingang "ModbusLink" referenziert die eigentliche ModbusTCP-Verbindung. Er wird vom Funktionsbaustein IL_ModbusTCPClient ausgegeben und ist gültig, solange die Verbindung besteht.
Fehlerbehandlung	Der Funktionsbaustein nutzt die Fehlertabelle F_RELATED_TABLE, 16#00000170. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
RESOURCE_ERROR	16#00001810	16#00000000	Der am Eingang ModbusLink angegebene Link ist nicht gültig.
INPUT_INVALID_ERROR	16#00001811	16#00000000	Die am Eingang "DataAddr" angegebene Adresse ist 0.
INPUT_INVALID_ERROR	16#00001812	16#00000000	Der Eingang "SizeOfData" ist 0
INPUT_INVALID_ERROR	16#00001813	16#00000000	Der Eingang "Quantity" ist 0

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#00001814	16#00000001	Eingang "Quantity" ist größer als 125
INPUT_INVALID_ERROR	16#00001816	16#00000000	Der Wert am Eingang "Quantity"*2 ist größer als "SizeOf-Data"
RESOURCE_ERROR	16#00001817	Modbus-Fehler-code	Die Modbus-Operation lieferte einen Fehlercode zurück

Tab. 5-54: IL_ModBusTCPWriteMultipleRegisters Fehlercodes

5.3.11 IL_ModBusTCPReadWriteMultipleRegisters

Kurzbeschreibung Der Funktionsbaustein IL_ModBusTCPReadWriteMultipleRegisters liest und schreibt zyklisch mehrere Holding-Register von einem ModbusTCP-Server. Hierfür wird eine durch den Funktionsbaustein IL_ModbusTCPClient verwaltete Verbindung benötigt.

Zielsystem	Bibliothek / Ordner
MLC 15VRS Basis-Funktionspaket PLC	RIL_ModbusTcp / Client

Tab. 5-55: Bibliothekszuordnung

Schnittstellenbeschreibung

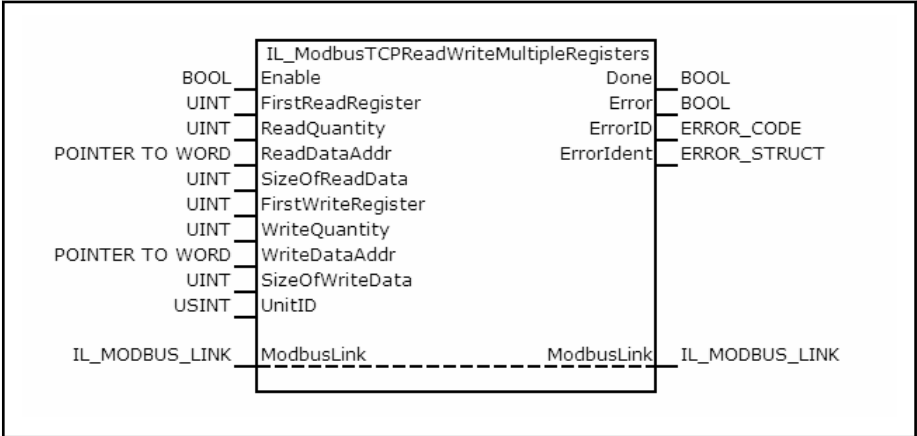


Abb. 5-26: Funktionsbaustein IL_ModBusTCPReadWriteMultipleRegisters

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (pegelgesteuert).
	FirstReadRegister	UINT	Erstes Register ab dem gelesen werden soll
	ReadQuantity	UINT	Anzahl der zu lesenden Register
	ReadDataAddr	POINTER TO WORD	Adresse des Datenbereichs in den die Register kopiert werden sollen
	SizeOfReadData	UINT	Größe des in ReadDataAddr angelegten Datenbereichs in Byte
	FirstWriteRegister	UINT	Erstes Register ab dem geschrieben werden soll
	WriteQuantity	UINT	Anzahl der zu schreibenden Register
	WriteDataAddr	POINTER TO WORD	Adresse des Datenbereichs aus dem die Register kopiert werden sollen
	SizeOfWriteData	UINT	Größe des in WriteDataAddr angelegten Datenbereichs in Byte
	UnitID	USINT	Unit-ID bzw. Slave-ID des Clients
VAR_OUTPUT	Done	BOOL	Ein Übertragungszyklus ist abgeschlossen. Die empfangenen Daten wurden in den in DataAddr angelegten Bereich übertragen
	Error	BOOL	Bei der Bearbeitung des Bausteins ist ein Fehler aufgetreten.
	ErrorID	ERROR_CODE	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang eine grobe Klassifizierung des Fehlers.
	ErrorIdent	ERROR_STRUCT	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang detaillierte Informationen über den Fehler. Siehe auch " Fehlerbehandlung " auf Seite 251.
VAR_IN_OUT	ModbusLink	IL_MODBUS_LINK	Link auf die bestehende Modbus-Verbindung, wird von Funktionsbaustein IL_ModbusTCPClient bereitgestellt

Tab. 5-56: Schnittstelle IL_ModbusTCPReadWriteMultipleRegisters

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
FirstReadRegister	UINT	0	65535	0	Steigende Flanke an "Enable"
ReadQuantity	UINT	1	125	0	Steigende Flanke an "Enable"
ReadDataAddr	POINTER TO WORD	0		0	Steigende Flanke an "Enable"
SizeOfReadData	UINT	2	250	0	Steigende Flanke an "Enable"
FirstWriteRegister	UINT	0	65535	0	Steigende Flanke an "Enable"

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
WriteQuantity	UINT	1	125	0	Steigende Flanke an "Enable"
WriteDataAddr	POINTER TO WORD	0		0	Steigende Flanke an "Enable"
SizeOfWriteData	UINT	2	250	0	Steigende Flanke an "Enable"
UnitID	USINT	0	255	255	Steigende Flanke an "Enable"

Tab. 5-57: *Eingangsverhalten IL_ModBusTCPReadWriteMultipleRegisters***Funktionsbeschreibung**

Der Funktionsbaustein IL_ModBusTCPReadWriteMultipleRegisters setzt den Modbus Function Code 23 um.

Er liest und schreibt zyklisch mehrere Holding-Register von einem ModbusTCP-Server.

Hierfür wird eine durch den Funktionsbaustein IL_ModbusTCPClient verwaltete Verbindung benötigt.

Enable, Done

Bei steigender Flanke an "Enable" prüft der Funktionsbaustein seine Eingänge. Danach sendet er zyklisch eine Anfrage über den an ModbusLink angegebenen Funktionsbaustein IL_ModbusTCPClient an den Modbus-Server und wartet auf Antwort. Nach jedem erfolgreichem Übertragungszyklus ist "Done" für einen SPS-Zyklus gesetzt. Bei einer fallenden Flanke an "Enable" wird der Funktionsbaustein deaktiviert. Eine eventuell noch laufende Übertragung wird durch den IL_ModbusTCPClient-Funktionsbaustein weiter bearbeitet und das Ergebnis verworfen.

Error, ErrorID, ErrorIdent

Tritt während der Abarbeitung des Funktionsbausteins ein Fehler auf, so wird der Ausgang "Error" gesetzt. "ErrorID" und "ErrorIdent" enthalten genauere Informationen über den aufgetretenen Fehler. Die Bearbeitung des Bausteins wird angehalten. Der Ausgang "Error" wird bei fallender Flanke an "Enable" zurückgesetzt.

Die Fehlerausgänge werden bei Fehlern an den Eingängen des Funktionsbausteins und bei ModbusTCP-Protokollfehlern gesetzt.

FirstReadRegister, ReadQuantity

Modbus stellt 66536 lesbare und schreibbare Holding-Register bereit. Jedes Register ist ein Wort (2 Byte) breit. Es kann einen Wert zwischen 0 und 65535 annehmen. Der Eingang "FirstReadRegister" gibt das erste zu lesende Register an. Ab diesem Register werden "ReadQuantity" Register gelesen. Es können maximal 125 Register mit einem Aufruf ausgelesen werden.

FirstWriteRegister, WriteQuantity

Modbus stellt 66536 lesbare und schreibbare Holding-Register bereit. Jedes Register ist ein Wort (2 Byte) breit. Es kann einen Wert zwischen 0 und 65535 annehmen. Der Eingang "FirstWriteRegister" gibt das erste zu schreibende Register an. Ab diesem Register werden "WriteQuantity" Register geschrieben. Es können maximal 125 Register mit einem Aufruf geschrieben werden.

ReadDataAddr, SizeOfReadData

Die empfangenen Daten werden in den in "ReadDataAddr" angegebenen Datenbereich abgelegt. Als Empfangspuffer wird ein Array of Word erwartet. Die in "SizeOfReadData" angegebene Array-Größe muss mindestens doppelt so groß sein, wie die Anzahl der zu lesenden Register. Es wird empfohlen, den SIZEOF()-Operator zu verwenden, um die Größe des Arrays zu bestimmen.

WriteDataAddr, SizeOfWriteData

Die zu sendenden Daten werden aus dem in "ReadDataAddr" angegebenen Datenbereich gelesen. Als Sendepuffer wird ein Array of Word erwartet. Die

in "SizeOfWriteData" angegebene Array-Größe muss mindestens doppelt so groß sein, wie die Anzahl der zu schreibenden Register. Es wird empfohlen, den SIZEOF()-Operator zu verwenden, um die Größe des Arrays zu bestimmen.

UnitID Die Unit-ID des ModbusTCP-Protokolls entspricht der Slave-ID von Modbus-RTU. Sie wird verwendet, um verschiedene Clients auf dem Bus zu unterscheiden, bzw. um auf verschiedene Adressbereiche zugreifen zu können.

ModBusLink Der Eingang "ModbusLink" referenziert die eigentliche ModbusTCP-Verbindung. Er wird vom Funktionsbaustein IL_ModbusTCPClient ausgegeben und ist gültig, solange die Verbindung besteht.

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle F_RELATED_TABLE, 16#0170. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
RESOURCE_ERROR	16#1810	16#0000	Der am Eingang ModbusLink angegebene Link ist nicht gültig.
INPUT_INVALID_ERROR	16#1811	16#0000	Die am Eingang "ReadDataAddr" angegebene Adresse ist 0.
INPUT_INVALID_ERROR	16#1811	16#0001	Die am Eingang "WriteDataAddr" angegebene Adresse ist 0.
INPUT_INVALID_ERROR	16#1812	16#0000	Der Eingang "SizeOfReadData" ist 0
INPUT_INVALID_ERROR	16#1812	16#0001	Der Eingang "SizeOfWriteData" ist 0
INPUT_INVALID_ERROR	16#1813	16#0000	Der Eingang "ReadQuantity" ist 0
INPUT_INVALID_ERROR	16#1813	16#0001	Der Eingang "WriteQuantity" ist 0
INPUT_RANGE_ERROR	16#1814	16#0001	Eingang "ReadQuantity" ist größer als 125
INPUT_RANGE_ERROR	16#1814	16#0002	Eingang "WriteQuantity" ist größer als 125
INPUT_INVALID_ERROR	16#1816	16#0000	Der Wert am Eingang "ReadQuantity"*2 ist größer als "SizeOfReadData"
INPUT_INVALID_ERROR	16#1816	16#0001	Der Wert am Eingang "WriteQuantity"*2 ist größer als "SizeOfWriteData"
RESOURCE_ERROR	16#1817	Modbus-Fehlercode	Die Modbus-Operation lieferte einen Fehlercode zurück

Tab. 5-58: IL_ModBusTCPReadWriteMultipleRegisters Fehlercodes

5.3.12 IL_ModbusTCPReadDeviceIdentification

Kurzbeschreibung Der Funktionsbaustein IL_ModbusTCPReadDeviceIdentification liest alle verfügbaren Geräte-Informationen über ein Modbus Client-Gerät, und gibt sie an seinen Ausgängen aus.

Hierfür wird eine durch den Funktionsbaustein IL_ModbusTCPClient verwaltete Verbindung benötigt.

Zielsystem	Bibliothek / Ordner
MLC 15VRS Basis-Funktionspaket PLC	RIL_ModbusTcp / Client

Tab. 5-59: Bibliothekszuordnung

Schnittstellenbeschreibung

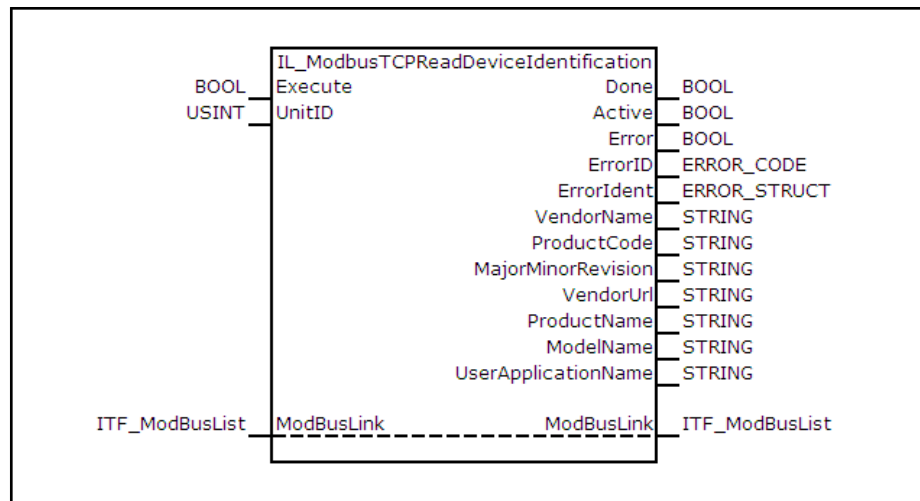


Abb. 5-27: Funktionsbaustein IL_ModbusTCPReadDeviceIdentification

	Name	Typ	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (flanken-gesteuert).
	UnitID	USINT	Unit-ID bzw. Slave-ID des Clients
VAR_OUTPUT	Done	BOOL	Ein Übertragungszyklus ist abgeschlossen. Die emp-fangenen Daten wurden in den in DataAddr angelegten Bereich übertragen
	Active	BOOL	Die Übertragung läuft
	Error	BOOL	Bei der Bearbeitung des Bausteins ist ein Fehler aufge-treten.
	ErrorID	ERROR_CODE	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang eine grobe Klassifizierung des Fehlers.
	ErrorIdent	ERROR_STRUCT	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang detaillierte Informationen über den Fehler. Siehe auch "Fehlerbehandlung" auf Seite 253 .
	VendorName	STRING	Name des Herstellers
	ProductCode	STRING	Produkt-Code
	MajorMinorRevision	STRING	Versionsnummer
	VendorUrl	STRING	Web-Adresse des Herstellers
	ProductName	STRING	Produktname
	ModelName	STRING	Modellname
	UserApplicationName	STRING	Name der Anwendung
VAR_IN_OUT	ModBusLink	IL_MODBUS_LINK	Link auf die bestehende Modbus-Verbindung, wird von Funktionsbaustein IL_ModbusTCPClient bereitgestellt

Tab. 5-60: Schnittstelle IL_ModbusTCPReadDeviceIdentification

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	kontinuierlich
UnitID	USINT	0	255	255	Steigende Flanke an "Execute"

Tab. 5-61: Eingangsverhalten IL_ModbusTCPReadDeviceIdentification

Funktionsbeschreibung	Der Funktionsbaustein IL_ModbusTCPReadDeviceIdentification liest alle verfügbaren Geräte-Informationen über ein Modbus Client-Gerät und gibt sie an seinen Ausgängen aus.
Execute, Active, Done	Bei steigender Flanke an "Execute" prüft der Funktionsbaustein seine Eingänge. Danach sendet er die Anfrage über den an ModbusLink angegebenen Funktionsbaustein IL_ModbusTCPClient an den Modbus-Server und wartet auf Antwort. Während dieser Zeit ist der Ausgang "Active" gesetzt. Nach erfolgreichem Datenempfang ist "Done" für gesetzt. Bei einer fallenden Flanke an "Execute" läuft der Funktionsbaustein noch bis Daten empfangen wurden oder ein Fehler auftritt.
Error, ErrorID, ErrorIdent	Tritt während der Abarbeitung des Funktionsbausteins ein Fehler auf, so wird der Ausgang "Error" gesetzt. "ErrorID" und "ErrorIdent" enthalten genauere Informationen über den aufgetretenen Fehler. Die Bearbeitung des Bausteins wird angehalten. Der Ausgang "Error" wird bei fallender Flanke an "Execute" zurückgesetzt. Die Fehlerausgänge werden bei Fehlern an den Eingängen des Funktionsbausteins und bei ModbusTCP-Protokollfehlern gesetzt.
UnitID	Die Unit-ID des ModbusTCP-Protokolls entspricht der Slave-ID von Modbus-RTU. Sie wird verwendet, um verschiedene Clients auf dem Bus zu unterscheiden, bzw. um auf verschiedene Adressbereiche zugreifen zu können.
ModBusLink	Der Eingang "ModbusLink" referenziert die eigentliche ModbusTCP-Verbindung. Er wird vom Funktionsbaustein IL_ModbusTCPClient ausgegeben und ist gültig, solange die Verbindung besteht.
Fehlerbehandlung	Der Funktionsbaustein nutzt die Fehlertabelle F_RELATED_TABLE, 16#00000170. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
RESOURCE_ERROR	16#00001810	16#00000000	Der am Eingang ModbusLink angegebene Link ist nicht gültig.
RESOURCE_ERROR	16#00001817	Modbus-Fehlercode	Die Modbus-Operation lieferte einen Fehlercode zurück

Tab. 5-62: IL_ModbusTCPReadDeviceIdentification Fehlercodes

6 RIL_SocketComm.library

6.1 Einführung und Überblick

Die RIL_SocketComm-Bibliothek enthält Funktionen und Funktionsbausteine zum Aufbau einer Verbindung und zum Datentransfer mit TCP (Transmission Control Protocol) und UDP (User Datagram Protocol).

TCP und UDP sind IP-basierte Internet-Daten-Transport-Protokolle.

Während TCP ein verbindungsorientiertes Protokoll ist, das Verbindungen aufbaut und diese aufrecht erhält, während die Anwendung Daten überträgt, ist UDP ein paketorientiertes verbindungsloses Protokoll, das Datenpakete innerhalb eines Netzwerks von einer Anwendung zur anderen sendet.

Die RIL_SocketComm unterstützt sowohl synchrone als auch asynchrone Kommunikation.

Die synchrone Kommunikation wird durch Funktionen unterstützt.

Für die asynchrone Kommunikation werden Funktionsbausteine verwendet. Es wird nicht empfohlen, Funktionen und Funktionsbausteine innerhalb derselben TCP- oder UDP-Verbindung zu verwenden.



Die IP-basierte Kommunikation ist nicht echtzeitfähig, d.h., es kann nicht garantiert werden, dass die Daten innerhalb eines bestimmten Zeitfensters ihr Ziel erreichen. Deshalb sollte sie nur für den reinen Datenaustausch verwendet werden.

Für die Synchronisation von Prozessen sowie der Steuerung von zeitkritischen Ein-/Ausgängen sollte sie nicht eingesetzt werden.

Die RIL_SocketComm-Bibliothek verwendet Funktionen der SysSocket-Bibliothek. Die IP-basierte Kommunikation wird grundsätzlich vom Betriebssystem der Steuerung durchgeführt. Deshalb können Unterschiede im Laufzeitverhalten auf verschiedenen Steuerungen auftreten.



Für die Bibliothek ist das Basis-Funktionspaket PLC des Systems MLC 15VRS ausreichend.

6.2 I/O Zeitverhalten der Funktionsbausteine

Jeder Funktionsbaustein unterstützt die gemeinsame Fehlerstruktur und das definierte Verhalten mit den Ausgängen "Error", "ErrorID" und "ErrorIdent".

Alle Funktionsbausteine mit "Enable"-Eingang und "Done"-Ausgang haben das gleiche pegelgesteuerte Laufzeitverhalten. Eine steigende Flanke am "Enable"-Eingang startet die Ausführung. Eine fallende Flanke stoppt die Ausführung und setzt alle binären Ausgänge zurück.

Ist "Done" TRUE, hat der Funktionsbaustein seine Aufgabe erfolgreich beendet. Der Funktionsbaustein bleibt in diesem Zustand stehen, bis "Enable" zurückgesetzt wird. Es werden keine weiteren Aktionen durchgeführt.

Im Fehlerfall wird "Error" gesetzt und "ErrorID" und "ErrorIdent" enthalten den Fehlercode. "Done" und "Error" werden zurückgesetzt, wenn "Enable" zurückgesetzt wird.

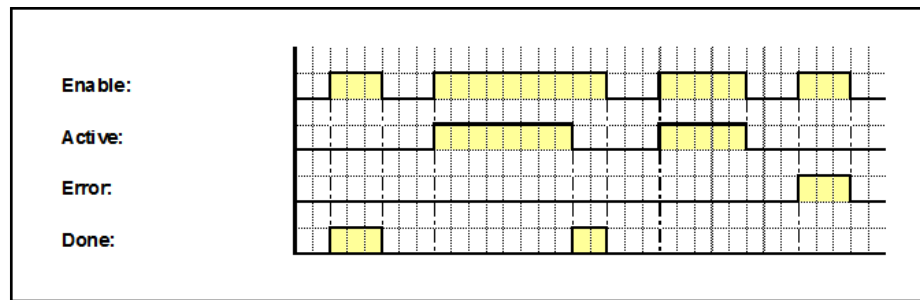


Abb. 6-1: Zeitdiagramm für Enable, Active, Error und Done

6.3 Funktionale Beschreibung

6.3.1 TCP

TCP ist ein verbindungsorientiertes IP-Protokoll. Bevor ein Datenaustausch stattfinden kann, muss sich der Client mit einem offenen TCP Server-Port verbinden. Der Datenfluss ist zeichenorientiert. Die gesendeten Daten werden vom TCP-Protokoll in Datenpakete aufgeteilt, übertragen und auf der Gegenseite wieder zu einem Zeichenstrom zusammengesetzt.

Die Reihenfolge der gesendeten Zeichen wird immer beibehalten. Treten während der Übertragung Fehler auf, wird das betreffende Paket automatisch wiederholt gesendet. Es können also keine Daten verloren gehen. Tritt ein Verbindungsverlust auf, wird dies beim nächsten Lese- oder Schreibzugriff mit einem Fehlercode gemeldet.

TCP eignet sich für nichtzyklische Kommunikation, bei der Daten den Empfänger sicher erreichen müssen. Ein gutes Beispiel ist eine Log-Datei. Es ist nicht wichtig, wann die Daten ihr Ziel erreichen, aber sie müssen an der Gegenstelle ankommen.

Das folgende Diagramm zeigt einen Verbindungsaufbau mit anschließender Datenübertragung über TCP.

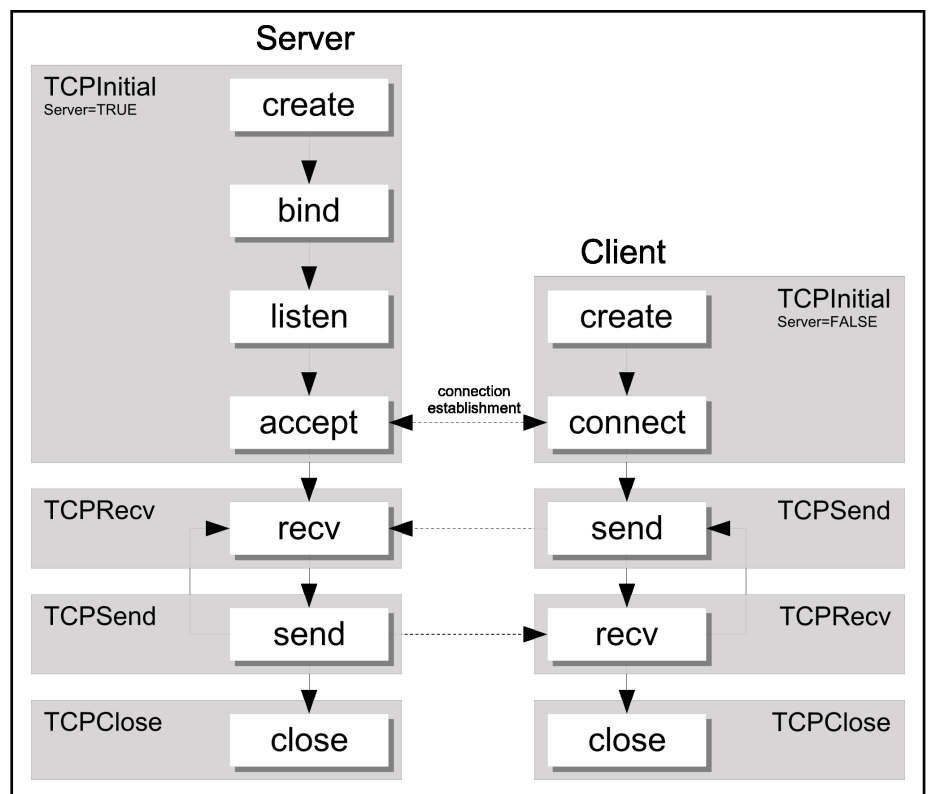


Abb. 6-2: TCP Server-Client Flussdiagramm

Asynchrone Funktionsbausteine für eine TCP-Verbindung

- [IL_TCPConnectionAsync](#) auf Seite 269
- [IL_TCPInitialAsync](#) auf Seite 273
- [IL_TCPSendAsync](#) auf Seite 277
- [IL_TCPRecvAsync](#) auf Seite 279
- [IL_TCPRecvPacketAsync](#) auf Seite 281
- [IL_TCPCloseAsync](#) auf Seite 283

Synchrone Funktionen für eine TCP-Verbindung

- [IL_TCPInitial](#) auf Seite 285
- [IL_TCPSend](#) auf Seite 286
- [IL_TCPRecv](#) auf Seite 287
- [IL_TCPRecvPacket](#) auf Seite 288
- [IL_TCPClose](#) auf Seite 290

Details zur Realisierung:

- [TCP Asynchrone Funktionsbausteine](#) auf Seite 269
- [TCP Synchrone Funktionen](#) auf Seite 284

TCP Server - Beispiel

Das folgende Beispiel zeigt eine einfache TCP-Serveranwendung, die einen ASCII-Text mit Hilfe der RIL_SocketComm-Bibliothek sendet.

Beim Start öffnet das Programm einen TCP-Server auf Port 4242 und wartet auf eine eingehende Verbindung. Nachdem die Verbindung aufgebaut wurde, wird der Text gesendet und die Verbindung wieder geschlossen. Danach öffnet das Programm den Server-Port erneut und wartet auf weitere eingehende Verbindungen.

Einfaches TCP-Server-Programm (Deklaration)

```
(* This very simple test program waits for an incoming TCP connection on port 4242. *)
(* After the connection is established, it sends the string "Hello World!"          *)
(* and closes the connection                                                       *)
```

```
PROGRAM PLC_PRG
VAR
    (* used function block instances *)
    fbTCPConnectionAsync: IL_TCPConnectionAsync;
    fbTCPSendAsync: IL_TCPSendAsync;

    (* variables *)
    udiSocket: UDINT; (* socket handle *)
    strHello: STRING := 'Hello World!$r$N'; (* text to send *)
END_VAR
```

Einfaches TCP-Server-Programm (Implementation)

```
(* Check if connection is established *)
IF fbTCPConnectionAsync.Done = FALSE THEN
    (* No connection established yet => setup server mode and wait *)
    fbTCPConnectionAsync(Enable:=TRUE, Server:=TRUE, Port:=4242, Socket=>udiSocket);
    IF fbTCPConnectionAsync.Error = TRUE THEN
        (* error while establishing connection => reset everything *)
        fbTCPConnectionAsync(Enable:=FALSE, Socket=>udiSocket);
    END_IF
ELSE
    (* connection established => send text *)
    fbTCPSendAsync(Enable:=TRUE, Socket:=udiSocket, ValueAdr:=ADR(strHello), NoOfBytes:=LEN(strHello));
    IF fbTCPSendAsync.Done=TRUE OR fbTCPSendAsync.Error=TRUE THEN
        (* text completely send or some error => close connection *)
        fbTCPSendAsync(Enable:=FALSE);
        fbTCPConnectionAsync(Enable:=FALSE, Socket=>udiSocket);
    END_IF
END_IF
```

Verwenden Sie das Windows-Programm "Telnet", um zur Steuerung zu verbinden. Der erste Parameter ist die IP-Adresse, der zweite Parameter ist der Port:

```
Telnet 192.168.1.1 4242
```

TCP Client - Beispiel

Das folgende Beispiel zeigt einen TCP-Client, der auf Port 4242 auf einen TCP-Server verbindet. Nach erfolgreichem Verbindungsaufbau wird der String "Hello" gesendet und auf Antwort gewartet. Nach Erhalt des Antwortpakets wird die Verbindung geschlossen.

Einfaches TCP-Client-Programm (Deklaration)

```
PROGRAM PLC_PRG
VAR
    fbConnection: IL_TCPConnectionAsync;
    fbSend: IL_TCPSendAsync;
    fbRecv: IL_TCPRecvAsync;

    udiSocket: UDINT := SOCKET_INVALID;

    strSend: STRING := 'Hello';
    strRecv: STRING;

    iState: INT := 0;
END_VAR
```

Einfaches TCP-Client-Programm (Implementation)

```
CASE iState OF
0:
    fbConnection(Enable:=FALSE);
    fbSend(Enable:=FALSE);
    fbRecv(Enable:=FALSE);
    diSocket := SOCKET_INVALID;
    iState := 1;
1:
```

```

fbConnection(Enable:=TRUE, Server:=FALSE, Address:='192.168.42.254', Port:=4242);
IF fbConnection.Done=TRUE THEN
  (* Connection established *)
  udiSocket := fbConnection.Socket;
  iState := 2;
END_IF

2:
  (* Send some data *)
  fbSend(Enable:=TRUE, Socket:=udiSocket, ValueAdr:=ADR(strSend), NoOfBytes:=LEN(strSend));
  IF fbSend.Done=TRUE THEN
    (* Send done *)
    fbSend(Enable:=FALSE);
    iState := 3;
  END_IF

3:
  (* Receive some data *)
  fbRecv(Enable:=TRUE, Socket:=udiSocket, ValueAdr:=ADR(strRecv), NoOfBytes:=SIZEOF(strRecv)-1);
  IF fbRecv.Done=TRUE THEN
    (* Received some data *)
    strRecv := LEFT(strRecv, UDINT_TO_INT(fbRecv.NoOfRecBytes)); (* end of string marker *)
    fbRecv(Enable:=FALSE);
    iState := 4;
  END_IF

4:
  (* Close the connection *)
  fbConnection(Enable:=FALSE);
  diSocket := SOCKET_INVALID;

END_CASE

```

TCP Cyclic - Beispiel

Das folgende Beispiel zeigt eine einfache TCP-Anwendung, die eine Datenstruktur mit Hilfe der RIL_SocketComm-Bibliothek zyklisch zwischen zwei Teilnehmern austauscht.

Beim Start öffnet das Programm einen TCP-Server auf Port 50001 und wartet auf eine eingehende Verbindung (Server) bzw. versucht eine Verbindung mit der Gegenstelle aufzubauen (Client). Nachdem die Verbindung aufgebaut wurde, wird die Datenstruktur zyklisch zwischen beiden Kommunikationspartnern ausgetauscht. Danach öffnet das Programm den Server-Port erneut und wartet auf weitere eingehende Verbindungen.

TCP-Server-Programm

```

PROGRAM Client
VAR
  fbIL_TCPCyclic: IL_TCPCyclic;

  SendData: MY_STRUCTURE;
  RecvData: MY_STRUCTURE;

  bEnable: BOOL;
  bInOperation: BOOL;
  bError: BOOL;
  nErrorCode: ERROR_CODE;
  nErrorIdent: ERROR_STRUCT;
  nStatus: IL_CONN_STATUS;
  tResponseTime: TIME;
  bDoneSend: BOOL;
  bDoneRecv: BOOL;
END_VAR

fbIL_TCPCyclic(
  Enable:= bEnable,
  Mode:= Cyclic_Client,
  IPAddress:= '192.168.0.10',
  Port:= 50001,
  SendDataAdr:= ADR(SendData),
  RecvDataAdr:= ADR(RecvData),
  SizeOfSendData:= SIZEOF(SendData),
  SizeOfRecvData:= SIZEOF(RecvData),
  Timeout:= T#1500MS,
  InOperation=> bInOperation,
  Error=> bError,
  ErrorID=> nErrorCode,
  ErrorIdent=> nErrorIdent,

```

```

Status=> nStatus,
ResponseTime=> tResponseTime,
DoneSend=> bDoneSend,
DoneRecv=> bDoneRecv);

```

TCP-Client-Programm

```

PROGRAM Server
VAR
    fbIL_TCPCyclic: IL_TCPCyclic;

    SendData: MY_STRUCTURE;
    RecvData: MY_STRUCTURE;

    bEnable: BOOL;
    bInOperation: BOOL;
    bError: BOOL;
    nErrorCode: ERROR_CODE;
    nErrorIdent: ERROR_STRUCT;
    nStatus: IL_CONN_STATUS;
    tResponseTime: TIME;
    bDoneSend: BOOL;
    bDoneRecv: BOOL;
END_VAR

fbIL_TCPCyclic(
    Enable:= bEnable,
    Mode:= Cyclic_Server,
    IpAddress:= '192.168.0.10',
    Port:= 50001,
    SendDataAdr:= ADR(SendData),
    RecvDataAdr:= ADR(RecvData),
    SizeOfSendData:= SIZEOF(SendData),
    SizeOfRecvData:= SIZEOF(RecvData),
    Timeout:= T#1500MS,
    InOperation=> bInOperation,
    Error=> bError,
    ErrorID=> nErrorCode,
    ErrorIdent=> nErrorIdent,
    Status=> nStatus,
    ResponseTime=> tResponseTime,
    DoneSend=> bDoneSend,
    DoneRecv=> bDoneRecv);

```

Datenstruktur

```

// Example of a data structure
TYPE MY_STRUCTURE :
STRUCT
    bTestBool    : BOOL;
    byTestByte   : BYTE;
    uiTestUInt   : UINT;
    rTestReal    : REAL;
    tTestTime    : TIME;
    strTest      : STRING(22) := 'Hello World';
    dwTest       : DWORD;
END_STRUCT
END_TYPE

```

6.3.2 UDP

UDP ist ein verbindungsloses IP-Protokoll. Im Gegensatz zum TCP-Protokoll gibt es keinen Verbindungsaufbau vor dem eigentlichen Datenaustausch. Die Kommunikation erfolgt paketbasiert. Empfangene Datenpakete werden zwischengespeichert, aber nicht zusammengefügt. Jeder Empfangsaufwurf holt genau ein Paket aus der Warteschlange.

Empfangene UDP-Datenpakete werden nicht bestätigt. Daher ist es nicht möglich, verloren gegangene Pakete zu erkennen. UDP bietet sich daher für zyklische Datenübertragung an, in der die Daten von eventuell verloren gegangenen Paketen durch spätere Pakete aktualisiert werden.

Das folgende Flussdiagramm zeigt eine typische unidirektionale UDP Client-Serverkonfiguration. Der Server öffnet eine UDP-Socket auf einem vorgegebenen Port, während der UDP Port des Clients unbekannt ist. In dieser Konstellation kann der Client nur senden, jedoch nicht empfangen.

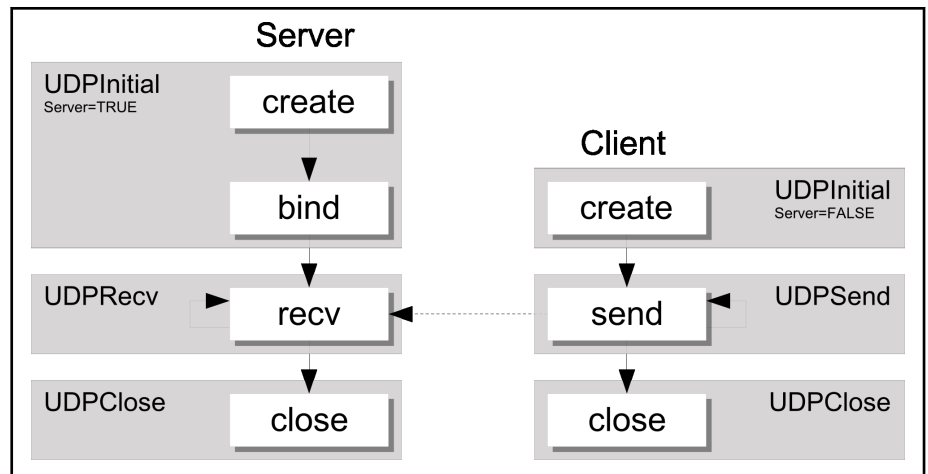


Abb. 6-3: UDP Server-Client Flussdiagramm

Das folgende Flussdiagramm zeigt eine typische bidirektionale UDP Server-Serverkommunikation. In dieser Konstellation können beide Kommunikationspartner Datenpakete senden und empfangen, da der jeweilige Zielport bekannt ist.

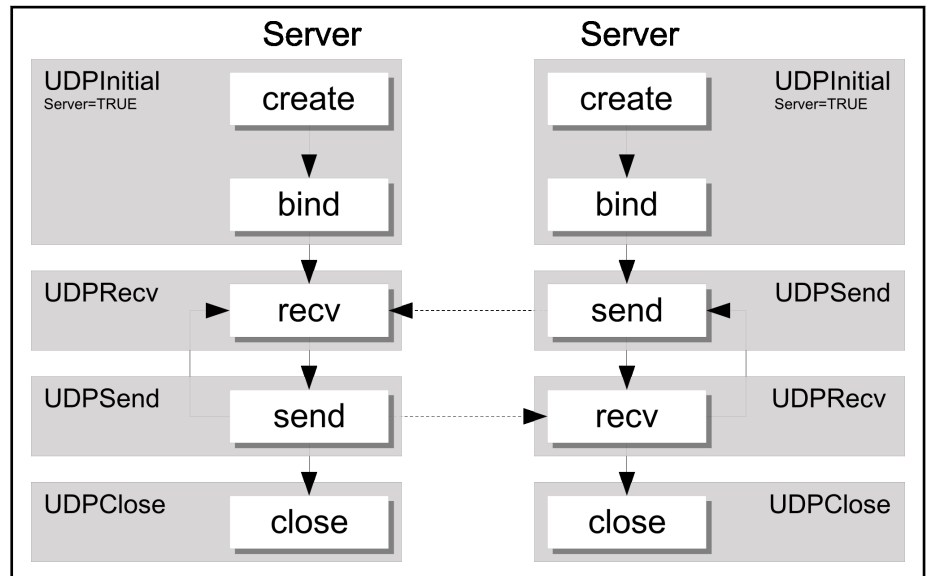


Abb. 6-4: UDP Server-Server-Flussdiagramm

Asynchrone Funktionsbausteine für eine UDP-Verbindung

- [IL_UDPSocketAsync](#) auf Seite 291
- [IL_UDPIInitialAsync](#) auf Seite 293
- [IL_UDPSendtoAsync](#) auf Seite 295
- [IL_UDPrecvfromAsync](#) auf Seite 297
- [IL_UDPCloseAsync](#) auf Seite 298

Synchrone Funktionen für eine UDP-Verbindung

- [IL_UDPIInitial](#) auf Seite 300
- [IL_UDPSendto](#) auf Seite 301
- [IL_UDPrecvfrom](#) auf Seite 302
- [IL_UDPClose](#) auf Seite 304

Details zur Realisierung:

- [UDP Asynchrone Funktionsbausteine auf Seite 290](#)
- [UDP Synchrone Funktionen auf Seite 299](#)

Nachstehend Beispielprogramme zum Datenaustausch auf UDP-Basis:

- [UDP-Client - Beispiel auf Seite 262](#)
- [UDP-Server, synchrone Übertragung auf Seite 263](#)
- [UDP-Client, synchrone Übertragung auf Seite 265](#)

UDP-Client - Beispiel

Das folgende Beispiel zeigt einen einfachen UDP-Client der zyklisch einmal pro Sekunde ein Datenpaket zu einem definierten UDP-Server IP- und Port-Paar schickt.

Typ-Definition einer UDP-Paket Datenstruktur:

```

TYPE UDP_PACKET :
STRUCT
  Sequence: UINT;
  State: INT;
  InfoTxt: STRING[20];
END_STRUCT
END_TYPE

```

Einfaches UDP-Clientprogramm (Deklaration)

```

(* This simple Example implements an UDP client. It sends an UDP packet once a second. *)
(* the packet can contain all process data and even strings. *)
(* A sequence number is implemented to allow for a missing data packets check *)

```

```

PROGRAM PLC_PRG
VAR CONSTANT
  (* communication settings *)
  strUDPServer: STRING := '192.168.1.254'; (* put in your remote UDP server IP here *)
  uiPort: UINT := 2323; (* remote UDP server port *)
END_VAR

VAR
  (* used function blocks *)
  fbUDPSocket: IL_UDPSocketAsync;
  fbUDPSend: IL_UDPSendtoAsync;

  (* variables *)
  udiSocket: UDINT; (* socket handle *)
  iCount: INT; (* send loop counter *)

  (* UDP packet data to send *)
  stUDPPacket: UDP_PACKET := (Sequence:=0, State:=1, InfoTxt:='Device Ready');
END_VAR

```

Einfaches UDP-Clientprogramm (Implementation)

```

(* check for UDP socket *)
IF fbUDPSocket.Done = FALSE THEN
  (* no UDP socket yet => create one *)
  fbUDPSocket(Enable:=TRUE, Server:=FALSE, Socket=>udiSocket);

  IF fbUDPSocket.Error = TRUE THEN
    (* an error occurred => reset *)
    fbUDPSocket(Enable:=FALSE, Socket=>udiSocket);
  END_IF
ELSE
  IF iCount <= 0 THEN
    (* every 100 loops send UDP packet (10ms task * 100 loops = 1s) *)
    IF fbUDPSend.Active = FALSE THEN
      (* not already about to send => increase sequence number of UDP packet *)
      stUDPPacket.Sequence := stUDPPacket.Sequence + 1;
    END_IF

    (* send the packet *)
    fbUDPSend(Enable:=TRUE, Socket:=udiSocket,
      ValueAdr:=ADR(stUDPPacket), NoOfBytes:=SIZEOF(stUDPPacket),

```

```

        Address:=strUDPServer, Port:=uiPort);

IF fbUDPSend.Done = TRUE THEN
    (* packet send => restart counter *)
    fbUDPSend(Enable:=FALSE);
    iCount := 100;
ELSIF fbUDPSend.Error = TRUE THEN
    (* an error occurred => reset everything *)
    fbUDPSend(Enable:=FALSE);
    fbUDPSocket(Enable:=FALSE);
END_IF
ELSE
    (* decrease loop counter *)
    iCount := iCount - 1;
END_IF
END_IF

```

UDP-Server, synchrone Übertragung

Im folgenden Beispiel wird ein UDP-Server implementiert. Der UDP-Server empfängt Daten von einem Client und sendet an diesen Daten zurück.

Danach wird die Verbindung geschlossen und das Socket freigegeben.

Das Programm setzt die Bibliotheken "RIL_SocketComm" und "cmperrors" voraus.



- Der Funktionsbaustein IL_UDPInitial darf nicht zyklisch aufgerufen werden. Der Baustein kehrt vom Aufruf erst zurück, wenn die Verbindung aufgebaut ist.
- IL_UDPInitial erzeugt bei jedem Aufruf ein Socket. Bitte die Begrenzung der Socket-Anzahl berücksichtigen.
- Das Auftreten des Fehlers "ERR SOCK WOULD BLOCK" beim Senden oder Empfangen von Daten bedeutet, dass der Send-Puffer voll oder der Receive-Puffer noch leer ist.

Der Fehler "ERR SOCK WOULD BLOCK" kann somit ignoriert und der Aufruf wiederholt werden.

Deklaration des Programms

```

PROGRAM UDPSync_Server_Prog
VAR
    arRecvData: ARRAY[1..10] OF BYTE;
    arSendData: ARRAY[1..10] OF BYTE;

    uiState:          UINT := 0;
    i:                INT;
    udiResult:        UDINT;
    bResult:          BOOL;
    udiSocket:        UDINT;
    udiErrorNum:      UDINT;
    uiClient_Port:    UINT;
    strClient_Address: STRING;
    udiResult_Latched: UDINT;
    udiErrorNum_Latched: UDINT;
END_VAR

VAR CONSTANT
    STATE_INIT:      UINT := 0;
    STATE_SERVER:    UINT := 1;
    STATE_RECEIVE:   UINT := 2;
    STATE_SEND:      UINT := 3;
    STATE_CLOSE:     UINT := 4;
    STATE_ERROR:     UINT := 5;
    STATE_FINISH:    UINT := 6;
END_VAR

```

Implementation

```

//The following program implements an UDP Server
//the UDP Server receives data from a Client
//then the Server sends data back to the Client
//after that close connection and release socket

CASE uiState OF

//Initialize senddata
STATE_INIT:
    FOR i := 1 TO 10 BY 1 DO
        arSendData[i] := arSendData[i] + 1;
    END FOR
    uiState := STATE_SERVER;

//UDP Server
STATE_SERVER:
    udiResult:= IL_UDPInitial(bServer:= TRUE, strAddress:= '', uiPort:= 6000, pudiSocket:= ADR(udiSocket),
        pudErrorNum:= ADR(udiErrorNum));

    //if no error, go to next state, else go to the error state
    IF udiResult = 0 AND udiErrorNum = 0 THEN
        uiState := STATE_RECEIVE;
    ELSE
        uiState := STATE_CLOSE;
    END_IF

//Receive data from Client
STATE_RECEIVE:
    udiResult:= IL_UDPRecvfrom(udiSocket:= udiSocket, pbyValueAdr:= ADR(arRecvData),
        pstSendSockAddr:= ADR(strClient_Address), puiSendPort:= ADR(uiClient_Port),
        pudErrorNum:= ADR(udiErrorNum), udiNoOfBytes:= SIZEOF(arRecvData));

    //go to the next state if the receive process is finished without error
    IF udiResult <> 0 AND udiErrorNum = Errors.ERR_OK THEN
        uiState := STATE_SEND;
    END_IF

    //if error is 'wouldblock', try it again; else go to the error state
    IF udiErrorNum <> 0 THEN
        IF udiErrorNum = Errors.ERR_SOCK_WOULDBLOCK THEN
            RETURN;
        ELSE
            uiState := STATE_CLOSE;
        END_IF
    END_IF

//Send data to client
STATE_SEND:
    udiResult:= IL_UDPSendto(udiSocket:= udiSocket, pbyValueAdr:= ADR(arSendData),
        strAddress:= strClient_Address, uiPort:= uiClient_Port,
        pudErrorNum:= ADR(udiErrorNum), udiNoOfBytes:= SIZEOF(arSendData));

    //go to the next state if the send process is finished without error
    IF udiResult <> 0 AND udiErrorNum = Errors.ERR_OK THEN
        uiState := STATE_CLOSE;
    END_IF

    //if error is wouldblock, try it again; else go to the error state
    IF udiErrorNum <> 0 THEN
        IF udiErrorNum = Errors.ERR_SOCK_WOULDBLOCK THEN
            RETURN;
        ELSE
            uiState := STATE_CLOSE;
        END_IF
    END_IF

//Release Socket
STATE_CLOSE:
    bResult := IL_UDPClose(udiSocket:= udiSocket, pudErrorNum:= ADR(strClient_Address));

    //store error information
    IF udiErrorNum <> 0 THEN
        udiErrorNum_Latched := udiErrorNum;
    END_IF
    uiState := STATE_FINISH;

    //store error information
STATE_ERROR:
    udiResult_Latched := udiResult;

```



```

udiErrorNum_Latched := udiErrorNum;
//go to the_close state to break the connection and release the socket
uiState := STATE_CLOSE;

//Finish, do nothing
STATE_FINISH:
;
END_CASE

```

UDP-Client, synchrone Übertragung

Im folgenden Beispiel wird ein UDP-Client implementiert. Der UDP-Client sendet Daten an einen Server und empfängt danach Daten von diesem. Danach wird die Verbindung geschlossen und das Socket freigegeben. Das Programm setzt die Bibliotheken "RIL_SocketComm" und "cmperrors" voraus.



- Der Funktionsbaustein IL_UDPInitial darf nicht zyklisch aufgerufen werden. Der Baustein kehrt vom Aufruf erst zurück, wenn die Verbindung aufgebaut ist.
- IL_UDPInitial erzeugt bei jedem Aufruf ein Socket. Bitte die Begrenzung der Socket-Anzahl berücksichtigen.
- Das Auftreten des Fehlers "ERR SOCK_WOULDBLOCK" beim Senden oder Empfangen von Daten bedeutet, dass der Send-Puffer voll oder der Receive-Puffer noch leer ist. Der Fehler "ERR SOCK_WOULDBLOCK" kann somit ignoriert und der Aufruf wiederholt werden.

Deklaration des Programms

```

PROGRAM UDPSync_Client_Prog
VAR
  arRecvData: ARRAY[1..10] OF BYTE;
  arSendData: ARRAY[1..10] OF BYTE;

  uiState:      UINT := 0;
  i:            INT;
  udiResult:    UDINT;
  bResult:      BOOL;
  udiSocket:    UDINT;
  udiErrorNum:  UDINT;
  uiRecv_Port:  UINT;
  strRecv_Address: STRING;
  udiResult_Latched: UDINT;
  udiErrorNum_Latched: UDINT;
END_VAR

VAR CONSTANT
  STATE_INIT:      UINT := 0;
  STATE_CLIENT:    UINT := 1;
  STATE_RECEIVE:   UINT := 2;
  STATE_SEND:      UINT := 3;
  STATE_CLOSE:     UINT := 4;
  STATE_ERROR:     UINT := 5;
  STATE_FINISH:   UINT := 6;
END_VAR

```

Implementation

```

//The following program implements a UDP Client
//the UDP Client sends data to a Server
//then it receives data from the Server
//after that close the connection and release socket

CASE uiState OF

//Initialize senddata
STATE_INIT:
  FOR i := 1 TO 10 BY 1 DO

```

RIL_SocketComm.library

```

        arSendData[i] := arSendData[i] + 1;
    END_FOR
    uiState := STATE_CLIENT;

//UDP Client
STATE_CLIENT:
    udiResult := IL_UDPInitial(bServer:= FALSE, strAddress := '192.168.42.11', uiPort:= 6000,
                               pudiSocket:= ADR(udiSocket), pudErrorNum:= ADR(udiErrorNum));

    //if no error, go to next state, else go to the error state
    IF udiResult = 0 AND udiErrorNum = Errors.ERR_OK THEN
        uiState := STATE_SEND;
    ELSE
        uiState := STATE_ERROR;
    END_IF

//Send data to Server
STATE_SEND:
    udiResult:= IL_UDPSendto(udiSocket:= udiSocket, pbyValueAdr:= ADR(arSendData),
                             strAddress:= '192.168.42.11', uiPort:= 6000,
                             pudErrorNum:= ADR(udiErrorNum) , udiNoOfBytes:= SIZEOF(arSendData));

    //go to the next state if the send process is finished without error
    IF udiResult <> 0 AND udiErrorNum = Errors.ERR_OK THEN
        uiState := STATE_RECEIVE;
    END_IF

    //if error is wouldblock, try it again; else go to the error state
    IF udiErrorNum <> 0 THEN
        IF udiErrorNum = Errors.ERR_SOCK_WOULDBLOCK THEN
            RETURN;
        ELSE
            uiState := STATE_ERROR;
        END_IF
    END_IF

//Receive data from Server
STATE_RECEIVE:
    udiResult:= IL_UDPRecvfrom(udiSocket:= udiSocket, pbyValueAdr:= ADR(arRecvData),
                                pstSendSockAddr:= ADR(strRecv_Address), puiSendPort:= ADR(uiRecv_Port),
                                pudErrorNum:= ADR(udiErrorNum), udiNoOfBytes:= SIZEOF(arRecvData));

    //go to the next state if the receive process is finished without error
    IF udiResult <> 0 AND udiErrorNum = Errors.ERR_OK THEN
        uiState := STATE_CLOSE;
    END_IF

    //if error is wouldblock, try it again; else go to the error state
    IF udiErrorNum <> 0 THEN
        IF udiErrorNum = Errors.ERR_SOCK_WOULDBLOCK THEN
            RETURN;
        ELSE
            uiState := STATE_ERROR;
        END_IF
    END_IF

//Release socket
STATE_CLOSE:
    bResult := IL_UDPClose(udiSocket:= udiSocket, pudErrorNum:= ADR(udiErrorNum));

    //store error information
    IF udiErrorNum <> 0 THEN
        udiErrorNum_Latched := udiErrorNum;
    END_IF
    uiState := STATE_FINISH;

    //store error information
STATE_ERROR:
    udiResult_Latched := udiResult;
    udiErrorNum_Latched := udiErrorNum;

    //go to the close state to break the connection and release the socket
    uiState := STATE_CLOSE;

//Finish, do nothing
STATE_FINISH:
    ;
END_CASE

```

6.3.3 ICMP

Das Internet Control Message Protocol (ICMP) dient in Netzwerken zum Austausch von Informations- und Fehlermeldungen über das Internet-Protokoll (IP). Es stellt unter anderem einen Mechanismus zum einfachen Testen von IP-Geräten. Dieses Ping sendet ein kurzes Paket an eine Gegenstelle. Die Gegenstelle antwortet mit dem gleichen Inhalt. Empfängt der Absender das Paket der Gegenstelle, war der Ping erfolgreich.

Asynchroner Funktionsbaustein für ICMP

- [IL_PingAsync auf Seite 304](#),

Synchrone Funktion für ICMP

- [IL_Ping auf Seite 306](#),

6.3.4 FTP

Das File Transfer Protocol (FTP) dient der Übertragung von Dateien zwischen zwei Netzwerkteilnehmer auf Basis des TCP-Protokoll. Ein FTP-Client verbindet zu einem auf dem Zielrechner laufenden FTP-Server. Nach erfolgreichem Verbindungsaufbau können Dateien sowohl gelesen, als auch geschrieben werden.

In allen Rexroth-Steuerungen wird ein FTP-Server beim Hochlauf der Firmware gestartet und steht somit als Gegenstelle zur Verfügung. Auf Windows basierten Rechnern ist normalerweise kein FTP-Server installiert.

Der FTP-Server läuft normalerweise auf TCP Port 21. Dieser muss von der Steuerung aus erreichbar sein. Eventuell im Netzwerk vorhandene Firewalls müssen entsprechend konfiguriert werden.

Asynchrone Funktionsbausteine für FTP:

- [IL_FTPGetAsync auf Seite 307](#)
- [IL_FTPPutAsync auf Seite 310](#)
- [IL_FTPGetFileAsync auf Seite 313](#)
- [IL_FTPPutFileAsync auf Seite 315](#)

Details zur Realisierung: [Asynchrone FTP-Funktionsbausteine auf Seite 307](#)

6.3.5 TFTP

Viele Peripherie-Geräte nutzen zum Übertragen von Firmware oder der Konfiguration das TFTP-Protokoll (Trivial File Transfer Protocol). Dieses Protokoll lässt sich besonders ressourcenschonend implementieren und wird daher von Geräten mit geringem Speicherausbau verwendet.

In allen Rexroth IndraDrive wird ein TFTP-Server beim Hochlauf gestartet und steht somit als Gegenstelle zur Verfügung. Dieser TFTP-Server wird für Firmwareupdates genutzt. Auf Windows basierten Rechnern ist normalerweise kein TFTP-Server installiert.

TFTP-Server laufen normalerweise auf UDP Port 69. Dieser muss von der Steuerung aus erreichbar sein. Eventuell im Netzwerk vorhandene Firewalls müssen entsprechend konfiguriert werden.

Asynchrone Funktionsbausteine für TFTP:

- [IL_TFTPGetFileAsync auf Seite 318](#)
- [IL_TFTPPutFileAsync auf Seite 321](#)

Details zur Realisierung: [Asynchrone TFTP-Funktionsbausteine auf Seite 318](#)

6.3.6 Zyklische Kommunikation

Die Funktionsbausteine unter *CyclicCommunication* ermöglichen es zyklisch Daten auszutauschen. Sie baut auf die TCP- und UDP-Kommunikationsbausteine der *RIL_SocketComm* auf.

Die auszutauschenden Daten können in eigenen Datenstrukturen angegeben werden.

Neben der zyklischen Kommunikation können Daten auch auf Anforderung ausgetauscht werden.

Asynchrone Funktionsbausteine für zyklischen Datenaustausch

- [IL_TCPcyclic auf Seite 323](#)
- [IL_UDPcyclic auf Seite 328](#)

Details zur Realisierung: [Asynchrone Kommunikationsbausteine zur zyklischen Kommunikation auf Seite 323](#)

6.3.7 Byte-Reihenfolge in IP-basierter Kommunikation

Die Byte-Reihenfolge (engl.: byte order) bezeichnet die Speicherorganisation für einfache Zahlenwerte, in erster Linie die Ablage von ganzzahligen Werten (Integer). Die kleinste adressierbare Einheit ein Byte. Die Speicherung einer Zahl erfolgt in mehreren Bytes, deren Speicheradressen direkt aufeinander folgen.

- Bei der **Big-Endian-First-Byte-Order** (wörtlich "Großes Ende zuerst") wird das Byte mit höchstwertigen Bits (d.h. die signifikantesten Stellen) zuerst gespeichert, d.h. an der kleinsten Speicheradresse abgelegt.
- Bei der **Little-Endian-First-Byte-Order** (wörtlich "Kleines Ende zuerst") wird dagegen das Byte mit den niederwertigsten Bits (d.h. die am wenigsten signifikanten Stellen) an der kleinsten Speicheradresse gespeichert.

Die Byte-Reihenfolge für IP-basierte Kommunikation ist auf Big-Endian festgelegt. Da Steuerungen die Daten entweder in Big-Endian oder in Little-Endian im Speicher ablegen, sind auf das System angepasste Konvertierungsfunktionen notwendig. Die Funktionen werden durch die SysSocket-Bibliothek bereitgestellt.

Die folgenden Funktionen sind verfügbar:

- SysSockHtons: Steuerung zu Netzwerk 16 Bit
- SysSockNtohs: Netzwerk zu Steuerung 16 Bit
- SysSockHtonl: Steuerung zu Netzwerk 32 Bit
- SysSockNtohl: Netzwerk zu Steuerung 32 Bit

Die folgenden Datentypen müssen zur Übertragung über TCP oder UDP konvertiert werden:

Datentyp	Steuerung zu Netzwerk	Netzwerk zu Steuerung
WORD, INT, UINT	SysSockHtons()	SysSockNtohs()
DWORD, DINT, UDINT, TIME	SysSockHtonl()	SysSockNtohl()

Die Datentypen BYTE, CHAR, STRING müssen nicht konvertiert werden.



Bei den Datentypen REAL und LREAL ist die Byte-Reihenfolge nicht vorgeschrieben. Hier kann die richtige Einstellung nur experimentell ermittelt werden.

6.3.8 Security - unsichere Protokolle abschalten

Siehe Dokumentation "MLC 15VRS Funktionsbeschreibung", Kap. "Protokolle - System Security".

6.4 TCP Asynchrone Funktionsbausteine

6.4.1 Einführung und Überblick

Asynchrone TCP-Funktionsbausteine werden in zyklischen Tasks verwendet und müssen zyklisch aufgerufen werden. Normalerweise werden mehrere Aufrufe benötigt, um eine Operation abzuschließen.

Ein Funktionsbaustein startet mit steigender Flanke an "Enable" und arbeitet bis entweder "Done" oder "Error" TRUE sind oder "Enable" zurückgesetzt wird. Um eine neue Operation zu starten, muss "Enable" zurückgesetzt und erneut gesetzt werden.

Eine neue TCP-Verbindung kann mit IL_TCPConnectionAsync oder IL_TCPIInitialAsync aufgebaut werden.

IL_TCPConnectionAsync ruft intern die Funktionsbausteine IL_TCPIInitialAsync und IL_TCPCloseAsync auf und hält das Socket-Handle der Verbindung.

Eine mit IL_TCPIInitialAsync erstellte Verbindung kann mit IL_TCPCloseAsync geschlossen werden.



IL_TCPCloseAsync darf nicht für Sockets aufgerufen werden, die mit IL_TCPConnectionAsync erstellt wurden.

Nachdem der Verbindungsaufbau erfolgt ist, können Daten mit IL_TCPSendAsync und IL_TCPRecvAsync über die Verbindung ausgetauscht werden.



TCP ist ein zeichenstrom-basiertes Protokoll, das die Reihenfolge und Konsistenz der übertragenen Daten sicherstellt. Empfangene Daten werden intern wieder zu einem Zeichenstrom zusammengefügt. Wird eine paket-basierte Kommunikation benötigt, ist der Anwender für das Erkennen des Paketkopfes verantwortlich.

Asynchrone Funktionsbausteine für eine TCP-Verbindung:

- [IL_TCPConnectionAsync auf Seite 269](#)
- [IL_TCPIInitialAsync auf Seite 273](#)
- [IL_TCPSendAsync auf Seite 277](#)
- [IL_TCPRecvAsync auf Seite 279](#)
- [IL_TCPRecvPacketAsync auf Seite 281](#)
- [IL_TCPCloseAsync auf Seite 283](#)

6.4.2 IL_TCPConnectionAsync

Kurzbeschreibung

Der Funktionsbaustein IL_TCPConnectionAsync erstellt entweder einen TCP-Server oder einen TCP-Client in Abhängigkeit des "Server"-Eingangs.

Schnittstellenbeschreibung

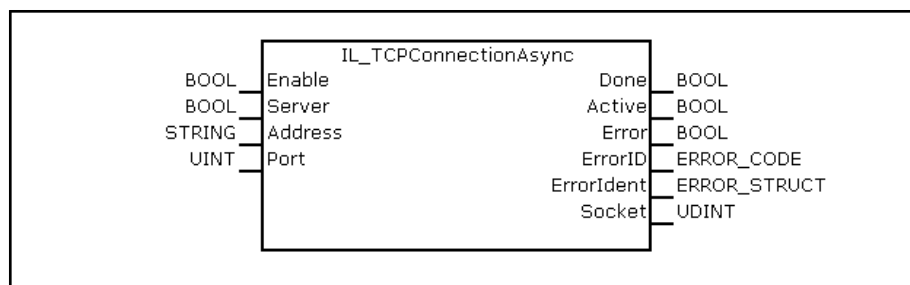


Abb. 6-5: Funktionsbaustein IL_TCPConnectionAsync

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Startet bei steigender Flanke. Binäre Ausgänge werden bei fallender Flanke zurückgesetzt.
	Server	BOOL	TRUE: ein TCP-Server wird erstellt. FALSE: ein TCP-Client wird erstellt.
	Address	STRING	Wenn "Server" gesetzt ist, wird dieser Eingang nicht verwendet und die eigene IP-Adresse benutzt. Ist "Server" FALSE, enthält dieser Eingang die IP-Adresse des zu verbindenden Geräts.
	Port	UINT	Der TCP-Port auf dem auf eingehende Verbindungen gewartet wird oder zu dem verbunden werden soll.
VAR_OUTPUT	Done	BOOL	TRUE zeigt an, dass die Verbindung erfolgreich aufgebaut wurde.
	Active	BOOL	TRUE zeigt an, dass der Funktionsbaustein auf den Verbindungsaufbau wartet.
	Error	BOOL	TRUE zeigt einen Fehler an.
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerinformation (siehe Fehlerbehandlung auf Seite 273)
	Socket	UDINT	Enthält den Socket-Handle der neuen Verbindung.

Tab. 6-1: Interface IL_TCPConnectionAsync

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
Server	BOOL			FALSE	Steigende Flanke an Enable
Address	STRING			"	Steigende Flanke an Enable
Port	UINT	0	65535	0	Steigende Flanke an Enable

Tab. 6-2: Min-, Max- und Default-Werte der Eingänge des Funktionsbausteins IL_TCPConnectionAsync

Funktionsbeschreibung

Der Funktionsbaustein IL_TCPConnectionAsync erstellt entweder einen TCP-Server oder einen TCP-Client in Abhängigkeit des "Server"-Eingangs. Die Abarbeitung beginnt bei steigender Flanke an "Enable". Der Funktionsbaustein muss zyklisch aufgerufen werden, bis "Done" oder "Error" TRUE sind.

Im Server-Modus öffnet der Funktionsbaustein eine Socket auf dem in "Port" angegebenen TCP-Port. Verbindet ein anderer TCP-Client mit diesem Port, wird die Verbindung angenommen. Der zur Verbindung gehörende Socket-

Handle erscheint dann am Ausgang "Socket" und der Ausgang "Done" wird gesetzt. Während auf eine eingehende Verbindung gewartet wird, ist der "Active"-Ausgang gesetzt.

Im Client-Modus verbindet der Funktionsbaustein zur in "Address" angegebenen IP-Adresse auf dem in "Port" angegebenen TCP-Zielpport. Wurde eine Verbindung erfolgreich aufgebaut, erscheint der Socket-Handle am Ausgang "Socket" und der Ausgang "Done" wird gesetzt. Während des Verbindungsaufbaus ist der "Active"-Ausgang gesetzt.

Nach dem Verbindungsaufbau enthält der Ausgang "Socket" den Socket-Handle der Verbindung.

Im Fehlerfall wird der Ausgang "Error" gesetzt. Am Ausgang "Socket" erscheint SOCKET_INVALID. Die Fehlerklasse und der genaue Fehlercode können an den Ausgängen "ErrorID" und "ErrorIdent" abgelesen werden.

Wird "Enable" auf FALSE gesetzt, bevor "Done" TRUE ist, wird keine Verbindung aufgebaut und alle internen Ressourcen frei gegeben. Wird "Enable" auf FALSE gesetzt, nachdem eine Verbindung aufgebaut wurde, wird die Verbindung geschlossen. Mögliche Fehler während des Verbindungsabbaus werden intern behandelt und nicht an den Fehlerausgängen ausgegeben. Nachdem "Enable" auf FALSE gesetzt wurde, wird der Ausgang "Socket" auf SOCKET_INVALID gesetzt.

Der Ausgang "Active" kann als Timeout-Indikator beim Verbindungsaufbau verwendet werden. Typische Timeouts für einen Verbindungsaufbau liegen zwischen 2 und 10 Sekunden.

IL_TCPCloseAsync sollte auf Sockets, die mit IL_TCPConnectionAsync erstellt wurden, nicht verwendet werden.

Signal-Zeit-Diagramme

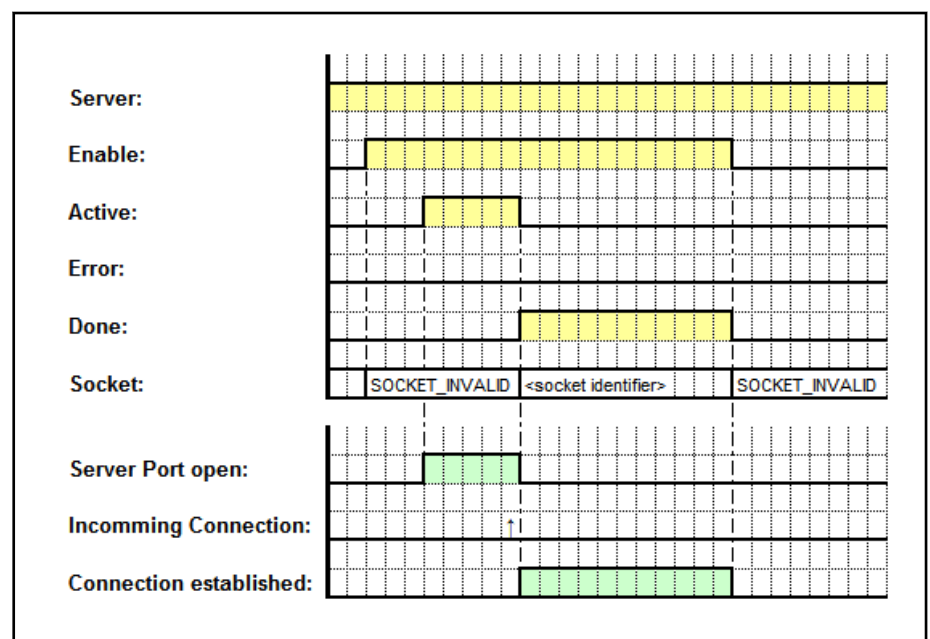


Abb. 6-6: Zeitdiagramm eines TCP-Servers mit eingehender Verbindung

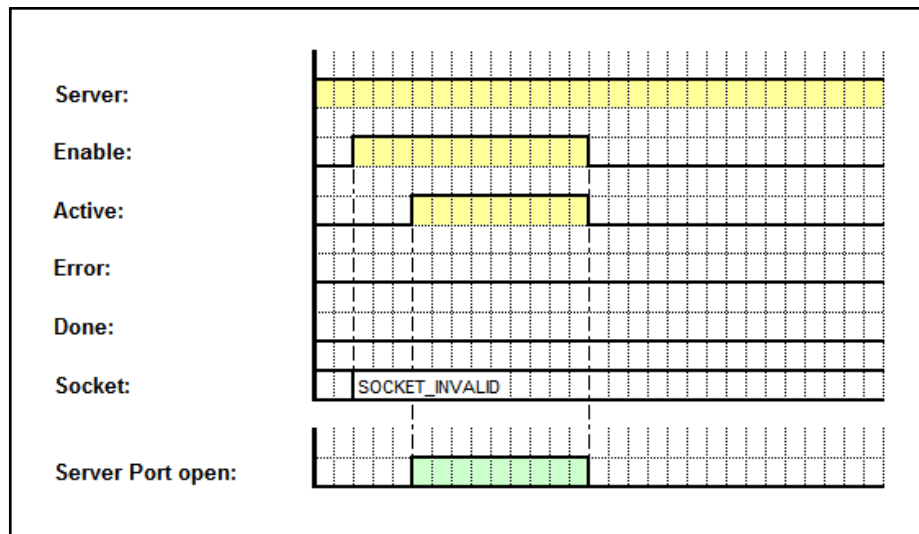


Abb. 6-7: Zeitdiagramm eines TCP-Servers ohne eingehende Verbindung

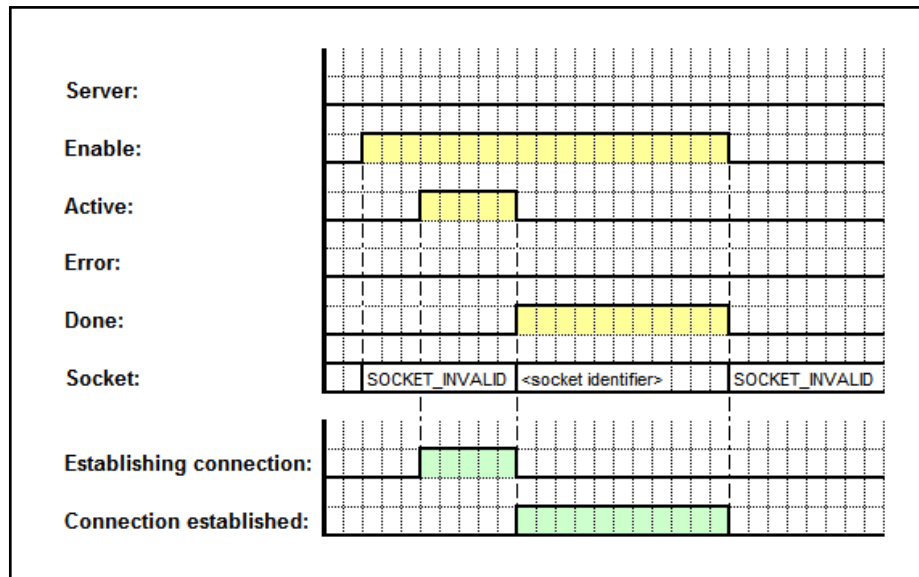


Abb. 6-8: Zeitdiagramm eines TCP-Clients, der auf einen anderen Server verbindet

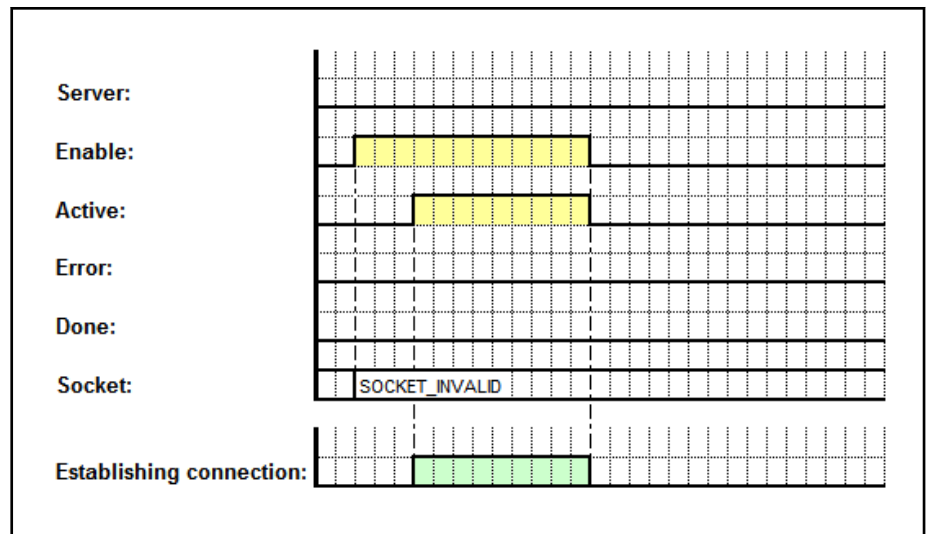


Abb. 6-9: Zeitdiagramm eines TCP-Clients, der nicht auf einen anderen Server verbinden kann

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

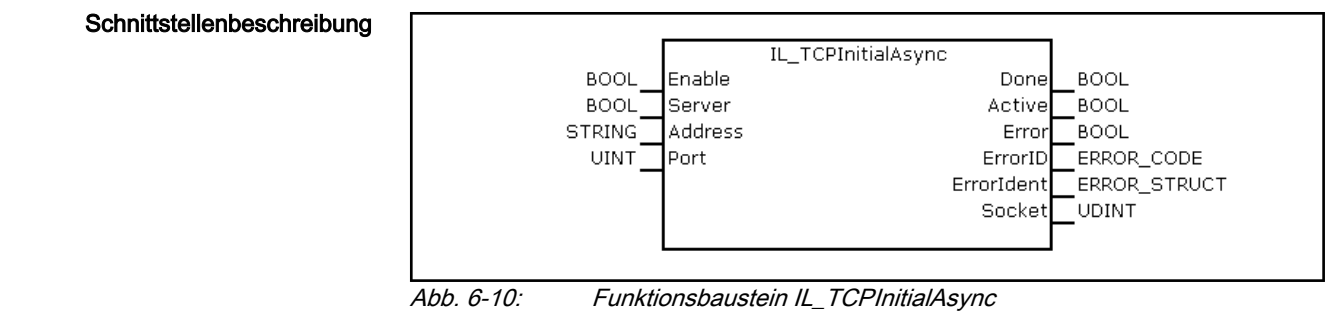
ErrorID	Additional1	Additional2	Beschreibung
OTHER_ERROR	16#00000040	x	Socket konnte nicht erstellt werden
OTHER_ERROR	16#00000041	x	Fehler beim Setzen des I/O Modus
OTHER_ERROR	16#00000042	x	Konnte Socket nicht an Port binden
OTHER_ERROR	16#00000043	x	Socket konnte nicht in den Listen-Modus versetzt werden
OTHER_ERROR	16#00000044	x	Eingehende Verbindung konnte nicht angenommen werden
OTHER_ERROR	16#00000045	x	Fehler beim Verbinden zu einem anderen Gerät
INPUT_INVALID_ERROR	16#00000208	16#00000000	Zieladresse ungültig
COMMUNICATION_ERROR	16#00000209	16#00000000	Die Gegenstelle hat den Verbindungsaufbau abgelehnt
COMMUNICATION_ERROR	16#0000020C	16#00000000	Keine Route zum Zielgerät

Tab. 6-3: Fehlercodes, die von IL_TCPConnectionAsync generiert werden

Weitere Fehlercodes sind in [ETHERNET_TABLE](#), [ERROR_TABLE 16#0150](#) auf Seite 334 aufgeführt.

6.4.3 IL_TCPInitialAsync

Kurzbeschreibung Der Funktionsbaustein IL_TCPInitialAsync erstellt entweder einen TCP-Server oder einen TCP-Client in Abhängigkeit des "Server"-Eingangs.



	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Startet bei steigender Flanke. Binäre Ausgänge werden bei fallender Flanke zurückgesetzt.
	Server	BOOL	TRUE: ein TCP-Server wird erstellt FALSE: ein TCP-Client wird erstellt
	Address	STRING	Wenn "Server" gesetzt ist, wird dieser Eingang nicht verwendet und die eigene IP-Adresse benutzt. Ist "Server" FALSE, enthält dieser Eingang die IP-Adresse des zu verbindenden Geräts.
	Port	UINT	Der TCP-Port auf dem auf eingehende Verbindungen gewartet wird oder zu dem verbunden werden soll.
VAR_OUTPUT	Done	BOOL	TRUE zeigt an, dass die Verbindung erfolgreich aufgebaut wurde.
	Active	BOOL	TRUE zeigt an, dass der Funktionsbaustein auf den Verbindungsaufbau wartet.
	Error	BOOL	TRUE zeigt einen Fehler an.
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerinformation (siehe Fehlerbehandlung auf Seite 277)
	Socket	UDINT	Enthält den Socket-Handle der neuen Verbindung.

Tab. 6-4: Interface IL_TCPInitialAsync

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
Server	BOOL			FALSE	Steigende Flanke an Enable
Address	STRING			"	Steigende Flanke an Enable
Port	UINT	0	65535	0	Steigende Flanke an Enable

Tab. 6-5: Min-, Max- und Default-Werte der Eingänge des Funktionsbausteins IL_TCPInitialAsync

Funktionsbeschreibung

Der Funktionsbaustein IL_TCPInitialAsync erstellt entweder einen TCP-Server oder einen TCP-Client in Abhängigkeit des "Server"-Eingangs. Die Abarbeitung beginnt bei steigender Flanke an "Enable". Der Funktionsbaustein muss zyklisch aufgerufen werden, bis "Done" oder "Error" TRUE sind.

Im Server-Modus öffnet der Funktionsbaustein eine Socket auf dem in "Port" angegebenen TCP-Port. Verbindet sich ein anderer TCP-Client mit diesem Port, wird die Verbindung angenommen. Der zur Verbindung gehörende So-

cket-Handle erscheint dann am Ausgang "Socket" und der Ausgang "Done" wird gesetzt. Während auf eine eingehende Verbindung gewartet wird, ist der "Active"-Ausgang gesetzt.

Im Client-Modus verbindet der Funktionsbaustein zur, in "Address", angegebenen IP-Adresse auf dem, in "Port", angegebenen TCP-Zielport. Wurde eine Verbindung erfolgreich aufgebaut, erscheint der Socket-Handle am Ausgang "Socket" und der Ausgang "Done" wird gesetzt. Während des Verbindungsaufbaus ist der "Active"-Ausgang gesetzt.

Nach dem Verbindungsaufbau enthält der Ausgang "Socket" den Socket-Handle der Verbindung.

Im Fehlerfall wird der Ausgang "Error" gesetzt. Am Ausgang "Socket" erscheint SOCKET_INVALID. Die Fehlerklasse und der genaue Fehlercode können an den Ausgängen "ErrorID" und "ErrorIdent" abgelesen werden.

Wird "Enable" auf FALSE gesetzt, bevor "Done" TRUE ist, wird keine Verbindung aufgebaut und alle internen Ressourcen frei gegeben. Nachdem "Enable" auf FALSE gesetzt wurde, wird der Ausgang "Socket" auf SOCKET_INVALID gesetzt.

Der Ausgang "Active" kann als Timeout-Indikator beim Verbindungsaufbau verwendet werden. Typische Timeouts für einen Verbindungsaufbau liegen zwischen 2 und 10 Sekunden.

Der Funktionsbaustein IL_TCPCloseAsync muss verwendet werden, um Verbindungen, die mit IL_TCPInitialAsync erzeugt wurden, zu schließen.

Signal-Zeit-Diagramme

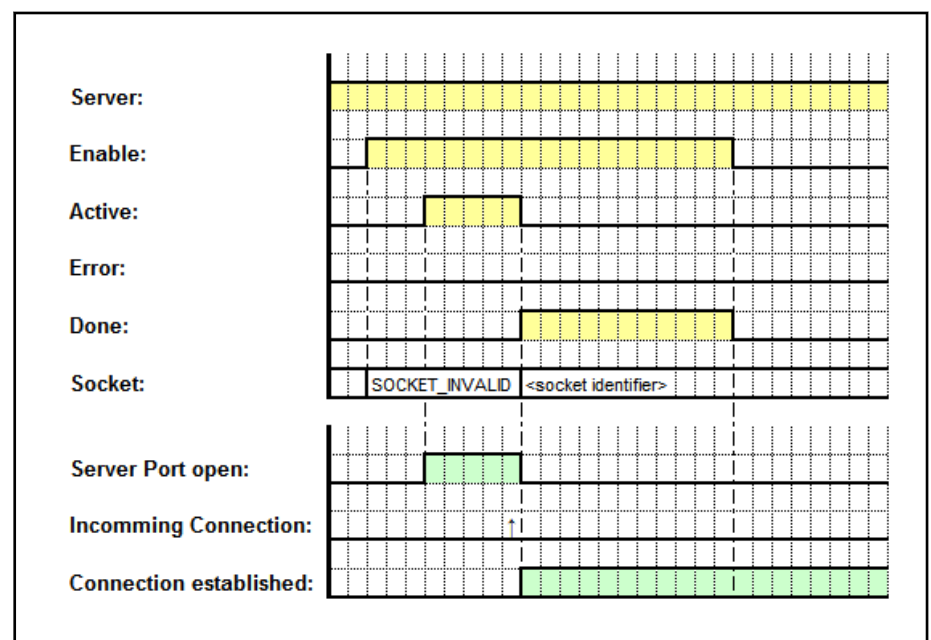


Abb. 6-11: Zeitdiagramm eines TCP-Servers mit eingehender Verbindung

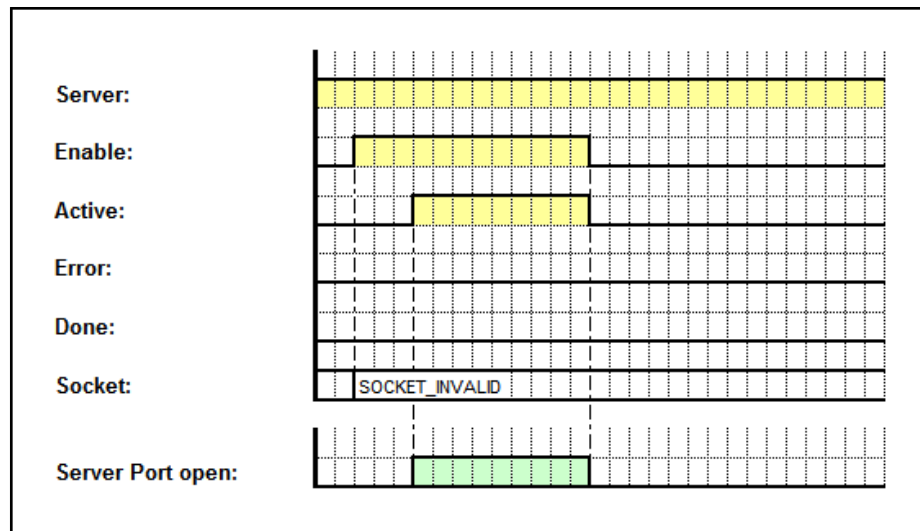


Abb. 6-12: Zeitdiagramm eines TCP-Servers ohne eingehende Verbindung

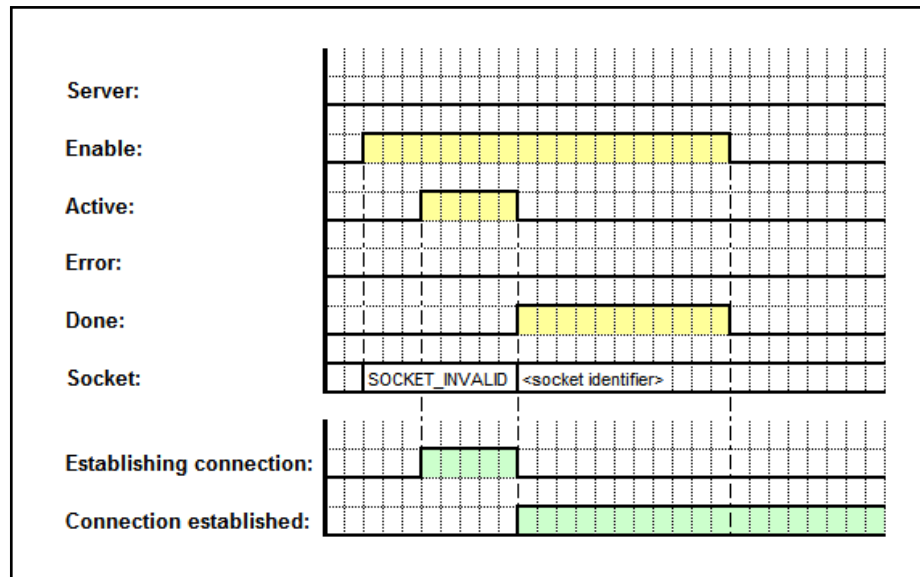


Abb. 6-13: Zeitdiagramm eines TCP-Clients, der auf einen anderen Server verbindet

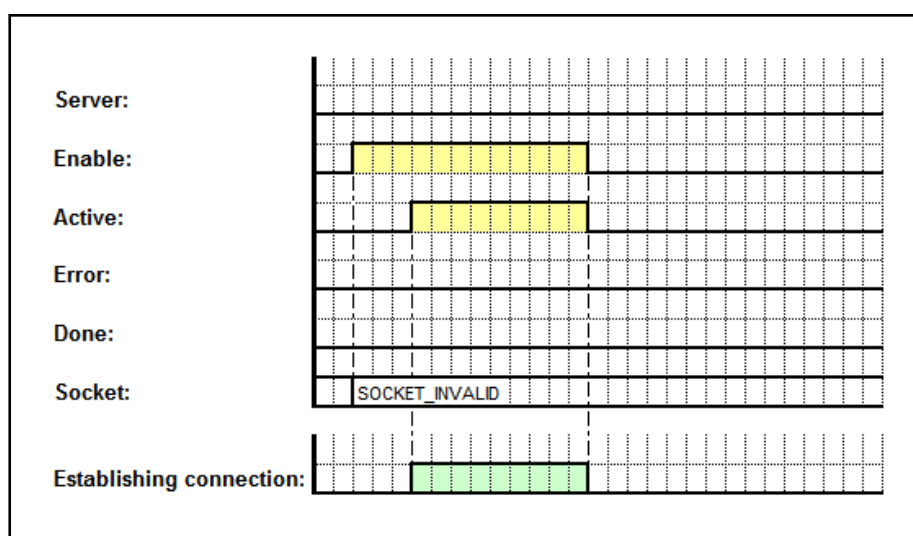


Abb. 6-14: Zeitdiagramm eines TCP-Clients, der nicht auf einen anderen Server verbinden kann

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
OTHER_ERROR	16#00000040	x	Socket konnte nicht erstellt werden
OTHER_ERROR	16#00000041	x	Fehler beim Setzen des I/O Modus
OTHER_ERROR	16#00000042	x	Konnte Socket nicht an Port binden
OTHER_ERROR	16#00000043	x	Socket konnte nicht in den Listen-Modus versetzt werden
OTHER_ERROR	16#00000044	x	Eingehende Verbindung konnte nicht angenommen werden.
OTHER_ERROR	16#00000045	x	Fehler beim Verbinden zu einem anderen Gerät
INPUT_INVALID_ERROR	16#00000208	16#00000000	Zieladresse ungültig
COMMUNICATION_ERROR	16#00000209	16#00000000	Gegenstelle hat den Verbindungsaufbau abgelehnt
COMMUNICATION_ERROR	16#0000020C	16#00000000	Keine Route zum Zielgerät

Tab. 6-6: Fehlercodes, die von IL_TCPInitialAsync generiert werden

Weitere Fehlercodes sind in [ETHERNET_TABLE](#), [ERROR_TABLE 16#0150](#) auf Seite 334 aufgeführt.

6.4.4 IL_TCPSendAsync

Kurzbeschreibung Der Funktionsbaustein IL_TCPSendAsync sendet Daten über eine Verbindung, die zuvor mit IL_TCPConnectionAsync oder IL_TCPInitialAsync aufgebaut wurde.

Schnittstellenbeschreibung

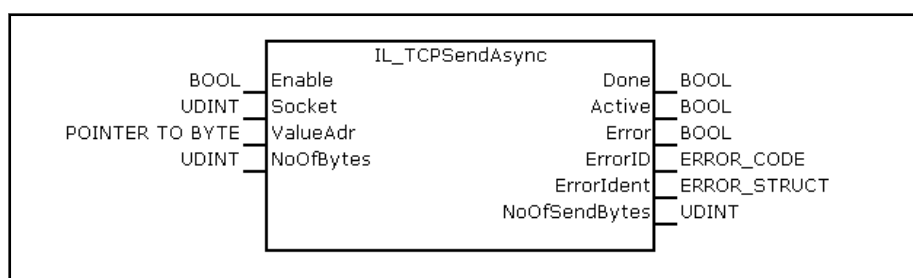


Abb. 6-15: Funktionsbaustein IL_TCPSendAsync

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Startet bei steigender Flanke Binäre Ausgänge werden bei fallender Flanke zurückgesetzt.
	Socket	UDINT	Socket-Handle von IL_TCPConnectionAsync oder IL_TCPIInitialAsync
	ValueAdr	POINTER TO BYTE	Startadresse der zu sendenden Daten
	NoOfBytes	UDINT	Anzahl der zu sendenden Zeichen
VAR_OUTPUT	Done	BOOL	TRUE zeigt an, dass alle Daten erfolgreich gesendet wurden.
	Active	BOOL	TRUE zeigt an, dass der Funktionsbaustein Daten sendet.
	Error	BOOL	TRUE zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerinformation (siehe Fehlerbehandlung auf Seite 279)
	NoOfSendBytes	UDINT	Anzahl der tatsächlich gesendeten Bytes.

Tab. 6-7: Interface IL_TCPSendAsync

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
Socket	UDINT	0	16#FFFFFFFF	SOCKET_INVALID	Steigende Flanke an Enable
ValueAdr	POINTER TO BYTE			0	Steigende Flanke an Enable
NoOfBytes	UDINT	0	16#FFFFFFFF	0	Steigende Flanke an Enable

Tab. 6-8: Min-, Max- und Default-Werte der Eingänge des Funktionsbausteins IL_TCPSendAsync

Funktionsbeschreibung

Der Funktionsbaustein IL_TCPSendAsync sendet Daten über eine Verbindung, die zuvor mit IL_TCPConnectionAsync oder IL_TCPIInitialAsync erstellt wurde.

Die Übertragung beginnt mit steigender Flanke an "Enable".



Der Funktionsbaustein muss zyklisch aufgerufen werden bis "Done" oder "Error" TRUE ist.

Der zu sendende Puffer wird an Eingang "ValueAdr" angelegt. Die Anzahl der zu sendenden Zeichen wird im Eingang "NoOfBytes" übergeben.

Wird "Enable" zurückgesetzt bevor "Done" TRUE ist, wird die Übertragung unterbrochen. Die Anzahl der bereits gesendeten Bytes kann am Ausgang "NoOfSentBytes" abgelesen werden.

Der Ausgang "Active" ist während der Übertragung der Daten gesetzt. "Active" kann als Timeout-Indikator verwendet werden. Typische Sende-Timeouts liegen zwischen 1 und 2 Sekunden. Während der Übertragung ("Active" ist TRUE), kann die Anzahl der bereits gesendeten Bytes am Ausgang "NoOfSentBytes" abgelesen werden. In der Regel werden alle Bytes in einem Aufruf gesendet.



IL_TCPSendAsync stellt sicher, dass alle Bytes gesendet wurden, bevor "Done" gesetzt wird.

IL_TCPSendAsync erkennt einen Verbindungsabbruch. Dieser wird mit einem entsprechenden Fehlercode angezeigt.

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#00000004	16#00000000	Eingang "ValueAdr" ist nicht gesetzt
INPUT_RANGE_ERROR	16#00000005	16#00000000	Eingang "NoOfBytes" ist 0
INPUT_INVALID_ERROR	16#00000202	16#00000000	Socket-Handle ist ungültig
COMMUNICATION_ERROR	16#0000020E	16#00000000	Verbindung wurde geschlossen
COMMUNICATION_ERROR	16#00000211	16#00000000	Gegenstelle hat die Verbindung geschlossen

Tab. 6-9: Fehlercodes, die von IL_TCPSendAsync generiert werden

Weitere Fehlercodes sind in [ETHERNET_TABLE](#), [ERROR_TABLE 16#0150](#) auf Seite 334 aufgeführt.

6.4.5 IL_TCPRecvAsync

Kurzbeschreibung Der Funktionsbaustein IL_TCPRecvAsync empfängt Daten über eine Verbindung, die zuvor mit IL_TCPConnectionAsync oder IL_TCPInitialAsync aufgebaut wurde.

Schnittstellenbeschreibung

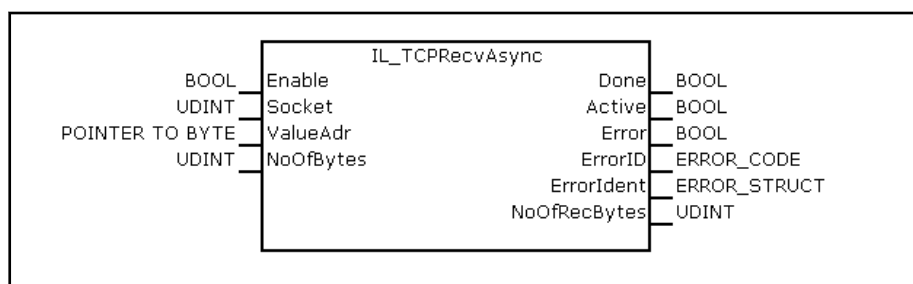


Abb. 6-16: Funktionsbaustein IL_TCPRecvAsync

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Startet bei steigender Flanke. Binäre Ausgänge werden bei fallender Flanke zurückgesetzt.
	Socket	UDINT	Socket-Handle von IL_TCPConnectionAsync oder IL_TCPInitialA- sync
	ValueAdr	POINTER TO BYTE	Startadresse des Empfangspuffers
	NoOfBytes	UDINT	Größe des Empfangsdatenpuffers

	Name	Typ	Beschreibung
VAR_OUTPUT	Done	BOOL	TRUE zeigt an, dass Daten erfolgreich empfangen wurden.
	Active	BOOL	TRUE zeigt an, dass der Funktionsbaustein auf ankommende Daten wartet.
	Error	BOOL	TRUE zeigt einen Fehler an.
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerinformation (siehe Fehlerbehandlung auf Seite 281)
	NoOfRecBytes	UDINT	Anzahl der empfangenen Bytes

Tab. 6-10: Interface IL_TCPRecvAsync

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
Socket	UDINT	0	16#FFFFFFFF	SOCKET_INVALID	Steigende Flanke an Enable
ValueAdr	POINTER TO BYTE			0	Steigende Flanke an Enable
NoOfBytes	UDINT	0	16#FFFFFFFF	0	Steigende Flanke an Enable

Tab. 6-11: Min-, Max- und Default-Werte der Eingänge des Funktionsbausteins IL_TCPRecvAsync

Funktionsbeschreibung

Der Funktionsbaustein IL_TCPRecvAsync empfängt Daten über eine Verbindung, die zuvor mit IL_TCPConnectionAsync oder IL_TCPInitialAsync aufgebaut wurde.

Die Übertragung beginnt mit steigender Flanke an "Enable".



Der Funktionsbaustein muss zyklisch aufgerufen werden bis "Done" oder "Error" TRUE ist.

Wird "Enable" zurückgesetzt bevor "Done" TRUE ist, wird die Übertragung unterbrochen. Bei einem solchen Abbruch werden keine Daten entgegengenommen. "NoOfRecBytes" ist demzufolge 0.

Der Ausgang "Active" ist während der Übertragung der Daten gesetzt.

Active bleibt TRUE bis Daten empfangen wurden.

Die Anzahl der empfangenen Bytes kann kleiner sein, als die Puffergröße.

Soll genau die Puffergröße empfangen werden, ist der Funktionsbaustein [IL_TCPRecvPacketAsync auf Seite 281](#) zu verwenden. "Active" kann als Timeout-Indikator verwendet werden.

Falls die Anzahl der von der Gegenstelle gesendeten Bytes größer als die Puffergröße ist, werden nur die in "NoOfBytes" eingegebene Anzahl an Bytes in den Puffer kopiert. Die übrigen Bytes werden beim nächsten Aufruf kopiert.



"Active" kann als Timeout-Indikator verwendet werden. Typische Empfangs-Timeouts liegen zwischen 1 und 2 Sekunden.

IL_TCPRecvAsync erkennt einen Verbindungsabbruch. Dieser wird mit einem entsprechenden Fehlercode angezeigt.

Signal-Zeit-Diagramm

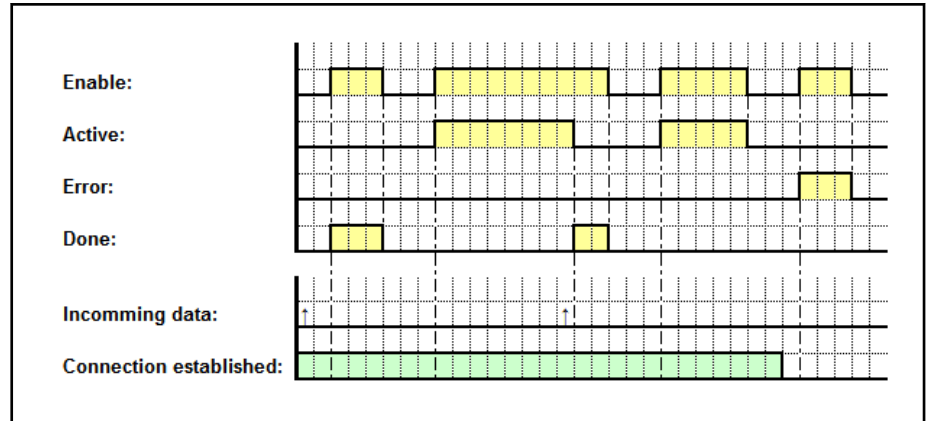


Abb. 6-17: Zeitdiagramm eines Datenempfangs

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#00000004	16#00000000	Eingang "ValueAdr" ist nicht gesetzt
INPUT_RANGE_ERROR	16#00000005	16#00000000	Eingang "NoOfBytes" ist 0
INPUT_INVALID_ERROR	16#00000202	16#00000000	Socket-Handle ist ungültig.
COMMUNICATION_ERROR	16#0000020E	16#00000000	Verbindung wurde geschlossen.
COMMUNICATION_ERROR	16#00000211	16#00000000	Gegenstelle hat die Verbindung geschlossen.

Tab. 6-12: Fehlercodes, die von IL_TCPRecvAsync generiert werden

Weitere Fehlercodes sind in [ETHERNET_TABLE](#), [ERROR_TABLE 16#0150](#) auf Seite 334 aufgeführt.

6.4.6 IL_TCPRecvPacketAsync

Kurzbeschreibung Der Funktionsbaustein IL_TCPRecvPacketAsync empfängt eine festgelegte Anzahl Bytes über eine Verbindung, die zuvor mit IL_TCPConnectionAsync oder IL_TCPInitialAsync aufgebaut wurde.

Schnittstellenbeschreibung

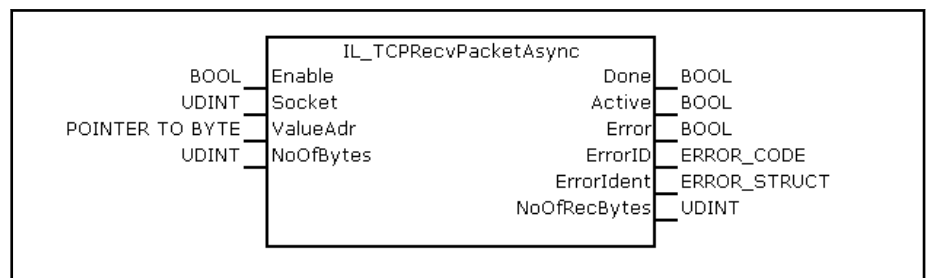


Abb. 6-18: Funktionsbaustein IL_TCPRecvPacketAsync

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Startet bei steigender Flanke Binäre Ausgänge werden bei fallender Flanke zurückgesetzt.
	Socket	UDINT	Socket-Handle von IL_TCPConnectionAsync oder IL_TCPInitialAsync
	ValueAdr	POINTER TO BYTE	Startadresse des Empfangspuffers
	NoOfBytes	UDINT	Anzahl der zu empfangenden Bytes
VAR_OUTPUT	Done	BOOL	TRUE zeigt an, dass Daten erfolgreich empfangen wurden.
	Active	BOOL	TRUE zeigt an, dass der Funktionsbaustein auf ankommende Daten wartet.
	Error	BOOL	TRUE zeigt einen Fehler an.
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerinformation (siehe Fehlerbehandlung auf Seite 283)
	NoOfRecBytes	UDINT	Anzahl der in den Empfangspuffer kopierten Bytes

Tab. 6-13: Interface IL_TCPRecvPacketAsync

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
Socket	UDINT	0	16#FFFFFFFF	SOCKET_INVALID	Steigende Flanke an Enable
ValueAdr	POINTER TO BYTE			0	Steigende Flanke an Enable
NoOfBytes	UDINT	0	16#FFFFFFFF	0	Steigende Flanke an Enable

Tab. 6-14: Min-, Max- und Default-Werte der Eingänge des Funktionsbausteins IL_TCPRecvPacketAsync

Funktionsbeschreibung

Der Funktionsbaustein IL_TCPRecvPacketAsync empfängt eine **festgelegte Anzahl** Bytes über eine Verbindung, die zuvor mit IL_TCPConnectionAsync oder IL_TCPInitialAsync aufgebaut wurde.

Die Übertragung beginnt mit steigender Flanke an "Enable".



Der Funktionsbaustein muss zyklisch aufgerufen werden bis "Done" oder "Error" TRUE ist.

Wird "Enable" zurückgesetzt, bevor "Done" TRUE ist, wird die Übertragung abgebrochen. Bei einem solchen Abbruch werden keine Daten entgegengenommen. "NoOfRecBytes" ist demzufolge 0.

Der Ausgang "Active" ist während der Übertragung der Daten gesetzt.

1. Ist die Datenmenge kleiner als die Puffergröße wird diese komplett übertragen, die tatsächliche Menge ist NoOfRecBytes zu entnehmen, ACTIVE bleibt TRUE, solange der NoOfRecBytes < NoOfBytes ist.
2. Ist die Datenmenge gleich der Puffergröße wird diese übertragen und NoOfRecBytes = NoOfBytes, ACTIVE wird auf FALSE und DONE auf TRUE gesetzt.

3. Ist die Datenmenge größer als die Puffergröße, werden nur NoOfBytes Bytes übertragen, ACTIVE wird auf FALSE und DONE auf TRUE gesetzt.

Beim nächsten Aufruf des Bausteines werden die Restbytes übertragen ggf. gefolgt von den Daten des nächsten zu übertragenden Paketes.

Achtung, es besteht die Gefahr des Pufferüberlaufes der gesamten Übertragungsstrecke!

IL_TCPRecvPacketAsync erkennt einen Verbindungsabbruch. Dieser wird mit einem entsprechenden Fehlercode angezeigt.

Signal-Zeit-Diagramm

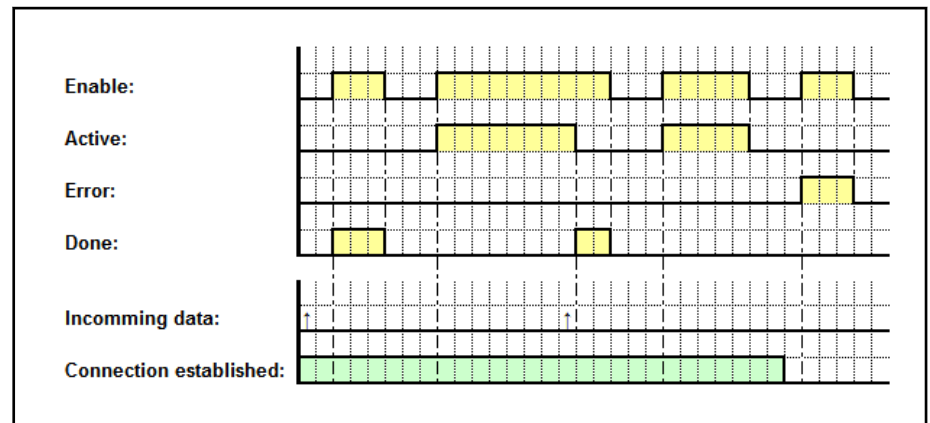


Abb. 6-19: Zeitdiagramm eines Datenempfangs

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#00000004	16#00000000	Eingang "ValueAdr" ist nicht gesetzt
INPUT_RANGE_ERROR	16#00000005	16#00000000	Eingang "NoOfBytes" ist 0
INPUT_INVALID_ERROR	16#00000202	16#00000000	Socket-Handle ist ungültig.
COMMUNICATION_ERROR	16#0000020E	16#00000000	Verbindung wurde geschlossen.
COMMUNICATION_ERROR	16#00000211	16#00000000	Gegenstelle hat die Verbindung geschlossen.

Tab. 6-15: Fehlercodes, die von IL_TCPRecvPacketAsync generiert werden

Weitere Fehlercodes sind in [ETHERNET_TABLE](#), [ERROR_TABLE 16#0150](#) auf Seite 334 aufgeführt.

6.4.7 IL_TCPCloseAsync

Kurzbeschreibung Der Funktionsbaustein IL_TCPCloseAsync schließt eine Verbindung, die zuvor mit IL_TCPInitialAsync erstellt wurde.

Schnittstellenbeschreibung

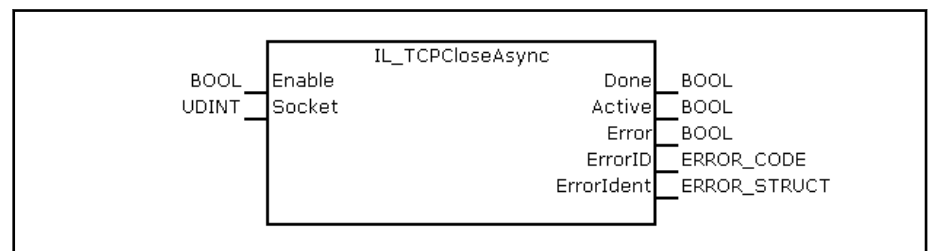


Abb. 6-20: IL_TCPCloseAsync Funktionsbaustein

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Startet bei steigender Flanke. Binäre Ausgänge werden bei fallender Flanke zurückgesetzt.
	Socket	UDINT	Socket-Handle von IL_TCPInitialAsync
VAR_OUTPUT	Done	BOOL	TRUE zeigt an, dass die Socket erfolgreich geschlossen wurde.
	Active	BOOL	TRUE zeigt an, dass die Socket geschlossen wird.
	Error	BOOL	TRUE zeigt einen Fehler an.
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerinformation (siehe Fehlerbehandlung auf Seite 284)

Tab. 6-16: Interface IL_TCPCloseAsync

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
Socket	UDINT	0	16#FFFFFFFF	SOCKET_INVALID	Steigende Flanke an Enable

Tab. 6-17: Min-, Max- und Default-Werte der Eingänge des Funktionsbausteins IL_TCPCloseAsync

Funktionsbeschreibung

Der Funktionsbaustein IL_TCPCloseAsync schließt eine Verbindung, die zuvor mit IL_TCPInitialAsync erstellt wurde.

Der Funktionsbaustein benötigt eventuell mehrere Aufrufe, um die Operation zu beenden. Die Operation ist erfolgreich abgeschlossen, wenn der Ausgang "Done" gesetzt ist. Während die Operation läuft, ist "Active" gesetzt. Tritt ein Fehler auf, so ist der Ausgang "Error" gesetzt. Die Ausgänge "ErrorID" und "ErrorIdent" enthalten dann nähere Informationen zu dem Fehler.



Der Funktionsbaustein muss nach steigender Flanke an "Enable" so lange aufgerufen werden, bis entweder "Done" oder "Error" gesetzt sind.

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR	16#00000202	16#00000000	Socket-Handle ist ungültig
COMMUNICATION_ERROR	16#00000211	16#00000000	Die Gegenstelle hat die Verbindung geschlossen

Tab. 6-18: Fehlercodes, die von IL_TCPCloseAsync generiert werden

Weitere Fehlercodes sind in [ETHERNET_TABLE](#), [ERROR_TABLE 16#0150](#) auf Seite 334 aufgeführt.

6.5 TCP Synchrone Funktionen

6.5.1 Einführung und Überblick

Die synchronen TCP-Funktionen sind als blockierende Aufrufe implementiert, d.h. sie kehren nicht zurück bis die Operation abgeschlossen wurde oder ein

Fehler auftrat. Diese Funktionen sollten nur in frei laufenden Tasks verwendet werden.

IL_TCPInitial wird verwendet, um eine Verbindung aufzubauen und gibt einen Socket-Handle zurück. Ist der Socket-Handle gültig, wurde die Verbindung aufgebaut. Nach dem Verbindungsaufbau können Daten mit IL_TCPSend und IL_TCPRecv gesendet und empfangen werden.

IL_TCPClose schließt eine bereits bestehende Verbindung und gibt den Socket-Handle frei.



TCP ist ein zeichenstrom-basiertes Protokoll, das die Reihenfolge und Konsistenz der übertragenen Daten sicherstellt. Empfangene Daten werden intern wieder zu einem Zeichenstrom zusammengefügt. Wird eine paket-basierte Kommunikation benötigt, ist der Anwender verantwortlich für das Erkennen des Paketkopfes.

Synchrone Funktionen für eine TCP-Verbindung:

- [IL_TCPInitial auf Seite 285](#)
- [IL_TCPSend auf Seite 286](#)
- [IL_TCPRecv auf Seite 287](#)
- [IL_TCPRecvPacket auf Seite 288](#)
- [IL_TCPClose auf Seite 290](#)

6.5.2 IL_TCPInitial

Kurzbeschreibung

Die Funktion IL_TCPInitial erstellt entweder einen TCP-Server oder einen TCP-Client in Abhängigkeit des "bServer"-Parameters.

Schnittstellenbeschreibung

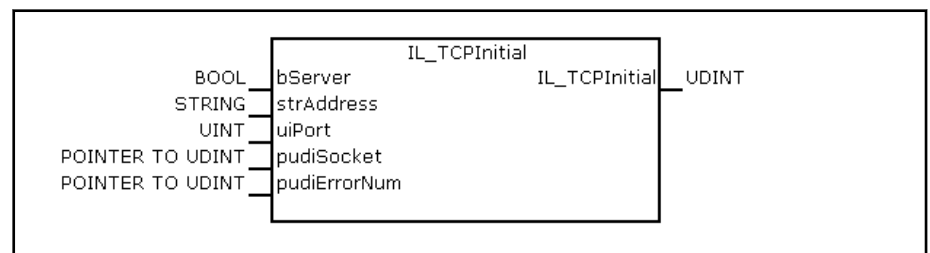


Abb. 6-21: IL_TCPInitial Funktion

	Name	Typ	Beschreibung
VAR_INPUT	bServer	BOOL	TRUE: ein TCP-Server wird erstellt FALSE: ein TCP-Client wird erstellt
	strAddress	STRING	Wenn "bServer" gesetzt ist, wird dieser Eingang nicht verwendet und die eigene IP-Adresse benutzt. Ist "bServer" FALSE, enthält dieser Eingang die IP-Adresse des zu verbindenden Geräts.
	uiPort	UINT	Der TCP-Port auf dem auf eingehende Verbindungen gewartet wird oder zu dem verbunden werden soll.
	pudiSocket	POINTER TO UDINT	Enthält den Socket-Handle der neuen Verbindung.
	pudiErrorNum	POINTER TO UDINT	Enthält den Fehlercode (siehe Fehlerbehandlung auf Seite 286).
RETURN		UDINT	0, falls eine Verbindung erfolgreich aufgebaut wurde; sonst INVALID_SOCKET.

Tab. 6-19: IL_TCPInitial Interface

Funktionsbeschreibung Die Funktion IL_TCPInitial erstellt entweder einen TCP-Server oder einen TCP-Client in Abhängigkeit des "bServer"-Parameters.

Im Server-Modus öffnet die Funktion eine Socket auf dem in "uiPort" angegebenen TCP-Port. Verbindet ein anderer TCP-Client mit diesem Port, wird die Verbindung angenommen. Der zur Verbindung gehörende Socket-Handle wird in "pdiSocket" gespeichert und die Funktion kehrt mit dem Rückgabewert 0 zurück.

Im Client-Modus verbindet die Funktion zur in "strAddress" angegebenen IP-Adresse auf dem in "uiPort" angegebenen TCP-Zielpport. Wurde eine Verbindung erfolgreich aufgebaut, wird der Socket-Handle in "pdiSocket" abgelegt und die Funktion kehrt mit dem Rückgabewert 0 zurück. Kann innerhalb von 2 Sekunden keine Verbindung aufgebaut werden, kehrt die Funktion mit dem Rückgabewert SOCKET_INVALID zurück und "pudiErrorNum" enthält einen entsprechenden Fehlercode.

Tritt ein Fehler auf, kehrt die Funktion mit SOCKET_INVALID zurück und legt den Fehlercode in "pudiErrorNum" ab.

Fehlerbehandlung Die Funktion nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150.

Die Funktion gibt die folgenden Fehlercodes in "pudiErrorNum" (Additional1) zurück:

ErrorID	Additional1	Additional2	Beschreibung
OTHER_ERROR	16#00000040	-	Socket konnte nicht erstellt werden
OTHER_ERROR	16#00000041	-	Fehler beim Setzen des I/O Modus
OTHER_ERROR	16#00000042	-	Konnte Socket nicht an Port binden
OTHER_ERROR	16#00000043	-	Socket konnte nicht in den Listen-Modus versetzt werden
OTHER_ERROR	16#00000044	-	Eingehende Verbindung konnte nicht angenommen werden.
OTHER_ERROR	16#00000045	-	Fehler beim Verbinden zu einem anderen Gerät
INPUT_INVALID_ERROR	16#00000208	-	Zieladresse ungültig
COMMUNICATION_ERROR	16#00000209	-	Gegenstelle hat den Verbindungsaufbau abgelehnt
COMMUNICATION_ERROR	16#0000020C	-	Keine Route zum Zielgerät

Tab. 6-20: IL_TCPInitial Fehlercodes

Weitere Fehlercodes sind in [ETHERNET_TABLE](#), [ERROR_TABLE 16#0150](#) auf Seite 334 aufgeführt.

6.5.3 IL_TCPSend

Kurzbeschreibung Die Funktion IL_TCPSend sendet Daten über eine Verbindung, die zuvor mit IL_TCPInitial erstellt wurde.

Diese Funktion kehrt nicht zurück, bis entweder alle Daten übertragen wurden oder ein Fehler auftritt.

Schnittstellenbeschreibung

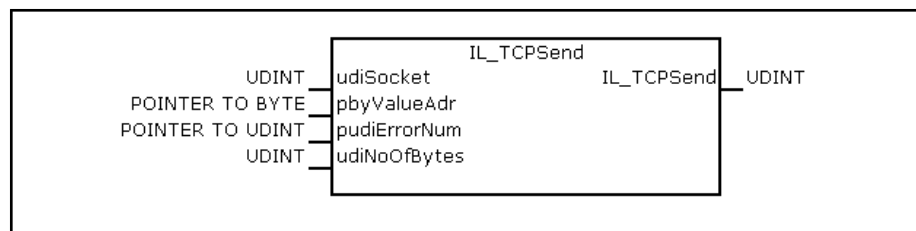


Abb. 6-22: IL_TCPSend Funktion

	Name	Typ	Beschreibung
VAR_INPUT	udiSocket	UDINT	Socket-Handle von IL_TCPInitial
	pbyValueAdr	POINTER TO BYTE	Startadresse der zu sendenden Daten.
	pudiErrorNum	POINTER TO UDINT	Enthält den Fehlercode (siehe Fehlerbehandlung auf Seite 287).
	udiNoOfBytes	UDINT	Anzahl der zu sendenden Zeichen
RETURN		UDINT	Anzahl der tatsächlich gesendeten Bytes. 0 zeigt an, dass die Verbindung geschlossen wurde oder ein Fehler auftrat.

Tab. 6-21: IL_TCPSend Interface

Funktionsbeschreibung Die Funktion IL_TCPSend sendet Daten über eine Verbindung, die zuvor mit IL_TCPInitial erstellt wurde. Diese Funktion kehrt nicht zurück, bis entweder alle Daten übertragen wurden oder ein Fehler auftrat.

Der zu sendende Puffer wird in "pbyValueAdr" angelegt. Die Anzahl der zu sendenden Zeichen wird in "udiNoOfBytes" übergeben.

Nach erfolgreichem Abschluss der Übertragung gibt diese Funktion die Anzahl der übertragenen Zeichen zurück. Wird 0 zurückgegeben, wurde die Verbindung geschlossen bevor Daten gesendet werden konnten. Im Fehlerfall wird der Fehlercode in "pudiErrorNum" abgelegt und die Funktion kehrt mit dem Rückgabewert 0 zurück.

Fehlerbehandlung Die Funktion nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150. Die Funktion gibt die folgenden Fehlercodes in "pudiErrorNum" (Additional1) zurück:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#00000004	-	Eingang "ValueAdr" ist 0
INPUT_RANGE_ERROR	16#00000005	-	Eingang "NoOfBytes" ist 0
INPUT_INVALID_ERROR	16#00000202	-	Socket-Handle ist ungültig.
COMMUNICATION_ERROR	16#0000020E	-	Verbindung wurde geschlossen.
COMMUNICATION_ERROR	16#00000211	-	Gegenstelle hat die Verbindung geschlossen.

Tab. 6-22: IL_TCPSend Fehlercodes

Weitere Fehlercodes sind in [ETHERNET_TABLE, ERROR_TABLE 16#0150 auf Seite 334](#) aufgeführt.

6.5.4 IL_TCPRecv

Kurzbeschreibung Die Funktion IL_TCPRecv empfängt Daten über eine Verbindung, die zuvor mit IL_TCPInitial erstellt wurde. Diese Funktion kehrt nicht zurück, bis entweder Daten empfangen wurden oder ein Fehler auftrat.

Schnittstellenbeschreibung

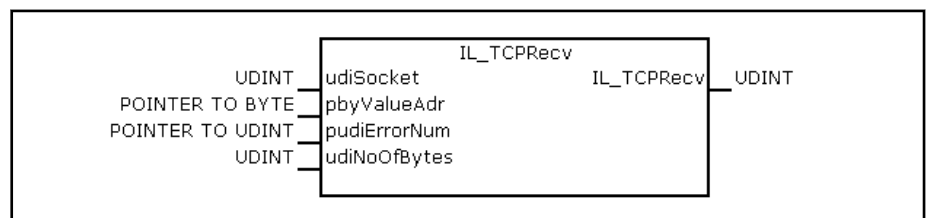


Abb. 6-23: IL_TCPRecv Funktion

	Name	Typ	Beschreibung
VAR_INPUT	udiSocket	UDINT	Socket-Handle von IL_TCPInitial
	pbyValueAdr	POINTER TO BYTE	Startadresse des Empfangsdatenpuffers
	pudiErrorNum	POINTER TO UDINT	Enthält den Fehlercode (siehe Fehlerbehandlung auf Seite 288).
	udiNoOfBytes	UDINT	Größe des Empfangsdatenpuffers
RETURN		UDINT	Anzahl der empfangenen Bytes. 0 zeigt an, dass die Verbindung geschlossen wurde oder ein Fehler auftrat.

Tab. 6-23: IL_TCPRecv Interface

Funktionsbeschreibung

Die Funktion IL_TCPRecv empfängt Daten über eine Verbindung, die zuvor mit IL_TCPInitial erstellt wurde. Diese Funktion kehrt nicht zurück, bis entweder Daten empfangen wurden oder ein Fehler auftrat.

Der Empfangspuffer wird in "pbyValueAdr" angelegt. Die Größe des Empfangspuffers wird in "udiNoOfBytes" übergeben.

Die Funktion IL_TCPRecv empfängt Zeichen bis zur maximalen Größe des angelegten Empfangspuffers. Es können aber auch weniger Zeichen empfangen werden. Soll immer der komplette Puffer gefüllt werden, ist die Funktion [IL_TCPRecvPacket auf Seite 288](#) zu verwenden. Nach erfolgreichem Abschluss der Übertragung gibt diese Funktion die Anzahl der empfangenen Zeichen zurück. Wird 0 zurückgegeben, wurde die Verbindung geschlossen bevor Daten empfangen werden konnten. Im Fehlerfall wird der Fehlercode in "pudiErrorNum" abgelegt und die Funktion kehrt mit dem Rückgabewert 0 zurück.

Werden innerhalb von einer Sekunde keine Daten empfangen, kehrt die Funktion mit dem Rückgabewert 0 zurück und legt den Fehlercode 16#0000020A in "pudiErrorNum" ab.

Fehlerbehandlung

Die Funktion nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150.

Die Funktion gibt die folgenden Fehlercodes in "pudiErrorNum" (Additional1) zurück:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#00000004	-	Eingang "ValueAdr" ist 0
INPUT_RANGE_ERROR	16#00000005	-	Eingang "NoOfBytes" ist 0
INPUT_INVALID_ERROR	16#00000202	-	Socket-Handle ist ungültig.
COMMUNICATION_ERROR	16#0000020A	-	Keine Daten innerhalb des Timeouts empfangen
COMMUNICATION_ERROR	16#0000020E	-	Verbindung wurde geschlossen.
COMMUNICATION_ERROR	16#00000211	-	Gegenstelle hat die Verbindung geschlossen.

Tab. 6-24: IL_TCPRecv Fehlercodes

Weitere Fehlercodes sind in [ETHERNET_TABLE, ERROR_TABLE 16#0150 auf Seite 334](#) aufgeführt.

6.5.5 IL_TCPRecvPacket

Kurzbeschreibung

Die Funktion IL_TCPRecvPacket empfängt Daten über eine Verbindung, die zuvor mit IL_TCPInitial erstellt wurde. Diese Funktion kehrt nicht zurück, bis entweder Daten empfangen wurden oder ein Fehler auftrat.

Schnittstellenbeschreibung

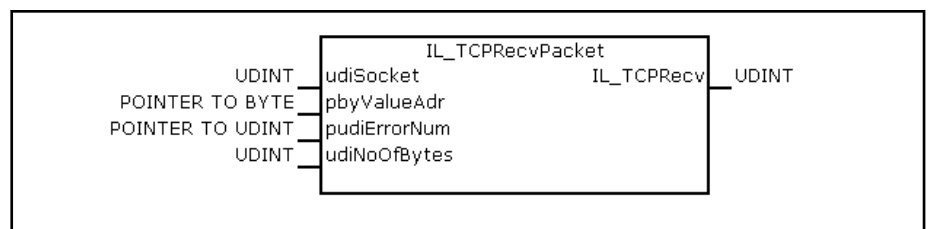


Abb. 6-24: IL_TCPRecvPacket Funktion

	Name	Typ	Beschreibung
VAR_INPUT	udiSocket	UDINT	Socket-Handle von IL_TCPInitial
	pbyValueAdr	POINTER TO BYTE	Startadresse des Empfangsdatenpuffers
	pudiErrorNum	POINTER TO UDINT	enthält den Fehlercode (siehe Fehlerbehandlung auf Seite 289).
	udiNoOfBytes	UDINT	Größe des Empfangsdatenpuffers
RETURN VA-LUE		UDINT	Anzahl der empfangenen Bytes, 0 zeigt an, dass die Verbindung geschlossen wurde oder ein Fehler auftrat.

Tab. 6-25: IL_TCPRecvPacket Interface

Funktionsbeschreibung

Die Funktion IL_TCPRecvPacket empfängt Daten über eine Verbindung, die zuvor mit IL_TCPInitial erstellt wurde. Diese Funktion kehrt nicht zurück, bis entweder Daten empfangen wurden oder ein Fehler auftrat.

Der Empfangspuffer wird in "pbyValueAdr" angelegt. Die Größe des Empfangspuffers wird in "udiNoOfBytes" übergeben.

Die Funktion IL_TCPRecvPacket empfängt Zeichen, bis der angelegte Empfangspuffer voll ist.

Sie kehrt zurück, wenn der Empfangspuffer voll oder ein Timeout aufgetreten ist. Nach erfolgreichem Abschluss der Übertragung gibt diese Funktion die Anzahl der empfangenen Zeichen zurück.

Wird 0 zurückgegeben, wurde die Verbindung geschlossen, bevor Daten empfangen werden konnten. Im Fehlerfall wird der Fehlercode in "pudiErrorNum" abgelegt, und die Funktion kehrt mit dem Rückgabewert 0 zurück.

Werden innerhalb von einer Sekunde keine Daten empfangen, kehrt die Funktion mit dem Rückgabewert 0 zurück und legt den Fehlercode 16#0000020A in "pudiErrorNum" ab.

Fehlerbehandlung

Die Funktion nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150.

Die Funktion gibt die folgenden Fehlercodes in "pudiErrorNum" (Additional1) zurück:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#00000004	-	Eingang "ValueAdr" ist 0
INPUT_RANGE_ERROR	16#00000005	-	Eingang "NoOfBytes" ist 0
INPUT_INVALID_ERROR	16#00000202	-	Socket-Handle ist ungültig.
COMMUNICATION_ERROR	16#0000020A	-	Keine Daten innerhalb des Timeouts empfangen
COMMUNICATION_ERROR	16#0000020E	-	Verbindung wurde geschlossen.
COMMUNICATION_ERROR	16#00000211	-	Gegenstelle hat die Verbindung geschlossen.

Tab. 6-26: IL_TCPRecvPacket Fehlercodes

Weitere Fehlercodes sind in [ETHERNET_TABLE](#), [ERROR_TABLE 16#0150](#) auf Seite 334 aufgeführt.

6.5.6 IL_TCPClose

Kurzbeschreibung

Die Funktion IL_TCPClose schließt eine Verbindung, die zuvor mit IL_TCPIinitial erstellt wurde. Diese Funktion kehrt sofort zurück.

Schnittstellenbeschreibung

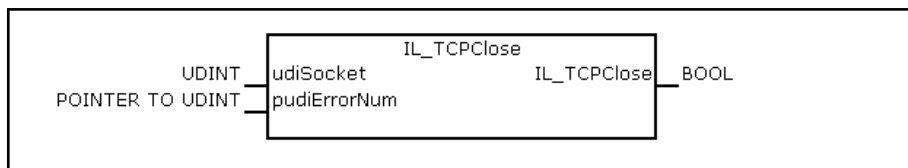


Abb. 6-25: IL_TCPClose Funktion

	Name	Typ	Beschreibung
VAR_INPUT	udiSocket	UDINT	Socket-Handle von IL_TCPIinitial
	pudiErrorNum	POINTER TO UDINT	Enthält den Fehlercode (siehe Fehlerbehandlung auf Seite 290).
RETURN		BOOL	TRUE: Die Socket wurde erfolgreich geschlossen. FALSE: Es ist ein Fehler beim Schließen der Socket aufgetreten.

Tab. 6-27: IL_TCPClose Interface

Funktionsbeschreibung

Die Funktion IL_TCPClose schließt eine Verbindung, die zuvor mit IL_TCPIinitial erstellt wurde. Diese Funktion kehrt sofort zurück.

Tritt ein Fehler auf, so wird der Fehlercode in der an pudiErrorNum angegebenen Variable abgelegt.

Fehlerbehandlung

Die Funktion nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150.

Die Funktion gibt die folgenden Fehlercodes in "pudiErrorNum" (Additional1) zurück:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR	16#00000202	-	Socket-Handle ist ungültig
COMMUNICATION_ERROR	16#00000211	-	Die Gegenstelle hat die Verbindung geschlossen

Tab. 6-28: IL_TCPClose Fehlercodes

Weitere Fehlercodes sind in [ETHERNET_TABLE, ERROR_TABLE 16#0150 auf Seite 334](#) aufgeführt.

6.6 UDP Asynchrone Funktionsbausteine

6.6.1 Einführung und Überblick

Asynchrone UDP-Funktionsbausteine werden in zyklischen Tasks verwendet und müssen zyklisch aufgerufen werden. Normalerweise werden mehrere Aufrufe benötigt, um eine Operation abzuschließen.

Ein Funktionsbaustein startet mit steigender Flanke an "Enable" und arbeitet bis entweder "Done" oder "Error" TRUE sind oder "Enable" zurückgesetzt wird. Um eine neue Operation zu starten, muss "Enable" zurückgesetzt und erneut gesetzt werden.

Eine neue UDP-Socket kann mit dem Funktionsbaustein IL_UDPSocketAsync oder IL_UDPIinitialAsync aufgebaut werden.

IL_UDPSocketAsync ruft intern die Funktionsbausteine IL_UDPIinitialAsync und IL_UDPCloseAsync auf und hält den Socket-Handle.

Eine mit IL_UDPIinitialAsync erstellte Socket kann mit IL_UDPCloseAsync geschlossen werden. Wird beim Erstellen der Socket "Server" auf FALSE gesetzt, kann die Socket keine Daten empfangen.



IL_UDPCloseAsync darf nicht für Sockets aufgerufen werden, die mit IL_UDPSocketAsync erstellt wurden.

Nachdem die Socket eingerichtet wurde, können Daten mit IL_UDPSendtoAsync und IL_UDPRecvfromAsync ausgetauscht werden.



UDP ist ein paket-basiertes Protokoll, das weder die Reihenfolge noch die Konsistenz der übertragenen Datenpakete sicherstellt.

Während der Übertragung können Pakete verloren gehen.

Empfangene Daten in der Empfangsreihenfolge in einer Warteschlange verwaltet.

Jeder Aufruf von IL_UDPRecvfromAsync oder IL_UDPRecvfrom gibt genau ein Paket aus der Warteschlange zurück.

Asynchrone Funktionsbausteine für eine UDP-Verbindung:

- [IL_UDPSocketAsync auf Seite 291](#)
- [IL_UDPInitialAsync auf Seite 293](#)
- [IL_UDPSendtoAsync auf Seite 295](#)
- [IL_UDPRecvfromAsync auf Seite 297](#)
- [IL_UDPCloseAsync auf Seite 298](#)

6.6.2 IL_UDPSocketAsync

Kurzbeschreibung

Der Funktionsbaustein IL_UDPSocketAsync verwaltet entweder einen UDP-Server oder einen UDP-Client, in Abhängigkeit des "Server"-Eingangs.

Schnittstellenbeschreibung

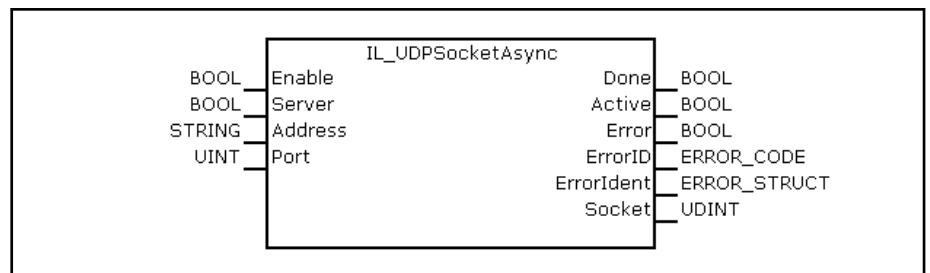


Abb. 6-26: IL_UDPSocketAsync Funktionsbaustein

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Startet bei steigender Flanke. Binäre Ausgänge werden bei fallender Flanke zurückgesetzt.
	Server	BOOL	TRUE: ein UDP-Server wird erstellt FALSE: ein UDP-Client wird erstellt
	Address	STRING	Wenn "Server" gesetzt ist, wird dieser Eingang nicht verwendet und die eigene IP-Adresse benutzt. Ist "Server" FALSE, enthält dieser Eingang die IP-Adresse des zu verbindenden Geräts.
	Port	UINT	Der UDP-Port auf dem auf eingehende Datenpakete gewartet wird oder zu dem Datenpakete gesendet werden sollen.

	Name	Typ	Beschreibung
VAR_OUTPUT	Done	BOOL	TRUE zeigt an, dass die Socket erfolgreich erstellt wurde.
	Active	BOOL	TRUE zeigt an, dass der Funktionsbaustein die Socket erstellt.
	Error	BOOL	TRUE zeigt einen Fehler an.
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerinformation (siehe Fehlerbehandlung auf Seite 293)
	Socket	UDINT	Enthält den Socket-Handle der neuen Socket.

Tab. 6-29: IL_UDPSocketAsync Interface

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
Server	BOOL			FALSE	Steigende Flanke an Enable
Address	STRING			"	Steigende Flanke an Enable
Port	UINT	0	65535	0	Steigende Flanke an Enable

Tab. 6-30: Min-, Max- und Default-Werte der Eingänge des Funktionsbausteins IL_UDPSocketAsync

Funktionsbeschreibung

Der Funktionsbaustein IL_UDPSocketAsync verwaltet entweder einen UDP-Server oder einen UDP-Client, in Abhängigkeit des "Server"-Eingangs. Die Abarbeitung beginnt bei steigender Flanke an "Enable". Der Funktionsbaustein muss zyklisch aufgerufen werden, bis "Done" oder "Error" TRUE sind.

Im Server-Modus öffnet der Funktionsbaustein eine Socket auf dem in "Port" angegebenen UDP-Port. Der zur Socket gehörende Socket-Handle erscheint dann am Ausgang "Socket" und der Ausgang "Done" wird gesetzt. Während die Socket erstellt wird, ist der "Active"-Ausgang gesetzt.

Im Client-Modus wird eine Socket angelegt, jedoch nicht mit einem Port verbunden. Dadurch ist die Portnummer unbekannt und es können keine Daten empfangen werden. Nach erfolgreicher Erstellung der Socket erscheint der Socket-Handle am Ausgang "Socket" und der Ausgang "Done" wird gesetzt. Während die Socket erstellt wird, ist der "Active"-Ausgang gesetzt.



Im Client-Modus können UDP-Pakete nur gesendet, jedoch nicht empfangen werden.

Im Fehlerfall wird der Ausgang "Error" gesetzt. Am Ausgang "Socket" erscheint SOCKET_INVALID. Die Fehlerklasse und der genaue Fehlercode können an den Ausgängen "ErrorID" und "ErrorIdent" abgelesen werden.

Wird "Enable" auf FALSE gesetzt, bevor "Done" TRUE ist, werden keine Socket angelegt und alle internen Ressourcen frei gegeben. Wird "Enable" auf FALSE gesetzt, nachdem die Socket angelegt wurde, wird die Socket geschlossen. Mögliche Fehler während des Schließens werden intern behandelt und nicht an den Fehlerausgängen ausgegeben. Nachdem "Enable" auf FALSE gesetzt wurde, wird der Ausgang "Socket" auf SOCKET_INVALID gesetzt.

Der Ausgang "Active" kann als Timeout-Indikator beim Verbindungsaufbau verwendet werden. Typische Timeouts für einen Verbindungsaufbau liegen zwischen 2 und 10 Sekunden.



IL_UDPCloseAsync sollte bei Sockets, die mit IL_UDPSocketAsync erstellt wurden, nicht verwendet werden.

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
OTHER_ERROR	16#00000040	x	Socket konnte nicht erstellt werden
OTHER_ERROR	16#00000041	x	Fehler beim Setzen des IO-Modus
OTHER_ERROR	16#00000042	x	Konnte Socket nicht an Port binden
INPUT_INVALID_ERROR	16#00000208	16#00000000	Zieladresse ungültig
COMMUNICATION_ERROR	16#0000020C	16#00000000	Keine Route zum Zielgerät

Tab. 6-31: IL_UDPSocketAsync Fehlercodes

Weitere Fehlercodes sind in [ETHERNET_TABLE](#), [ERROR_TABLE 16#0150](#) auf Seite 334 aufgeführt.

6.6.3 IL_UDPInitialAsync

Kurzbeschreibung

Der Funktionsbaustein IL_UDPInitialAsync erstellt entweder einen UDP-Server oder einen UDP-Client in Abhängigkeit des "Server"-Eingangs.

Schnittstellenbeschreibung

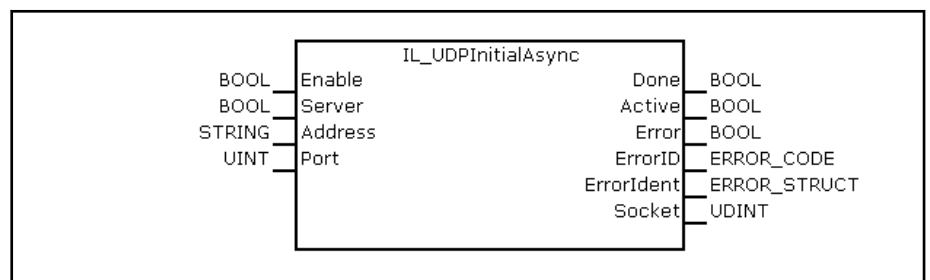


Abb. 6-27: IL_UDPInitialAsync Funktionsbaustein

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Startet bei steigender Flanke. Binäre Ausgänge werden bei fallender Flanke zurückgesetzt.
	Server	BOOL	TRUE: ein UDP-Server wird erstellt. FALSE: ein UDP-Client wird erstellt.
	Address	STRING	Wenn "Server" gesetzt ist, wird dieser Eingang nicht verwendet und die eigene IP-Adresse benutzt. Ist "Server" FALSE, enthält dieser Eingang die IP-Adresse des zu verbindenden Geräts.
	Port	UINT	Der UDP-Port auf dem auf eingehende Datenpakete gewartet wird oder zu dem Datenpakete gesendet werden sollen.

	Name	Typ	Beschreibung
VAR_OUTPUT	Done	BOOL	TRUE zeigt an, dass die Socket erfolgreich erstellt wurde.
	Active	BOOL	TRUE zeigt an, dass der Funktionsbaustein die Socket erstellt.
	Error	BOOL	TRUE bedeutet, dass im Funktionsbaustein ein Fehler aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerinformation (siehe Fehlerbehandlung auf Seite 294)
	Socket	UDINT	Enthält den Socket-Handle der neuen Socket.

Tab. 6-32: IL_UDPInitialAsync Interface

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
Server	BOOL			FALSE	Steigende Flanke an Enable
Address	STRING			"	Steigende Flanke an Enable
Port	UINT	0	65535	0	Steigende Flanke an Enable

Tab. 6-33: Min-, Max- und Default-Werte der Eingänge des Funktionsbausteins IL_UDPInitialAsync

Funktionsbeschreibung

Der Funktionsbaustein IL_UDPInitialAsync erstellt entweder einen UDP-Server oder einen UDP-Client, in Abhängigkeit des "Server"-Eingangs. Die Abarbeitung beginnt bei steigender Flanke an "Enable". Der Funktionsbaustein muss zyklisch aufgerufen werden, bis "Done" oder "Error" TRUE sind.

Im Server-Modus öffnet der Funktionsbaustein eine Socket auf dem, in "Port" angegebenen UDP-Port. Der zur Socket gehörende Socket-Handle erscheint dann am Ausgang "Socket" und der Ausgang "Done" wird gesetzt. Während die Socket erstellt wird, ist der "Active"-Ausgang gesetzt.

Im Client-Modus wird eine Socket angelegt, jedoch nicht mit einem Port verbunden. Dadurch ist die Portnummer unbekannt und es können keine Daten empfangen werden. Nach erfolgreicher Erstellung der Socket erscheint der Socket-Handle am Ausgang "Socket" und der Ausgang "Done" wird gesetzt. Während die Socket erstellt wird, ist der "Active"-Ausgang gesetzt.



Im Client-Modus können UDP-Pakete nur gesendet, jedoch nicht empfangen werden.

Im Fehlerfall wird der Ausgang "Error" gesetzt. Am Ausgang "Socket" erscheint SOCKET_INVALID. Die Fehlerklasse und der genaue Fehlercode können an den Ausgängen "ErrorID" und "ErrorIdent" abgelesen werden.

Wird "Enable" auf FALSE gesetzt, bevor "Done" TRUE ist, werden keine Socket angelegt und alle internen Ressourcen frei gegeben. Wird "Enable" auf FALSE gesetzt, nachdem die Socket angelegt wurde, wird die Socket im Gegensatz zu IL_UDPSocketAsync nicht geschlossen. Nachdem "Enable" auf FALSE gesetzt wurde, wird der Ausgang "Socket" auf SOCKET_INVALID gesetzt.

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
OTHER_ERROR	16#00000040	x	Socket konnte nicht erstellt werden
OTHER_ERROR	16#00000041	x	Fehler beim Setzen des I/O Modus
OTHER_ERROR	16#00000042	x	Konnte Socket nicht an Port binden
INPUT_INVALID_ERROR	16#00000208	16#00000000	Zieladresse ungültig
COMMUNICATION_ERROR	16#0000020C	16#00000000	Keine Route zum Zielgerät

Tab. 6-34: IL_UDPInitialAsync Fehlercodes

Weitere Fehlercodes sind in [ETHERNET_TABLE](#), [ERROR_TABLE 16#0150](#) auf Seite 334 aufgeführt.

6.6.4 IL_UDPSendtoAsync

Kurzbeschreibung Der Funktionsbaustein IL_UDPSendtoAsync sendet Datenpakete über eine Socket, die zuvor mit IL_UDPSocketAsync oder IL_UDPInitialAsync erstellt wurde.

Schnittstellenbeschreibung

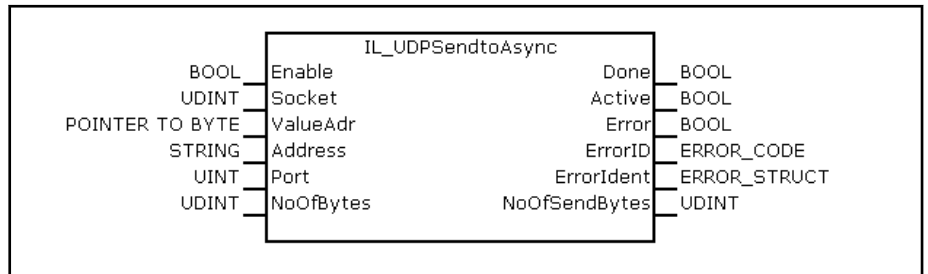


Abb. 6-28: IL_UDPSendtoAsync Funktionsbaustein

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Startet bei steigender Flanke Binäre Ausgänge werden bei fallender Flanke zurückgesetzt
	Socket	UDINT	Socket-Handle von IL_UDPSocketAsync oder IL_UDPInitialAsync
	ValueAdr	POINTER TO BYTE	Startadresse der zu sendenden Daten
	Address	STRING	Ziel IP-Adresse des Empfängers (z.B. '192.168.1.2')
	Port	UINT	UDP-Empfangsport der Gegenstelle
	NoOfBytes	UDINT	Anzahl der zu sendenden Zeichen
VAR_OUTPUT	Done	BOOL	TRUE zeigt an, dass das Datenpaket erfolgreich gesendet wurde
	Active	BOOL	TRUE zeigt an, dass der Funktionsbaustein Daten sendet
	Error	BOOL	TRUE zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerinformation (siehe Fehlerbehandlung auf Seite 296)
	NoOfSendBytes	UDINT	Anzahl der gesendeten Bytes, 0 zeigt an, dass ein Fehler aufgetreten ist oder die Socket geschlossen wurde.

Tab. 6-35: IL_UDPSendtoAsync Interface

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
Socket	UDINT	0	16#FFFFFFFF	SOCKET_INVALID	Steigende Flanke an Enable
ValueAdr	POINTER TO BYTE			0	Steigende Flanke an Enable
Address	STRING			"	Steigende Flanke an Enable
Port	UINT	1	65535	0	Steigende Flanke an Enable
NoOfBytes	UDINT	0	16#FFFFFFFF	0	Steigende Flanke an Enable

Tab. 6-36: Min-, Max- und Default-Werte der Eingänge des Funktionsbausteins IL_UDPSendtoAsync

Funktionsbeschreibung

Der Funktionsbaustein IL_UDPSendtoAsync sendet Datenpakete über eine Socket, die zuvor mit IL_UDPSocketAsync oder IL_UDPInitialAsync erstellt wurde. Die Übertragung beginnt mit steigender Flanke an "Enable". Der Funktionsbaustein muss zyklisch aufgerufen werden bis "Done" oder "Error" TRUE ist.

Der zu sendende Puffer wird an Eingang "ValueAdr" angelegt. Die Anzahl der zu sendenden Zeichen wird im Eingang "NoOfBytes" übergeben.



Bei einigen Steuerungen ist die maximale Größe von UDP-Datenpaketen auf 1430 Bytes beschränkt.

Wird "Enable" zurückgesetzt bevor "Done" TRUE ist, wird die Übertragung unterbrochen. Es werden keine Daten gesendet, und der Ausgang "NoOfSentBytes" ist 0.

Der Ausgang "Active" ist während der Übertragung des Datenpakets gesetzt. "Active" kann als Timeout-Indikator verwendet werden. Typische Sendetimeouts liegen zwischen 1 und 2 Sekunden. Während der Übertragung ("Active" ist TRUE), kann die Anzahl der gesendeten Bytes am Ausgang "NoOfSentBytes" abgelesen werden. In der Regel wird das gesamte Paket in einem Aufruf gesendet.

Die IP-Adresse der Gegenstelle sowie der Zielport muss in den Eingängen "Address" und "Port" angegeben werden.

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#00000004	16#00000000	ValueAdr ist nicht gesetzt
INPUT_RANGE_ERROR	16#00000005	16#00000000	NoOfBytes ist 0
INPUT_INVALID_ERROR	16#00000202	16#00000000	Socket-Handle ist ungültig
COMMUNICATION_ERROR	16#00000208	16#00000000	Zieladresse ungültig
COMMUNICATION_ERROR	16#0000020C	16#00000000	Kein Netzwerkpfad zum Zielgerät gefunden

Tab. 6-37: IL_UDPSendtoAsync Fehlercodes

Weitere Fehlercodes sind in [ETHERNET_TABLE](#), [ERROR_TABLE 16#0150](#) auf Seite 334 aufgeführt.

6.6.5 IL_UDPRecvfromAsync

Kurzbeschreibung Der Funktionsbaustein IL_UDPRecvfromAsync empfängt Datenpakete über eine Socket, die zuvor mit IL_UDPSocketAsync oder IL_UDPInitialAsync erstellt wurde.

Schnittstellenbeschreibung

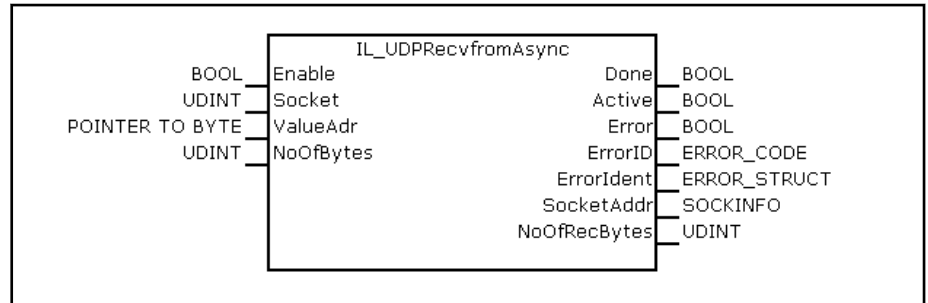


Abb. 6-29: IL_UDPRecvfromAsync Funktionsbaustein

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Startet bei steigender Flanke. Binäre Ausgänge werden bei fallender Flanke zurückgesetzt.
	Socket	UDINT	Socket-Handle von IL_UDPSocketAsync oder IL_UDPInitialAsync
	ValueAdr	POINTER TO BYTE	Startadresse des Empfangsdatenpuffers
	NoOfBytes	UDINT	Größe des Empfangsdatenpuffers
VAR_OUTPUT	Done	BOOL	TRUE zeigt an, dass ein Datenpaket erfolgreich empfangen wurde.
	Active	BOOL	TRUE zeigt an, dass der Funktionsbaustein auf eingehende Datenpakete wartet.
	Error	BOOL	TRUE zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerinformation (siehe Fehlerbehandlung auf Seite 298)
	SocketAddr	SOCKINFO	IP-Adresse und Port-Nummer des Senders
	NoOfRecBytes	UDINT	Anzahl der in den Empfangsdatenpuffer kopierten Bytes 0 zeigt an, dass ein Fehler aufgetreten ist oder die Socket geschlossen wurde.

Tab. 6-38: IL_UDPRecvfromAsync Interface

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
Socket	UDINT	0	16#FFFFFFFF	SOCKET_INVALID	Steigende Flanke an Enable
ValueAdr	POINTER TO BYTE			0	Steigende Flanke an Enable
NoOfBytes	UDINT	0	16#FFFFFFFF	0	Steigende Flanke an Enable

Tab. 6-39: Min-, Max- und Default-Werte der Eingänge des Funktionsbausteins IL_UDPRecvfromAsync

Funktionsbeschreibung

Der Funktionsbaustein IL_UDPRecvAsync empfängt Datenpakete über eine Socket, die zuvor mit IL_UDPSocketAsync oder IL_UDPInitialAsync erstellt wurde.

Die Übertragung beginnt mit steigender Flanke an "Enable".

Der Funktionsbaustein muss zyklisch aufgerufen werden bis "Done" oder "Error" TRUE ist.

Wird "Enable" zurückgesetzt bevor "Done" TRUE ist, wird die Übertragung abgebrochen. Bei einem solchen Abbruch werden keine Daten entgegengenommen. "NoOfRecBytes" ist demzufolge 0.

Der Ausgang "Active" ist während der Übertragung der Daten gesetzt. "Active" kann als Timeout-Indikator verwendet werden.

Typische Empfangs-Timeouts liegen zwischen 1 und 2 Sekunden.

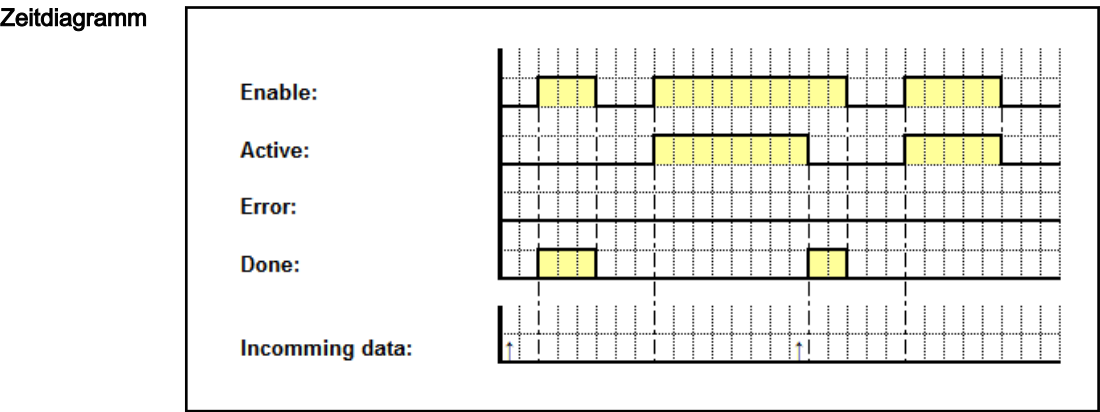


Abb. 6-30: Zeitdiagramm eines UDP-Datenpaketempfangs

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#00000004	16#00000000	ValueAdr ist nicht gesetzt
INPUT_RANGE_ERROR	16#00000005	16#00000000	NoOfBytes ist 0
INPUT_INVALID_ERROR	16#00000202	16#00000000	Socket-Handle ist ungültig.

Tab. 6-40: IL_UDPRecvfromAsync Fehlercodes

Weitere Fehlercodes sind in [ETHERNET_TABLE](#), [ERROR_TABLE 16#0150](#) auf Seite 334 aufgeführt.

6.6.6 IL_UDPCloseAsync

Kurzbeschreibung

Der Funktionsbaustein IL_UDPCloseAsync schließt eine Socket, die zuvor mit IL_UDPInitialAsync erstellt wurde.

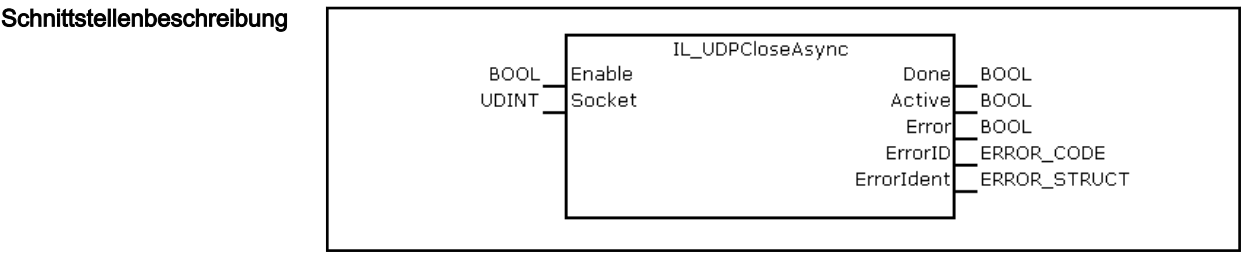


Abb. 6-31: IL_UDPCloseAsync Funktionsbaustein

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Startet bei steigender Flanke. Binäre Ausgänge werden bei fallender Flanke zurückgesetzt.
	Socket	UDINT	Socket-Handle von IL_UDPInitialAsync
VAR_OUTPUT	Done	BOOL	TRUE zeigt an, dass die Socket erfolgreich geschlossen wurde.
	Active	BOOL	TRUE zeigt an, dass die Socket geschlossen wird.
	Error	BOOL	TRUE zeigt einen Fehler an.
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerinformation (siehe Fehlerbehandlung auf Seite 299)

Tab. 6-41: IL_UDPCloseAsync Interface

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
Socket	UDINT	0	16#FFFFFFFF	SOCKET_INVALID	Steigende Flanke an Enable

Tab. 6-42: Min-, Max- und Default-Werte der Eingänge des Funktionsbausteins IL_UDPCloseAsync

Funktionsbeschreibung

Der Funktionsbaustein IL_UDPCloseAsync schließt eine Verbindung, die zuvor mit IL_UDPInitialAsync erstellt wurde.

Der Funktionsbaustein benötigt eventuell mehrere Aufrufe, um die Operation zu beenden. Die Operation ist erfolgreich abgeschlossen, wenn der Ausgang "Done" gesetzt ist. Während die Operation läuft, ist "Active" gesetzt. Tritt ein Fehler auf, so ist der Ausgang "Error" gesetzt. Die Ausgänge "ErrorID" und "ErrorIdent" enthalten dann nähere Informationen zu dem Fehler. Der Funktionsbaustein muss nach steigender Flanke an "Enable" so lange aufgerufen werden, bis entweder "Done" oder "Error" gesetzt sind oder "Enable" zurückgesetzt wird.

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR	16#00000202	16#00000000	Socket-Handle ist ungültig.
COMMUNICATION_ERROR	16#00000211	16#00000000	Gegenstelle hat die Verbindung geschlossen.

Tab. 6-43: IL_UDPCloseAsync Fehlercodes

Weitere Fehlercodes sind in [ETHERNET_TABLE](#), [ERROR_TABLE 16#0150](#) auf Seite 334 aufgeführt.

6.7 UDP Synchrone Funktionen

6.7.1 Einführung und Überblick

Die synchronen UDP-Funktionen sind als blockierende Aufrufe implementiert, d.h., sie kehren nicht zurück, bis die Operation abgeschlossen wurde oder ein Fehler auftrat. Diese Funktionen sollten nur in frei laufenden Tasks verwendet werden.

IL_UDPInitial wird verwendet, um eine neue UDP-Socket anzulegen und gibt einen Socket-Handle zurück.

Nach erfolgreicher Erstellung der UDP-Socket können Daten mit IL_UDP-Sendto und IL_UDPRecvfrom gesendet und empfangen werden.

Wird beim Erstellen der Socket "bServer" auf FALSE gesetzt, kann die Socket keine Daten empfangen

IL_UDPClose schließt eine UDP-Socket und gibt den Socket-Handle frei.



UDP ist ein paket-basiertes Protokoll, das weder die Reihenfolge noch die Konsistenz der übertragenen Datenpakete sicherstellt. Während der Übertragung können Pakete verloren gehen.

Empfangene Daten in der Empfangsreihenfolge werden in einer Warteschlange verwaltet.

Jeder Aufruf von IL_UDPRecvfromAsync oder IL_UDPRecvfrom gibt genau ein Paket aus der Warteschlange zurück.

Synchrone Funktionen für eine UDP-Verbindung:

- [IL_UDPInitial auf Seite 300](#)
- [IL_UDPSendto auf Seite 301](#)
- [IL_UDPRecvfrom auf Seite 302](#)
- [IL_UDPClose auf Seite 304](#)

6.7.2 IL_UDPInitial

Kurzbeschreibung Die Funktion IL_UDPInitial erstellt entweder einen UDP-Server oder einen UDP-Client in Abhängigkeit des "bServer"-Parameters.

Schnittstellenbeschreibung

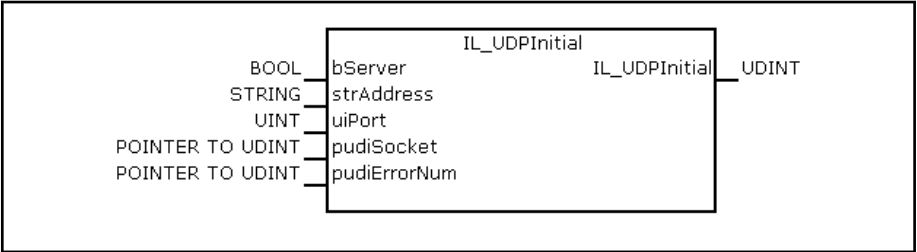


Abb. 6-32: IL_UDPInitial Funktion

	Name	Typ	Beschreibung
VAR_INPUT	bServer	BOOL	TRUE: ein UDP-Server wird erstellt. FALSE: ein UDP-Client wird erstellt.
	strAddress	STRING	Wenn "bServer" gesetzt ist, wird dieser Eingang nicht verwendet und die eigene IP-Adresse benutzt. Ist "bServer" FALSE, enthält dieser Eingang die IP-Adresse des zu verbindenden Geräts.
	uiPort	UINT	Der UDP-Port auf dem auf eingehende Datenpakete gewartet wird oder zu dem Datenpakete gesendet werden sollen.
	pudiSocket	POINTER TO UDINT	Enthält den Socket-Handle der neuen Socket.
	pudiErrorNum	POINTER TO UDINT	Enthält den Fehlercode (siehe Fehlerbehandlung auf Seite 301).
RETURN		UDINT	0, falls die Socket erfolgreich erstellt wurde; sonst INVALID_SOCKET.

Tab. 6-44: IL_UDPInitial Interface

Funktionsbeschreibung Die Funktion IL_UDPInitial erstellt entweder einen UDP-Server oder einen UDP-Client in Abhängigkeit des "bServer"-Parameters.

Im Server-Modus öffnet die Funktion eine Socket auf dem, in "uiPort" angegebenen UDP-Port. Der zur Socket gehörende Socket-Handle erscheint dann in "pudiSocket" und die Funktion kehrt mit dem Rückgabewert 0 zurück.

Im Client-Modus wird eine Socket angelegt, jedoch nicht mit einem Port verbunden. Dadurch ist die Portnummer unbekannt und es können keine Daten empfangen werden. Nach erfolgreicher Erstellung der Socket erscheint der Socket-Handle in "pudiSocket" und die Funktion kehrt mit dem Rückgabewert 0 zurück.



Im Client-Modus können UDP-Pakete nur gesendet, jedoch nicht empfangen werden.

Tritt ein Fehler auf, kehrt die Funktion mit SOCKET_INVALID zurück und legt den Fehlercode in "pudiErrorNum" ab.

Fehlerbehandlung Die Funktion nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150.

Die Funktion gibt die folgenden Fehlercodes in "pudiErrorNum" (Additional1) zurück:

ErrorID	Additional1	Additional2	Beschreibung
OTHER_ERROR	16#00000040	-	Socket konnte nicht erstellt werden
OTHER_ERROR	16#00000041	-	Fehler beim Setzen des I/O Modus
OTHER_ERROR	16#00000042	-	Konnte Socket nicht an Port binden
INPUT_INVALID_ERROR	16#00000208	-	Zieladresse ungültig
COMMUNICATION_ERROR	16#0000020C	-	Keine Route zum Zielgerät

Tab. 6-45: IL_UDPInitial Fehlercodes

Weitere Fehlercodes sind in [ETHERNET_TABLE](#), [ERROR_TABLE 16#0150](#) auf [Seite 334](#) aufgeführt.

6.7.3 IL_UDPSendto

Kurzbeschreibung Die Funktion IL_UDPSendto sendet ein Datenpaket über eine Socket, die zuvor mit IL_UDPInitial erstellt wurde. Diese Funktion kehrt nicht zurück, bis entweder alle Daten übertragen wurden oder ein Fehler auftrat.

Schnittstellenbeschreibung

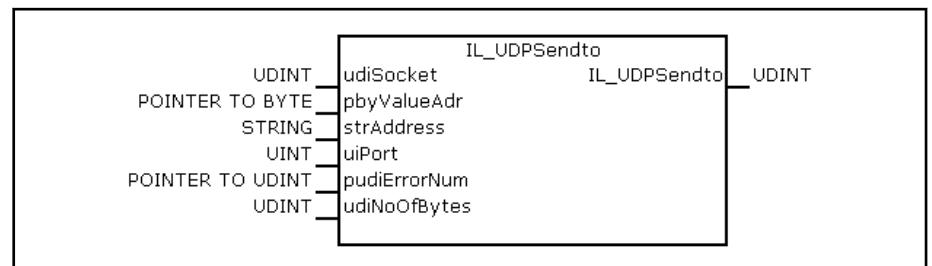


Abb. 6-33: IL_UDPSendto Funktion

	Name	Typ	Beschreibung
VAR_INPUT	udiSocket	UDINT	Socket-Handle von IL_UDPInitial
	pbyValueAdr	POINTER TO BYTE	Startadresse der zu sendenden Daten
	strAddress	STRING	Ziel IP-Adresse des Empfängers (z.B. '192.168.1.2')
	uiPort	UINT	UDP-Empfangsport der Gegenstelle
	pudiErrorNum	POINTER TO UDINT	Enthält den Fehlercode (siehe Fehlerbehandlung auf Seite 302)
	udiNoOfBytes	UDINT	Anzahl der zu sendenden Zeichen
RETURN		UDINT	Anzahl der tatsächlich gesendeten Bytes, 0 zeigt an, dass die Socket geschlossen wurde oder ein Fehler auftrat

Tab. 6-46: IL_UDPSendto Interface

Funktionsbeschreibung

Die Funktion IL_UDPSendto sendet ein Datenpaket über eine Socket, die zuvor mit IL_UDPInitial erstellt wurde. Diese Funktion kehrt nicht zurück, bis entweder alle Daten übertragen wurden oder ein Fehler auftrat.

Der zu sendende Puffer wird in "pbyValueAdr" angelegt. Die Anzahl der zu sendenden Zeichen wird in "udiNoOfBytes" übergeben.

Nach erfolgreichem Abschluss der Übertragung gibt diese Funktion die Anzahl der übertragenen Zeichen zurück. Wird 0 zurückgegeben, wurde die Socket geschlossen bevor das Datenpaket gesendet werden konnte. Im Fehlerfall wird der Fehlercode in "pudiErrorNum" abgelegt und die Funktion kehrt mit dem Rückgabewert 0 zurück.

Fehlerbehandlung

Die Funktion nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150.

Die Funktion gibt die folgenden Fehlercodes in "pudiErrorNum" (Additional1) zurück:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#00000004	-	ValueAdr ist nicht gesetzt
INPUT_RANGE_ERROR	16#00000005	-	NoOfBytes ist 0
INPUT_INVALID_ERROR	16#00000202	-	Socket-Handle ist ungültig
COMMUNICATION_ERROR	16#00000208	-	Zieladresse ungültig
COMMUNICATION_ERROR	16#0000020C	-	Kein Netzwerkpfad zum Zielgerät gefunden

Tab. 6-47: IL_UDPSendto Fehlercodes

Weitere Fehlercodes sind in [ETHERNET_TABLE, ERROR_TABLE 16#0150 auf Seite 334](#) aufgeführt.

6.7.4 IL_UDPRecvfrom

Kurzbeschreibung

Die Funktion IL_UDPRecvfrom empfängt Datenpakete über eine Socket, die zuvor mit IL_UDPInitial erstellt wurde. Diese Funktion kehrt nicht zurück, bis entweder ein Datenpaket empfangen wurde oder ein Fehler auftrat.

Schnittstellenbeschreibung

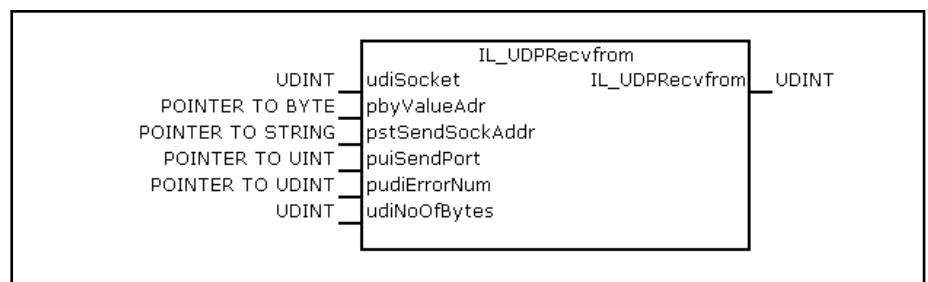


Abb. 6-34: IL_UDPRecvfrom Funktion

	Name	Typ	Beschreibung
VAR_INPUT	udiSocket	UDINT	Socket-Handle von IL_UDPIinitial
	pbyValueAdr	POINTER TO BYTE	Startadresse des Empfangsdatenpuffers
	pstrSendSockAddr	POINTER TO STRING	IP-Adresse des Senders.
	poiSendPort	POINTER TO UINT	Port-Nummer des Senders
	pudiErrorNum	POINTER TO UDINT	Enthält den Fehlercode (siehe Fehlerbehandlung auf Seite 303).
	udiNoOfBytes	UDINT	Größe des Empfangsdatenpuffers
RETURN		UDINT	Anzahl der empfangenen Bytes, 0 zeigt an, dass die Socket geschlossen wurde oder ein Fehler auftrat.

Tab. 6-48: IL_UDPRecvfrom Interface

Funktionsbeschreibung

Die Funktion IL_UDPRecvfrom empfängt Datenpakete über eine Socket, die zuvor mit IL_UDPIinitial erstellt wurde. Diese Funktion kehrt nicht zurück, bis entweder ein Datenpaket empfangen wurde oder ein Fehler auftrat.

Der Empfangspuffer wird in "pbyValueAdr" angelegt. Die Größe des Empfangspuffers wird in "udiNoOfBytes" übergeben.

Nach erfolgreichem Abschluss der Übertragung gibt diese Funktion die Anzahl der empfangenen Zeichen zurück. Wird 0 zurückgegeben, wurde die Socket geschlossen bevor ein Datenpaket empfangen werden konnten. Im Fehlerfall wird der Fehlercode in "pudiErrorNum" abgelegt und die Funktion kehrt mit dem Rückgabewert 0 zurück.

Werden innerhalb von einer Sekunde keine Daten empfangen, kehrt die Funktion mit dem Rückgabewert 0 zurück und legt den Fehlercode 16#0000020A in "pudiErrorNum" ab.

Fehlerbehandlung

Die Funktion nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150.

Die Funktion gibt die folgenden Fehlercodes in "pudiErrorNum" (Additional1) zurück:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#00000004	-	ValueAdr ist nicht gesetzt
INPUT_RANGE_ERROR	16#00000005	-	NoOfBytes ist 0
INPUT_INVALID_ERROR	16#00000202	-	Socket-Handle ist ungültig.

Tab. 6-49: IL_UDPRecvfromAsync Fehlercodes

Weitere Fehlercodes sind in [ETHERNET_TABLE](#), [ERROR_TABLE 16#0150](#) auf [Seite 334](#) aufgeführt.

6.7.5 IL_UDPClose

Kurzbeschreibung

Die Funktion IL_UDPClose schließt eine Socket, die zuvor mit IL_UDPInitial erstellt wurde. Diese Funktion kehrt sofort zurück.

Schnittstellenbeschreibung

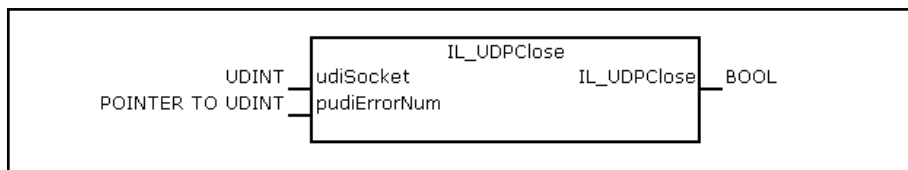


Abb. 6-35: IL_UDPClose Funktion

	Name	Typ	Beschreibung
VAR_INPUT	udiSocket	UDINT	Socket-Handle von IL_UDPInitial
	pudiErrorNum	POINTER TO UDINT	enthält den Fehlercode (siehe Fehlerbehandlung auf Seite 304).
RETURN		BOOL	TRUE: Die Socket wurde erfolgreich geschlossen. FALSE: Es ist ein Fehler beim Schließen der Socket aufgetreten.

Tab. 6-50: IL_UDPClose Interface

Funktionsbeschreibung

Die Funktion IL_UDPClose schließt eine Socket, die zuvor mit IL_UDPInitial erstellt wurde. Diese Funktion kehrt sofort zurück.

Fehlerbehandlung

Die Funktion nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150.

Die Funktion gibt die folgenden Fehlercodes in "pudiErrorNum" (Additional1) zurück:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR	16#00000202	-	Socket-Handle ist ungültig.
COMMUNICATION_ERROR	16#00000211	-	Gegenstelle hat die Verbindung geschlossen.

Tab. 6-51: IL_UDPClose Fehlercodes

Weitere Fehlercodes sind in [ETHERNET_TABLE, ERROR_TABLE 16#0150 auf Seite 334](#) aufgeführt.

6.8 Asynchrone ICMP-Funktionsbausteine

6.8.1 IL_PingAsync

Kurzbeschreibung

Der Funktionsbaustein IL_PingAsync führt einen ICMP-Ping auf die angegebene IP-Adresse aus und wartet auf die Antwort. Die Laufzeit des Pings erscheint am Ausgang "EchoTime".

Schnittstellenbeschreibung

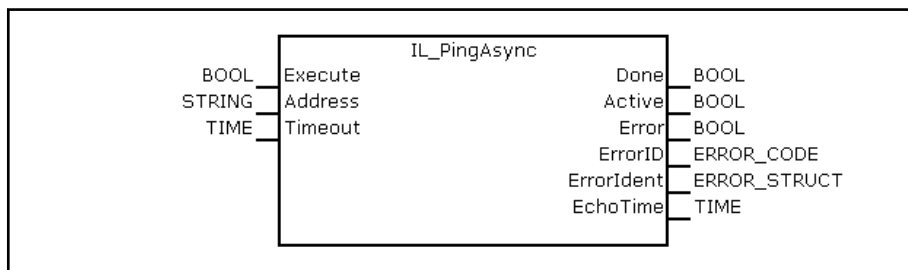


Abb. 6-36: IL_PingAsync Funktionsbaustein

	Name	Typ	Beschreibung
VAR_INPUT	Execute	BOOL	Steigende Flanke startet den Ping
	Address	STRING	IP-Adresse des zu überprüfenden Geräts
	Timeout	TIME	maximale Laufzeit des Pings
VAR_OUTPUT	Done	BOOL	TRUE: Der Ping wurde erfolgreich durchgeführt.
	Active	BOOL	TRUE: Der Ping wird gerade durchgeführt.
	Error	BOOL	Es ist ein Fehler aufgetreten.
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerinformation (siehe Fehlerbehandlung auf Seite 305)
	EchoTime	TIME	Gemessene Laufzeit des Pings

Tab. 6-52: IL_PingAsync Interface

Min-, Max-, Default-Werte

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	kontinuierlich
Address	STRING			""	steigende Flanke an Execute
Timeout	TIME	0s	n. def.	3s	steigende Flanke an Execute

Tab. 6-53: Min-, Max-, Default-Werte des Funktionsbausteins IL_PingAsync

Funktionsbeschreibung

Dieser Funktionsbaustein dient der Prüfung der Erreichbarkeit eines Geräts im Ethernet.

Bei steigender Flanke an "Execute" schickt der Funktionsbaustein IL_PingAsync ein kurzes ICMP-Pingpaket an das in "Address" angegebene Ethernet-Gerät.

Sofern dieses Gerät erreichbar ist und das ICMP-Protokoll unterstützt, sendet dieses Gerät einen Ping-Reply mit demselben Paketinhalt zurück zum Absender.

Die Laufzeit wird vom Funktionsbaustein gemessen und am Ausgang "EchoTime" ausgegeben.

Antwortet das Gerät nicht in der am Eingang "Timeout" angegebenen Zeitspanne, wird eine Fehlermeldung an den Error-Ausgängen generiert. Solange der Funktionsbaustein auf Antwort wartet ist "Active" gesetzt. Wurde die Antwort empfangen, wird "Done" gesetzt und "EchoTime" enthält die Laufzeit.

Der "EchoTime"-Ausgang hat die Auflösung der aufrufenden Task. Die minimale "EchoTime" einer zyklischen 2 ms Task ist somit 2 ms.

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
COMMUNICATION_ERROR	16#0000020A	16#00000000	Timeout, angesprochenes Gerät hat nicht geantwortet
COMMUNICATION_ERROR	16#0000020B	16#00000000	Gegenstelle wurde nicht gefunden
COMMUNICATION_ERROR	16#0000020C	-	Fehlercode für System MLD: ERR_NOTIMPLEMEN- TED: Die Funktion ist nicht implementiert. Nutzen Sie bitte die Funktion SysSockPing, Bibliothek SysSocket

Tab. 6-54: Fehlercodes, die von IL_PingAsync generiert werden

Weitere Fehlercodes sind in [ETHERNET_TABLE](#), [ERROR_TABLE 16#0150](#) auf Seite 334 aufgeführt.

6.9 Synchrone ICMP-Funktionen

6.9.1 IL_Ping

Kurzbeschreibung

Die Funktion IL_Ping führt einen ICMP-Ping auf die angegebene IP-Adresse aus und wartet auf die Antwort. Die Funktion gibt die Laufzeit des Pings zurück.

Schnittstellenbeschreibung

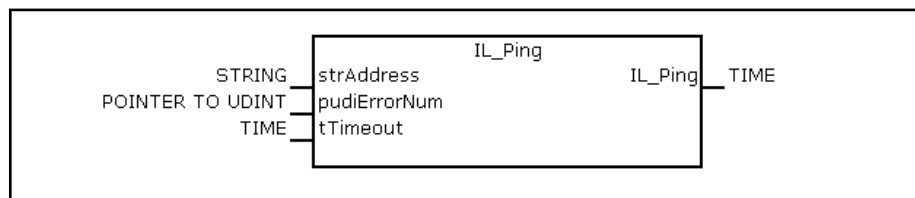


Abb. 6-37: IL_Ping Funktion

	Name	Typ	Beschreibung
VAR_INPUT	strAddress	STRING	IP-Adresse des zu überprüfenden Geräts
	pudiErrorNum	POINTER TO UDINT	Enthält den Fehlercode (siehe Fehlerbehandlung auf Seite 306).
	tTimeout	TIME	Maximale Laufzeit des Pings
RETURN		TIME	Gemessene Laufzeit des Pings

Tab. 6-55: IL_Ping Interface

Funktionsbeschreibung

Diese Funktion dient der Prüfung der Erreichbarkeit eines Geräts im Ethernet.

Bei Aufruf schickt die Funktion IL_Ping ein kurzes ICMP-Pingpaket an das in "Address" angegebene Ethernet-Gerät. Sofern dieses Gerät erreichbar ist und das ICMP-Protokoll unterstützt, sendet dieses Gerät einen Ping-Reply mit demselben Paketinhalt zurück zum Absender. Die Laufzeit wird von der Funktion gemessen und auf den Rückgabewert gespeichert. Antwortet das Gerät nicht in der am Eingang "Timeout" angegebenen Zeitspanne, wird eine Fehlermeldung in "pudiErrorNum" geschrieben.

Der Funktion arbeitet synchron, d.h. sie kehrt erst zurück, wenn entweder das Gerät geantwortet hat, ein Timeout auftrat oder ein anderer Fehler aufgetreten ist.

Fehlerbehandlung

Die Funktion nutzt die Fehlertabelle [ETHERNET_TABLE](#), 16#0150.

Die Funktion gibt die folgenden Fehlercodes in "pudiErrorNum" (Additional1) zurück:

ErrorID	Additional1	Additional2	Beschreibung
COMMUNICATION_ERROR	16#0000020A	-	Timeout, angesprochenes Gerät hat nicht geantwortet
COMMUNICATION_ERROR	16#0000020B	-	Gegenstelle wurde nicht gefunden
COMMUNICATION_ERROR	16#0000020C	-	Fehlercode für System MLD: ERR_NOTIMPLEMEN- TED: Die Funktion ist nicht implementiert. Nutzen Sie bitte die Funktion SysSockPing, Bibliothek SysSocket

Tab. 6-56: Fehlercodes, die von IL_Ping generiert werden

Weitere Fehlercodes sind in [ETHERNET_TABLE](#), [ERROR_TABLE 16#0150](#) auf Seite 334 aufgeführt.

6.10 Asynchrone FTP-Funktionsbausteine

6.10.1 Einführung und Überblick

FTP (File Transfer Protocol) implementiert den Transfer einzelner Dateien über IP. Es setzt auf das TCP-Protokoll auf.

Um eine Datei zu übertragen, muss sich ein FTP-Client vorher auf einen FTP-Server anmelden. Danach kann die Datei entweder vom Server an den Client (get) oder vom Client an den Server (put) übertragen werden. Der FTP-Server wartet in der Regel auf TCP-Port 21 auf eine eingehende Client-Verbindung.

Die Datenübertragung erfolgt im Passiv-Modus. Dadurch werden mögliche Probleme durch Firewalls und NAT (Network Address Translation) umgangen.

Unterstützt der FTP-Server den Passiv-Modus nicht, so wird der Aktiv-Modus verwendet. Die Daten werden immer binär übertragen. Eine Übersetzung der Zeilenendekennung bei Texten zwischen Windows und Unix-basierten Systemen findet demzufolge nicht statt.

Asynchrone Funktionsbausteine für FTP:

- [IL_FTPGetAsync](#) auf Seite 307
- [IL_FTPPutAsync](#) auf Seite 310
- [IL_FTPGetFileAsync](#) auf Seite 313
- [IL_FTPPutFileAsync](#) auf Seite 315

6.10.2 IL_FTPGetAsync

Kurzbeschreibung

Der Funktionsbaustein IL_FTPGetAsync liest eine Datei von einem FTP-Server und legt sie in einem Datenbereich ab.

Schnittstellenbeschreibung

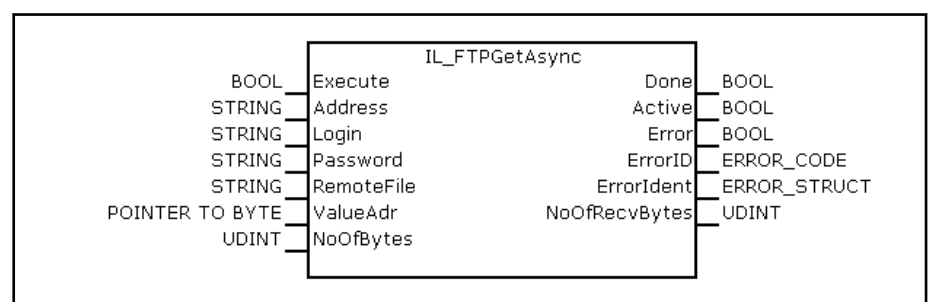


Abb. 6-38: Funktionsbaustein IL_FTPGetAsync

	Name	Typ	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flanken-gesteuert)
	Address	STRING	Name oder IP-Adresse des FTP-Servers
	Login	STRING	Login auf dem FTP-Server, leer bedeutet Anonymus Zugriff
	Password	STRING	Passwort für das Login auf dem FTP-Server
	RemoteFile	STRING	Dateiname inklusive Pfad auf dem FTP-Server
	ValueAdr	POINTER TO BYTE	Startadresse des Speicherbereichs, in dem die Daten abgelegt werden sollen
	NoOfBytes	UDINT	Größe des in "ValueAdr" angegebenen Speicherbereichs
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen
	Active	BOOL	Bearbeitung läuft
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerinformation (siehe Fehlerbehandlung auf Seite 309)
	NoOfRecBytes	UDINT	Anzahl der tatsächlich übertragenen Bytes.

Tab. 6-57: Interface IL_FTPGetAsync

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
Address	STRING			"	Steigende Flanke an Execute
Login	STRING			"	Steigende Flanke an Execute
Password	STRING			"	Steigende Flanke an Execute
RemoteFile	STRING			"	Steigende Flanke an Execute
ValueAdr	POINTER TO BYTE			0	Steigende Flanke an Execute
NoOfBytes	UDINT	0	65535	0	Steigende Flanke an Execute

Tab. 6-58: Min-, Max- und Default-Werte der Eingänge des Funktionsbausteins IL_FTPGetAsync

Funktionsbeschreibung

Der Funktionsbaustein IL_FTPGetAsync liest eine Datei von einem FTP-Server und legt sie im RAM der Steuerung in einem Datenbereich ab.

Bei steigender Flanke an "Execute" werden alle Eingänge übernommen und auf ihre Gültigkeit überprüft.

Danach beginnt die Verbindungsaufnahme mit dem in "Address" angegebenen FTP-Server. Als Port wird immer der für FTP registrierte TCP-Port 21 verwendet.

Nach erfolgreichem Verbindungsaufbau werden "Login" und "Password" übermittelt. Ist der Eingang "Login" leer, so wird versucht ein Login über den Nutzer "Anonymus" auszuführen.

War das Einloggen erfolgreich, wird der Dateitransfer gestartet.

Die in "RemoteFile" angegebene Datei wird vom FTP-Server in den in "ValueAdr" angegebenen Speicherbereich übertragen. Ist die Länge der Datei größer als der in "NoOfBytes" angegebene Wert, wird der Vorgang abgebrochen. Der Speicherbereich an "ValueAdr" enthält dann nicht die gesamte Datei. Es wird eine entsprechende Fehlermeldung generiert. Während der Übertragung kann der Fortschritt am Ausgang "NoOfRecBytes" abgelesen werden.

Die Daten werden immer im Binärmodus übertragen. Für die Datenübertragung wird vorzugsweise Passive-FTP verwendet, bei dem der FTP-Client eine zweite Verbindung zum FTP-Server aufbaut. Wird dieser Modus nicht unterstützt, so wird das so genannte Active-FTP verwendet, bei dem der FTP-Server für die Datenübertragung eine Verbindung zu FTP-Client aufbaut.

Der Funktionsbaustein benötigt mehrere Aufrufe, um die Operation zu beenden.

Die Übertragung ist erst erfolgreich abgeschlossen, wenn der Ausgang "Done" gesetzt ist. Während der Übertragung ist "Active" gesetzt. Tritt ein Fehler auf, so ist der Ausgang "Error" gesetzt. Die Ausgänge "ErrorID" und "ErrorIdent" enthalten dann nähere Informationen zu dem Fehler.



Der Funktionsbaustein muss nach steigender Flanke an "Execute" so lange aufgerufen werden, bis entweder "Done" oder "Error" gesetzt sind.

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#00000004	16#00000000	Eingang "ValueAdr" ist nicht gesetzt
INPUT_RANGE_ERROR	16#00000005	16#00000000	Eingang "NoOfBytes" ist 0
OTHER_ERROR	16#00000040	x	Socket konnte nicht erstellt werden
OTHER_ERROR	16#00000041	x	Fehler beim Setzen des I/O Modus
OTHER_ERROR	16#00000042	x	konnte Socket nicht an Port binden
OTHER_ERROR	16#00000043	x	Socket konnte nicht in den Listen-Modus versetzt werden.
OTHER_ERROR	16#00000044	x	Eingehende Verbindung konnte nicht angenommen werden.
OTHER_ERROR	16#00000045	x	Fehler beim Verbinden zu einem anderen Gerät.
INPUT_INVALID_ERROR	16#00000208	16#00000000	Zieladresse ungültig
COMMUNICATION_ERROR	16#00000209	16#00000000	Die Gegenstelle hat den Verbindungsaufbau abgelehnt
COMMUNICATION_ERROR	16#0000020C	16#00000000	Kein Netzwerkpfad zum Zielgerät gefunden
COMMUNICATION_ERROR	16#00002001	16#00000000	Eingang "RemoteFile" ist ungültig
COMMUNICATION_ERROR	16#00002002	16#00000000	Zielpuffer ist zu klein für zu übertragene Datei
COMMUNICATION_ERROR	16#00002003	16#00000078	FTP-Server ist nicht bereit
COMMUNICATION_ERROR	16#00002003	16#0000014C	Account für Login benötigt
COMMUNICATION_ERROR	16#00002003	16#000001A5	Dienst nicht bereit
COMMUNICATION_ERROR	16#00002003	16#000001A9	Kann Datenverbindung nicht öffnen

ErrorID	Additional1	Additional2	Beschreibung
COMMUNICATION_ERROR	16#00002003	16#000001AA	Verbindung geschlossen, Dateitransfer abgebrochen
COMMUNICATION_ERROR	16#00002003	16#000001C2	Datei nicht verfügbar
COMMUNICATION_ERROR	16#00002003	16#000001C3	Fehler beim Verarbeiten der Datei auf dem FTP-Server
COMMUNICATION_ERROR	16#00002003	16#000001C4	Nicht genug Speicherplatz auf FTP-Server
COMMUNICATION_ERROR	16#00002003	16#000001F4	Fehler im Kommandosyntax
COMMUNICATION_ERROR	16#00002003	16#000001F5	Falsche Kommandoparameter
COMMUNICATION_ERROR	16#00002003	16#000001F6	Kommando nicht implementiert
COMMUNICATION_ERROR	16#00002003	16#000001F7	Falsche Kommandoreihenfolge
COMMUNICATION_ERROR	16#00002003	16#000001F8	Kommando mit diesem Parameter nicht implementiert
COMMUNICATION_ERROR	16#00002003	16#00000212	Login oder Passwort inkorrekt
COMMUNICATION_ERROR	16#00002003	16#00000226	Datei nicht verfügbar
COMMUNICATION_ERROR	16#00002003	16#00000227	Unbekannter Page-Typ
COMMUNICATION_ERROR	16#00002003	16#00000228	Nicht genug Speicherplatz vorhanden
COMMUNICATION_ERROR	16#00002003	16#00000229	Dateiname nicht erlaubt
COMMUNICATION_ERROR	16#00002003	16#0000FFFE	Unerwartete Antwort vom FTP-Server
COMMUNICATION_ERROR	16#00002003	16#0000FFFF	Unbekannte Antwort vom FTP-Server

Tab. 6-59: Fehlercodes, die von IL_FTPGetAsync generiert werden

Weitere Fehlercodes sind in [ETHERNET_TABLE](#), [ERROR_TABLE 16#0150](#) auf Seite 334 aufgeführt.

6.10.3 IL_FTPPutAsync

Kurzbeschreibung Der Funktionsbaustein IL_FTPPutAsync legt einen Datenbereich als Datei auf einem FTP-Server ab.

Schnittstellenbeschreibung

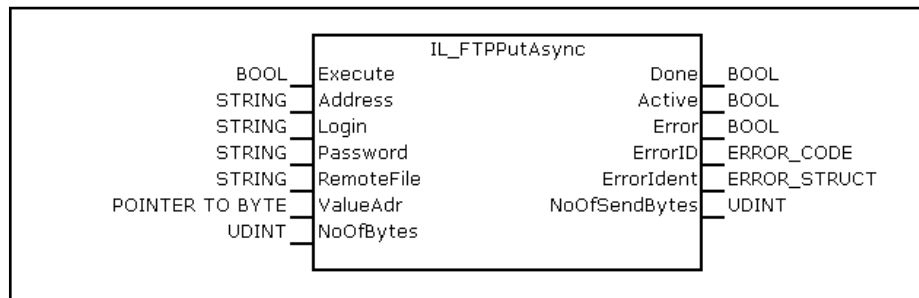


Abb. 6-39: Funktionsbaustein IL_FTPPutAsync

	Name	Typ	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	Address	STRING	Name oder IP-Adresse des FTP-Servers
	Login	STRING	Login auf dem FTP-Server, leer bedeutet Anonymus Zugriff
	Password	STRING	Passwort für das Login auf dem FTP-Server
	RemoteFile	STRING	Dateiname inklusive Pfad auf dem FTP-Server
	ValueAdr	POINTER TO BYTE	Startadresse der zu übertragenden Daten
	NoOfBytes	UDINT	Anzahl der zu übertragenden Zeichen

	Name	Typ	Beschreibung
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen
	Active	BOOL	Bearbeitung läuft
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerinformation (siehe Fehlerbehandlung auf Seite 312)
	NoOfSendBytes	UDINT	Anzahl der tatsächlich übertragenen Bytes.

Tab. 6-60: Interface IL_FTPPutAsync

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
Address	STRING			"	Steigende Flanke an Execute
Login	STRING			"	Steigende Flanke an Execute
Password	STRING			"	Steigende Flanke an Execute
RemoteFile	STRING			"	Steigende Flanke an Execute
ValueAdr	POINTER TO BYTE			0	Steigende Flanke an Execute
NoOfBytes	UDINT	0	65535	0	Steigende Flanke an Execute

Tab. 6-61: Min-, Max- und Default-Werte der Eingänge des Funktionsbausteins IL_FTPPutAsync

Funktionsbeschreibung

Der Funktionsbaustein IL_FTPPutAsync überträgt Daten aus einem Datenbereich im RAM der Steuerung auf einen FTP-Server und legt sie dort als Datei ab.

Bei steigender Flanke an "Execute" werden alle Eingänge übernommen und auf ihre Gültigkeit überprüft. Danach beginnt die Verbindungsaufnahme mit dem in "Address" angegebenen FTP-Server.

Als Port wird immer der für FTP registrierte TCP-Port 21 verwendet.

Nach erfolgreichem Verbindungsaufbau werden "Login" und "Password" übermittelt. Ist der Eingang "Login" leer, so wird versucht ein Login über den Nutzer "Anonymus" auszuführen.

War das Einloggen erfolgreich, wird der Dateitransfer gestartet. Es werden "NoOfBytes" Bytes vom in "ValueAdr" angegebenen Speicherbereich an den FTP-Server übertragen und dort als Datei unter dem Namen, der in "RemoteFile" angegeben wurde, abgelegt. Während der Übertragung kann der Fortschritt am Ausgang "NoOfSendBytes" abgelesen werden.

Die Daten werden immer im Binärmodus übertragen. Für die Datenübertragung wird vorzugsweise Passive-FTP verwendet, bei dem der FTP-Client eine zweite Verbindung zum FTP-Server aufbaut. Wird dieser Modus nicht unterstützt, so wird das so genannte Active-FTP verwendet, bei dem der FTP-Server für die Datenübertragung eine Verbindung zum FTP-Client aufbaut.

Der Funktionsbaustein benötigt mehrere Aufrufe, um die Operation zu beenden. Die Übertragung ist erst erfolgreich abgeschlossen, wenn der Ausgang "Done" gesetzt ist. Während der Übertragung ist "Active" gesetzt. Tritt ein Fehler auf, so ist der Ausgang "Error" gesetzt. Die Ausgänge "ErrorID" und "ErrorIdent" enthalten dann nähere Informationen zu dem Fehler.



Der Funktionsbaustein muss nach steigender Flanke an "Execute" so lange aufgerufen werden, bis entweder "Done" oder "Error" gesetzt sind.

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#00000004	16#00000000	Eingang "ValueAdr" ist nicht gesetzt
INPUT_RANGE_ERROR	16#00000005	16#00000000	Eingang "NoOfBytes" ist 0
OTHER_ERROR	16#00000040	x	Socket konnte nicht erstellt werden
OTHER_ERROR	16#00000041	x	Fehler beim Setzen des I/O Modus
OTHER_ERROR	16#00000042	x	Konnte Socket nicht an Port binden
OTHER_ERROR	16#00000043	x	Socket konnte nicht in den Listen-Modus versetzt werden.
OTHER_ERROR	16#00000044	x	Die eingehende Verbindung konnte nicht angenommen werden.
OTHER_ERROR	16#00000045	x	Fehler beim Verbinden zu einem anderen Gerät.
INPUT_INVALID_ERROR	16#00000208	16#00000000	Zieladresse ungültig
COMMUNICATION_ERROR	16#00000209	16#00000000	Die Gegenstelle hat den Verbindungsaufbau abgelehnt
COMMUNICATION_ERROR	16#0000020C	16#00000000	Kein Netzwerkpfad zum Zielgerät gefunden
COMMUNICATION_ERROR	16#00002001	16#00000000	Eingang "RemoteFile" ist ungültig
COMMUNICATION_ERROR	16#00002003	16#00000078	FTP-Server ist nicht bereit
COMMUNICATION_ERROR	16#00002003	16#0000014C	Account für Login benötigt
COMMUNICATION_ERROR	16#00002003	16#000001A5	Dienst nicht bereit
COMMUNICATION_ERROR	16#00002003	16#000001A9	Kann Datenverbindung nicht öffnen
COMMUNICATION_ERROR	16#00002003	16#000001AA	Verbindung geschlossen, Dateitransfer abgebrochen
COMMUNICATION_ERROR	16#00002003	16#000001C2	Datei nicht verfügbar
COMMUNICATION_ERROR	16#00002003	16#000001C3	Fehler beim Verarbeiten der Datei auf dem FTP-Server
COMMUNICATION_ERROR	16#00002003	16#000001C4	Nicht genug Speicherplatz auf FTP-Server
COMMUNICATION_ERROR	16#00002003	16#000001F4	Fehler im Kommandosyntax
COMMUNICATION_ERROR	16#00002003	16#000001F5	Falsche Kommandoparameter
COMMUNICATION_ERROR	16#00002003	16#000001F6	Kommando nicht implementiert
COMMUNICATION_ERROR	16#00002003	16#000001F7	Falsche Kommandoreihenfolge
COMMUNICATION_ERROR	16#00002003	16#000001F8	Kommando mit diesem Parameter nicht implementiert
COMMUNICATION_ERROR	16#00002003	16#00000212	Login oder Passwort inkorrekt
COMMUNICATION_ERROR	16#00002003	16#00000226	Datei nicht verfügbar
COMMUNICATION_ERROR	16#00002003	16#00000227	Unbekannter Page-Typ

ErrorID	Additional1	Additional2	Beschreibung
COMMUNICATION_ERROR	16#00002003	16#00000228	Nicht genug Speicherplatz
COMMUNICATION_ERROR	16#00002003	16#00000229	Dateiname nicht erlaubt
COMMUNICATION_ERROR	16#00002003	16#0000FFFE	Unerwartete Antwort vom FTP-Server
COMMUNICATION_ERROR	16#00002003	16#0000FFFF	Unbekannte Antwort vom FTP-Server

Tab. 6-62: Fehlercodes, die von IL_FTPPutAsync generiert werden

Weitere Fehlercodes sind in [ETHERNET_TABLE](#), [ERROR_TABLE 16#0150](#) auf Seite 334 aufgeführt.

6.10.4 IL_FTPGetFileAsync

Kurzbeschreibung Der Funktionsbaustein IL_FTPGetFileAsync liest eine Datei von einem FTP-Server und legt sie auf der Flash-Karte ab.

Schnittstellenbeschreibung

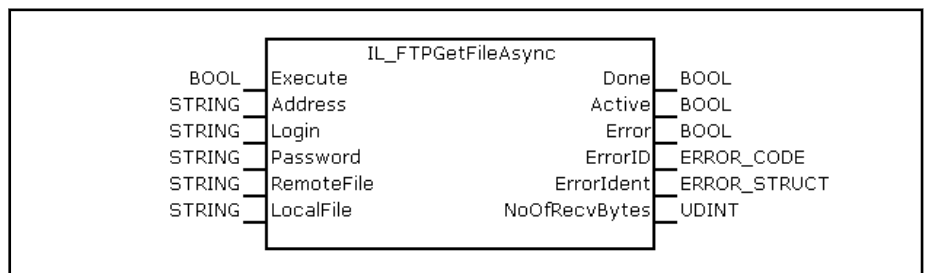


Abb. 6-40: Funktionsbaustein IL_FTPGetFileAsync

	Name	Typ	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	Address	STRING	Name oder IP-Adresse des FTP-Servers
	Login	STRING	Login auf dem FTP-Server, leer bedeutet Anonymus Zugriff
	Password	STRING	Passwort für das Login auf dem FTP-Server
	RemoteFile	STRING	Dateiname inklusive Pfad auf dem FTP-Server
	LocalFile	STRING	Dateiname inklusive Pfad auf der Flash-Karte
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen
	Active	BOOL	Bearbeitung läuft
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerinformation (siehe Fehlerbehandlung auf Seite 314)
	NoOfRecvBytes	UDINT	Anzahl der tatsächlich übertragenen Bytes.

Tab. 6-63: Interface IL_FTPGetFileAsync

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
Address	STRING			"	Steigende Flanke an Execute
Login	STRING			"	Steigende Flanke an Execute

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Password	STRING			"	Steigende Flanke an Execute
RemoteFile	STRING			"	Steigende Flanke an Execute
LocalFile	STRING			"	Steigende Flanke an Execute

Tab. 6-64: Min-, Max- und Default-Werte der Eingänge des Funktionsbausteins IL_FTPGetFileAsync

Funktionsbeschreibung

Der Funktionsbaustein IL_FTPGetAsync liest eine Datei von einem FTP-Server und legt sie auf der Flash-Karte der Steuerung ab.

Bei steigender Flanke an "Execute" werden alle Eingänge übernommen und auf ihre Gültigkeit überprüft. Danach beginnt die Verbindungsaufnahme mit dem in "Address" angegebenen FTP-Server. Als Port wird immer der für FTP registrierte TCP-Port 21 verwendet.

Nach erfolgreichem Verbindungsaufbau werden "Login" und "Password" übermittelt. Ist der Eingang "Login" leer, so wird versucht ein Login über die Nutzer "Anonymus" auszuführen.

War das Einloggen erfolgreich, so wird der Dateitransfer gestartet. Die in "RemoteFile" angegebene Datei wird vom FTP-Server auf die Flash-Karte übertragen und unter dem in "LocalFile" angegebenen Dateinamen abgelegt. Während der Übertragung kann der Fortschritt am Ausgang "NoOfRecBytes" abgelesen werden.

Die Daten werden immer im Binärmodus übertragen. Für die Datenübertragung wird vorzugsweise Passive-FTP verwendet, bei dem der FTP-Client eine zweite Verbindung zum FTP-Server aufbaut. Wird dieser Modus nicht unterstützt, so wird das Active-FTP verwendet, bei dem der FTP-Server für die Datenübertragung eine Verbindung zu FTP-Client aufbaut.

Der Funktionsbaustein benötigt mehrere Aufrufe, um die Operation zu beenden. Die Übertragung ist erst erfolgreich abgeschlossen, wenn der Ausgang "Done" gesetzt ist. Während der Übertragung ist "Active" gesetzt. Tritt ein Fehler auf, so ist der Ausgang "Error" gesetzt. Die Ausgänge "ErrorID" und "ErrorIdent" enthalten dann nähere Informationen zu dem Fehler.



Der Funktionsbaustein muss nach steigender Flanke an "Execute" so lange aufgerufen werden, bis entweder "Done" oder "Error" gesetzt sind.

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#00000004	16#00000000	Eingang "ValueAdr" ist nicht gesetzt
INPUT_RANGE_ERROR	16#00000005	16#00000000	Eingang "NoOfBytes" ist 0
OTHER_ERROR	16#00000040	x	Socket konnte nicht erstellt werden
OTHER_ERROR	16#00000041	x	Fehler beim Setzen des I/O Modus
OTHER_ERROR	16#00000042	x	konnte Socket nicht an Port binden
OTHER_ERROR	16#00000043	x	Socket konnte nicht in den Listen-Modus versetzt werden.
OTHER_ERROR	16#00000044	x	Eingehende Verbindung konnte nicht angenommen werden.
OTHER_ERROR	16#00000045	x	Fehler beim Verbinden zu einem anderen Gerät.

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR	16#00000208	16#00000000	Zieladresse ungültig
COMMUNICATION_ERROR	16#00000209	16#00000000	Die Gegenstelle hat den Verbindungsaufbau abgelehnt
COMMUNICATION_ERROR	16#0000020C	16#00000000	Kein Netzwerkpfad zum Zielgerät gefunden
COMMUNICATION_ERROR	16#00002001	16#00000000	Eingang "RemoteFile" ist ungültig
COMMUNICATION_ERROR	16#00002002	16#00000000	Zielpuffer ist zu klein für zu übertragene Datei
COMMUNICATION_ERROR	16#00002003	16#00000078	FTP-Server ist nicht bereit
COMMUNICATION_ERROR	16#00002003	16#0000014C	Account für Login benötigt
COMMUNICATION_ERROR	16#00002003	16#000001A5	Dienst nicht bereit
COMMUNICATION_ERROR	16#00002003	16#000001A9	Kann Datenverbindung nicht öffnen
COMMUNICATION_ERROR	16#00002003	16#000001AA	Verbindung geschlossen, Dateitransfer abgebrochen
COMMUNICATION_ERROR	16#00002003	16#000001C2	Datei nicht verfügbar
COMMUNICATION_ERROR	16#00002003	16#000001C3	Fehler beim Verarbeiten der Datei auf dem FTP-Server
COMMUNICATION_ERROR	16#00002003	16#000001C4	Nicht genug Speicherplatz auf FTP-Server
COMMUNICATION_ERROR	16#00002003	16#000001F4	Fehler im Kommandosyntax
COMMUNICATION_ERROR	16#00002003	16#000001F5	Falsche Kommandoparameter
COMMUNICATION_ERROR	16#00002003	16#000001F6	Kommando nicht implementiert
COMMUNICATION_ERROR	16#00002003	16#000001F7	Falsche Kommandoreihenfolge
COMMUNICATION_ERROR	16#00002003	16#000001F8	Kommando mit diesem Parameter nicht implementiert
COMMUNICATION_ERROR	16#00002003	16#00000212	Login oder Passwort inkorrekt
COMMUNICATION_ERROR	16#00002003	16#00000226	Datei nicht verfügbar
COMMUNICATION_ERROR	16#00002003	16#00000227	Unbekannter Page-Typ
COMMUNICATION_ERROR	16#00002003	16#00000228	Nicht genug Speicherplatz vorhanden
COMMUNICATION_ERROR	16#00002003	16#00000229	Dateiname nicht erlaubt
COMMUNICATION_ERROR	16#00002003	16#0000FFFE	Unerwartete Antwort vom FTP-Server
COMMUNICATION_ERROR	16#00002003	16#0000FFFF	Unbekannte Antwort vom FTP-Server
ACCESS_ERROR	16#00002004	16#00000000	LocalFile konnte nicht angelegt werden
ACCESS_ERROR	16#00002004	16#00000002	LocalFile konnte nicht geschrieben werden
ACCESS_ERROR	16#00002004	16#00000003	LocalFile konnte nicht geschlossen werden

Tab. 6-65: Fehlercodes, die von IL_FTPGetFileAsync generiert werden

Weitere Fehlercodes sind in [ETHERNET_TABLE](#), [ERROR_TABLE 16#0150](#) auf Seite 334 aufgeführt.

6.10.5 IL_FTPPutFileAsync

Kurzbeschreibung Der Funktionsbaustein IL_FTPPutFileAsync kopiert eine Datei von der Flash-Karte der Steuerung auf einen FTP-Server.

Schnittstellenbeschreibung

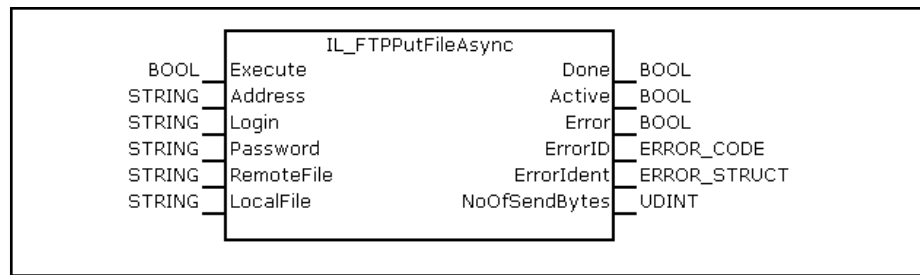


Abb. 6-41: Funktionsbaustein IL_FTPPutFileAsync

	Name	Typ	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flanken-gesteuert)
	Address	STRING	Name oder IP-Adresse des FTP-Servers
	Login	STRING	Login auf dem FTP-Server, leer bedeutet Anonymus Zugriff
	Password	STRING	Passwort für das Login auf dem FTP-Server
	RemoteFile	STRING	Dateiname inklusive Pfad auf dem FTP-Server
	LocalFile	STRING	Dateiname inklusive Pfad auf der Flash-Karte
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen
	Active	BOOL	Bearbeitung läuft
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerinformation (siehe Fehlerbehandlung auf Seite 317)
	NoOfSendBytes	UDINT	Anzahl der tatsächlich übertragenen Bytes.

Tab. 6-66: Interface IL_FTPPutFileAsync

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
Address	STRING			"	Steigende Flanke an Execute
Login	STRING			"	Steigende Flanke an Execute
Password	STRING			"	Steigende Flanke an Execute
RemoteFile	STRING			"	Steigende Flanke an Execute
LocalFile	STRING			"	Steigende Flanke an Execute

Tab. 6-67: Min-, Max- und Default-Werte der Eingänge des Funktionsbausteins IL_FTPPutFileAsync

Funktionsbeschreibung

Der Funktionsbaustein IL_FTPPutFileAsync kopiert eine Datei von der Flash-Karte der Steuerung auf einem FTP-Server.

Bei steigender Flanke an "Execute" werden alle Eingänge übernommen und auf ihre Gültigkeit überprüft. Danach beginnt die Verbindungsaufnahme mit dem in "Address" angegebenen FTP-Server.

Als Port wird immer der für FTP registrierte TCP-Port 21 verwendet.

Nach erfolgreichem Verbindungsaufbau werden "Login" und "Password" übermittelt. Ist der Eingang "Login" leer, so wird versucht ein Login über die Nutzer "Anonymus" auszuführen.

War das Einloggen erfolgreich, so wird der Dateitransfer gestartet. Die in "LocaleFile" angegebene Datei wird von der Flash-Karte der Steuerung übertragen und unter dem in "RemoteFile" angegebenen Dateinamen auf dem FTP-Server abgelegt. Während der Übertragung kann der Fortschritt am Ausgang "NoOfSendBytes" abgelesen werden.

Die Daten werden immer im Binärmodus übertragen. Für die Datenübertragung wird vorzugsweise Passive-FTP verwendet, bei dem der FTP-Client eine zweite Verbindung zum FTP-Server aufbaut. Wird dieser Modus nicht unterstützt, so wird das so genannte Active-FTP verwendet, bei dem der FTP-Server für die Datenübertragung eine Verbindung zu FTP-Client aufbaut.

Der Funktionsbaustein benötigt mehrere Aufrufe, um die Operation zu beenden. Die Übertragung ist erst erfolgreich abgeschlossen, wenn der Ausgang "Done" gesetzt ist. Während der Übertragung ist "Active" gesetzt.

Tritt ein Fehler auf, so ist der Ausgang "Error" gesetzt. Die Ausgänge "ErrorID" und "ErrorIdent" enthalten dann nähere Informationen zu dem Fehler.



Der Funktionsbaustein muss nach steigender Flanke an "Execute" so lange aufgerufen werden, bis entweder "Done" oder "Error" gesetzt sind.

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#00000004	16#00000000	Eingang "ValueAdr" ist nicht gesetzt
INPUT_RANGE_ERROR	16#00000005	16#00000000	Eingang "NoOfBytes" ist 0
OTHER_ERROR	16#00000040	x	Socket konnte nicht erstellt werden
OTHER_ERROR	16#00000041	x	Fehler beim Setzen des I/O Modus
OTHER_ERROR	16#00000042	x	konnte Socket nicht an Port binden
OTHER_ERROR	16#00000043	x	Socket konnte nicht in den Listen-Modus versetzt werden.
OTHER_ERROR	16#00000044	x	Eingehende Verbindung konnte nicht angenommen werden.
OTHER_ERROR	16#00000045	x	Fehler beim Verbinden zu einem anderen Gerät.
INPUT_INVALID_ERROR	16#00000208	16#00000000	Zieladresse ungültig
COMMUNICATION_ERROR	16#00000209	16#00000000	Die Gegenstelle hat den Verbindungsaufbau abgelehnt
COMMUNICATION_ERROR	16#0000020C	16#00000000	Kein Netzwerkpfad zum Zielgerät gefunden
COMMUNICATION_ERROR	16#00002001	16#00000000	Eingang "RemoteFile" ist ungültig
COMMUNICATION_ERROR	16#00002003	16#00000078	FTP-Server ist nicht bereit
COMMUNICATION_ERROR	16#00002003	16#0000014C	Account für Login benötigt
COMMUNICATION_ERROR	16#00002003	16#000001A5	Dienst nicht bereit
COMMUNICATION_ERROR	16#00002003	16#000001A9	Kann Datenverbindung nicht öffnen
COMMUNICATION_ERROR	16#00002003	16#000001AA	Verbindung geschlossen, Dateitransfer abgebrochen
COMMUNICATION_ERROR	16#00002003	16#000001C2	Datei nicht verfügbar

ErrorID	Additional1	Additional2	Beschreibung
COMMUNICATION_ERROR	16#00002003	16#000001C3	Fehler beim Verarbeiten der Datei auf dem FTP-Server
COMMUNICATION_ERROR	16#00002003	16#000001C4	Nicht genug Speicherplatz auf FTP-Server
COMMUNICATION_ERROR	16#00002003	16#000001F4	Fehler im Kommandosyntax
COMMUNICATION_ERROR	16#00002003	16#000001F5	Falsche Kommandoparameter
COMMUNICATION_ERROR	16#00002003	16#000001F6	Kommando nicht implementiert
COMMUNICATION_ERROR	16#00002003	16#000001F7	Falsche Kommandoreihenfolge
COMMUNICATION_ERROR	16#00002003	16#000001F8	Kommando mit diesem Parameter nicht implementiert
COMMUNICATION_ERROR	16#00002003	16#00000212	Login oder Passwort inkorrekt
COMMUNICATION_ERROR	16#00002003	16#00000226	Datei nicht verfügbar
COMMUNICATION_ERROR	16#00002003	16#00000227	Unbekannter Page-Typ
COMMUNICATION_ERROR	16#00002003	16#00000228	Nicht genug Speicherplatz vorhanden
COMMUNICATION_ERROR	16#00002003	16#00000229	Dateiname nicht erlaubt
COMMUNICATION_ERROR	16#00002003	16#0000FFFE	Unerwartete Antwort vom FTP-Server
COMMUNICATION_ERROR	16#00002003	16#0000FFFF	Unbekannte Antwort vom FTP-Server
ACCESS_ERROR	16#00002004	16#00000000	LocalFile konnte nicht angelegt werden
ACCESS_ERROR	16#00002004	16#00000001	LocalFile konnte nicht gelesen werden
ACCESS_ERROR	16#00002004	16#00000003	LocalFile konnte nicht geschlossen werden

Tab. 6-68: Fehlercodes, die von IL_FTTPutFileAsync generiert werden

Weitere Fehlercodes sind in [ETHERNET_TABLE](#), [ERROR_TABLE 16#0150](#) auf Seite 334 aufgeführt.

6.11 Asynchrone TFTP-Funktionsbausteine

6.11.1 Einführung und Überblick

Viele Peripherie-Geräte nutzen zum Übertragen von Firmware oder der Konfiguration das TFTP-Protokoll (Trivial File Transfer Protocol). Dieses Protokoll lässt sich besonders ressourcenschonend implementieren und wird daher von Geräten mit geringem Speicherausbau verwendet.

Die beiden Funktionsbausteine IL_TFTPGetFileAsync und IL_TFTPPutFileAsync ermöglichen die Nutzung des TFTP-Protokolls zur Dateiübertragung aus der SPS heraus.

Dateien können zwischen der Flash-Karte der Steuerung und dem Zielgerät übertragen werden.

Beide Bausteine arbeiten asynchron, d.h. sie schließen die Operation nicht beim ersten Aufruf ab, sondern müssen mehrfach aufgerufen werden, bis die Operation abgeschlossen ist oder ein Fehler gemeldet wird.

Asynchrone Funktionsbausteine für TFTP:

- [IL_TFTPGetFileAsync](#) auf Seite 318
- [IL_TFTPPutFileAsync](#) auf Seite 321

6.11.2 IL_TFTPGetFileAsync

Kurzbeschreibung Der Funktionsbaustein IL_TFTPGetFileAsync überträgt eine Datei von einem entfernten Gerät auf die CompactFlash-Karte der Steuerung unter Nutzung des TFTP-Protokolls.

Schnittstellenbeschreibung

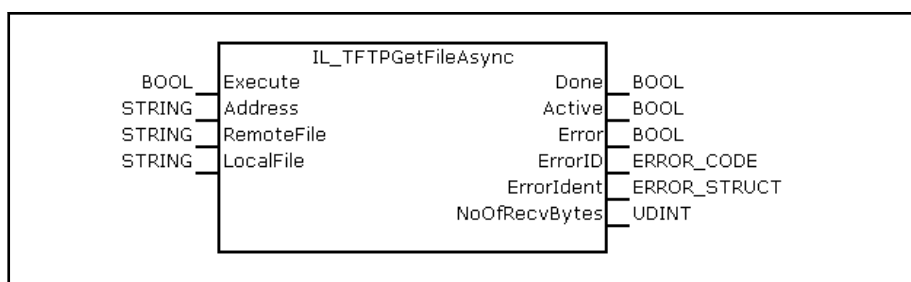


Abb. 6-42: Funktionsbaustein IL_TFTPGetFileAsync

	Name	Typ	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	Address	STRING	Name oder IP-Adresse des Zielgeräts
	RemoteFile	STRING	Dateiname inklusive Pfad auf dem Zielgerät
	LocalFile	STRING	Dateiname inklusive Pfad auf der Flash-Karte
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen
	Active	BOOL	Bearbeitung läuft
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerinformation (siehe Fehlerbehandlung auf Seite 320)
	NoOfRecvBytes	UDINT	Anzahl der tatsächlich übertragenen Bytes.

Tab. 6-69: Interface IL_TFTPGetFileAsync

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
Address	STRING			"	Steigende Flanke an Execute
RemoteFile	STRING			"	Steigende Flanke an Execute
LocalFile	STRING			"	Steigende Flanke an Execute

Tab. 6-70: Min-, Max- und Default-Werte der Eingänge des Funktionsbausteins IL_TFTPGetFileAsync

Funktionsbeschreibung

Der Funktionsbaustein IL_TFTPGetFileAsync überträgt die in "RemoteFile" angegebene Datei von einem entfernten Gerät unter Nutzung des TFTP-Protokolls und legt sie auf der auf die CompactFlash-Karte der Steuerung unter dem Dateinamen "LocalFile" ab.

Voraussetzungen

Das Zielgerät muss von der Steuerung aus über das Internet-Protokoll (IPv4) erreichbar sein. Dies ist der Fall, wenn das Gerät von der Steuerung aus über Ping (s. [IL_PingAsync auf Seite 304](#)) erreichbar ist. Es ist jedoch nicht notwendig, dies explizit durch das SPS-Programm zu prüfen. Das Zielgerät muss einen lesenden Zugriff über das TFTP-Protokoll unterstützen.

Address

Der Eingang "Address" beinhaltet die IP-Adresse des Zielgeräts. Diese muss in der für IPv4 üblichen Notation aus vier, durch Punkte getrennte, Dezimalzahlen zwischen 0 und 255 bestehen.

Beispiel:

"192.168.1.42"

RemoteFile

Der Eingang "RemoteFile" gibt den Dateinamen auf dem Zielgerät an. Dadurch kann eine bestimmte Datei vom Zielgerät abgerufen werden. Nicht jedes Zielgerät unterstützt die Angabe eines Dateinamens auf dem Zielgerät. In diesem Fall wird der Eingang "RemoteFile" ignoriert.

LocalFile

Der Eingang "LocalFile" gibt den Dateiname auf der lokalen CompactFlash-Karte der Steuerung an. Er kann auch einen kompletten Pfad enthalten. Das Standardverzeichnis ist die USER-Partition der Steuerung.

Zeitverhalten Der Funktionsbaustein IL_TFTPGetFileAsync arbeitet asynchron. Nach dem Start durch eine steigende Flanke am Eingang "Execute" liest er die Eingänge "Address", "RemoteFile" und "LocalFile".



Der Funktionsbaustein muss so lange zyklisch aufgerufen werden, bis die erfolgreiche Übertragung durch den Ausgang "Done" signalisiert wird, oder der Ausgang "Error" einen Fehler anzeigt. Während dieser Zeit ist der Ausgang "Active" gesetzt.

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
OTHER_ERROR	16#00000040	x	Socket konnte nicht erstellt werden
OTHER_ERROR	16#00000041	x	Fehler beim Setzen des I/O Modus
OTHER_ERROR	16#00000042	x	konnte Socket nicht an Port binden
INPUT_INVALID_ERROR	16#00000208	16#00000000	Zieladresse ungültig
COMMUNICATION_ERROR	16#0000020C	16#00000000	Kein Netzwerkpfad zum Zielgerät gefunden
INPUT_INVALID_ERROR	16#00002101	16#00000000	Eingang "RemoteFile" ist ungültig
INPUT_INVALID_ERROR	16#00002102	16#00000000	Eingang "LocalFile" ist ungültig
INPUT_INVALID_ERROR	16#00002103	16#00000000	"LocalFile" nicht gefunden
ACCESS_ERROR	16#00002104	16#00000000	"LocalFile" konnte nicht geschrieben werden
COMMUNICATION_ERROR	16#00002105	16#00000000	Fehler beim Verbindungsaufbau
COMMUNICATION_ERROR	16#00002106	16#00000000	Zeitüberschreitung während der Übertragung
COMMUNICATION_ERROR	16#00002107	16#00000000	Unerwarteter Kommunikationsfehler
ACCESS_ERROR	16#00002107	16#00000001	"RemoteFile" nicht gefunden
ACCESS_ERROR	16#00002107	16#00000002	Zugriffsverletzung
ACCESS_ERROR	16#00002107	16#00000003	Nicht genug Speicherplatz auf FTP-Server
COMMUNICATION_ERROR	16#00002107	16#00000004	Unerlaubte TFTP-Operation
COMMUNICATION_ERROR	16#00002107	16#00000005	Unbekannte Transfer ID
COMMUNICATION_ERROR	16#00002107	16#00000006	Datei existiert bereits

ErrorID	Additional1	Additional2	Beschreibung
ACCESS_ERROR	16#00002107	16#00000007	Unbekannter Nutzer
ACCESS_ERROR	16#00002108	16#00000000	"LocalFile" konnte nicht geöffnet werden.
ACCESS_ERROR	16#00002108	16#00000002	"LocalFile" konnte nicht geschrieben werden.
ACCESS_ERROR	16#00002108	16#00000003	"LocalFile" konnte nicht geschlossen werden.

Tab. 6-71: Fehlercodes, die von IL_TFTPGetFileAsync generiert werden

Weitere Fehlercodes sind in [ETHERNET_TABLE](#), [ERROR_TABLE 16#0150](#) auf Seite 334 aufgeführt.

6.11.3 IL_TFTPPutFileAsync

Kurzbeschreibung

Der Funktionsbaustein IL_TFTPPutFileAsync überträgt eine Datei von der CompactFlash-Karte der Steuerung auf ein entferntes Gerät unter Nutzung des TFTP-Protokolls.

Schnittstellenbeschreibung

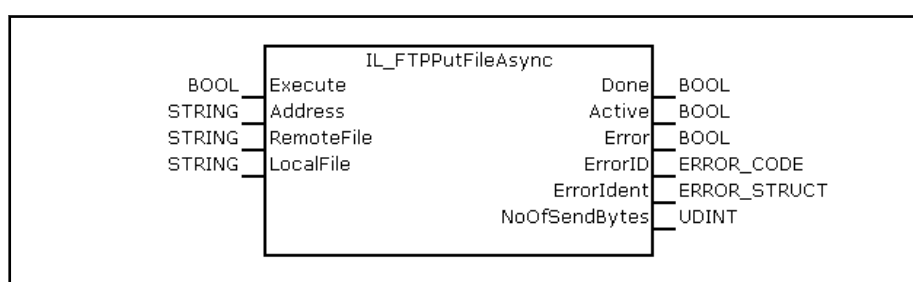


Abb. 6-43: Funktionsbaustein IL_TFTPPutFileAsync

	Name	Typ	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	Address	STRING	Name oder IP-Adresse des Zielgeräts
	RemoteFile	STRING	Dateiname inklusive Pfad auf dem Zielgerät
	LocalFile	STRING	Dateiname inklusive Pfad auf der Flash-Karte
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen
	Active	BOOL	Bearbeitung läuft
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerinformation (siehe Fehlerbehandlung auf Seite 322)
	NoOfSendBytes	UDINT	Anzahl der tatsächlich übertragenen Bytes.

Tab. 6-72: Interface IL_TFTPPutFileAsync

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
Address	STRING			"	Steigende Flanke an Execute

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
RemoteFile	STRING			"	Steigende Flanke an Execute
LocalFile	STRING			"	Steigende Flanke an Execute

Tab. 6-73: Min-, Max- und Default-Werte der Eingänge des Funktionsbausteins IL_TFTPPutFileAsync

Funktionsbeschreibung

Der Funktionsbaustein IL_TFTPPutFileAsync überträgt die in "LocalFile" angegebene Datei von der CompactFlash-Karte der Steuerung auf ein entferntes Gerät unter Nutzung des TFTP-Protokolls und legt sie unter dem Dateinamen "RemoteFile" ab.

Voraussetzungen

Das Zielgerät muss von der Steuerung aus über das Internet-Protokoll (IPv4) erreichbar sein. Dies ist der Fall, wenn das Gerät von der Steuerung aus über Ping ([IL_PingAsync auf Seite 304](#)) erreichbar ist. Es ist jedoch nicht notwendig, dies explizit durch das SPS-Programm zu prüfen. Das Zielgerät muss einen schreibenden Zugriff über das TFTP-Protokoll unterstützen.

Address

Der Eingang "Address" beinhaltet die IP-Adresse des Zielgeräts. Diese muss in der für IPv4 üblichen Notation aus vier, durch Punkte getrennte, Dezimalzahlen zwischen 0 und 255 bestehen.

Beispiel:

"192.168.1.42"

RemoteFile

Der Eingang "RemoteFile" gibt den Dateinamen auf dem Zielgerät an. Dadurch kann eine bestimmte Datei vom Zielgerät abgerufen werden. Nicht jedes Zielgerät unterstützt die Angabe eines Dateinamens auf dem Zielgerät. In diesem Fall wird der Eingang "RemoteFile" ignoriert.

LocalFile

Der Eingang "LocalFile" gibt den Dateinamen auf der lokalen CompactFlash-Karte der Steuerung an. Er kann auch einen kompletten Pfad enthalten. Das Standardverzeichnis ist die USER-Partition der Steuerung.

Zeitverhalten

Der Funktionsbaustein IL_TFTPPutFileAsync arbeitet asynchron. Nach dem Start durch eine steigende Flanke am Eingang "Execute" liest er die Eingänge "Address", "RemoteFile" und "LocalFile".



Der Funktionsbaustein muss so lange zyklisch aufgerufen werden, bis die erfolgreiche Übertragung durch den Ausgang "Done" signalisiert wird, oder der Ausgang "Error" einen Fehler anzeigt. Während dieser Zeit ist der Ausgang "Active" gesetzt.

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
OTHER_ERROR	16#00000040	x	Socket konnte nicht erstellt werden
OTHER_ERROR	16#00000041	x	Fehler beim Setzen des I/O Modus
OTHER_ERROR	16#00000042	x	konnte Socket nicht an Port binden
INPUT_INVALID_ERROR	16#00000208	16#00000000	Zieladresse ungültig

ErrorID	Additional1	Additional2	Beschreibung
COMMUNICATION_ERROR	16#000020C	16#00000000	Kein Netzwerkpfad zum Zielgerät gefunden
INPUT_INVALID_ERROR	16#00002101	16#00000000	Eingang "RemoteFile" ist ungültig
INPUT_INVALID_ERROR	16#00002102	16#00000000	Eingang "LocalFile" ist ungültig
ACCESS_ERROR	16#00002104	16#00000000	"LocalFile" kann nicht geschrieben werden
COMMUNICATION_ERROR	16#00002105	16#00000000	Fehler beim Verbindungsaufbau
COMMUNICATION_ERROR	16#00002106	16#00000000	Zeitüberschreitung während der Übertragung
COMMUNICATION_ERROR	16#00002107	16#00000000	Unerwarteter Kommunikationsfehler
ACCESS_ERROR	16#00002107	16#00000001	"RemoteFile" nicht gefunden
ACCESS_ERROR	16#00002107	16#00000002	Zugriffsverletzung
COMMUNICATION_ERROR	16#00002107	16#00000004	Unerlaubte TFTP-Operation
COMMUNICATION_ERROR	16#00002107	16#00000005	Unbekannte Transfer ID
ACCESS_ERROR	16#00002107	16#00000007	Unbekannter Nutzer
ACCESS_ERROR	16#00002108	16#00000000	"LocalFile" konnte nicht geöffnet werden.
ACCESS_ERROR	16#00002108	16#00000001	"LocalFile" konnte nicht gelesen werden.
ACCESS_ERROR	16#00002108	16#00000003	"LocalFile" konnte nicht geschlossen werden.

Tab. 6-74: Fehlercodes, die von IL_TFTPPutFileAsync generiert werden

Weitere Fehlercodes sind in [ETHERNET_TABLE](#), [ERROR_TABLE 16#0150](#) auf Seite 334 aufgeführt.

6.12 Asynchrone Kommunikationsbausteine zur zyklische Kommunikation

6.12.1 Einführung und Überblick

Die Funktionsbausteine unter CyclicCommunication ermöglichen einen zyklisch Datenaustausch. Sie bauen auf die TCP- und UDP-Kommunikationsbausteine der RIL_SocketComm auf.

Die auszutauschenden Daten können in eigenen Datenstrukturen angegeben werden. Neben der zyklischen Kommunikation können Daten auch auf Anforderung ausgetauscht werden.

Das Anwendungsgebiet für zyklische Kommunikation ist der zyklische Datenaustausch zwischen zwei Steuerungen oder einem anderen Kommunikationspartner. Eventgesteuerte Kommunikation ist bei Abfragen von Daten, z.B. einer Kamera, notwendig. Beide Kommunikationsarten werden sowohl über das UDP- als auch das TCP-Protokoll unterstützt.

Asynchrone Funktionsbausteine für zyklischen Datenaustausch:

- [IL_TCPCyclic auf Seite 323](#)
- [IL_UDPCyclic auf Seite 328](#)

6.12.2 IL_TCPCyclic

Kurzbeschreibung

Der Funktionsbaustein **IL_TCPCyclic** sendet und empfängt Daten über eine TCP-basierende Verbindung.

Es werden der zyklische bidirektionale Betrieb und der eventgesteuerte Sendebetrieb unterstützt.

Um eine Verbindung aufzubauen, muss ein Kommunikationspartner die Rolle des Servers einnehmen, d.h. er hört auf eingehende Verbindungen. Der andere Kommunikationspartner nimmt die Rolle des Clients ein, d.h. er versucht eine TCP-Verbindung zum Server aufzubauen. Empfangende Pakete können mit jedem Baustein-Aufruf gelesen werden. Der Funktionsbaustein wird mit steigender Flanke an "Enable" aktiviert und muss zyklisch aufgerufen werden. "InOperation" zeigt eine bestehende Verbindung an.

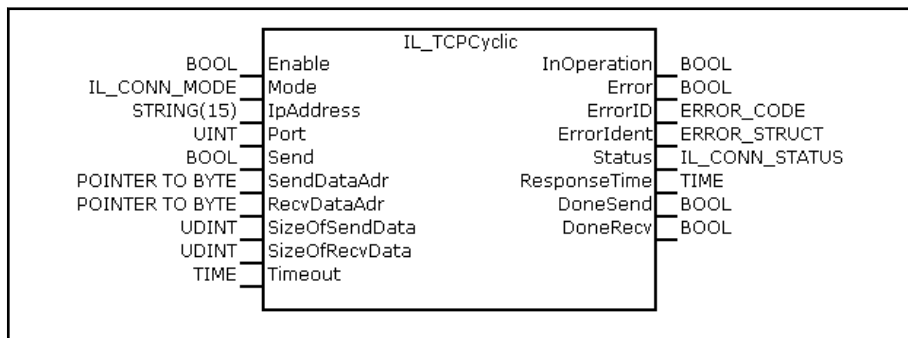
Schnittstelle

Abb. 6-44: Schnittstellenbeschreibung Funktionsbaustein IL_TCPCyclic

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Startet bei steigender Flanke
	Mode	IL_CONN_MODE	Legt den Verbindungsmodus fest, siehe IL_CONN_MODE auf Seite 332 .
	IpAddress	STRING(15)	IP-Adresse vom Kommunikationspartner in der Form aaa.bbb.ccc.ddd
	Port	UINT	Kommunikationsport (Default: Port 50000). Der Port sollte im Bereich von 50000 bis 65535 liegen. Kommunikationspartner haben identische Portnummern
	Send	BOOL	Steigende Flanke sendet Daten; ist nur bei den Modi "Cyclic_Event_Server", "Cyclic_Event_Client", "Cyclic_Event_Server_Protocol", "Cyclic_Event_Client_Protocol" verfügbar.
	SendDataAdr	POINTER TO BYTE	Pointer auf die zu sendenden Daten. Maximal 1400 Byte Daten können gesendet werden.
	RecvDataAdr	POINTER TO BYTE	Pointer auf die zu empfangenden Daten. Maximal 1400 Byte Daten können empfangen werden.
	SizeOfSendData	UDINT	Länge der zu sendenden Daten
	SizeOfRecvData	UDINT	Länge der zu empfangenden Daten
	Timeout	TIME	Optional; Timeout der Kommunikation (Default: 1500 ms)
VAR_OUTPUT	InOperation	BOOL	TRUE zeigt an, dass der Funktionsbaustein bereit ist, Daten zu senden oder empfangen.
	Error	BOOL	TRUE zeigt einen Fehler an.
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerinformation
	Status	IL_CONN_STATUS	Statusinformation über den Zustand des Funktionsbausteins, siehe IL_CONN_STATUS auf Seite 333 .

	Name	Typ	Beschreibung
	ResponseTime	TIME	Zeit die vom Senden bis zum erfolgreichen Empfangen benötigt wird.
	DoneSend	BOOL	TRUE zeigt für einen Zyklus an, dass Daten erfolgreich gesendet worden
	DoneRecv	BOOL	TRUE zeigt für einen Zyklus an, dass Daten erfolgreich empfangen worden

Tab. 6-75: Ein-/Ausgänge Funktionsbaustein IL_TCPcyclic

Min-/Max- und Default-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte der Bausteineingänge aufgelistet.

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Mode	IL_CONN_MODE	Cyclic_Server	Cyclic_Event_Client_Protocol	Cyclic_Server	steigende Flanke an "Enable"
IpAddress	STRING(15)	'0.0.0.0'	'255.255.255.255'		steigende Flanke an "Enable"
Port	UINT	0	65535	50000	steigende Flanke an "Enable"
Send	BOOL			FALSE	kontinuierlich, nur bei den Modes "Cyclic_Event_Server", "Cyclic_Event_Client", "Cyclic_Event_Server_Protocol", "Cyclic_Event_Client_Protocol"
SendDataAdr	POINTER TO BYTE	>0			kontinuierlich
RecvDataAdr	POINTER TO BYTE	>0			kontinuierlich
SizeOfSendData	UDINT	1	1400		kontinuierlich
SizeOfRecvData	UDINT	1	1400		kontinuierlich
Timeout	TIME			T#1500MS	kontinuierlich

Tab. 6-76: Eingangsverhalten IL_TCPcyclic

Verbindungsmodus

Der Eingang "Mode" gibt an, wie die Verbindung zwischen zwei Kommunikationspartnern aufgebaut wird. Weiterhin legt dieser Eingang fest, ob die Daten zyklisch oder eventgesteuert übertragen werden und ob die Länge der empfangenen Daten berücksichtigt wird.

Mögliche Werte sind im Aufzählungstyp [IL_CONN_MODE](#) auf Seite 332 beschrieben.

Die Länge der Übertragungsdaten wird bei den Modi "Server_Client", "Cyclic_Client", "Cyclic_Event_Server", "Cyclic_Event_Client" nicht berücksichtigt.

Falls die Anzahl der von der Gegenstelle gesendeten Bytes größer ist als der Puffer "RecvDataAdr", werden nur "SizeOfRecvData" Bytes in den angelegten Empfangspuffer „RecvDataAdr“ kopiert.

Funktionsbeschreibung

Die übrigen und neu empfangenen Bytes werden beim nächsten Zyklus weiter kopiert.

Bei den Modi "Cyclic_Server_Protocol", "Cyclic_Client_Protocol", "Cyclic_Event_Server_Protocol" und "Cyclic_Event_Client_Protocol" wird die Länge der Übertragungsdaten berücksichtigt.

Falls die Anzahl der von der Gegenstelle gesendeten Bytes größer als die konfigurierte Puffergröße ist, wird eine Fehler ausgegeben.

Der Funktionsbaustein IL_TCPCyclic sendet und empfängt Daten über eine TCP/IP-basierende Verbindung. Die Reihenfolge der gesendeten Zeichen wird immer beibehalten. Treten während der Übertragung Fehler auf, so wird das betreffende Paket automatisch wiederholt gesendet. Es können also keine Daten verloren gehen. Tritt ein Verbindungsverlust auf, so wird dies beim nächsten Lese- oder Schreibzugriff mit einem Fehlercode gemeldet. Grundsätzlich wird zwischen einer "aktiven" und einer "passiven" Verbindung unterschieden:

- Die "passive" Verbindung besteht darin, einen Port anzugeben und dann auf diesem Port auf Anfragen zu warten (das sog. "Listening"). Diese wird von Servern (Webserver, Mailserver etc.) eingesetzt.
- Für eine "aktive" Verbindung wird eine IP-Adresse der Gegenstelle und ein Port benötigt - dieser muss allerdings "offen" sein, d. h. auf dem Zielgerät muss ein Client Programm auf eben diesem Port eine passive Verbindung bestehen haben.

Nachdem die Verbindung aufgebaut wurde, können Daten in beide Richtungen übertragen werden. Der Funktionsbaustein IL_TCPCyclic realisiert eine zyklische Server / Client Ethernet basierende Netzwerkkommunikation zwischen Steuerungen.

Die Kommunikation zwischen Server und Client läuft azyklisch ab. Im Regelfall stehen die übertragenen Daten (1x hin + 1x zurück; also mit Quittierung) innerhalb von 2 SPS-Zyklen im jeweiligen SPS-Programm zur Weiterverarbeitung zur Verfügung.

Beispiel: Bei 10ms SPS-Zykluszeit dauert zwar ein Übertragungszyklus weniger als 20ms, die Daten stehen aber frühestens mit dem 2. SPS Zyklus zur Verfügung

Unterstützte Funktionen:

- Zyklische Kommunikation
- Verwendung als aktive (Client) und passive (Server) Verbindung
- Max. Telegrammlänge: 1400 Bytes
- Eventgesteuertes Senden
- Timeout-Überprüfung
- Überprüfung auf zweifachen Telegrammausfall beim Empfangen oder Senden
- Überprüfung auf Verbindungsunterbrechung

Der Ablauf der zyklischen Kommunikation ist wie folgt umgesetzt:

- Der Server öffnet eine Socket auf dem Port und wartet auf Anfragen
- Der Client baut eine Verbindung verbindet zum Port des Servers auf
- **Cyclic_Client:** Der Client fängt mit dem Datenaustausch an.
Er sendet ein Telegramm an den Server, startet die Zeitüberwachung (TimeOut) und wartet auf die Antwort
- **Cyclic_Client_Protocol:** Der Client fängt mit dem Datenaustausch an.

Ein Header mit Magicnummer 4711 und Datenlänge werden zuerst an den Server gesendet. Dann sendet der Client noch die Daten, startet die Zeitüberwachung (TimeOut) und wartet auf die Antwort.

- **Cyclic_Server:** Der Server empfängt die Daten vom Client.

Dann antwortet der Server mit einem Telegramm, startet die Zeitüberwachung und wartet auf die Antwort.

Cyclic_Server_Protocol: Der Server empfängt die Daten vom Client.

Der von Client gesendete Header wird zuerst erwartet. Die Magicnummer mit 4711 wird geprüft. Dann empfängt der Server die von Client gesendeten Daten. Die Datenlänge der empfangenen Daten wird gegen die im Header enthaltene Datenlänge geprüft. Danach antwortet der Server mit einem Telegramm, startet die Zeitüberwachung und wartet auf die Antwort.

- Zyklischen Wechsel Server - Client
- Für jedes gesendete Telegramm wird eine Antwort erwartet

Der Ablauf der eventgesteuerte Kommunikation ist wie folgt umgesetzt:

- Der Server öffnet eine Socket auf dem Port und wartet auf Anfragen
- Der Client baut eine Verbindung zum Port des Servers auf
- **Cyclic_Event_Client:** Falls eine steigende Flanke an dem Eingang „Send“ des Client eingegeben wird, sendet Client ein Telegramm an den Server und wartet auf die Antwort vom Server.

Cyclic_Event_Client_Protocol: Falls eine steigende Flanke am Eingang „Send“ des Client eingegeben wird, werden ein Header mit der Magicnummer 4711 und die Datenlänge zuerst an den Server gesendet. Dann sendet der Client noch die Daten und wartet auf die Antwort vom Server

- **Cyclic_Event_Server:** Der Server empfängt die Daten von Client.

Bei einer steigenden Flanke an dem Eingang „Send“ antwortet Server mit einem Telegramm und wartet auf die Antwort vom Client.

Cyclic_Event_Server_Protocol: Der Server empfängt die Daten von Client.

Der vom Client gesendete Header wird zuerst erwartet. Die Magicnummer wird mit 4711 geprüft. Dann empfängt der Server die von Client gesendeten Daten. Die Datenlänge wird gegen die im Header enthaltene Datenlänge geprüft. Danach, bei einer steigenden Flanke an dem Eingang „Send“, antwortet der Server mit einem Telegramm (mit der Header) und wartet auf die Antwort vom Client.

Diagnosen Der Ausgang "Status" zeigt den Zustand der Verbindung zwischen zwei Kommunikationspartnern an.

Mögliche Werte sind im Aufzählungstyp [IL_CONN_STATUS](#) auf Seite 333 beschrieben.

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150.

ErrorID	Additional1	Additional2	Beschreibung
OTHER_ERROR	16#00000040	x	Socket konnte nicht erstellt werden
OTHER_ERROR	16#00000041	x	Fehler beim Setzen des I/O Modus
OTHER_ERROR	16#00000042	x	konnte Socket nicht an Port binden
OTHER_ERROR	16#00000043	x	Socket konnte nicht in den Listen-Modus versetzt werden.

ErrorID	Additional1	Additional2	Beschreibung
OTHER_ERROR	16#00000044	x	Eingehende Verbindung konnte nicht angenommen werden.
OTHER_ERROR	16#00000045	x	Fehler beim Verbinden zu einem anderen Gerät.
INPUT_INVALID_ERROR	16#00000202	16#00000000	Der Socket-Handle ist ungültig.
INPUT_INVALID_ERROR	16#00000204	16#00000000	Die Portnummer liegt außerhalb des gültigen Bereichs
COMMUNICATION_ERROR	16#00000207	16#00000000	Der Port ist bereits belegt.
INPUT_INVALID_ERROR	16#00000208	16#00000000	Zieladresse ungültig
COMMUNICATION_ERROR	16#00000209	16#00000000	Die Gegenstelle hat den Verbindungsaufbau abgelehnt
COMMUNICATION_ERROR	16#0000020A	16#00000000	Es ist ein Timeout aufgetreten.
COMMUNICATION_ERROR	16#0000020B	16#00000000	Die Gegenstelle wurde nicht gefunden.
COMMUNICATION_ERROR	16#0000020C	16#00000000	Kein Netzwerkpfad zum Zielgerät gefunden
COMMUNICATION_ERROR	16#0000020E	16#00000000	Die Verbindung wurde geschlossen.
COMMUNICATION_ERROR	16#0000020F	16#00000000	Die Verbindungsabbruch durch Software
INPUT_RANGE_ERROR	16#00000210	16#00000000	Die Adresse in SendDataAdr darf nicht 0 oder kleiner sein
INPUT_RANGE_ERROR	16#00000210	16#00000001	Die Größe von SizeOfSendData muss mindestens 1 sein.
INPUT_RANGE_ERROR	16#00000210	16#00000002	SizeOfSendData darf maximal 1400 sein. Es können maximal 1400 Byte übertragen werden.
INPUT_RANGE_ERROR	16#00000210	16#00000003	Die Adresse in RecvDataAdr darf nicht 0 oder kleiner sein.
INPUT_RANGE_ERROR	16#00000210	16#00000004	Die Größe von SizeOfRecvData muss mindestens 1 sein.
INPUT_RANGE_ERROR	16#00000210	16#00000005	SizeOfRecvData darf maximal 1400 sein. Es können maximal 1400 Byte empfangen werden.
INPUT_RANGE_ERROR	16#00000210	16#00000006	SizeOfRecvData ist kleiner als Menge der empfangenen Daten
COMMUNICATION_ERROR	16#00000211	16#00000000	Die Gegenstelle hat die Verbindung geschlossen.
COMMUNICATION_ERROR	16#00000212	16#00000000	Ungültiges Protokoll
COMMUNICATION_ERROR	16#00001001	16#00000000	Datenübertragungsfehler, gesendete Daten entsprechen nicht den empfangenen Daten
COMMUNICATION_ERROR	16#00001002	16#00000000	Unbekannter Verbindungsmodus am Eingang Mode.

Tab. 6-77: Fehlercodes, die von IL_TCPCyclic generiert werden

Weitere Fehlercodes sind in [ETHERNET_TABLE](#), [ERROR_TABLE 16#0150](#) auf Seite 334 aufgeführt.

6.12.3 IL_UDPCyclic

Kurzbeschreibung

Der Funktionsbaustein **IL_UDPCyclic** sendet und empfängt Daten über eine UDP-basierende Verbindung.

Es werden der zyklische bidirektionale Betrieb und der eventgesteuerte Sendebetrieb unterstützt.

Schnittstelle

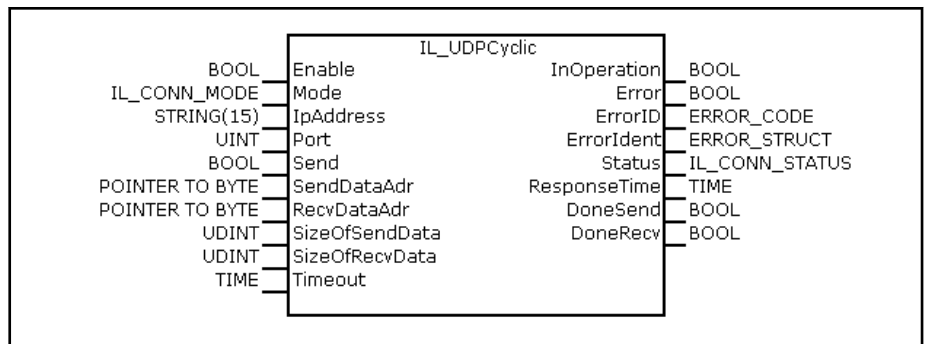


Abb. 6-45: Schnittstellenbeschreibung Funktionsbaustein IL_UDPCyclic

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Startet bei steigender Flanke
	Mode	IL_CONN_MODE	Legt den Verbindungsmodus fest, siehe IL_CONN_MODE auf Seite 332 .
	IpAddress	STRING(15)	IP-Adresse vom Kommunikationspartner in der Form aaa.bbb.ccc.ddd
	Port	UINT	Kommunikationsport (Default: Port 50000). Der Port sollte im Bereich von 50000 bis 65535 liegen. Kommunikationspartner haben einen identischen Port
	Send	BOOL	Steigende Flanke sendet Daten; ist nur bei den Modi "Cyclic_Event_Server" und "Cyclic_Event_Client" verfügbar.
	SendDataAdr	POINTER TO BYTE	Pointer auf die zu sendenden Daten. Maximal 1400 Byte Daten können gesendet werden.
	RecvDataAdr	POINTER TO BYTE	Pointer auf die zu empfangenden Daten. Maximal 1400 Byte Daten können empfangen werden.
	SizeOfSendData	UDINT	Länge der zu sendenden Daten
	SizeOfRecvData	UDINT	Länge der zu empfangenden Daten
	Timeout	TIME	Optional; Timeout der Kommunikation (Default: 1500 ms)
VAR_OUTPUT	InOperation	BOOL	TRUE zeigt an, dass der Funktionsbaustein bereit ist, Daten zu senden oder empfangen.
	Error	BOOL	TRUE zeigt einen Fehler an.
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerinformation
	Status	IL_CONN_STATUS	Status Information über den Zustand des Funktionsbausteins, siehe IL_CONN_STATUS auf Seite 333
	ResponseTime	TIME	Zeit die vom Senden bis zum erfolgreichen Empfangen benötigt wird.
	DoneSend	BOOL	TRUE zeigt für einen Zyklus an, dass Daten erfolgreich gesendet worden
	DoneRecv	BOOL	TRUE zeigt für einen Zyklus an, dass Daten erfolgreich empfangen worden

Tab. 6-78: Ein-/Ausgänge Funktionsbaustein IL_UDPCyclic

Min-/Max- und Default-Werte In der folgenden Tabelle sind die Min-/Max- und Default-Werte der Bausteineingänge aufgelistet.

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Mode	IL_CONN_MODE	Cyclic_Server	Cyclic_Event_Client	Cyclic_Server	steigende Flanke an "Enable"
IpAddress	STRING(15)	'0.0.0.0'	'255.255.255.255'		steigende Flanke an "Enable"
Port	UINT	0	65535	50000	steigende Flanke an "Enable"
Send	BOOL			FALSE	kontinuierlich, nur bei den Modes "Cyclic_Event_Server" und "Cyclic_Event_Client"
SendDataAdr	POINTER TO BYTE	>0			kontinuierlich
RecvDataAdr	POINTER TO BYTE	>0			kontinuierlich
SizeOfSendData	UDINT	1	1400		kontinuierlich
SizeOfRecvData	UDINT	1	1400		kontinuierlich
Timeout	TIME			T#1500MS	kontinuierlich

Tab. 6-79: Eingangsverhalten IL_UDPCyclic

Verbindungsmodus Der Eingang Mode gibt an, wie die Verbindung zwischen zwei Kommunikationspartnern aufgebaut wird. Weiterhin legt dieser Eingang fest, ob die Daten zyklisch oder eventgesteuert übertragen werden.

Mögliche Werte sind im Aufzählungstyp [IL_CONN_MODE](#) auf Seite 332 beschrieben.

Funktionsbeschreibung Der Funktionsbaustein **IL_UDPCyclic** sendet und empfängt Daten über eine UDP-basierende Verbindung.

Es werden der zyklische bidirektionale Betrieb und der eventgesteuerte Sendebetrieb unterstützt.

UDP ist ein verbindungsloses IP-Protokoll. Im Gegensatz zum TCP-Protokoll gibt es keinen Verbindungsaufbau vor dem eigentlichen Datenaustausch. Die Kommunikation erfolgt paketbasiert. Empfangende Datenpakete werden zwischengespeichert, aber nicht zusammengefügt. Jeder Empfangsaufruf holt genau ein Paket aus der Warteschlange. Empfangene UDP-Datenpakete werden nicht bestätigt. Daher ist es nicht möglich verloren gegangene Pakete zu erkennen. UDP bietet sich daher für zyklische Datenübertragung an, in der die Daten von eventuell verloren gegangenen Paketen durch spätere Pakete aktualisiert werden.

Zum Kommunizieren ist es notwendig, dass ein Server einen offenen Port für einen Client zur Verfügung stellt. Empfangende Pakete können innerhalb eines Baustein-Aufrufes empfangen und gelesen werden. Der Funktionsbaustein wird mit steigender Flanke an "Enable" aktiviert und muss zyklisch aufgerufen werden.

Der Server öffnet einen UDP-Socket auf einem vorgegebenen Port, während der UDP-Port des Clients unbekannt ist. In dieser Konstellation kann der Client nur senden, jedoch nicht empfangen. Mit der Konstellation Server-Server

ist es möglich, eine bidirektionale UDP Server-Server Kommunikation zu realisieren. In dieser Konstellation können beide Kommunikationspartner Datenpakete senden und empfangen, sofern der jeweilige Zielport angegeben ist.

Aus Sicht des SPS-Programms läuft die Kommunikation zwischen Server und Client azyklisch ab. Im Regelfall stehen die Übertragenen Daten (1x hin + 1x zurück; also mit Quittierung) innerhalb von 2 SPS-Zyklen im jeweiligen SPS-Programm zur Weiterverarbeitung zur Verfügung. Beispiel: Bei 10ms SPS-Zykluszeit dauert zwar ein Übertragungszyklus weniger als 20ms, die Daten stehen aber frühestens mit dem 2. SPS-Zyklus zur Verfügung

Unterstützte Funktionen:

- Zyklische Kommunikation
- Verwendung als aktive (Client) und passive (Server) Verbindung
- Telegrammlänge: 1400 Bytes
- Eventgesteuertes Senden
- Timeout-Überprüfung
- Zweifacher Telegrammausfall: Überprüfung beim Empfangen oder Senden
- Verbindungsunterbrechung-Überprüfung

Das zyklische Verhalten wurde durch folgende Maßnahmen umgesetzt:

- Server öffnet einen Socket auf dem Port und wartet auf die Anfragen
- Client verbindet sich zu dem Socket auf dem Port auf dem Server
- Client fängt mit dem Datenaustausch an, sendet ein Telegramm an den Server, startet die Zeitüberwachung (TimeOut) und wartet auf die Antwort
- Server empfängt die Daten vom Client, antwortet mit einem Telegramm, startet die Zeitüberwachung und wartet auf die Antwort
- Client empfängt die Daten vom Server, antwortet mit einem Telegramm, startet die Zeitüberwachung und wartet auf die Antwort
- Zyklischen Wechsel Server - Client, wie oben beschrieben
- Es wird eine Antwort auf jedes gesendete Telegramm erwartet

Diagnosen Der Ausgang "Status" gibt den Zustand der Verbindung zwischen zwei Kommunikationspartnern aus.

Mögliche Werte sind im Aufzählungstyp [IL_CONN_STATUS](#) auf Seite 333 beschrieben.

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle ETHERNET_TABLE, 16#0150.

ErrorID	Additional1	Additional2	Beschreibung
OTHER_ERROR	16#00000040	x	Socket konnte nicht erstellt werden
OTHER_ERROR	16#00000041	x	Fehler beim Setzen des I/O Modus
OTHER_ERROR	16#00000042	x	konnte Socket nicht an Port binden
OTHER_ERROR	16#00000043	x	Socket konnte nicht in den Listen-Modus versetzt werden.
OTHER_ERROR	16#00000044	x	Eingehende Verbindung konnte nicht angenommen werden.
OTHER_ERROR	16#00000045	x	Fehler beim Verbinden zu einem anderen Gerät.
INPUT_INVALID_ERROR	16#00000202	16#00000000	Der Socket-Handle ist ungültig.

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR	16#00000204	16#00000000	Die Portnummer liegt außerhalb des gültigen Bereichs
COMMUNICATION_ERROR	16#00000207	16#00000000	Der Port ist bereits belegt.
INPUT_INVALID_ERROR	16#00000208	16#00000000	Zieladresse ungültig
COMMUNICATION_ERROR	16#00000209	16#00000000	Die Gegenstelle hat den Verbindungsaufbau abgelehnt
COMMUNICATION_ERROR	16#0000020A	16#00000000	Es ist ein Timeout aufgetreten.
COMMUNICATION_ERROR	16#0000020B	16#00000000	Die Gegenstelle wurde nicht gefunden.
COMMUNICATION_ERROR	16#0000020C	16#00000000	Kein Netzwerkpfad zum Zielgerät gefunden
COMMUNICATION_ERROR	16#0000020E	16#00000000	Die Verbindung wurde geschlossen.
COMMUNICATION_ERROR	16#0000020F	16#00000000	Die Verbindungsabbruch durch Software
INPUT_RANGE_ERROR	16#00000210	16#00000000	Die Adresse in SendDataAdr darf nicht 0 oder kleiner sein.
INPUT_RANGE_ERROR	16#00000210	16#00000001	Die Größe von SizeOfSendData muss mindestens 1 sein.
INPUT_RANGE_ERROR	16#00000210	16#00000002	SizeOfSendData darf maximal 1400 sein. Es können maximal 1400 Byte übertragen werden.
INPUT_RANGE_ERROR	16#00000210	16#00000003	Die Adresse in RecvDataAdr darf nicht 0 oder kleiner sein.
INPUT_RANGE_ERROR	16#00000210	16#00000004	Die Größe von SizeOfRecvData muss mindestens 1 sein.
INPUT_RANGE_ERROR	16#00000210	16#00000005	SizeOfRecvData darf maximal 1400 sein. Es können maximal 1400 Byte empfangen werden.
COMMUNICATION_ERROR	16#00000211	16#00000000	Die Gegenstelle hat die Verbindung geschlossen.
COMMUNICATION_ERROR	16#00001001	16#00000000	Datenübertragungsfehler, gesendete Daten entsprechen nicht den empfangenen Daten
COMMUNICATION_ERROR	16#00001002	16#00000000	Unbekannter Verbindungsmodus am Eingang Mode.

Tab. 6-80: Fehlercodes, die von IL_UDPCyclic generiert werden

Weitere Fehlercodes sind in [ETHERNET_TABLE](#), [ERROR_TABLE 16#0150](#) auf Seite 334 aufgeführt.

6.12.4 IL_CONN_MODE

Der Aufzählungsdatentyp IL_CONN_MODE gibt an, wie die Verbindung zwischen zwei Kommunikationpartnern aufgebaut wird.

Weiterhin legt er fest, ob die Daten zyklisch oder eventgesteuert mit oder ohne Protokoll übertragen werden.

ENUM	Beschreibung
Cyclic_Server	Server, zyklischer Betrieb, wartet auf das erste Telegramm vom Client
Cyclic_Client	Client, zyklischer Betrieb, startet den Datenaustausch
Cyclic_Event_Server	wie Cyclic_Server, über eine Flanke an "Send" können Daten eventgesteuert gesendet werden
Cyclic_Event_Client	wie Cyclic_Client, über eine Flanke an "Send" können Daten eventgesteuert gesendet werden
Cyclic_Server_Protocol	Server, zyklischer Betrieb, wartet auf das erste Telegramm vom Client. Magicnummer und Übertragungsdatenlänge sind berücksichtigt (nur für IL_TCPCyclic auf Seite 323 verfügbar)
Cyclic_Client_Protocol	Client, zyklischer Betrieb, startet den Datenaustausch. Magicnummer und Übertragungsdatenlänge werden berücksichtigt (nur für IL_TCPCyclic auf Seite 323 verfügbar)

ENUM	Beschreibung
Cyclic_Event_Server_Protocol	wie Cyclic_Server_Protocol, über eine Flanke an "Send" können Daten eventgesteuert gesendet werden. Magicnummer und Übertragungsdatenlänge werden berücksichtigt (nur für IL_TCPCyclic auf Seite 323 verfügbar)
Cyclic_Event_Server_Protocol	wie Cyclic_Client_Protocol, über eine Flanke an "Send" können Daten eventgesteuert gesendet werden. Magicnummer und Übertragungsdatenlänge werden berücksichtigt (nur für IL_TCPCyclic auf Seite 323 verfügbar)

Tab. 6-81: Verbindungsmodus der Funktionsbausteine IL_TCPCyclic und IL_UDPCyclic



- Die Kommunikation der Funktionsbausteine ist bidirektional aufgebaut.
- Beim Empfangen wird immer genau die unter "SizeOfRecvData" angegebene Anzahl an Byte erwartet (gilt für Mode = Cyclic_Server, Mode = Cyclic_Client, Mode = Cyclic_Server_Protocol oder Mode = Cyclic_Client_Protocol).
- Bei Mode = Cyclic_Event_* ist es möglich weniger Daten zu empfangen.
- Bei der Kommunikation zwischen verschiedenen Steuerungsvarianten oder Fremdsteuerungen ist auf "natural alignment" und "packed alignment" zu achten, d.h. der Speicherbereich wird fortlaufend in WORD-Größe beschrieben und kann daher Lücken, sogenannte Füllbytes, enthalten.

Empfehlung:

Mit den großen Datentypen beginnen, z.B. DWORD - WORD - BYTE - BOOL.

STRING ist vergleichbar mit BYTE

- Für die Modi "Cyclic_Server_Protocol", "Cyclic_Client_Protocol", "Cyclic_Event_Server_Protocol" und "Cyclic_Event_Client_Protocol":

Header und die Übertragungsdaten werden getrennt gesendet / empfangen.

Beim Send:

Der Header mit zwei Elemente "MagicNumber (4711, WORD)" und "DataSize (UINT) = "SizeOfSendData" der Gegenstelle" wird zuerst gesendet.

Dann werden die Daten gesendet.

Beim Receive:

Erwartet wird zunächst der Header.

Die zwei Elemente des Headers werden geprüft.

Wenn MagicNumber (WORD) ungleich 4711 ist oder "DataSize (UINT)" größer ist als der Eingangswert "SizeOfRecvData", wird ein Fehler ausgegeben.

Tritt kein Fehler auf, werden die Daten empfangen.

6.12.5 IL_CONN_STATUS

Der Aufzählungsdatentyp IL_CONN_STATUS gibt den aktuellen Verbindungsstatus zwischen zwei Kommunikationpartnern an.

ENUM	Beschreibung
NotConnected	Der Funktionsbaustein ist inaktiv. Es besteht keine Verbindung
WaitingForConnection	Es wird eine Verbindung aufgebaut bzw. es wird auf eine eingehende Verbindung gewartet
Running	Es besteht eine aktive Kommunikation zwischen Server und Client (Default-Zustand im Betrieb; enthält die einzelnen Status Receiving, Received, Transmitting, Transmitted)
Receiving	Daten werden empfangen
Received	Datenempfangen ist erfolgreich abgeschlossen
Transmitting	Daten werden gesendet
Transmitted	Datensenden ist erfolgreich abgeschlossen
ConnectionTest	Ping-Verbindungstest wird durchgeführt
Waiting	Zustand warten
ReadyToSend	Daten können gesendet werden
TimeOutRx	Der konfigurierte (Default = 1500ms) Timeout-Wert wurde erreicht
TimeOutTx	Der konfigurierte (Default = 1500ms) Timeout-Wert wurde erreicht
NotReadyToSend	Daten können nicht gesendet werden
DataLengthError	Das Datenpaket ist zu groß, d.h. die SendData bzw. RecvData ist größer als 1400 Byte
ConnectionError	Fehler in der Verbindung, detaillierte Informationen sind in der Dokumentation der Bibliothek "RIL_SocketComm.lib" zu finden
ChecksumError	Es sind fehlerhafte Daten übermittelt worden. Die gesendeten Daten stimmen nicht mit den empfangenen Daten überein
DataTransmittingError	Fehler im Senden von Daten, detaillierte Informationen sind in der Dokumentation der Bibliothek "RIL_SocketComm.lib" zu finden
DataReceivingError	Fehler im Empfangen von Daten, detaillierte Informationen sind in der Dokumentation zu der Bibliothek "RIL_SocketComm.lib" zu finden

Tab. 6-82: Diagnosestatus der Funktionsbausteine IL_TCPcyclic und IL_UDP-Cyclic

6.12.6 ETHERNET_TABLE, ERROR_TABLE 16#0150

ETHERNET_TABLE, Allgemeines

ETHERNET_TABLE - ERROR_TABLE, Tabellennummer 16#0150

Die Tabelle enthält die Fehlernummern, die ausgegeben werden von:

- Funktionen und Funktionsbausteinen der RIL_SocketComm-Bibliothek
Die RIL_SocketComm Bibliothek enthält Funktionen und Funktionsbausteine zum Aufbau einer Verbindung und zum Datentransfer mit TCP (Transmission Control Protocol) und UDP (User Datagram Protocol).
Die RIL_SocketComm-Bibliothek verwendet Funktionen der SysSockets-Bibliothek, Runtime-Fehlercodes (CmpErrors.library).
- Funktionsbaustein IL_UdpKeys, RIL_HMI_Uutilities.library
Der Funktionsbaustein IL_UdpKeys ermöglicht dem Anwender Tastendrucke von einem HMI-Gerät sicher an die Steuerung zu übertragen. Die Verbindung erfolgt über UDP.
- Funktionsbaustein IL_ModbusTCPClient, RIL_ModbusTCP.library

Der Funktionsbaustein IL_ModbusTCPClient implementiert einen Modbus TCP-Master. Er dient dem Aufbau einer Verbindung zu einem ModbusTCP-Server.

Die Tabelle ist in aufsteigender Nummerierung von ERROR_STRUCT/ Additional1 geordnet.

ETHERNET_TABLE, General, 16#00000001-16#00000212

Additional1	Additional2	ErrorID	Ursache/ Lösung
16#00000001	16#00000000	STATE_MACHINE_ERROR	Ungültiger Zustand der Zustandsmaschine
16#00000004	16#00000000	INPUT_RANGE_ERROR	Keine Adresse am Eingang 'ValueAdr'
16#00000005	16#00000000	INPUT_RANGE_ERROR	Eingang 'NoOfByte' ist 0
16#00000010	16#00000000	INPUT_INVALID_ERROR	Falsche IP-Syntax
16#00000011	16#00000000	COMMUNICATION_ERROR	Unbekannte IP-Adresse oder keine Verbindung
16#00000012	16#00000000	DEVICE_ERROR	Ungültige interne Puffergröße
16#00000040	x	OTHER_ERROR	Socket konnte nicht erstellt werden
16#00000041	x	OTHER_ERROR	Fehler beim Setzen des I/O Modus
16#00000042	x	OTHER_ERROR	Konnte Socket nicht an Port binden
16#00000043	x	OTHER_ERROR	Socket konnte nicht in den Listen-Modus versetzt werden
16#00000044	x	OTHER_ERROR	Die eingehende Verbindung konnte nicht angenommen werden
16#00000045	x	OTHER_ERROR	Fehler beim Verbinden zu einem anderen Gerät
16#00000045	16#00000209	OTHER_ERROR	Verbindungsaufbau wurde von der Gegenstelle abgelehnt. Die Gegenstelle erwartet keine eingehenden Verbindungen auf diesem Port
16#00000045	16#0000020A	OTHER_ERROR	Timeout beim Verbindungsaufbau zu einem anderen Gerät. Das Gerät antwortet nicht
16#00000045	16#0000020B	OTHER_ERROR	Timeout beim Verbindungsaufbau zu einem anderen Gerät. Das Gerät wurde nicht gefunden
16#00000045	16#0000020C	OTHER_ERROR	Verbindungsaufbau nicht möglich. Kein Netzwerkpfad zum Zielgerät
16#00000202	16#00000000	INPUT_INVALID_ERROR	Der Socket-Handle ist ungültig
16#00000204	16#00000000	INPUT_INVALID_ERROR	Die Portnummer liegt außerhalb des gültigen Bereichs
16#00000207	16#00000000	COMMUNICATION_ERROR	Der Port ist bereits belegt
16#00000208	16#00000000	INPUT_INVALID_ERROR	Zieladresse ungültig
16#00000209	16#00000000	COMMUNICATION_ERROR	Die Gegenstelle hat den Verbindungsaufbau abgelehnt
16#0000020A	16#00000000	COMMUNICATION_ERROR	Es ist ein Timeout aufgetreten
16#0000020B	16#00000000	COMMUNICATION_ERROR	Die Gegenstelle wurde nicht gefunden
16#0000020C	16#00000000	COMMUNICATION_ERROR	Kein Netzwerkpfad zum Zielgerät gefunden
16#0000020E	16#00000000	COMMUNICATION_ERROR	Die Verbindung wurde geschlossen
16#0000020F	16#00000000	COMMUNICATION_ERROR	Die Verbindungsabbruch durch Software

Additional1	Additional2	ErrorID	Ursache/ Lösung
16#00000210	16#00000000	COMMUNICATION_ERROR	Datenpaket zu groß, Adresse in SendDataAdr darf nicht 0 oder kleiner sein
16#00000210	16#00000001	COMMUNICATION_ERROR	Die Größe von SizeOfSendData muss mindestens 1 sein
16#00000210	16#00000002	COMMUNICATION_ERROR	SizeOfSendData darf maximal 1400 sein. Es können maximal 1400 Byte übertragen werden
16#00000210	16#00000003	COMMUNICATION_ERROR	Datenpaket ist zu groß, Adresse in RecvDataAdr darf nicht 0 oder kleiner sein
16#00000210	16#00000004	COMMUNICATION_ERROR	Größe von SizeOfRecvData muss mindestens 1 sein
16#00000210	16#00000005	COMMUNICATION_ERROR	SizeOfRecvData darf maximal 1400 sein. Es können maximal 1400 Byte empfangen werden
16#00000210	16#00000006	COMMUNICATION_ERROR	SizeOfRecvData ist kleiner als Menge der empfangenen Daten
16#00000211	16#00000000	COMMUNICATION_ERROR	Die Gegenstelle hat die Verbindung geschlossen
16#00000212	16#00000000	COMMUNICATION_ERROR	Ungültiges Protokoll

ETHERNET_TABLE, IL_TcpCyclic, IL_UdpCyclic, 16#00001001-16#00001002

Additional1	Additional2	ErrorID	Ursache/ Lösung
16#00001001	16#00000000	COMMUNICATION_ERROR	Datenübertragungsfehler, gesendete Daten entsprechen nicht den empfangenen Daten
16#00001002	16#00000000	COMMUNICATION_ERROR	Unbekannter Verbindungsmodus am Eingang Mode

ETHERNET_TABLE, IL_FTPGetAsync, IL_FTPPutAsync, IL_FTPGetFileAsync, IL_FTP_PutFileAsync, 16#00002001-16#00002004

Additional1	Additional2	ErrorID	Ursache/ Lösung
16#00002001	16#00000000	COMMUNICATION_ERROR	Eingang 'RemoteFile' ist ungültig
16#00002002	16#00000000	COMMUNICATION_ERROR	Zielpuffer ist zu klein für zu übertragene Datei
16#00002003	16#00000078	COMMUNICATION_ERROR	FTP-Server ist nicht bereit
16#00002003	16#0000014C	COMMUNICATION_ERROR	Account für Login benötigt
16#00002003	16#000001A5	COMMUNICATION_ERROR	Dienst nicht bereit
16#00002003	16#000001A9	COMMUNICATION_ERROR	Kann Datenverbindung nicht öffnen
16#00002003	16#000001AA	COMMUNICATION_ERROR	Verbindung geschlossen, Dateitransfer abgebrochen
16#00002003	16#000001C2	COMMUNICATION_ERROR	Datei nicht verfügbar
16#00002003	16#000001C3	COMMUNICATION_ERROR	Fehler beim Verarbeiten der Datei auf dem FTP-Server
16#00002003	16#000001C4	COMMUNICATION_ERROR	Nicht genug Speicherplatz auf FTP-Server
16#00002003	16#000001F4	COMMUNICATION_ERROR	Fehler im Kommandosyntax
16#00002003	16#000001F5	COMMUNICATION_ERROR	Falscher Kommandoparameter
16#00002003	16#000001F6	COMMUNICATION_ERROR	Kommando nicht implementiert
16#00002003	16#000001F7	COMMUNICATION_ERROR	Falsche Kommandoreihenfolge

Additional1	Additional2	ErrorID	Ursache/ Lösung
16#00002003	16#000001F8	COMMUNICATION_ERROR	Kommando mit diesem Parameter nicht implementiert
16#00002003	16#00000212	COMMUNICATION_ERROR	Login oder Passwort inkorrekt
16#00002003	16#00000226	COMMUNICATION_ERROR	Datei nicht verfügbar
16#00002003	16#00000227	COMMUNICATION_ERROR	Unbekannter Page-Typ
16#00002003	16#00000228	COMMUNICATION_ERROR	Nicht genug Speicherplatz
16#00002003	16#00000229	COMMUNICATION_ERROR	Dateiname nicht erlaubt
16#00002003	16#0000FFFE	COMMUNICATION_ERROR	Unerwartete Antwort vom FTP-Server
16#00002003	16#0000FFFF	COMMUNICATION_ERROR	Unbekannte Antwort vom FTP-Server
16#00002004	16#00000000	ACCESS_ERROR	LocalFile konnte nicht angelegt werden
16#00002004	16#00000001	ACCESS_ERROR	LocalFile konnte nicht gelesen werden
16#00002004	16#00000002	ACCESS_ERROR	LocalFile konnte nicht geschrieben werden
16#00002004	16#00000003	ACCESS_ERROR	LocalFile konnte nicht geschlossen werden

ETHERNET_TABLE, IL_TFTPGetFileAsync, IL_TFTPPutFileAsync, 16#00002101-16#00002108

Additional1	Additional2	ErrorID	Ursache/ Lösung
16#00002101	16#00000000	COMMUNICATION_ERROR	Eingang 'RemoteFile' ist ungültig
16#00002102	16#00000000	COMMUNICATION_ERROR	Eingang 'LocalFile' ist ungültig
16#00002103	16#00000000	COMMUNICATION_ERROR	LocalFile nicht gefunden
16#00002104	16#00000000	COMMUNICATION_ERROR	LocalFile kann nicht geschrieben werden
16#00002105	16#00000000	COMMUNICATION_ERROR	Fehler beim Verbindungsaufbau
16#00002106	16#00000000	COMMUNICATION_ERROR	Zeitüberschreitung während der Übertragung
16#00002107	16#00000000	COMMUNICATION_ERROR	Unerwarteter Kommunikationsfehler
16#00002107	16#00000001	COMMUNICATION_ERROR	RemoteFile nicht gefunden
16#00002107	16#00000002	COMMUNICATION_ERROR	Zugriffsverletzung
16#00002107	16#00000003	COMMUNICATION_ERROR	Nicht genug Speicherplatz auf FTP-Server
16#00002107	16#00000004	COMMUNICATION_ERROR	Unerlaubte TFTP-Operation
16#00002107	16#00000005	COMMUNICATION_ERROR	Unbekannte Transfer ID
16#00002107	16#00000006	COMMUNICATION_ERROR	Datei existiert bereits
16#00002107	16#00000007	COMMUNICATION_ERROR	Unbekannter Nutzer
16#00002108	16#00000000	ACCESS_ERROR	LocalFile konnte nicht angelegt werden
16#00002108	16#00000001	ACCESS_ERROR	LocalFile konnte nicht gelesen werden
16#00002108	16#00000002	ACCESS_ERROR	LocalFile konnte nicht geschrieben werden
16#00002108	16#00000003	ACCESS_ERROR	LocalFile konnte nicht geschlossen werden

7 RIL_LoopControl.library

7.1 Einführung und Übersicht

Die RIL_LoopControl stellt Grundelemente und Regler der Regelungstechnik zur Verfügung.

Die meisten Bausteine sind sowohl als zeitdiskrete, als auch als eventgesteuerte Version verfügbar.

Alle Funktionsbausteine sind hardwareunabhängig nach IEC 61131-3 implementiert.

Die Bibliotheken RIL_CommonTypes und RIL_Uutilities werden vorausgesetzt.



Für die Bibliothek ist das Basis-Funktionspaket PLC des Systems MLC 15VRS ausreichend.

Anwendungsgebiete und Funktionen / Funktionsbausteine von RIL_LoopControl

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine	Beschreibung
Regler (Controllers)	
IL_PIDType01, Seite 343	PID-Regler mit Anti-Reset-Wind-up-Integrator und Begrenzer, zeitdiskret
IL_PIDType02, Seite 347	PID-Regler mit Anti-Reset-Wind-up-Integrator und Begrenzer, eventgesteuert
IL_PIDType03, Seite 351	2DOF (2 degree of freedom) PID Regler mit Anti-Reset-Wind-up-Integrator und Begrenzer, zeitdiskret
IL_PIDType04, Seite 355	2DOF (2 degree of freedom) PID Regler mit Anti-Reset-Wind-up-Integrator und Begrenzer, eventgesteuert
IL_PIType01, Seite 360	PI-Regler mit geglättetem Feedback-Eingang, bewerteter Regelabweichung, Anti-Reset-Wind-Up Integrator und Begrenzer am Ausgang, zeitdiskret
IL_PIType02, Seite 364	PI-Regler mit geglättetem Feedback-Eingang, bewerteter Regelabweichung, Anti-Reset-Wind-Up Integrator und Begrenzer am Ausgang, eventgesteuert
Grundelemente (Elements)	
IL_DeadtimeType01, Seite 369	Totzeitglied (zeitvariabel), zeitdiskret
IL_DeadtimeType02, Seite 372	Totzeitglied (zeitvariabel), eventgesteuert
IL_DeviationResponseCurve, Seite 375	Bewertungskurve
IL_DT1Type01, Seite 377	Diffenzierglied mit Verzögerung erster Ordnung, zeitdiskret
IL_DT1Type02, Seite 379	Diffenzierglied mit Verzögerung erster Ordnung, eventgesteuert
IL_PDT1Type01, Seite 381	Digitales Filter zum Anheben bzw. Absenken der Übertragungsfunktion ab einer bestimmten Frequenz, zeitdiskret
IL_PDT1Type02, Seite 385	Digitales Filter zum Anheben bzw. Absenken der Übertragungsfunktion ab einer bestimmten Frequenz, eventgesteuert
IL_PT1Type01, Seite 388	Verzögerungsglied erster Ordnung, zeitdiskret

IL_PT1Type02, Seite 390	Verzögerungsglied erster Ordnung, eventgesteuert
IL_TransportDelayType01, Seite 393	Totzeitglied (laufzeitvariabel), zeitdiskret
IL_TransportDelayType02, Seite 396	Totzeitglied (laufzeitvariabel), eventgesteuert
IL_ValueLimit, Seite 399	Wertebegrenzung
IL_ValueWindow, Seite 401	Wertebereichsprüfung
Filter (Filters)	
IL_FIRType01, Seite 403	Digitales FIR-Filter für Hoch-, Tief- und Bandpass-Verwendung, zeitdiskret
IL_FIRType02, Seite 405	Digitales FIR-Filter für Hoch-, Tief- und Bandpass-Verwendung, eventgesteuert
IL_IIRType01, Seite 407	Digitales IIR-Filter für Hoch-, Tief- und Bandpass-Verwendung, zeitdiskret
IL_IIRType02, Seite 409	Digitales IIR-Filter für Hoch-, Tief- und Bandpass-Verwendung, eventgesteuert
IL_StateVariableType01, Seite 412	IIR-Filter (Infinite Impulse Response) zweiter Ordnung für Hoch-, Tief- und Bandpass-Verwendung, sowie zum Einsatz als Bandsperre, zeitdiskret
IL_StateVariableType02, Seite 414	IIR-Filter (Infinite Impulse Response) zweiter Ordnung für Hoch-, Tief- und Bandpass-Verwendung, sowie zum Einsatz als Bandsperre, eventgesteuert
Generatoren (Generators)	
IL_NoiseGeneratorType01, Seite 419	Pseudozufallszahl und Rauschgenerator, zeitdiskret
IL_NoiseGeneratorType02, Seite 420	Rauschgenerator für weißes Rauschen, zeitdiskret
IL_PRBSGeneratorType01, Seite 422	Generator für pseudozufällige Binärzahlen, zeitdiskret
IL_PRBSGeneratorType02, Seite 424	Generator für pseudozufällige Binärzahlen, eventgesteuert
Signalgeneratoren (Generators/IL_Signalgenerator)	
IL_AngleGeneratorType01, Seite 426	Winkelgenerator, zeitdiskret
IL_AngleGeneratorType02, Seite 428	Winkelgenerator, eventgesteuert
IL_PhaseController, Seite 429	Phasenregler
IL_SignalGeneratorType01, Seite 431	Signalgenerator, zeitdiskret
IL_SignalGeneratorType02, Seite 435	Signalgenerator, eventgesteuert
IL_AmplitudeControllerType01, Seite 439	Amplitudenregler, zeitdiskret
IL_AmplitudeControllerType02, Seite 442	Amplitudenregler, eventgesteuert
Statistikbausteine (Statistics)	
IL_HistogramType01, Seite 447	sortiert Werte in Kategorien, zeitdiskret
IL_HistogramType02, Seite 449	sortiert Werte in Kategorien, eventgesteuert
IL_StatisticType01, Seite 451	ermittelt gleitenden Mittel-, Min- und Max-Wert, zeitdiskret
IL_StatisticType02, Seite 453	ermittelt gleitenden Mittel-, Min- und Max-Wert, eventgesteuert
IL_SuppressionType01, Seite 455	unterdrückt Messwerte, die eine exzessive Abweichung zum Mittelwert der Messwerte besitzen (1 Schranke), zeitdiskret

IL_SuppressionType02, Seite 460	unterdrückt Messwerte, die eine exzessive Abweichung zum Mittelwert der Messwerte besitzen (1 Schranke), eventgesteuert
IL_SuppressionType03, Seite 465	unterdrückt Messwerte, die eine exzessive Abweichung zum Mittelwert der Messwerte besitzen (2 Schranken), zeitdiskret
IL_SuppressionType04, Seite 470	unterdrückt Messwerte, die eine exzessive Abweichung zum Mittelwert der Messwerte besitzen (2 Schranken), eventgesteuert
Temperaturregler (TemperatureController)	
IL_TempControlType01, Seite 477	Regler für Temperaturstrecke mit automatischer offline Reglereinstellung
IL_TempControlType02, Seite 495	Regler für Temperaturstrecke mit automatischer online Reglereinstellung
Zweipunktregler (TwoPosCtrl)	
IL_TwoPosCtrlType01, Seite 509	Zweipunktregler mit Hysterese, ist für einfache Regelungen geeignet
Utilities (Utilities)	
IL_GetCurrentTimeUs	Dokumentation in Vorbereitung

Tab. 7-1: RIL_LoopControl-Bibliothek

7.2 Zeitdiskrete Funktionsbausteine

Diese Funktionsbausteine berechnen bei jedem Aufruf einen neuen Wert, wenn der "Enable"-Eingang aktiv ist. Die Zeit zwischen zwei Aufrufen wird gemessen und steht am Ausgang "ActScanTime" zur Verfügung. Alle Berechnungen beziehen sich auf die gemessene Zykluszeit.

Zeitdiskrete Funktionsbausteine sollten in einer zyklischen, besser noch, einer *sercos* synchronen Task aufgerufen werden. Dies kann eine signifikante Auswirkung auf die Genauigkeit des Ausgabewertes haben.

Die meisten Eingänge der Funktionsbausteine werden sofort beim nächsten Aufruf übernommen. Es ist somit möglich, Regelparameter im laufenden Betrieb zu ändern.

Ein aktiver "Pause"-Eingang wird durch Setzen des Ausgangs "InPause" quittiert. Die Ausgangswerte sind dann eingefroren und es finden keine internen Berechnungen mehr statt. "ActScanTime" wird auch bei aktivem "Pause" weiterhin aktualisiert.

Die Übernahme eines Wertes am "PresetValue"-Eingang auf den Ausgang "ControlValue" oder "OutputValue" erfolgt bei steigender Flanke an "Preset". Nach erfolgter Übernahme wird der Ausgang "PresetAck" aktiv. Ist "Preset" bei steigender Flanke an "Enable" aktiv, wird der Wert in "PresetValue" ebenfalls übernommen. Ein aktives "Preset" hat Priorität über den "Pause"-Eingang.

Zeitdiskrete Funktionsbausteine können direkt miteinander verkettet werden, solange sich alle Funktionsbausteine innerhalb derselben Task befinden. Der "InOperation"-Ausgang des vorhergehenden Funktionsbausteins wird mit dem "Enable"-Eingang des folgenden Funktionsbausteins verbunden. In gleicher Weise wird "InPause" auf "Pause" und "OutputValue" auf "Value" weitergegeben. Die Reihenfolge der Verkettung muss bei der Reihenfolge der Funktionsbausteinaufrufe beachtet werden.

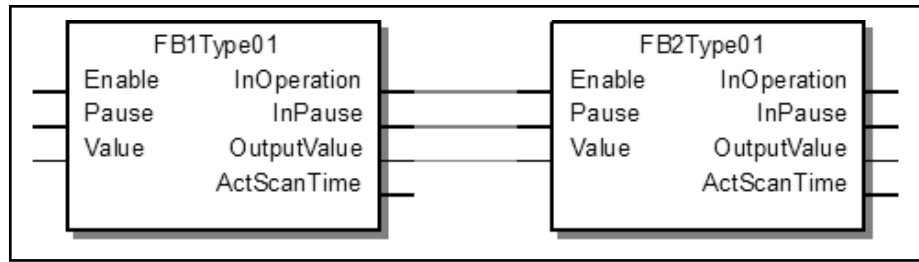


Abb. 7-1: Verkettung von zeitdiskreten Funktionsbausteinen

7.3 Eventgesteuerte Funktionsbausteine

Diese Funktionsbausteine berechnen nur einen neuen Wert, wenn der "Enable"-Eingang aktiv ist und sich der Wert des "Step"-Eingangs ändert. Der zuletzt bearbeitete Schritt steht am Ausgang "ActStep" zur Verfügung. Durch Verbinden des "ActStep"-Ausgangs des vorherigen Funktionsbausteins mit dem "Step"-Eingang des nächsten Bausteins, können mehrere eventgesteuerte Funktionsbausteine kaskadiert werden.

Eine Übernahme von Eingängen findet nur bei einer Änderung am "Step"-Eingang statt. Somit wird sichergestellt, dass sich ein einmal berechneter Wert nicht nachträglich ändert. Eine Parametrierung im laufenden Betrieb ist somit möglich. Die Eingänge "Preset" und "Pause" wirken jedoch unabhängig von einer Änderung am "Step"-Eingang.

Ein aktiver "Pause"-Eingang wird durch Setzen des Ausgangs "InPause" quittiert. Die Ausgangswerte sind dann eingefroren und es finden keine internen Berechnungen mehr statt. "ActStep" wird auch bei aktivem "Pause" weiterhin aktualisiert.

Die Übernahme eines Wertes am "PresetValue"-Eingang auf den Ausgang "ControlValue" oder "OutputValue" erfolgt bei steigender Flanke an "Preset". Nach erfolgter Übernahme wird der Ausgang "PresetAck" aktiv. Ist "Preset" bei steigender Flanke an "Enable" aktiv, wird der Wert in "PresetValue" ebenfalls übernommen. Ein aktives "Preset" hat Priorität über den "Pause"-Eingang.

Eventgesteuerte Funktionsbausteine können direkt miteinander verkettet werden. Der "InOperation"-Ausgang des vorhergehenden Funktionsbausteins wird mit dem "Enable"-Eingang des folgenden Funktionsbausteins verbunden. In gleicher Weise wird "InPause" auf "Pause", "ActStep" auf "Step" und "OutputValue" auf "Value" weitergegeben. Die Reihenfolge der Verkettung sollte bei der Reihenfolge der Funktionsbausteinaufrufe beachtet werden. Dies ist hier aber nicht unbedingt notwendig, da die Einheit von "ActStep" und "OutputValue" immer eine korrekte Abarbeitung sicherstellt.

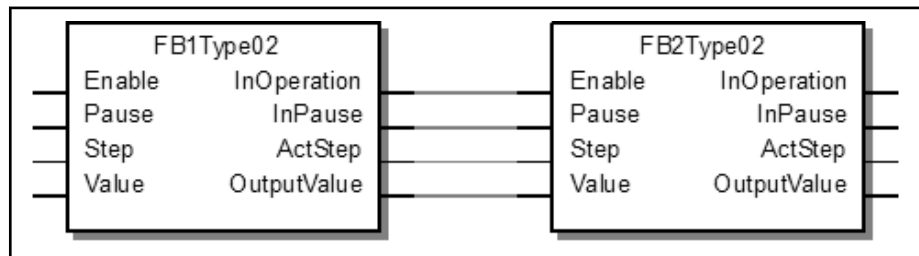


Abb. 7-2: Verkettung von eventgesteuerten Funktionbausteinen

7.4 Regler (Controllers)

7.4.1 Einleitung und Überblick

Die RIL_LoopControl stellt die folgenden Regler zur Verfügung:

Anwendungsgebiete und Funktionen / Funktionsbausteine von RIL_LoopControl

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine	Beschreibung
Regler (Controllers)	
IL_PIDType01, Seite 343	PID-Regler mit Anti-Reset-Wind-up-Integrator und Begrenzer, zeitdiskret
IL_PIDType02, Seite 347	PID-Regler mit Anti-Reset-Wind-up-Integrator und Begrenzer, eventgesteuert
IL_PIDType03, Seite 351	2DOF (2 degree of freedom) PID Regler mit Anti-Reset-Wind-up-Integrator und Begrenzer, zeitdiskret
IL_PIDType04, Seite 355	2DOF (2 degree of freedom) PID Regler mit Anti-Reset-Wind-up-Integrator und Begrenzer, eventgesteuert
IL_PIType01, Seite 360	PI-Regler mit geglättetem Feedback-Eingang, bewerteter Regelabweichung, Anti-Reset-Wind-Up Integrator und Begrenzer am Ausgang, zeitdiskret
IL_PIType02, Seite 364	PI-Regler mit geglättetem Feedback-Eingang, bewerteter Regelabweichung, Anti-Reset-Wind-Up Integrator und Begrenzer am Ausgang, eventgesteuert

Tab. 7-2: RIL_LoopControl, Regler

7.4.2 IL_PIDType01

Kurzbeschreibung

Der PID-Regler beinhaltet einen Anti-Reset-Wind-up-Integrator und einen Begrenzer am Ausgang. Zusätzlich kann ein voreingestellter Wert gesetzt und der Regler angehalten werden. In diesem Fall bleibt der Reglerausgang konstant.

Der PID-Reglerbaustein sollte zyklisch aufgerufen werden. Da er als digitaler Regler implementiert ist, kann dies eine signifikante Auswirkung auf die Regelgenauigkeit haben. Der Funktionsbaustein misst die Zeit zwischen den Aufrufen (Abtastzeit) und berechnet alle Reglerparameter neu, wenn die Abtastzeit sich wesentlich ändert oder wenn die Reglerparameter sich ändern. Die gemessene Abtastzeit steht auch als Ausgang zur Verfügung.

Schnittstellenbeschreibung

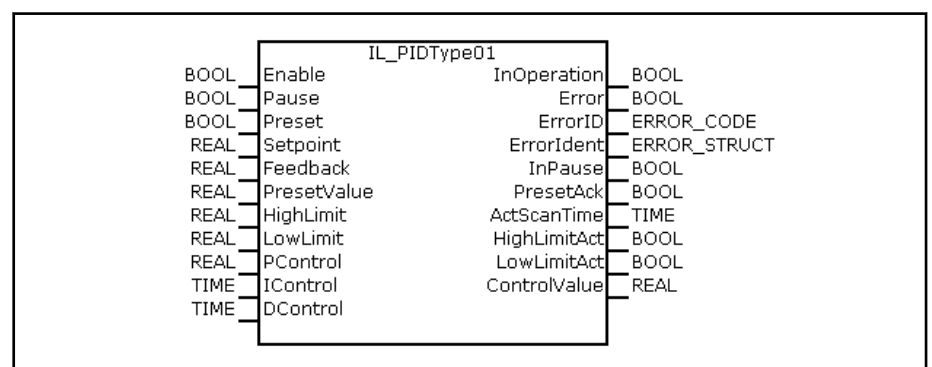


Abb. 7-3: Funktionsbaustein IL_PIDType01

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Wenn TRUE, wird "ControlValue" entweder auf den voreingestellten Wert von "PresetValue" (wenn an "Preset" eine steigende Flanke vorhanden ist), oder auf den berechneten Wert gesetzt
	Pause	BOOL	Wenn TRUE, wird der Wert von "ControlValue" vom Regler nicht verändert
	Preset	BOOL	Bei der steigenden Flanke dieses Eingangs wird der Reglerausgang auf den in "PresetValue" parametrierbaren Wert gesetzt
	Setpoint	REAL	Der gewünschte "Sollwert" für den geregelten Prozess
	Feedback	REAL	Der aktuelle Istwert des Prozesses
	PresetValue	REAL	Voreingestellter Wert, der bei steigender Flanke an "Preset" als "ControlValue" erscheint
	HighLimit	REAL	Die obere Grenze für "ControlValue", wenn "ControlValue" diesen Wert überschreitet, wird er auf diesen Wert begrenzt
	LowLimit	REAL	Die untere Grenze für "ControlValue", wenn "ControlValue" diesen Wert unterschreitet, wird er auf diesen Wert begrenzt
	PControl	REAL	Proportionaler Wert für den Verstärkungsanteil des Reglers.
	IControl	TIME	Integrationszeit "Tn" für den Integrationsanteil des Reglers.
	DControl	TIME	Differenzierzeit "Tv" für den Differenzanteil des Reglers
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, dann arbeitet der PID-Regler und der Reglerausgang "ControlValue" ist gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes , Seite 346.
	InPause	BOOL	"Pause" ist aktiv und der Wert an "ControlValue" ist eingefroren
	PresetAck	BOOL	Der Wert an "PresetValue" wurde übernommen
	ActScan-Time	TIME	Die vergangene Zeit seit dem letzten Aufruf des Funktionsbausteins
	HighLimitAct	BOOL	"ControlValue" hat obere Grenze erreicht
	LowLimitAct	BOOL	"ControlValue" hat untere Grenze erreicht
	ControlValue	REAL	Berechneter Reglerausgangswert

Tab. 7-3: Funktionsbaustein IL_PIDType01

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Preset	BOOL			FALSE	kontinuierlich
Setpoint	REAL	n.def.	n.def.	0.0	kontinuierlich
Feedback	REAL	n.def.	n.def.	0.0	kontinuierlich
PresetValue	REAL	n.def.	n.def.	0.0	steigende Flanke von Preset
HighLimit	REAL	LowLimit	n.def.	0.0	kontinuierlich
LowLimit	REAL	n.def.	HighLimit	0.0	kontinuierlich
PControl	REAL	n.def.	n.def.	0.0	kontinuierlich
IControl	TIME	0s	n.def.	0s	kontinuierlich
DControl	TIME	0s	n.def.	0s	kontinuierlich

Tab. 7-4: Min-, Max- und Default-Werte der Eingänge des IL_PIDType01

PID-Reglerkonfiguration

Das gewünschte P-, I-, PI-, PD- oder PID-Verhalten kann durch Zuweisen der entsprechenden Werte für die "PControl", "IControl" und "DControl" Eingänge erreicht werden. Die Reglerparameter können "on the fly" geändert werden. D. h. bei eingeschaltetem Regler werden die Werte sofort übernommen. Die folgende Tabelle kann benutzt werden, um den gewünschten Regelalgorithmus zu erstellen.

Eine besondere Konfiguration besteht wenn alle drei Werte für "PControl", "IControl" und "DControl" auf 0 gesetzt werden. In diesem Fall ist der Regler abgeschaltet (kein P-, I- und D-Anteil) und der Ausgang "ControlValue" wird fest auf 0 gesetzt. Allerdings bleiben die Funktionalität für "Preset" und "Pause" weiterhin bestehen, d.h. es können mit Hilfe dieser Eingänge auch andere Werte am Ausgang erzwungen werden.

Reglertyp	PControl	IControl	DControl
abgeschaltet	0.0	0s	0s
P-Regler	Gültiger Wert	0s	0s
I-Regler	0.0	Gültiger Wert	0s
PI-Regler	Gültiger Wert	Gültiger Wert	0s
PD-Regler	Gültiger Wert	0s	Gültiger Wert
PID-Regler	Gültiger Wert	Gültiger Wert	Gültiger Wert

Tab. 7-5: PID-Regler-Konfigurationsoptionen

Der Regelalgorithmus kann als folgendes Blockdiagramm dargestellt werden:

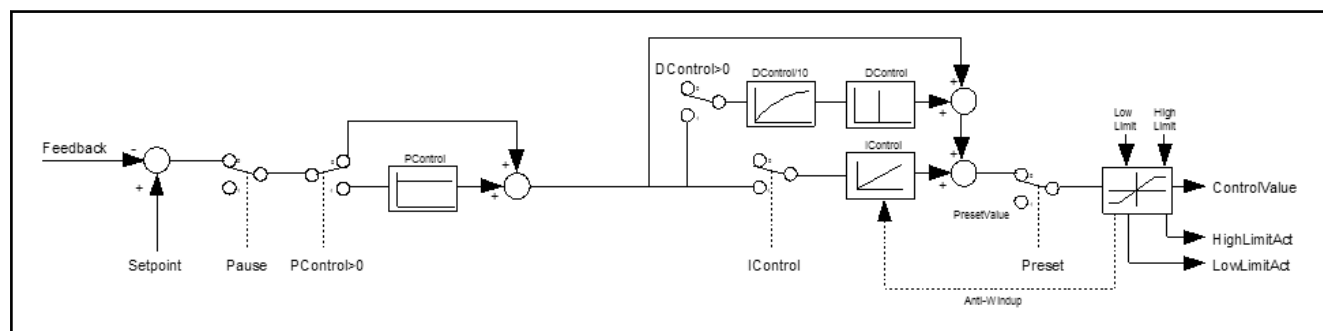


Abb. 7-4: PID-Regler-Blockstruktur

Funktionsbeschreibung Initialisierung

Während der ersten Ausführung des Funktionsbausteins werden die Variablen $x_{d,k}$, $x_{d,k-1}$, $x_{d,k-2}$, y_k und y_{k-1} mit dem Wert "PresetValue" initialisiert. Während der Initialisierung müssen die Eingänge "Setpoint" und "Feedback" mit gültigen Werten belegt sein.

Preset-Wert

Wenn der Anwender einen störungsfreien Start des Reglerausgangs "ControlValue" wünscht, dann muss er den "Preset"-Eingang setzen, wenn "Enable" aktiviert wird. In diesem Fall setzt der PID-Regler die Werte

$$x_{d,k} = x_{d,k-1} = x_{d,k-2} = \text{Setpoint} - \text{Feedback}$$

und

$$y_k = y_{k-1} = \text{PresetValue}$$

Während dieser Zeit müssen die Eingänge "Setpoint" und "Feedback" mit gültigen Werten belegt sein.

Zusätzlich ist die Überprüfung der Ober-/Untergrenzen noch aktiv, um eine Begrenzung des "ControlValue" zu gewährleisten, wenn der vorgegebene Wert ("PresetValue") außerhalb der Grenzen von "HighLimit" oder "LowLimit" ist.

"PresetAck" wird TRUE, sobald der Wert an "PresetValue" übernommen wurde.

"Preset" hat Priorität über den "Pause"-Eingang.

Pause

Der Eingang "Pause" wird nur ausgewertet, solange der "Enable"-Eingang aktiv ist. In diesem Fall wird die Regelabweichung ("Setpoint" - "Feedback") auf 0 gesetzt. Dies bewirkt, dass "ControlValue" fixiert ist. Dies kann dazu benutzt werden, um den Ausgang konstant in einem stabilen Zustand zu halten.

Der Ausgang "InPause" ist gesetzt, solange "Pause" aktiv ist.

Der Ausgang "ActScanTime" wird weiterhin aktualisiert.

Normalbetrieb

Wenn "ControlValue" den Wert von "HighLimit" oder "LowLimit" über-, bzw. unterschreitet, dann wird "ControlValue" auf den jeweiligen Wert begrenzt. Wenn der Regler einen Integrations-Anteil hat, wird zusätzlich ein Anti-Reset-Windup durchgeführt. D.h. der integrale Anteil wird nicht weiter aufsummiert. In diesem Fall werden die Ausgänge "HighLimitAct" und "LowLimitAct" gesetzt.

Die Reglerkoeffizienten werden neu berechnet, wenn sich die Reglerparameter oder die Abtastzeit ändern. Das gibt dem Anwender die Möglichkeit, durch Einstellungen der Reglerparameter eine Feinabstimmung durchzuführen. Die Abtastzeit wird bei jedem Aufruf des Funktionsbausteins gemessen. Weicht die Abtastzeit mehr als 20% von der ersten gemessenen ab, dann wird angenommen, dass sich die Abtastzeit geändert hat, und alle Koeffizienten werden neu berechnet. Der Funktionsbaustein sollte nur in einer zyklischen Task benutzt werden, da die Berechnungen zeitintensiv sind.

Fehlercodes Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR, 16#0001	16#00000C01	16#00000001	Eingangswert "LowLimit" ist größer als Eingangswert "HighLimit"
INPUT_INVALID_ERROR, 16#0001	16#00000C01	16#00000002	Auswahl des Reglertyps wird nicht unterstützt. Eingänge für "PControl", "IControl" und "DControl" überprüfen.
INPUT_RANGE_ERROR, 16#0006	16#00000C02	16#00000002	Der Eingang "DControl" ist ungültig
INPUT_RANGE_ERROR, 16#0006	16#00000C02	16#00000003	Der Eingang "IControl" ist ungültig
INPUT_RANGE_ERROR, 16#0006	16#00000C02	16#00000004	Der Eingang "PControl" ist ungültig
STATE_MACHINE_ERROR, 16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.

Tab. 7-6: Generierte Fehlernummern des Funktionsbausteins IL_PIDType01

7.4.3 IL_PIDType02

Kurzbeschreibung

Der PID-Regler beinhaltet einen Anti-Reset-Wind-up-Integrator und einen Begrenzer am Ausgang. Zusätzlich kann ein voreingestellter Wert gesetzt und der Regler angehalten werden. In diesem Fall bleibt der Reglerausgang konstant.

Der PID-Reglerbaustein sollte zyklisch aufgerufen werden. Eine Berechnung wird nur ausgeführt wenn sich der Wert am Step-Eingang ändert. Die Schrittweite sollte möglichst konstant sein. Der letzte bearbeitete Schritt steht auch als Ausgang zur Verfügung.

Schnittstellenbeschreibung

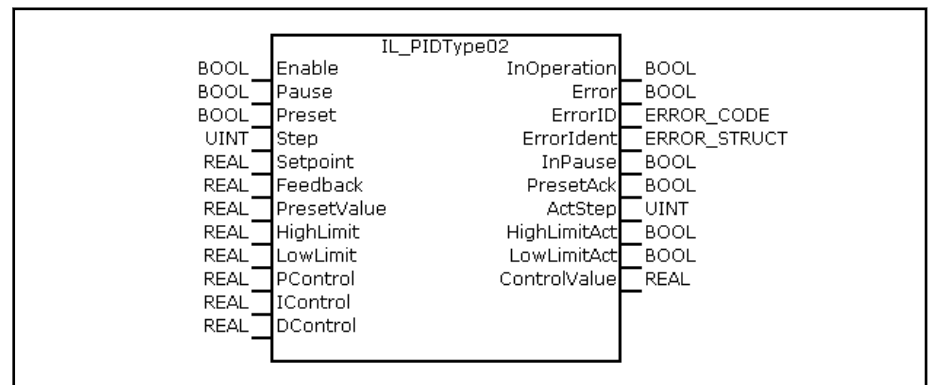


Abb. 7-5: Funktionsbaustein IL_PIDType02

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Wenn TRUE, wird "ControlValue" entweder auf den voreingestellten Wert von "PresetValue" (wenn an "Preset" eine steigende Flanke vorhanden ist), oder auf den berechneten Wert gesetzt.
	Pause	BOOL	Wenn TRUE, wird der Wert von "ControlValue" vom Regler nicht verändert
	Preset	BOOL	Bei der steigenden Flanke dieses Eingangs wird der Reglerausgang auf den in "PresetValue" parametrierbaren Wert gesetzt
	Step	UINT	Verarbeitungsschritt
	Setpoint	REAL	Der gewünschte "Sollwert" für den geregelten Prozess
	Feedback	REAL	Der aktuelle Istwert des Prozesses

	Name	Typ	Kommentar
	PresetValue	REAL	Ein voreingestellter Wert, der als "ControlValue" erscheint, wenn Preset eine steigende Flanke besitzt
	HighLimit	REAL	Die obere Grenze für "ControlValue", wenn "ControlValue" diesen Wert überschreitet, wird er auf diesen Wert begrenzt
	LowLimit	REAL	Die untere Grenze für "ControlValue", wenn "ControlValue" diesen Wert unterschreitet, wird er auf diesen Wert begrenzt
	PControl	REAL	Proportionaler Wert für den Verstärkungsanteil des Reglers
	IControl	REAL	Integrationskonstante T _n für den Integrationsanteil des Reglers
	DControl	REAL	Differenzierkonstante T _v für den Differenzanteil des Reglers
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, dann arbeitet der PID-Regler und der Reglerausgang "ControlValue" ist gültig
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes, Seite 350 .
	InPause	BOOL	"Pause" ist aktiv und der Wert an "ControlValue" ist eingefroren
	PresetAck	BOOL	Der Wert an "PresetValue" wurde übernommen
	ActStep	UINT	Letzter verarbeiteter Schritt
	HighLimitAct	BOOL	"ControlValue" hat obere Grenze erreicht
	LowLimitAct	BOOL	"ControlValue" hat untere Grenze erreicht
	ControlValue	REAL	Berechneter Reglerausgangswert

Tab. 7-7: Funktionsbaustein IL_PIDType02

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Preset	BOOL			FALSE	kontinuierlich
Step	UINT	0	65335	0	kontinuierlich
Setpoint	REAL	n.def.	n.def.	0.0	kontinuierlich
Feedback	REAL	n.def.	n.def.	0.0	kontinuierlich
PresetValue	REAL	n.def.	n.def.	0.0	steigende Flanke von Preset
HighLimit	REAL	LowLimit	n.def.	0.0	kontinuierlich
LowLimit	REAL	n.def.	HighLimit	0.0	kontinuierlich
PControl	REAL	n.def.	n.def.	0.0	kontinuierlich

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
IControl	REAL	0.0	n.def.	0.0	kontinuierlich
DControl	REAL	0.0	n.def.	0.0	kontinuierlich

Tab. 7-8: Min-, Max- und Default-Werte der Eingänge des IL_PIDType02

PID-Reglerkonfiguration

Das gewünschte P-, I-, PI-, PD- oder PID-Verhalten kann durch Zuweisen der entsprechenden Werte für die "PControl", "IControl" und "DControl" Eingänge erreicht werden. Die Reglerparameter können "on the fly" geändert werden. D. h. bei eingeschaltetem Regler werden die Werte sofort übernommen. Die folgende Tabelle kann benutzt werden, um den gewünschten Regelalgorithmus zu erstellen.

Eine besondere Konfiguration besteht wenn alle drei Werte für "PControl", "IControl" und "DControl" auf 0 gesetzt werden. In diesem Fall ist der Regler abgeschaltet (kein P-, I- und D-Anteil) und der Ausgang "ControlValue" wird fest auf 0 gesetzt. Allerdings bleiben die Funktionalität für "Preset" und "Pause" weiterhin bestehen, d.h. es können mit Hilfe dieser Eingänge auch andere Werte am Ausgang erzwungen werden.

Reglertyp	PControl	IControl	DControl
abgeschaltet	0.0	0.0	0.0
P-Regler	Gültiger Wert	0.0	0.0
I-Regler	0.0	Gültiger Wert	0.0
PI-Regler	Gültiger Wert	Gültiger Wert	0.0
PD-Regler	Gültiger Wert	0.0	Gültiger Wert
PID-Regler	Gültiger Wert	Gültiger Wert	Gültiger Wert

Tab. 7-9: PID-Regler-Konfigurationsoptionen

Der Regelalgorithmus kann als folgendes Blockdiagramm dargestellt werden:

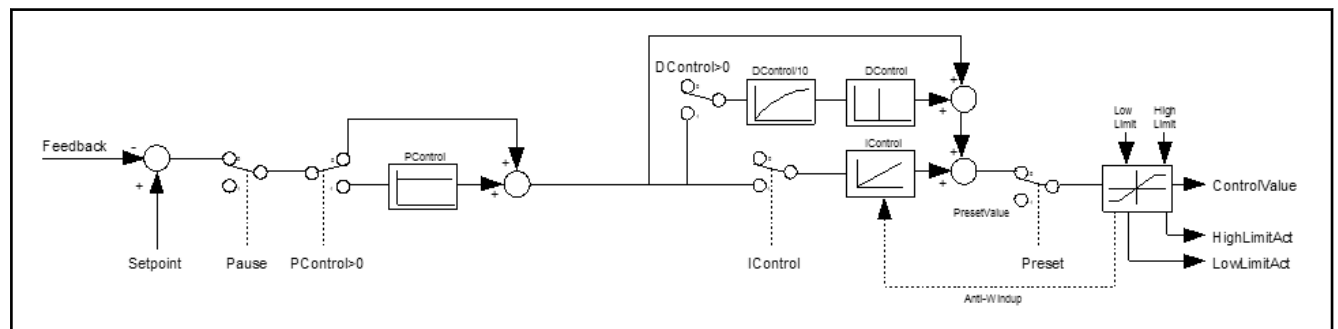


Abb. 7-6: PID-Regler-Blockstruktur

Funktionsbeschreibung**Initialisierung**

Während der ersten Ausführung des Funktionsbausteins werden die Variablen $x_{d,k}$, $x_{d,k-1}$, $x_{d,k-2}$, y_k und y_{k-1} mit dem Wert "PresetValue" initialisiert. Während der Initialisierung müssen die Eingänge "Setpoint" und "Feedback" mit gültigen Werten belegt sein.

Preset-Wert

Wenn der Anwender einen störungsfreien Start des Reglerausgangs ("ControlValue") wünscht, dann muss er den "Preset"-Eingang setzen, wenn "Enable" aktiviert wird. In diesem Fall setzt der PID-Regler die Werte

$$x_{d,k} = x_{d,k-1} = x_{d,k-2} = \text{Setpoint} - \text{Feedback}$$

und

$$y_k = y_{k-1} = \text{PresetValue}$$

Während dieser Zeit müssen die Eingänge "Setpoint" und "Feedback" mit gültige Werten belegt sein.

Zusätzlich ist die Überprüfung der Ober-/Untergrenzen noch aktiv, um eine Begrenzung des "ControlValue" zu gewährleisten, wenn der vorgegebene Wert ("PresetValue") außerhalb der Grenzen von "HighLimit" oder "LowLimit" ist.

"PresetAck" wird TRUE, sobald der Wert an "PresetValue" übernommen wurde.

"Preset" hat Priorität über den "Pause"-Eingang.

Pause

Der Eingang "Pause" wird nur ausgewertet, solange der "Enable"-Eingang aktiv ist. In diesem Fall wird die Regelabweichung ("Setpoint" - "Feedback") auf 0 gesetzt. Dies bewirkt, dass "ControlValue" fixiert ist. Dies kann dazu benutzt werden, um den Ausgang konstant in einem stabilen Zustand zu halten.

Der Ausgang "InPause" ist gesetzt, solange "Pause" aktiv ist.

Der Ausgang "ActStep" wird weiterhin aktualisiert.

Normalbetrieb

Eine Berechnung von "ControlValue" findet nur bei Änderung des "Step"-Eingangs statt.

Wenn "ControlValue" den Wert von "HighLimit" oder "LowLimit" über-, bzw. unterschreitet, dann wird "ControlValue" auf den jeweiligen Wert begrenzt. Wenn der Regler einen Integrations-Anteil hat, wird zusätzlich ein Anti-Reset-Windup durchgeführt. D.h. der integrale Anteil wird nicht weiter aufsummiert. In diesem Fall werden die Ausgänge "HighLimitAct" und "LowLimitAct" gesetzt.

Die Reglerkoeffizienten werden neu berechnet, wenn sich die Reglerparameter oder die Schrittweite ändern. Das gibt dem Anwender die Möglichkeit, durch Einstellungen der Reglerparameter eine Feinabstimmung durchzuführen. Die Schrittweite wird bei jeder Änderung des "Step"-Eingangs gemessen. Weicht die Schrittweite mehr als 20% von der ersten gemessenen ab, werden alle Koeffizienten neu berechnet. Der Funktionsbaustein sollte nur in einer zyklischen Task benutzt werden, da die Berechnungen zeitintensiv sind.

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR, 16#0001	16#00000C01	16#00000001	Eingangswert "LowLimit" ist größer als Eingangswert "HighLimit"
INPUT_INVALID_ERROR, 16#0001	16#00000C01	16#00000002	Auswahl des Reglertyps wird nicht unterstützt. Eingänge für "PControl", "IControl" und "DControl" überprüfen.
INPUT_RANGE_ERROR, 16#0006	16#00000C02	16#00000002	Der Eingang "DControl" ist ungültig
INPUT_RANGE_ERROR, 16#0006	16#00000C02	16#00000003	Der Eingang "IControl" ist ungültig

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR, 16#0006	16#00000C02	16#00000004	Der Eingang "PControl" ist ungültig
STATE_MACHINE_ERROR, 16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.

Tab. 7-10: Generierte Fehlernummern des Funktionsbausteins IL_PIDType02

7.4.4 IL_PIDType03

Kurzbeschreibung IL_PIDType03 ist der 2DOF (2 degree of freedom) PID-Regler.

Die Regeldifferenz wird separat für den proportionalen, integralen und differenziellen Anteil des Reglers gebildet.

Die Sollwerte für den proportionalen und den differenziellen Anteil werden jeweils mit den eingegebenen Gewichtungsfaktoren "bControl" und "cControl" gewichtet. Daher können Stör- und Führungsgrößenverhalten unabhängig voneinander festgelegt werden. Die Werte der "bControl" und "cControl" können von 0 bis 1 gesetzt werden.

Wenn beide (bControl und cControl) auf "1.0" gesetzt werden, verhält sich der Regler gleich wie ein klassischer 1DOF-PID-Regler (gleiche Verhalten wie IL_PIDType01).

Außerdem beinhalten die Regler einen Anti-Reset-Wind-up-Integrator und einen Begrenzer am Ausgang. Zusätzlich kann ein voreingestellter Wert gesetzt und der Regler angehalten werden. In diesem Fall bleibt der Reglerausgang konstant.

Der PID-Reglerbaustein sollte zyklisch aufgerufen werden. Da er als digitaler Regler implementiert ist, kann dies eine signifikante Auswirkung auf die Regelgenauigkeit haben.

Der Funktionsbaustein misst die Zeit zwischen den Aufrufen (Abtastzeit) und berechnet alle Reglerparameter neu, wenn die Abtastzeit sich wesentlich ändert oder wenn die Reglerparameter sich ändern. Die gemessene Abtastzeit steht auch als Ausgang zur Verfügung.

Die Regelkoeffizienten (P-Anteil, I-Anteil und D-Anteil) können jeweils durch die Properties "PCorr", "ICorr", "DCorr" gelesen werden.

Schnittstellenbeschreibung

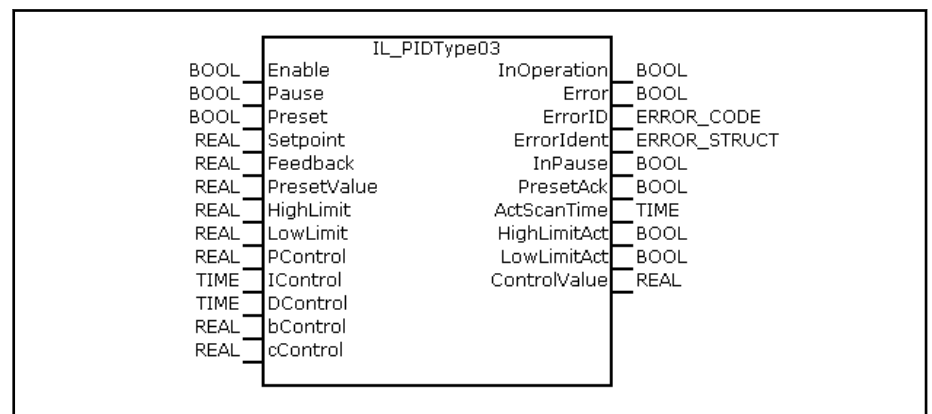


Abb. 7-7: Funktionsbaustein IL_PIDType03

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Wenn TRUE, wird "ControlValue" entweder auf den voreingestellten Wert von "PresetValue" (wenn an "Preset" eine steigende Flanke vorhanden ist), oder auf den berechneten Wert gesetzt
	Pause	BOOL	Wenn TRUE, wird der Wert von "ControlValue" vom Regler nicht verändert
	Preset	BOOL	Bei der steigenden Flanke dieses Eingangs wird der Reglerausgang auf den in "PresetValue" parametrierbaren Wert gesetzt
	Setpoint	REAL	Der gewünschte "Sollwert" für den geregelten Prozess
	Feedback	REAL	Der aktuelle Istwert des Prozesses
	PresetValue	REAL	Voreingestellter Wert, der bei steigender Flanke an "Preset" als "ControlValue" erscheint
	HighLimit	REAL	Die obere Grenze für "ControlValue", wenn "ControlValue" diesen Wert überschreitet, wird er auf diesen Wert begrenzt
	LowLimit	REAL	Die untere Grenze für "ControlValue", wenn "ControlValue" diesen Wert unterschreitet, wird er auf diesen Wert begrenzt
	PControl	REAL	Der proportionale Wert für den Verstärkungsanteil des Reglers
	IControl	TIME	Die Integrationszeit "Tn" für den Integrationsanteil des Reglers
	DControl	TIME	Die Differenzierzeit "Tv" für den Differenzanteil des Reglers
	bControl	REAL	Gewichtungsfaktor für P-Anteil
	cControl	REAL	Gewichtungsfaktor für D-Anteil
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, dann arbeitet der PID-Regler, und der Reglerausgang "ControlValue" ist gültig
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes , Seite 355
	InPause	BOOL	"Pause" ist aktiv und der Wert an "ControlValue" ist eingefroren.
	PresetAck	BOOL	Der Wert an "PresetValue" wurde übernommen.
	ActScan-Time	TIME	Die vergangene Zeit seit dem letzten Aufruf des Funktionsbausteins.
	HighLimitAct	BOOL	"ControlValue" hat obere Grenze erreicht
	LowLimitAct	BOOL	"ControlValue" hat untere Grenze erreicht
	ControlValue	REAL	Der berechnete Reglerausgangswert

Tab. 7-11: Funktionsbaustein IL_PIDType03

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Preset	BOOL			FALSE	kontinuierlich
Setpoint	REAL	n.def.	n.def.	0.0	kontinuierlich
Feedback	REAL	n.def.	n.def.	0.0	kontinuierlich
PresetValue	REAL	n.def.	n.def.	0.0	steigende Flanke von Preset
HighLimit	REAL	LowLimit	n.def.	0.0	kontinuierlich
LowLimit	REAL	n.def.	HighLimit	0.0	kontinuierlich
PControl	REAL	n.def.	n.def.	0.0	kontinuierlich
IControl	TIME	0s	n.def.	0s	kontinuierlich
DControl	TIME	0s	n.def.	0s	kontinuierlich
bControl	REAL	0	1	1	kontinuierlich
cControl	REAL	0	1	1	kontinuierlich

Tab. 7-12: Min-, Max- und Default-Werte der Eingänge des IL_PIDType03

PID-Reglerkonfiguration

Das gewünschte P-, I-, PI-, PD- oder PID-Verhalten kann durch Zuweisen der entsprechenden Werte für die "PControl", "IControl" und "DControl" Eingänge erreicht werden. Die Reglerparameter können "on the fly" geändert werden. D. h. bei eingeschaltetem Regler werden die Werte sofort übernommen. Die folgende Tabelle kann benutzt werden, um den gewünschten Regelalgorithmus zu erstellen.

Eine besondere Konfiguration besteht wenn alle drei Werte für "PControl", "IControl" und "DControl" auf 0 gesetzt werden. In diesem Fall ist der Regler abgeschaltet (kein P-, I- und D-Anteil) und der Ausgang "ControlValue" wird fest auf 0 gesetzt. Allerdings bleiben die Funktionalität für "Preset" und "Pause" weiterhin bestehen, d.h. es können mit Hilfe dieser Eingänge auch andere Werte am Ausgang erzwungen werden.

Reglertyp	PControl	IControl	DControl
abgeschaltet	0.0	0s	0s
P-Regler	Gültiger Wert	0s	0s
I-Regler	0.0	Gültiger Wert	0s
PI-Regler	Gültiger Wert	Gültiger Wert	0s
PD-Regler	Gültiger Wert	0s	Gültiger Wert
PID-Regler	Gültiger Wert	Gültiger Wert	Gültiger Wert

Tab. 7-13: PID-Regler-Konfigurationsoptionen

Der Regelalgorithmus kann als folgendes Blockdiagramm dargestellt werden:

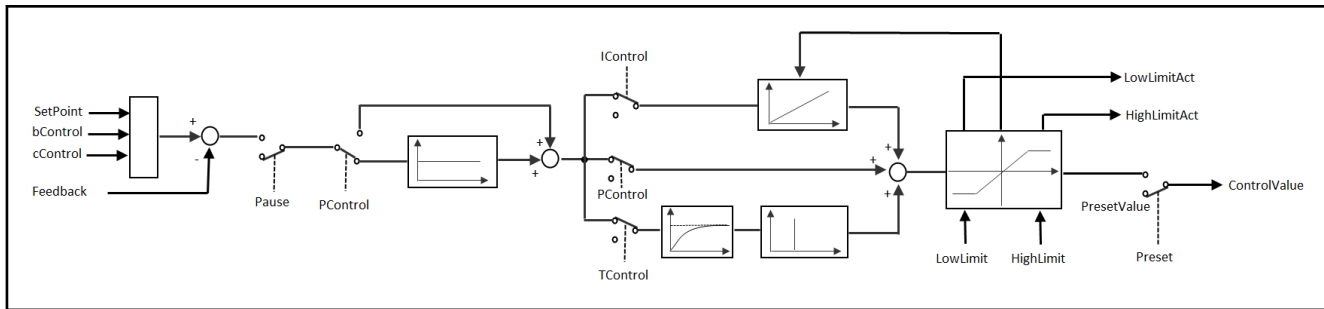


Abb. 7-8: 2DOF-PID-Regler-Blockstruktur

Funktionsbeschreibung**Initialisierung**

Während der Initialisierung müssen die Eingänge "Setpoint" und "Feedback" mit gültigen Werten belegt sein. Falls der Eingang "Preset" "TRUE" gesetzt, wird der Ausgang "ControlValue" mit dem Eingang "PresetValue" initialisiert, sonst wird der Ausgang "ControlValue" mit "0" initialisiert.

Preset-Wert

Wenn der Anwender einen störungsfreien Start des Reglerausgangs "ControlValue" wünscht, dann muss er den "Preset"-Eingang setzen, wenn "Enable" aktiviert wird. In diesem Fall nimmt der „ControlValue“ den „Preset-Value“ über. Während dieser Zeit müssen die Eingänge "Setpoint" und "Feedback" mit gültigen Werten belegt sein.

Zusätzlich ist die Überprüfung der Ober-/Untergrenzen noch aktiv, um eine Begrenzung des "ControlValue" zu gewährleisten, wenn der vorgegebene Wert ("PresetValue") außerhalb der Grenzen von "HighLimit" oder "LowLimit" ist.

"PresetAck" wird TRUE, sobald der Wert an "PresetValue" übernommen wurde.

"Preset" hat Priorität über den "Pause"-Eingang.

Pause

Der Eingang "Pause" wird nur ausgewertet, solange der "Enable"-Eingang aktiv ist. In diesem Fall ist "ControlValue" fixiert. Dies kann dazu benutzt werden, um den Ausgang konstant in einem stabilen Zustand zu halten.

Der Ausgang "InPause" ist gesetzt, solange "Pause" aktiv ist.

Der Ausgang "ActScanTime" wird weiterhin aktualisiert.

Normalbetrieb

Wenn "ControlValue" den Wert von "HighLimit" oder "LowLimit" über-, bzw. unterschreitet, dann wird "ControlValue" auf den jeweiligen Wert begrenzt. In diesem Fall werden die Ausgänge "HighLimitAct" und "LowLimitAct" gesetzt. Wenn der Regler einen Integrations-Anteil hat, wird zusätzlich ein Anti-Reset-Windup durchgeführt. D.h. der integrale Anteil wird nicht weiter aufsummiert.

Die Reglerkoeffizienten werden neu berechnet, wenn sich die Reglerparameter oder die Abtastzeit ändern. Das gibt dem Anwender die Möglichkeit, durch Einstellungen der Reglerparameter eine Feinabstimmung durchzuführen. Die Abtastzeit wird bei jedem Aufruf des Funktionsbausteins gemessen. Weicht die Abtastzeit mehr als 20% von der ersten gemessenen ab, dann wird angenommen, dass sich die Abtastzeit geändert hat, und alle Koeffizienten werden neu berechnet. Der Funktionsbaustein sollte nur in einer zyklischen Task benutzt werden, da die Berechnungen zeitintensiv sind.

Beispiel für die Auswahl des "bControl"

Folgende Bild zeigt die Sprungantwort eines PI-Reglers mit unterschiedlichem "bControl".

Die Führungsverhalten des PI-Reglers kann durch "bControl" geändert werden:

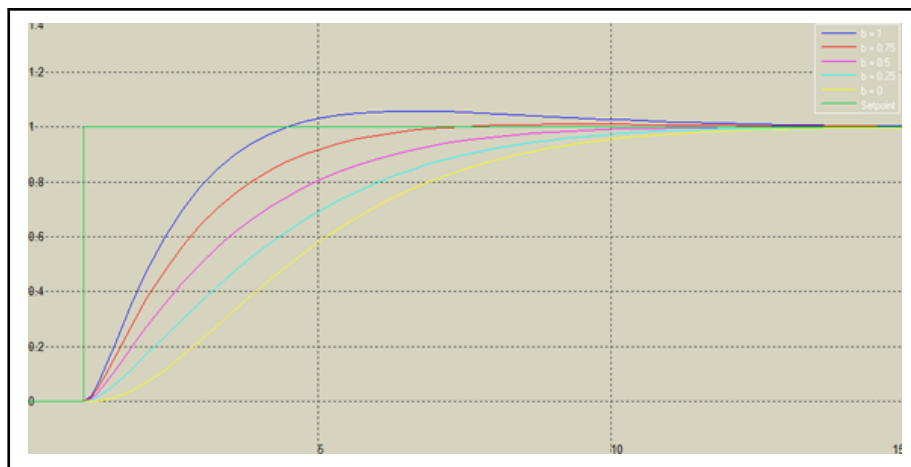


Abb. 7-9: Sprungantwort eines PI-Reglers mit unterschiedlichem bControl

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR, 16#0001	16#0C01	16#0001	Eingangswert "LowLimit" ist größer als Eingangswert "HighLimit"
INPUT_INVALID_ERROR, 16#0001	16#0C01	16#0002	Auswahl des Reglertyps wird nicht unterstützt. Eingänge für "PControl", "IControl" und "DControl" überprüfen.
INPUT_RANGE_ERROR, 16#0006	16#0C02	16#0002	Der Eingang "DControl" ist ungültig
INPUT_RANGE_ERROR, 16#0006	16#0C02	16#0003	Der Eingang "IControl" ist ungültig
INPUT_RANGE_ERROR, 16#0006	16#0C02	16#0004	Der Eingang "PControl" ist ungültig
STATE_MACHINE_ERROR, 16#0005	16#0006	16#0000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.
INPUT_INVALID_ERROR, 16#0001	16#0C12	16#0000	Der Eingang "bControl" ist ungültig
INPUT_INVALID_ERROR, 16#0001	16#0C13	16#0000	Der Eingang "cControl" ist ungültig

Tab. 7-14: Generierte Fehlernummern des Funktionsbausteins IL_PIDType03

7.4.5 IL_PIDType04

Kurzbeschreibung

IL_PIDType04 ist der 2DOF (2 degree of freedom)-PID-Regler.

Die Regeldifferenz wird separat für den proportionalen, integralen und differenziellen Anteil des Reglers gebildet.

Die Sollwerte beim proportionalen und differenziellen Anteil werden jeweilig von den Gewichtungsfaktoren "bControl" und "cControl" gewichtet. Daher können Stör- und Führungsgrößenverhalten unabhängig voneinander festgelegt werden. Die Werte der "bControl" und "cControl" können von 0 bis 1 gesetzt werden. Wenn beide (bControl und cControl) auf "1.0" gesetzt werden, verhält sich der Regler gleich wie ein klassischer 1DOF-PID-Regler (gleiches Verhalten wie IL_PIDType02).

Außerdem beinhalten die Regler einen Anti-Reset-Wind-up-Integrator und einen Begrenzer am Ausgang. Zusätzlich kann ein voreingestellter Wert gesetzt und der Regler angehalten werden. In diesem Fall bleibt der Reglerausgang konstant.

IL_PIDType04 ist schrittabhängiger Regler. Der PID-Reglerbaustein muss zyklisch aufgerufen werden. Eine Berechnung wird nur ausgeführt, wenn sich der Wert am Step-Eingang ändert. Die Schrittweite sollte möglichst konstant sein. Der letzte bearbeitete Schritt steht auch als Ausgang zur Verfügung.

Die Regelkoeffizienten (P-Anteil, I-Anteil und D-Anteil) können jeweils durch die Properties "PCorr", "ICorr", "DCorr" gelesen werden.

Schnittstellenbeschreibung

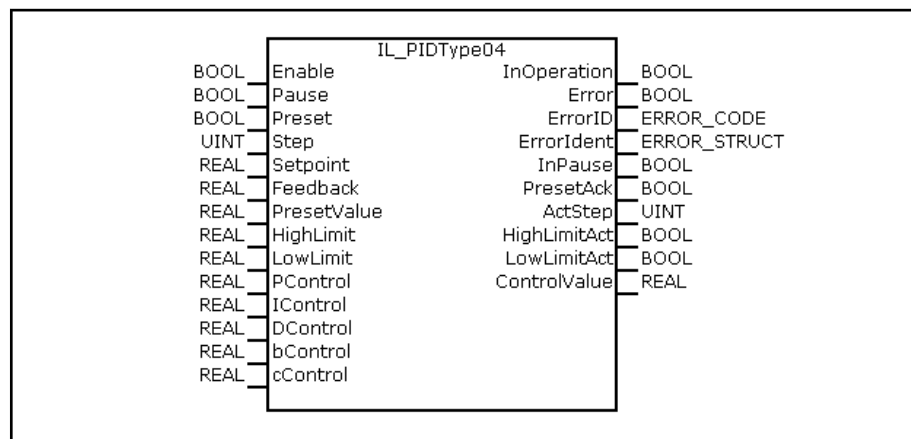


Abb. 7-10: Funktionsbaustein IL_PIDType04

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Wenn TRUE, wird "ControlValue" entweder auf den voreingestellten Wert von "PresetValue" (wenn an "Preset" eine steigende Flanke vorhanden ist), oder auf den berechneten Wert gesetzt.
	Pause	BOOL	Wenn TRUE, wird der Wert von "ControlValue" vom Regler nicht verändert.
	Preset	BOOL	Bei der steigenden Flanke dieses Eingangs wird der Reglerausgang auf den in "PresetValue" parametrierbaren Wert gesetzt.
	Step	UINT	Verarbeitungsschritt
	Setpoint	REAL	Der gewünschte "Sollwert" für den geregelten Prozess.
	Feedback	REAL	Der aktuelle Istwert des Prozesses.
	PresetValue	REAL	Ein voreingestellter Wert, der als "ControlValue" erscheint, wenn Preset eine steigende Flanke besitzt.
	HighLimit	REAL	Die obere Grenze für "ControlValue". Wenn "ControlValue" diesen Wert überschreitet, wird er auf diesen Wert begrenzt.
	LowLimit	REAL	Die untere Grenze für "ControlValue". Wenn "ControlValue" diesen Wert unterschreitet, wird er auf diesen Wert begrenzt.
	PControl	REAL	Der proportionale Wert für den Verstärkungsanteil des Reglers.
	IControl	REAL	Die Integrationskonstante "Tn" für den Integrationsanteil des Reglers.
	DControl	REAL	Die Differenzierkonstante "Tv" für den Differenzanteil des Reglers.

	Name	Typ	Kommentar
	bControl	REAL	Gewichtungsfaktor für P Anteil.
	cControl	REAL	Gewichtungsfaktor für D Anteil.
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, dann arbeitet der PID-Regler, und der Reglerausgang "ControlValue" ist gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes , Seite 359
	InPause	BOOL	"Pause" ist aktiv und der Wert an "ControlValue" ist eingefroren.
	PresetAck	BOOL	Der Wert an "PresetValue" wurde übernommen.
	ActStep	UINT	Letzter verarbeiteter Schritt.
	HighLimitAct	BOOL	"ControlValue" hat obere Grenze erreicht
	LowLimitAct	BOOL	"ControlValue" hat untere Grenze erreicht
	ControlValue	REAL	Der berechnete Reglerausgangswert

Tab. 7-15: Funktionsbaustein IL_PIDType02

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Preset	BOOL			FALSE	kontinuierlich
Step	UINT	0	65335	0	kontinuierlich
Setpoint	REAL	n.def.	n.def.	0.0	kontinuierlich
Feedback	REAL	n.def.	n.def.	0.0	kontinuierlich
PresetValue	REAL	n.def.	n.def.	0.0	steigende Flanke von Preset
HighLimit	REAL	LowLimit	n.def.	0.0	kontinuierlich
LowLimit	REAL	n.def.	HighLimit	0.0	kontinuierlich
PControl	REAL	n.def.	n.def.	0.0	kontinuierlich
IControl	REAL	0.0	n.def.	0.0	kontinuierlich
DControl	REAL	0.0	n.def.	0.0	kontinuierlich
bControl	REAL	0.0	1.0	1.0	kontinuierlich
cControl	REAL	0.0	1.0	1.0	kontinuierlich

Tab. 7-16: Min-, Max- und Default-Werte der Eingänge des IL_PIDType04

PID-Reglerkonfiguration

Das gewünschte P-, I-, PI-, PD- oder PID-Verhalten kann durch Zuweisen der entsprechenden Werte für die "PControl", "IControl" und "DControl" Eingänge erreicht werden. Die Reglerparameter können "on the fly" geändert werden. D. h. bei eingeschaltetem Regler werden die Werte sofort übernommen. Die

folgende Tabelle kann benutzt werden, um den gewünschten Regelalgorithmus zu erstellen.

Eine besondere Konfiguration entsteht, wenn alle drei Werte für "PControl", "IControl" und "DControl" auf "0" gesetzt werden. In diesem Fall ist der Regler abgeschaltet (kein P-, I- und D-Anteil) und der Ausgang "ControlValue" wird fest auf 0 gesetzt. Allerdings bleiben die Funktionalität für "Preset" und "Pause" weiterhin bestehen, d.h. es können mit Hilfe dieser Eingänge auch andere Werte am Ausgang erzwungen werden.

Reglertyp	PControl	IControl	DControl
abgeschaltet	0.0	0.0	0.0
P-Regler	Gültiger Wert	0.0	0.0
I-Regler	0.0	Gültiger Wert	0.0
PI-Regler	Gültiger Wert	Gültiger Wert	0.0
PD-Regler	Gültiger Wert	0.0	Gültiger Wert
PID-Regler	Gültiger Wert	Gültiger Wert	Gültiger Wert

Tab. 7-17: PID-Regler-Konfigurationsoptionen

Der Regelalgorithmus kann als folgendes Blockdiagramm dargestellt werden:

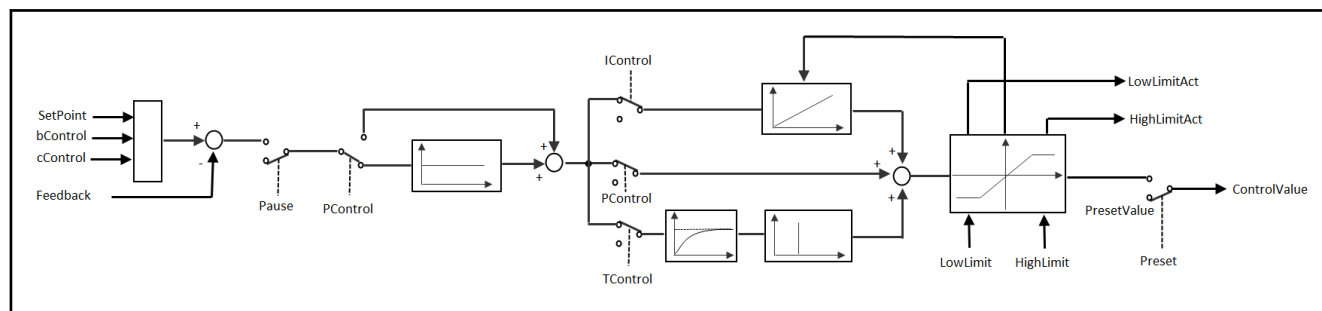


Abb. 7-11: 2DOF-PID-Regler-Blockstruktur

Funktionsbeschreibung

Initialisierung

Während der Initialisierung müssen die Eingänge "Setpoint" und "Feedback" mit gültigen Werten belegt sein. Falls der Eingang "Preset" "TRUE" gesetzt, wird der Ausgang "ControlValue" mit dem Eingang "PresetValue" initialisiert, sonst wird der Ausgang "ControlValue" mit "0" initialisiert.

Preset-Wert

Wenn der Anwender einen störungsfreien Start des Reglerausgangs "ControlValue" wünscht, dann muss er den "Preset"-Eingang setzen, wenn "Enable" aktiviert wird. In diesem Fall nimmt der „ControlValue“ den „Preset-Value“ über. Während dieser Zeit müssen die Eingänge "Setpoint" und "Feedback" mit gültigen Werten belegt sein. Zusätzlich ist die Überprüfung der Ober-/Untergrenzen noch aktiv, um eine Begrenzung des "ControlValue" zu gewährleisten, wenn der vorgegebene Wert ("PresetValue") außerhalb der Grenzen von "HighLimit" oder "LowLimit" ist.

"PresetAck" wird TRUE, sobald der Wert an "PresetValue" übernommen wurde.

"Preset" hat Priorität über den "Pause"-Eingang.

Pause

Der Eingang "Pause" wird nur ausgewertet, solange der "Enable"-Eingang aktiv ist. In diesem Fall ist "ControlValue" fixiert. Dies kann dazu benutzt wer-

den, um den Ausgang konstant in einem stabilen Zustand zu halten. In Pause werden die Regelkoeffizienten intern noch aktualisiert.

Der Ausgang "InPause" ist gesetzt, solange "Pause" aktiv ist.

Der Ausgang "ActStep" wird weiterhin aktualisiert.

Normalbetrieb

Eine Berechnung von "ControlValue" findet nur bei Änderung des "Step"-Eingangs statt. Wenn "ControlValue" den Wert von "HighLimit" oder "LowLimit" über-, bzw. unterschreitet, dann wird "ControlValue" auf den jeweiligen Wert begrenzt. Wenn der Regler einen Integrations-Anteil hat, wird zusätzlich ein Anti-Reset-Windup durchgeführt. D.h. der integrale Anteil wird nicht weiter aufsummiert. In diesem Fall werden die Ausgänge "HighLimitAct" und "LowLimitAct" gesetzt.

Die Reglerkoeffizienten werden neu berechnet, wenn sich die Reglerparameter oder die Schrittweite ändern. Das gibt dem Anwender die Möglichkeit, durch Einstellungen der Reglerparameter eine Feinabstimmung durchzuführen. Die Schrittweite wird bei jeder Änderung des "Step"-Eingangs gemessen.

Beispiel für die Auswahl des "bControl"

Folgende Bild zeigt die Sprungantwort eines PI-Reglers mit unterschiedlichem "bControl".

Die Führungsverhalten des PI-Reglers kann durch "bControl" geändert werden:

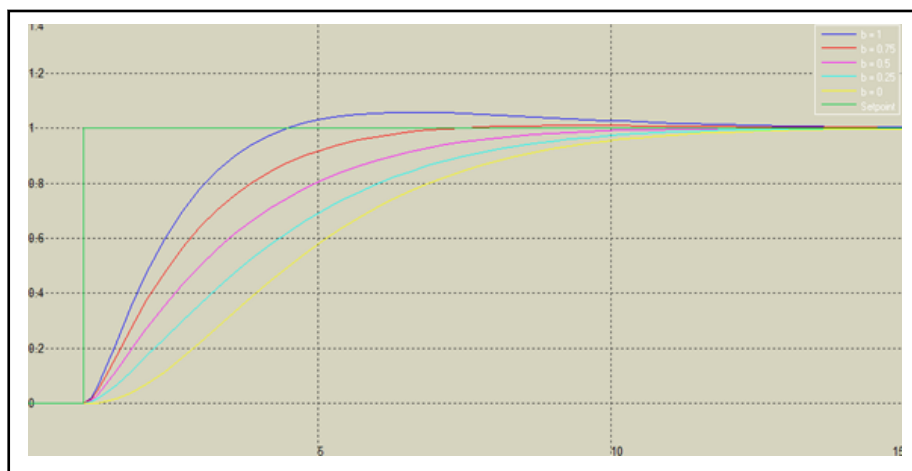


Abb. 7-12: Sprungantwort eines PI-Reglers mit unterschiedlichem "bControl"

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR, 16#0001	16#00000C01	16#00000001	Eingangswert "LowLimit" ist größer als Eingangswert "HighLimit"
INPUT_INVALID_ERROR, 16#0001	16#00000C01	16#00000002	Auswahl des Reglertyps wird nicht unterstützt. Eingänge für "PControl", "IControl" und "DControl" überprüfen.
INPUT_RANGE_ERROR, 16#0006	16#00000C02	16#00000002	Der Eingang "DControl" ist ungültig
INPUT_RANGE_ERROR, 16#0006	16#00000C02	16#00000003	Der Eingang "IControl" ist ungültig
INPUT_RANGE_ERROR, 16#0006	16#00000C02	16#00000004	Der Eingang "PControl" ist ungültig

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR, 16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.
INPUT_INVALID_ERROR, 16#0001	16#00000C12	16#00000000	Der Eingang "bControl" ist ungültig
INPUT_INVALID_ERROR, 16#0001	16#00000C13	16#00000000	Der Eingang "cControl" ist ungültig

Tab. 7-18: Generierte Fehlernummern des Funktionsbausteins IL_PIDType04

7.4.6 IL_PIDType01

Kurzbeschreibung

Der PI-Regler beinhaltet einen Anti-Reset-Wind-up-Integrator und einen Begrenzer am Ausgang. Der Istwert-Eingang wird durch ein einstellbares PT1-Glied gefiltert. Die Differenz aus Sollwert und Istwert wird durch eine Sollwertkurve bewertet. Zusätzlich kann ein voreingestellter Wert gesetzt und der Regler angehalten werden. In diesem Fall bleibt der Reglerausgang konstant.

Der PI-Reglerbaustein sollte zyklisch aufgerufen werden. Da er als digitaler Regler implementiert ist, kann dies eine signifikante Auswirkung auf die Regelgenauigkeit haben. Der Funktionsbaustein misst die Zeit zwischen den Aufrufen (Abtastzeit) und berechnet alle Reglerparameter neu, wenn die Abtastzeit sich wesentlich ändert oder wenn die Reglerparameter sich ändern. Die gemessene Abtastzeit steht als Ausgang zur Verfügung.

Schnittstellenbeschreibung

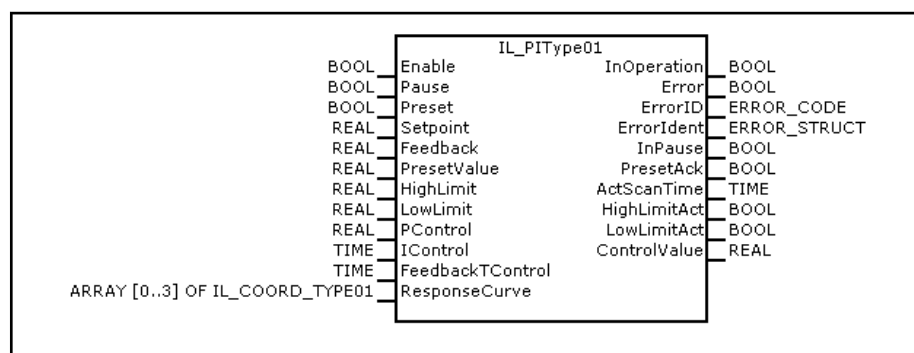


Abb. 7-13: Funktionsbaustein IL_PIDType01

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Wenn TRUE, wird "ControlValue" entweder auf den voreingestellten Wert von "PresetValue" (wenn an "Preset" eine steigende Flanke gesetzt ist), oder auf den berechneten Wert gesetzt.
	Pause	BOOL	Wenn TRUE, wird der Wert von "ControlValue" vom Regler nicht verändert.
	Preset	BOOL	Bei der steigenden Flanke dieses Eingangs wird der Reglerausgang auf den in "PresetValue" parametrierbaren Wert gesetzt.
	Setpoint	REAL	Der gewünschte Sollwert oder Führungsgröße für den geregelten Prozess.
	Feedback	REAL	Der aktuelle Istwert des Prozesses.
	PresetValue	REAL	Voreingestellter Wert, der bei steigender Flanke an "Preset" als "ControlValue" erscheint.

	Name	Typ	Kommentar
	HighLimit	REAL	Die obere Grenze für "ControlValue". Wenn "ControlValue" diesen Wert überschreitet, wird er auf diesen Wert begrenzt.
	LowLimit	REAL	Die untere Grenze für "ControlValue". Wenn "ControlValue" diesen Wert unterschreitet, wird er auf diesen Wert begrenzt.
	PControl	REAL	Proportionaler Wert für den Verstärkungsanteil des Reglers.
	IControl	TIME	Integrationszeit "Tn" für den Integrationsanteil des Reglers.
	FeedbackT-Control	TIME	Filterzeitkonstante des PT1-Glieds, das den Istwert-Eingang "Feedback" glättet.
	Response-Curve	ARRAY[0..3] OF IL_COORD_TYPE01	Sollwertkurve für die Bewertung der Differenz aus geglättetem Istwert "Feedback" und Sollwert "Setpoint". IL_COORD_TYPE01, Seite 511
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, dann arbeitet der PI-Regler und der Reglerausgang "ControlValue" ist gültig
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes, Seite 363 .
	InPause	BOOL	"Pause" ist aktiv und der Wert an "ControlValue" ist eingefroren.
	PresetAck	BOOL	Wert an "PresetValue" wurde übernommen
	ActScan-Time	TIME	Die vergangene Zeit seit dem letzten Aufruf des Funktionsbausteins
	HighLimitAct	BOOL	"ControlValue" hat obere Grenze erreicht
	LowLimitAct	BOOL	"ControlValue" hat untere Grenze erreicht
	ControlValue	REAL	Der berechnete Reglerausgangswert

Tab. 7-19: Funktionsbaustein IL_PType01

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Preset	BOOL			FALSE	kontinuierlich
Setpoint	REAL	n.def.	n.def.	0.0	kontinuierlich
Feedback	REAL	n.def.	n.def.	0.0	kontinuierlich
PresetValue	REAL	n.def.	n.def.	0.0	steigende Flanke an Preset
HighLimit	REAL	LowLimit	n.def.	0.0	kontinuierlich
LowLimit	REAL	n.def.	HighLimit	0.0	kontinuierlich

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
PControl	REAL	n.def.	n.def.	0.0	kontinuierlich
IControl	TIME	0s	n.def.	0s	kontinuierlich
FeedbackTControl	TIME	0s	n.def.	0s	kontinuierlich
ResponseCurve	ARRAY[0..3] OF IL_COORD_TY-PE01			0	steigende Flanke an Enable

Tab. 7-20: Min-, Max- und Default-Werte der Eingänge des IL_PType01

Reglerkonfiguration

Das Signal am "Feedback"-Eingang wird über ein Verzögerungsglied erster Ordnung geglättet. Die dazugehörige Filterzeitkonstante wird über "FeedbackTControl" eingestellt. Ist "FeedbackTControl" 0s, wird der Filter deaktiviert. Das Differenzsignal aus gefiltertem "Feedback"-Eingang und "Setpoint" wird über die in "ResponseCurve" angegebene Kennlinie gewichtet. Die Kennlinie ist punktsymmetrisch zum Ursprung. Die X-Werte der Kennlinie entsprechen dem Differenzsignal, die Y-Werte dem gewichteten Differenzsignal. Es können bis zu vier Punkte angegeben werden. Die Teilstücke zwischen den Punkten werden linear interpoliert. Wird "ResponseCurve" nicht angelegt, so entfällt die Wichtung.

Das gewünschte P-, I-, PI-Verhalten kann durch Zuweisen der entsprechenden Werte für die "PControl" und "IControl" Eingänge erreicht werden. Die Reglerparameter können "on the fly" geändert werden. D.h. bei eingeschaltetem Regler werden die Werte sofort übernommen. Die folgende Tabelle kann benutzt werden, um den gewünschten Regelalgorithmus zu erstellen. Dabei entspricht ein ungültiger Wert der Zahl 0, alle anderen Werte sind gültig.

Eine besondere Konfiguration besteht, wenn alle drei Werte für "PControl" und "IControl" auf 0 bzw. 0s gesetzt werden. In diesem Fall ist der Regler abgeschaltet (kein P- und I-Anteil) und der Ausgang "ControlValue" wird fest auf 0 gesetzt. Allerdings bleiben die Funktionalität für "Preset" und "Pause" weiterhin bestehen, d.h. es können mit Hilfe dieser Eingänge auch andere Werte am Ausgang erzwungen werden.

Reglertyp	PControl	IControl
abgeschaltet	0.0	0s
P-Regler	Gültiger Wert	0s
I-Regler	0.0	Gültiger Wert
PI-Regler	Gültiger Wert	Gültiger Wert

Tab. 7-21: PI-Regler-Konfigurationsoptionen

Der Regelalgorithmus kann als folgendes Blockdiagramm dargestellt werden:

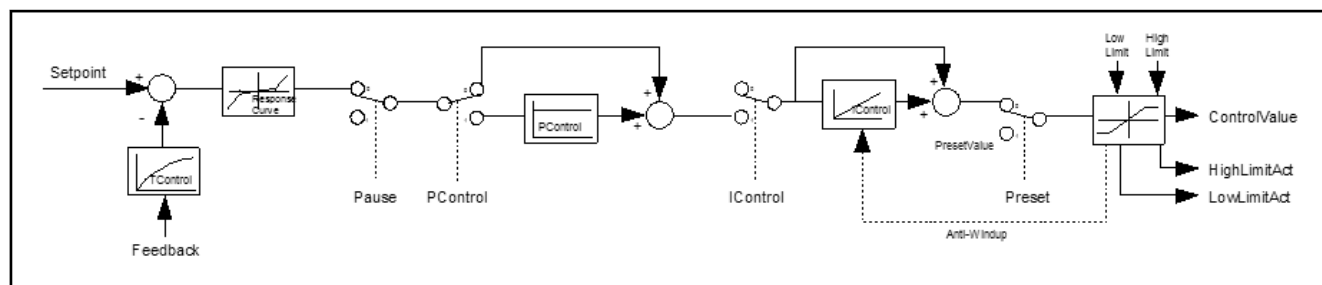


Abb. 7-14: PI-Regler Blockstruktur

Funktionsbeschreibung**Initialisierung**

Während der ersten Ausführung des Funktionsbausteins werden die Variablen $x_{d,k}$, $x_{d,k-1}$, $x_{d,k-2}$, y_k und y_{k-1} mit dem Wert "PresetValue" initialisiert. Während der Initialisierung müssen die Eingänge "Setpoint" und "Feedback" gültige Daten haben.

Bei steigender Flanke an „Enable“ wird das Verzögerungsglied mit dem an "Feedback" anliegenden Wert initialisiert, d.h. "Feedback" erscheint sofort am Ausgang des Verzögerungsgliedes.

Preset-Wert

Wenn der Anwender einen störungsfreien Start des Reglerausgangs "ControlValue" wünscht, dann muss er den "Preset"-Eingang setzen, wenn "Enable" aktiviert wird. In diesem Fall werden die internen Integratoren auf die Differenz von "Setpoint" und durch "ResponseCurve" gewichtetem "Feedback" initialisiert. Das Ergebnis erscheint sofort an "ControlValue".

Während dieser Zeit müssen die Eingänge "Setpoint" und "Feedback" gültige Daten haben.

Zusätzlich ist die Überprüfung der Ober-/Untergrenzen noch aktiv, um eine Begrenzung des "ControlValue" zu gewährleisten, wenn der vorgegebene Wert "PresetValue" außerhalb der Grenzen von "HighLimit" oder "LowLimit" ist.

"PresetAck" wird TRUE, sobald der Wert an "PresetValue" übernommen wurde.

"Preset" hat Priorität über den "Pause"-Eingang.

Pause

Der Eingang "Pause" wird nur ausgewertet, solange der "Enable"-Eingang aktiv ist. In diesem Fall wird die Regelabweichung ("Setpoint" - gewichtetes "Feedback") auf 0 gesetzt. Dies bewirkt, dass "ControlValue" fixiert ist. Dies kann dazu benutzt werden, um den Ausgang konstant in einem stabilen Zustand zu halten.

Der Ausgang "InPause" ist gesetzt, solange "Pause" aktiv ist.

Der Ausgang "ActScanTime" wird weiterhin aktualisiert.

Normalbetrieb

Wenn "ControlValue" den Wert von "HighLimit" oder "LowLimit" über-, bzw. unterschreitet, dann wird "ControlValue" auf den jeweiligen Wert begrenzt. Wenn der Regler einen Integrations-Anteil hat, wird zusätzlich ein Anti-Reset-Windup durchgeführt. D.h. der integrale Anteil wird nicht weiter aufsummiert. In diesem Fall werden die Ausgänge "HighLimitAct" und "LowLimitAct" gesetzt.

Die Reglerkoeffizienten werden neu berechnet, wenn sich die Reglerparameter oder die Abtastzeit ändern. Das gibt dem Anwender die Möglichkeit, durch Einstellungen der Reglerparameter eine Feinabstimmung durchzuführen. Die Abtastzeit wird bei jedem Aufruf des Funktionsbausteins gemessen. Weicht die Abtastzeit mehr als 20% von der ersten gemessenen ab, dann wird angenommen, dass sich die Abtastzeit geändert hat, und alle Koeffizienten werden neu berechnet. Der Funktionsbaustein sollte nur in einer zyklischen Task benutzt werden, da die Berechnungen zeitintensiv sind.

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR, 16#0001	16#00000C01	16#00000001	Eingangswert "LowLimit" ist größer als Eingangswert "HighLimit"
INPUT_INVALID_ERROR, 16#0001	16#00000C01	16#00000002	Auswahl des Reglertyps wird nicht unterstützt. Eingänge für "PControl" und "IControl" überprüfen.
INPUT_RANGE_ERROR, 16#0006	16#00000C02	16#00000001	Der Eingang "FeedbackTControl" ist ungültig.
INPUT_RANGE_ERROR, 16#0006	16#00000C02	16#00000003	Der Eingang "IControl" ist ungültig
INPUT_RANGE_ERROR, 16#0006	16#00000C02	16#00000004	Der Eingang "PControl" ist ungültig
INPUT_INVALID_ERROR, 16#0001	16#00000C05	16#00000001	X-Werte in "ResponseCurve" nicht aufsteigend
INPUT_INVALID_ERROR, 16#0001	16#00000C05	16#00000002	Mehr als einen Werte für eine X-Position in "ResponseCurve"
INPUT_INVALID_ERROR, 16#0001	16#00000C05	16#00000003	Negativer X-Wert in "ResponseCurve"
STATE_MACHINE_ERROR, 16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.

Tab. 7-22: Generierte Fehlernummern des Funktionsbausteins IL_PIType01

7.4.7 IL_PIType02

Kurzbeschreibung

Der PI-Regler beinhaltet einen Anti-Reset-Wind-up-Integrator und einen Begrenzer am Ausgang. Der Istwert-Eingang wird durch ein einstellbares PT1-Glied gefiltert. Die Differenz aus Sollwert und Istwert wird durch eine Sollwertkurve bewertet. Zusätzlich kann ein voreingestellter Wert gesetzt und der Regler angehalten werden. In diesem Fall bleibt der Reglerausgang konstant.

Eine Berechnung findet nur statt, wenn sich der "Step"-Eingang ändert. Der zuletzt bearbeitete Schritt steht als Ausgang zur Verfügung.

Schnittstellenbeschreibung

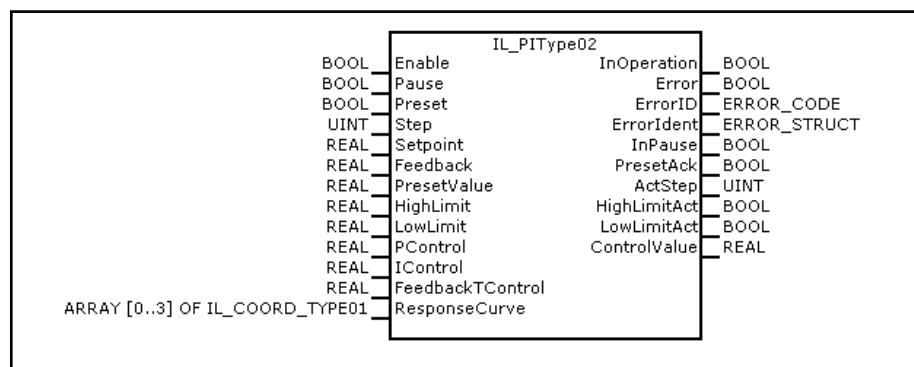


Abb. 7-15: Funktionsbaustein IL_PIType02

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Wenn TRUE, wird "ControlValue" entweder auf den voreingestellten Wert von "PresetValue" (wenn an "Preset" eine steigende Flanke gesetzt ist), oder auf den berechneten Wert gesetzt.
	Pause	BOOL	Wenn TRUE, wird der Wert von "ControlValue" vom Regler nicht verändert.
	Preset	BOOL	Bei der steigenden Flanke dieses Eingangs wird der Reglerausgang auf den in "PresetValue" parametrierbaren Wert gesetzt.

	Name	Typ	Kommentar
	Setpoint	REAL	Der gewünschte Sollwert oder Führungsgröße für den geregelten Prozess.
	Feedback	REAL	Der aktuelle Istwert des Prozesses.
	PresetValue	REAL	Voreingestellter Wert, der bei steigender Flanke an "Preset" als "ControlValue" erscheint.
	Step	UINT	Taktschritt des Reglers. Eine Berechnung von "ControlValue" erfolgt nur, wenn sich "Step" ändert.
	HighLimit	REAL	Die obere Grenze für "ControlValue", wenn "ControlValue" diesen Wert überschreitet, wird er auf diesen Wert begrenzt
	LowLimit	REAL	Die untere Grenze für "ControlValue", wenn "ControlValue" diesen Wert unterschreitet, wird er auf diesen Wert begrenzt
	PControl	REAL	Proportionaler Wert für den Verstärkungsanteil des Reglers
	IControl	TIME	Integrationszeit T _n für den Integrationsanteil des Reglers
	FeedbackT-Control	TIME	Filterzeitkonstante des PT1-Glieds, das den Istwert-Eingang "Feedback" glättet.
	Response-Curve	ARRAY[0..3] OF IL_COORD_TY-PE01	Sollwertkurve für die Bewertung der Differenz aus geglättetem Istwert "Feedback" und Sollwert "Setpoint". IL_COORD_TYPE01, Seite 511
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, dann arbeitet der PI-Regler und der Reglerausgang "ControlValue" ist gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes, Seite 368 .
	InPause	BOOL	"Pause" ist aktiv und der Wert an "ControlValue" ist eingefroren.
	PresetAck	BOOL	Der Wert an "PresetValue" wurde übernommen.
	ActStep	UINT	Step Wert, auf den sich "OutputValue" bezieht.
	HighLimitAct	BOOL	"ControlValue" hat obere Grenze erreicht
	LowLimitAct	BOOL	"ControlValue" hat untere Grenze erreicht
	ControlValue	REAL	Der berechnete Reglerausgangswert

Tab. 7-23: Funktionsbaustein IL_PIType02

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Preset	BOOL			FALSE	kontinuierlich

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Step	UINT	0	65335	0	kontinuierlich
Setpoint	REAL	n.def.	n.def.	0.0	kontinuierlich
Feedback	REAL	n.def.	n.def.	0.0	kontinuierlich
PresetValue	REAL	n.def.	n.def.	0.0	steigende Flanke an Preset
HighLimit	REAL	LowLimit	n.def.	0.0	kontinuierlich
LowLimit	REAL	n.def.	HighLimit	0.0	kontinuierlich
PControl	REAL	n.def.	n.def.	0.0	kontinuierlich
IControl	REAL	0.0	n.def.	0.0	kontinuierlich
FeedbackTControl	REAL	0.0	n.def.	0.0	kontinuierlich
ResponseCurve	ARRAY[0..3] OF IL_COORD_TY-PE01			0	steigende Flanke an Enable

Tab. 7-24: Min-, Max- und Default-Werte der Eingänge des IL_PIType02

Reglerkonfiguration

Das Signal am "Feedback"-Eingang wird über ein Verzögerungsglied erster Ordnung geglättet. Die dazugehörige Filterzeitkonstante wird über "FeedbackTControl" eingestellt. Ist "FeedbackTControl" 0s, wird der Filter deaktiviert. Das Differenzsignal aus gefiltertem "Feedback"-Eingang und "Setpoint" wird über die in "ResponseCurve" angegebene Kennlinie gewichtet. Die Kennlinie ist punktsymmetrisch zum Ursprung. Die X-Werte der Kennlinie entsprechen dem Differenzsignal, die Y-Werte dem gewichteten Differenzsignal. Es können bis zu vier Punkte angegeben werden. Die Teilstücke zwischen den Punkten werden linear interpoliert. Wird "ResponseCurve" nicht angelegt, so entfällt die Wichtung.

Das gewünschte P-, I-, PI-Verhalten kann durch Zuweisen der entsprechenden Werte für die "PControl" und "IControl" Eingänge erreicht werden. Die Reglerparameter können "on the fly" geändert werden. D.h. bei eingeschaltetem Regler werden die Werte sofort übernommen. Die folgende Tabelle kann benutzt werden, um den gewünschten Regelalgorithmus zu erstellen.

Eine besondere Konfiguration besteht, wenn alle drei Werte für "PControl" und "IControl" auf 0.0 gesetzt werden. In diesem Fall ist der Regler abgeschaltet (kein P- und I-Anteil) und der Ausgang "ControlValue" wird fest auf 0.0 gesetzt. Allerdings bleiben die Funktionalität für "Preset" und "Pause" weiterhin bestehen, d.h. es können mit Hilfe dieser Eingänge auch andere Werte am Ausgang erzwungen werden.

Reglertyp	PControl	IControl
abgeschaltet	0.0	0.0
P-Regler	Gültiger Wert	0.0
I-Regler	0.0	Gültiger Wert
PI-Regler	Gültiger Wert	Gültiger Wert

Tab. 7-25: PI-Regler-Konfigurationsoptionen

Der Regelalgorithmus kann als folgendes Blockdiagramm dargestellt werden:

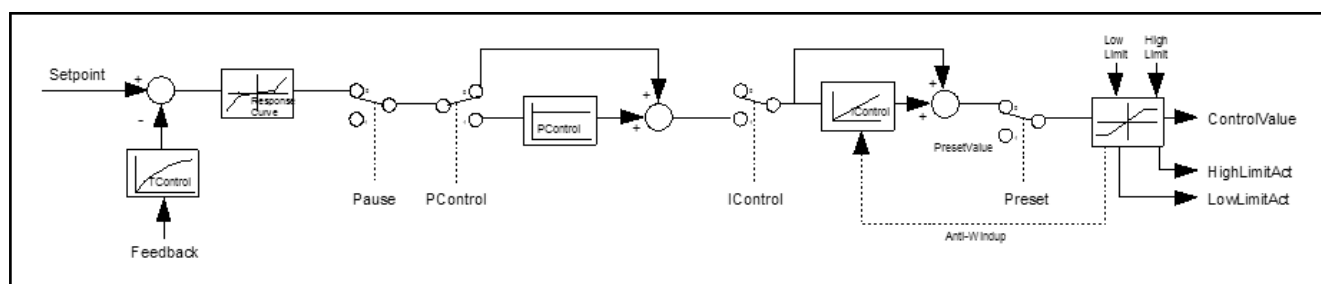


Abb. 7-16: PI-Regler Blockstruktur

Funktionsbeschreibung**Initialisierung**

Während der ersten Ausführung des Funktionsbausteins werden die Variablen $x_{d,k}$, $x_{d,k-1}$, $x_{d,k-2}$, y_k und y_{k-1} mit dem Wert "PresetValue" initialisiert. Während der Initialisierung müssen die Eingänge "Setpoint" und "Feedback" gültige Daten haben.

Bei steigender Flanke an „Enable“ wird das Verzögerungsglied mit dem an "Feedback" anliegenden Wert initialisiert, d.h. "Feedback" erscheint sofort am Ausgang des Verzögerungsgliedes.

Der Wert des "Step"-Eingangs wird auf "ActStep" kopiert.

Preset-Wert

Wenn der Anwender einen störungsfreien Start des Reglerausgangs "ControlValue" wünscht, dann muss er den "Preset"-Eingang setzen, wenn "Enable" aktiviert wird. In diesem Fall werden die internen Integratoren auf die Differenz von "Setpoint" und durch "ResponseCurve" gewichtetem "Feedback" initialisiert. Das Ergebnis erscheint sofort an "ControlValue".

Während dieser Zeit müssen die Eingänge "Setpoint" und "Feedback" gültige Daten haben.

Zusätzlich ist die Überprüfung der Ober-/Untergrenzen noch aktiv, um eine Begrenzung des "ControlValue" zu gewährleisten, wenn der vorgegebene Wert "PresetValue" außerhalb der Grenzen von "HighLimit" oder "LowLimit" ist.

"PresetAck" wird TRUE, sobald der Wert an "PresetValue" übernommen wurde.

"Preset" hat Priorität über den "Pause"-Eingang.

Pause

Der Eingang "Pause" wird nur ausgewertet, solange der "Enable"-Eingang aktiv ist. In diesem Fall wird die Regelabweichung ("Setpoint" - gewichtetes "Feedback") auf 0 gesetzt. Dies bewirkt, dass "ControlValue" fixiert ist. Dies kann dazu benutzt werden, um den Ausgang konstant in einem stabilen Zustand zu halten.

Der Ausgang "InPause" ist gesetzt, solange "Pause" aktiv ist.

Der Ausgang "ActStep" wird weiterhin aktualisiert.

Normalbetrieb

Der Regler rechnet nur, wenn sich der Wert am "Step"-Eingang ändert. Die Schrittdifferenz zwischen zwei Funktionsbausteinaufrufen wird beachtet. "IControl" und "FeedbackTControl" haben Schritte als Zeitbasis.

Wenn "ControlValue" den Wert von "HighLimit" oder "LowLimit" über-, bzw. unterschreitet, dann wird "ControlValue" auf einen dieser Werte begrenzt. Wenn der Regler einen Integrations-Anteil hat, wird zusätzlich ein Anti-Reset-Windup durchgeführt. D.h. der integrale Anteil wird nicht weiter aufsummiert.

In diesem Fall werden die Ausgänge "HighLimitAct" und "LowLimitAct" gesetzt.

Die Reglerkoeffizienten werden neu berechnet, wenn sich die Reglerparameter oder die Schrittdifferenz ändern. Das gibt dem Anwender die Möglichkeit, durch Einstellungen der Reglerparameter eine Feinabstimmung durchzuführen. Die Schrittdifferenz wird bei jedem Aufruf des Funktionsbausteins gemessen. Weicht die Schrittdifferenz mehr als 20% von der ersten gemessenen ab und ist nicht 0, werden alle Koeffizienten neu berechnet.

Fehlercodes Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR, 16#0001	16#00000C01	16#00000001	Eingangswert "LowLimit" ist größer als Eingangswert "HighLimit"
INPUT_INVALID_ERROR, 16#0001	16#00000C01	16#00000002	Auswahl des Reglertyps wird nicht unterstützt. Eingänge für "PControl" und "IControl" überprüfen.
INPUT_RANGE_ERROR, 16#0006	16#00000C02	16#00000001	Der Eingang "FeedbackTControl" ist ungültig.
INPUT_RANGE_ERROR, 16#0006	16#00000C02	16#00000003	Der Eingang "IControl" ist ungültig
INPUT_RANGE_ERROR, 16#0006	16#00000C02	16#00000004	Der Eingang "PControl" ist ungültig
INPUT_INVALID_ERROR, 16#0001	16#00000C05	16#00000001	X-Werte in "ResponseCurve" nicht aufsteigend
INPUT_INVALID_ERROR, 16#0001	16#00000C05	16#00000002	Mehr als einen Werte für eine X-Position in "ResponseCurve"
INPUT_INVALID_ERROR, 16#0001	16#00000C05	16#00000003	Negativer X-Wert in "ResponseCurve"
STATE_MACHINE_ERROR, 16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.

Tab. 7-26: Generierte Fehlernummern des Funktionsbausteins IL_PIType02

7.5 Grundelemente (Elements)

7.5.1 Einführung und Übersicht

Die RIL_LoopControl stellt die folgenden regelungstechnische Grundelemente zur Verfügung:

zeitdiskret	eventgesteuert	Beschreibung
IL_DeadtimeType01, Seite 369	IL_DeadtimeType02, Seite 372	Totzeitglied (zeitvariabel)
IL_DT1Type01, Seite 377	IL_DT1Type02, Seite 379	Differenzglied mit Verzögerung erster Ordnung
IL_PDT1Type01, Seite 381	IL_PDT1Type02, Seite 385	Digitales Filter zum Anheben bzw. Absenken der Übertragungsfunktion ab einer bestimmten Frequenz
IL_PT1Type01, Seite 388	IL_PT1Type02, Seite 390	Verzögerungsglied erster Ordnung
IL_TransportDelayType01, Seite 393	IL_TransportDelayType02, Seite 396	Totzeitglied (laufzeitvariabel)
IL_DeviationResponseCurve, Seite 375		Bewertungskurve

zeitdiskret	eventgesteuert	Beschreibung
IL_ValueLimit, Seite 399		Wertebegrenzung
IL_ValueWindow, Seite 401		Wertebereichsprüfung

Tab. 7-27: Filter der RIL_LoopControl

7.5.2 IL_DeadtimeType01

Kurzbeschreibung Dieser Funktionsbaustein implementiert ein zeitdiskretes Totzeitglied mit variabler Totzeit. Die Totzeit kann kontinuierlich verändert werden. Die tatsächliche Totzeit wird fließend an den vorgegebenen Wert angepasst. Die Anpassung an die neue Totzeit erfolgt nach dem "variable time delay"-Schema.

Der Totzeitbaustein sollte zyklisch aufgerufen werden. Da er als digitales Element implementiert ist, kann dies eine signifikante Auswirkung auf die Genauigkeit des Ausgangs haben. Der Funktionsbaustein misst die Zeit zwischen den Aufrufen (Abtastzeit) und führt die resultierende Totzeit fließend nach, wenn die Abtastzeit oder die eingestellte Totzeit sich ändert. Die gemessene Abtastzeit steht auch als Ausgang zur Verfügung.

Schnittstellenbeschreibung

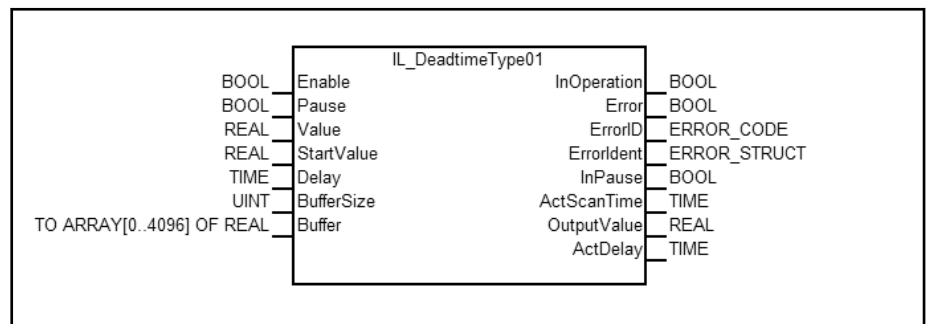


Abb. 7-17: Funktionsbaustein IL_DeadtimeType01

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Bei steigender Flanke wird "OutputValue" auf den voreingestellten Wert von "StartValue" gesetzt und der Baustein aktiviert.
	Pause	BOOL	Wenn TRUE, wird der Wert von "OutputValue" nicht verändert.
	Value	REAL	Eingangswert, der um "Delay" verzögert wieder ausgegeben werden soll.
	StartValue	REAL	Ein voreingestellter Wert der bei steigender Flanke an "Enable" an "OutputValue" erscheint.
	Delay	TIME	Eingestellte Totzeit
	BufferSize	UINT	Größe des in "Buffer" angegebenen Puffers in Elementen
	Buffer	POINTER TO REAL	Puffer zur Zwischenspeicherung der Werte
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, dann arbeitet der Funktionsbaustein und die Ausgänge "OutputValue" und "ActDelay" sind gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes, Seite 371 .

	Name	Typ	Kommentar
	InPause	BOOL	"Pause" ist aktiv und die Wert an "OutputValue" und "ActDelay" sind eingefroren.
	ActScan-Time	TIME	Die vergangene Zeit seit der Funktionsbaustein das letzte Mal aufgerufen wurde.
	OutputValue	REAL	Der um "ActDelay" verzögerte Wert
	ActDelay	TIME	Derzeitige Verzögerung

Tab. 7-28: Funktionsbaustein IL_DeadtimeType01

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Value	REAL	n.def.	n.def.	0.0	kontinuierlich
StartValue	REAL	n.def.	n.def.	0.0	steigende Flanke von Enable
Delay	TIME	0s	n.def.	0s	kontinuierlich
BufferSize	UINT	0	4096	0	steigende Flanke von Enable
Buffer	POINTER TO REAL			0	steigende Flanke von Enable

Tab. 7-29: Min-, Max- und Default-Werte der Eingänge des IL_DeadtimeType01

Funktionsbeschreibung

Initialisierung

Bei steigender Flanke an "Enable" wird nach Prüfung von "BufferSize" und "Buffer" der Puffer mit dem Wert von "StartValue" gefüllt. "ActDelay" wird mit 0s initialisiert. "StartValue" erscheint sofort am Ausgang "OutputValue" und bleibt konstant, bis "ActDelay" gleich "Delay" ist. Erst danach erscheinen die ersten Werte von "Value" am Ausgang "OutputValue".

Normalbetrieb

Bei aktivem "Enable"-Eingang arbeitet der Funktionsbaustein. Bei jedem Aufruf wird die Zeit zum vergangenen Aufruf gemessen und am Ausgang "ActScanTime" ausgegeben. "ActScanTime" wird außerdem für die Berechnung der Pufferverwaltung verwendet.

Der Funktionsbaustein speichert bei jedem Aufruf den Wert an "Value" und gibt diesen um "ActDelay" verzögert wieder an "OutputValue" aus. Sollte die Größe "BufferSize" des Puffers "Buffer" nicht ausreichen, um alle Werte für die in "Delay" angegebene Zeit zwischenspeichern, so werden Werte ausgelassen. Auf die resultierende Totzeit hat dies keinen Einfluss.

Der Ausgang "ActDelay" gibt die derzeitige Verzögerung des Wertes an "OutputValue" an. Bei steigender Flanke an Enable oder bei Änderung an "Delay" wird "ActDelay" fließend auf den Wert in "Delay" nachgeführt. Wird „Delay“ geändert, während der Baustein aktiv ist, so wird die Totzeit fließend nach dem Schema der variablen Zeitverzögerung angepasst. Das Anpassen der Totzeit dauert maximal "BufferSize" * "ActScanTime".

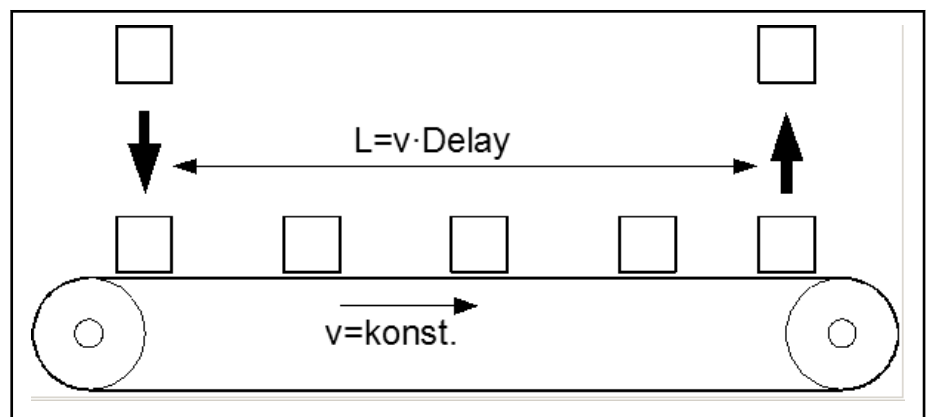


Abb. 7-18: Zur Veranschaulichung der Totzeitanpassung läuft ein Transportband mit konstanter Geschwindigkeit. Die Pakete werden an einem festen Punkt auf das Band gelegt und nach dem Verstreichen der Totzeit vom Band genommen. Ändert sich die Totzeit, wandert der Punkt, an dem die Pakete vom Band genommen werden.

Pause

Der Eingang "Pause" wird nur ausgewertet, wenn der "Enable"-Eingang aktiv ist. Der Funktionsbaustein kann über "Pause" angehalten werden. In diesem Fall werden keine weiteren Werte gespeichert und "OutputValue" und "Act-Delay" bleiben konstant.

Der Ausgang "InPause" ist gesetzt, solange "Pause" aktiv ist.

Der Ausgang "ActScanTime" wird weiterhin aktualisiert.

Konfiguration

Die Größe des in "Buffer" angelegten Puffers beeinflusst die Genauigkeit der Ausgangskurve. Es ist möglich, an "Buffer" einen kleineren Puffer als 4096 Werte anzulegen. Im Idealfall ist der Puffer größer, als die maximale benötigte Verzögerung geteilt durch die Aufrufzykluszeit des Funktionsbausteins. "BufferSize" kann auch kleiner sein, als die Verzögerung. In diesem Fall steht nicht bei jedem Zyklus ein neuer Wert am Ausgang "OutputValue" an. Die zeitliche Auflösung sinkt.

Notwendige Puffergröße:

"BufferSize" = $1.4 \cdot \max(\text{"Delay"}) / \text{benötigte Auflösung}$

Hinreichende Puffergröße:

"BufferSize" = $1.4 \cdot \max(\text{"Delay"}) / \text{"ActScanTime"}$

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR, 16#0001	16#00000C03	16#00000001	Der Eingang "Delay" ist ungültig
INPUT_RANGE_ERROR, 16#0006	16#00000C03	16#00000002	Die in "Delay" angegebene Verzögerung ist zu groß
INPUT_INVALID_ERROR, 16#0001	16#00000C03	16#00000003	Der Eingang "Buffer" ist nicht belegt
INPUT_RANGE_ERROR, 16#0006	16#00000C03	16#00000004	"BufferSize" ist zu klein

Tab. 7-30: Generierte Fehlernummern des Funktionsbausteins IL_Deadtime-Type01

7.5.3 IL_DeadtimeType02

Kurzbeschreibung Dieser Funktionsbaustein implementiert ein eventgesteuertes Totzeitglied mit variabler Totzeit. Die Totzeit kann kontinuierlich verändert werden. Die tatsächlichen Totzeitschritte wird fließend an den vorgegebenen Wert angepasst. Die Anpassung an die neue Totzeit erfolgt nach dem "variable time delay"-Schema.

Die Schrittweite des "Step"-Eingangs sollte möglichst konstant sein. Der letzte bearbeitete Schritt steht auch als Ausgang zur Verfügung.

Schnittstellenbeschreibung

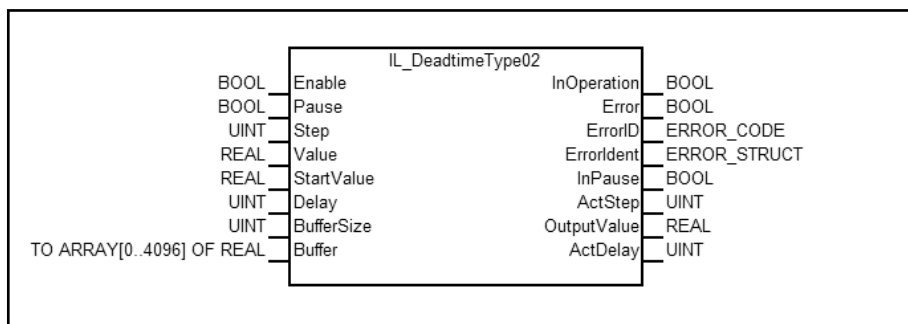


Abb. 7-19: Funktionsbaustein IL_DeadtimeType02

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Bei steigender Flanke wird "OutputValue" auf den voreingestellten Wert von "StartValue" gesetzt und der Baustein aktiviert.
	Pause	BOOL	Wenn TRUE, wird der Wert von "OutputValue" nicht verändert.
	Step	UINT	Verarbeitungsschritt
	Value	REAL	Eingangswert, der um "Delay" verzögert wieder ausgegeben werden soll.
	StartValue	REAL	Ein voreingestellter Wert der bei steigender Flanke an "Enable" an "OutputValue" erscheint.
	Delay	UINT	Anzahl der zu verzögernden Schritte
	BufferSize	UINT	Größe des in "Buffer" angegebenen Puffers in Elementen
	Buffer	POINTER TO REAL	Puffer zur Zwischenspeicherung der Werte
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, dann arbeitet der Funktionsbaustein und die Ausgänge "OutputValue" und "ActDelay" sind gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes, Seite 374 .
	InPause	BOOL	"Pause" ist aktiv und die Wert an "OutputValue" und "ActDelay" sind eingefroren.
	ActStep	UINT	Letzter bearbeiteter Schritt.
	OutputValue	REAL	Der um "ActDelay" verzögerte Wert
	ActDelay	UINT	Tatsächliche Verzögerung in Schritten

Tab. 7-31: Funktionsbaustein IL_DeadtimeType02

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Step	UINT	0	65335	0	kontinuierlich
Value	REAL	n.def.	n.def.	0.0	kontinuierlich
StartValue	REAL	n.def.	n.def.	0.0	steigende Flanke von Enable
Delay	UINT	0	65335	0	kontinuierlich
BufferSize	UINT	0	4096	0	steigende Flanke von Enable
Buffer	POINTER TO REAL			0	steigende Flanke von Enable

Tab. 7-32: Min-, Max- und Default-Werte der Eingänge des IL_DeadtimeType02

Funktionsbeschreibung

Initialisierung

Bei steigender Flanke an "Enable" wird nach Prüfung von "BufferSize" und "Buffer" der Puffer mit dem Wert von "StartValue" gefüllt. "ActDelay" wird mit 0 initialisiert. "StartValue" erscheint sofort am Ausgang "OutputValue" und bleibt konstant, bis "ActDelay" gleich "Delay" ist. Erst danach erscheinen die ersten Werte von "Value" am Ausgang "OutputValue".

Normalbetrieb

Bei aktivem "Enable"-Eingang arbeitet der Funktionsbaustein. Bei jedem Aufruf wird "ActStep" auf "Step" aktualisiert. Eine Bearbeitung findet nur statt, wenn sich der "Step"-Eingang ändert. Die Differenz zwischen den Schritten wird außerdem für die Pufferverwaltung verwendet.

Der Funktionsbaustein speichert bei jedem Aufruf den Wert an "Value" und gibt diesen um "ActDelay" Schritte verzögert wieder an "OutputValue" aus. Sollte die Größe "BufferSize" des Puffers "Buffer" nicht ausreichen, um alle Werte für die in "Delay" angegebene Schrittzahl zwischenspeichern, so werden Werte ausgelassen. Auf die resultierende Totzeit hat dies keinen Einfluss.

Der Ausgang "ActDelay" gibt die derzeitige Verzögerung des Wertes an "OutputValue" an. Bei steigender Flanke an Enable oder bei Änderung an "Delay" wird die "ActDelay" fließend auf den Wert in "Delay" nachgeführt. Wird „Delay“ geändert, während der Baustein aktiv ist, so wird die Totzeit fließend nach dem Schema der variablen Zeitverzögerung angepasst. Das Anpassen der Totzeit dauert maximal "BufferSize" Schritte.

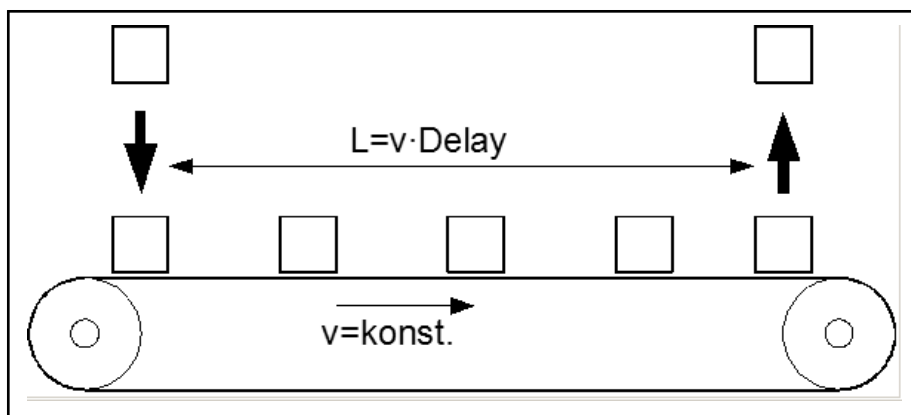


Abb. 7-20: Zur Veranschaulichung der Totzeitanpassung läuft ein Transportband mit konstanter Geschwindigkeit. Die Pakete werden an einem festen Punkt auf das Band gelegt und nach dem Verstreichen der Totzeit vom Band genommen. Ändert sich die Totzeit, wandert der Punkt, an dem die Pakete vom Band genommen werden.

Pause

Der Eingang "Pause" wird nur ausgewertet, wenn der "Enable"-Eingang aktiv ist. Der Funktionsbaustein kann über "Pause" angehalten werden. In diesem Fall werden keine weiteren Werte gespeichert und "OutputValue" und "Act-Delay" bleiben konstant.

Der Ausgang "InPause" ist gesetzt, solange "Pause" aktiv ist.

Der Ausgang "ActStep" wird weiterhin aktualisiert

Konfiguration

Die Größe des in "Buffer" angelegten Puffers beeinflusst die Genauigkeit der Ausgangskurve. Es ist möglich, an "Buffer" einen kleineren Puffer als 4096 Werte anzulegen. Im Idealfall ist der Puffer größer, als die maximale benötigte Verzögerung in Schritten. "BufferSize" kann auch kleiner sein, als die Verzögerung. In diesem Fall steht nicht bei jedem Zyklus ein neuer Wert am Ausgang "OutputValue" an. Die Auflösung, bezogen auf die Schritte, sinkt.

Notwendige Puffergröße:

"BufferSize" = 1.4 * max("Delay") / benötigte Auflösung

Hinreichende Puffergröße:

"BufferSize" = 1.4 * max("Delay")

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR, 16#0001	16#00000C03	16#00000001	Der Eingang "Delay" ist ungültig
INPUT_RANGE_ERROR, 16#0006	16#00000C03	16#00000002	Die in "Delay" angegebene Verzögerung ist zu groß
INPUT_INVALID_ERROR, 16#0001	16#00000C03	16#00000003	Der Eingang "Buffer" ist nicht belegt
INPUT_RANGE_ERROR, 16#0006	16#00000C03	16#00000004	"BufferSize" ist zu klein oder zu groß

Tab. 7-33: Generierte Fehlernummern des Funktionsbausteins IL_Deadtime-Type02

7.5.4 IL_DeviationResponseCurve

Kurzbeschreibung Dieser Funktionsbaustein setzt ein Eingangssignal punktsymmetrisch zum Ursprung auf eine Sollwertkurve um. Die Sollwertkurve wird durch bis zu 4 Punkte vorgegeben. Zwischen den Punkten wird linear interpoliert. Die erste Strecke geht vom Ursprung bis zum ersten Punkt. Vom letzten Punkt an bis unendlich gilt eine Steigung von 1.0. Die X-Koordinate der Punkte muss in positiver Richtung ansteigen und darf nicht kleiner 0 sein.

Schnittstellenbeschreibung

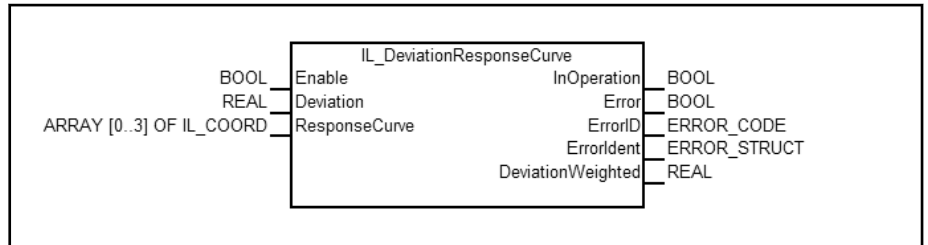


Abb. 7-21: Funktionsbaustein IL_DeviationResponseCurve

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Wenn TRUE, ist der Baustein aktiv.
	Deviation	REAL	Eingangswert, der durch die in "ResponseCurve" angegebene Kurve bewertet werden soll.
	Response-Curve	ARRAY[0..3] OF IL_COORD_TY-PE01	Sollwertkurve bestehend aus Wertepaaren IL_COORD_TYPE01, Seite 511
VAR_OUTPUT	InOperation	BOOL	Der Baustein hat die Initialisierung abgeschlossen und "Deviation-Weighted" ist gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes, Seite 376
	Deviation-Weighted	REAL	Durch "ResponseCurve" gewichtete Abweichung

Tab. 7-34: Funktionsbaustein IL_DeviationResponseCurve

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Deviation	REAL	n.def.	n.def.	0.0	kontinuierlich
ResponseCurve	ARRAY[0..3] OF IL_COORD_TY-PE01				steigende Flanke von Enable

Tab. 7-35: Min-, Max- und Default-Werte der Eingänge des IL_DeviationResponseCurve

Funktionsbeschreibung Initialisierung

Bei steigender Flanke an "Enable" wird eine Prüfung von "ResponseCurve" durchgeführt. nach erfolgreicher Prüfung wird "InOperation" gesetzt. Der Wert in "DeviationWeighted" in dann gültig.

Normalbetrieb

Bei aktivem "Enable"-Eingang arbeitet der Funktionsbaustein. Bei jedem Aufruf wird der Wert in "Deviation" anhand der in "ResponseCurve" angelegten Sollwertkurve übertragen und an "DeviationWeighted" ausgegeben.

Konfiguration

Die in "ResponseCurve" definierte Kennlinie besteht aus vier Punkten. Negative Abweichungen werden durch Punktspiegelung der Kurve am Ursprung behandelt. Daraus ergeben sich insgesamt neun lineare Sektionen. Die Sektion s5 verläuft durch den Ursprung. Die Sektionen s1 und s9 haben einen Anstieg von 1 und laufen jeweils ins positive oder negative Unendliche.

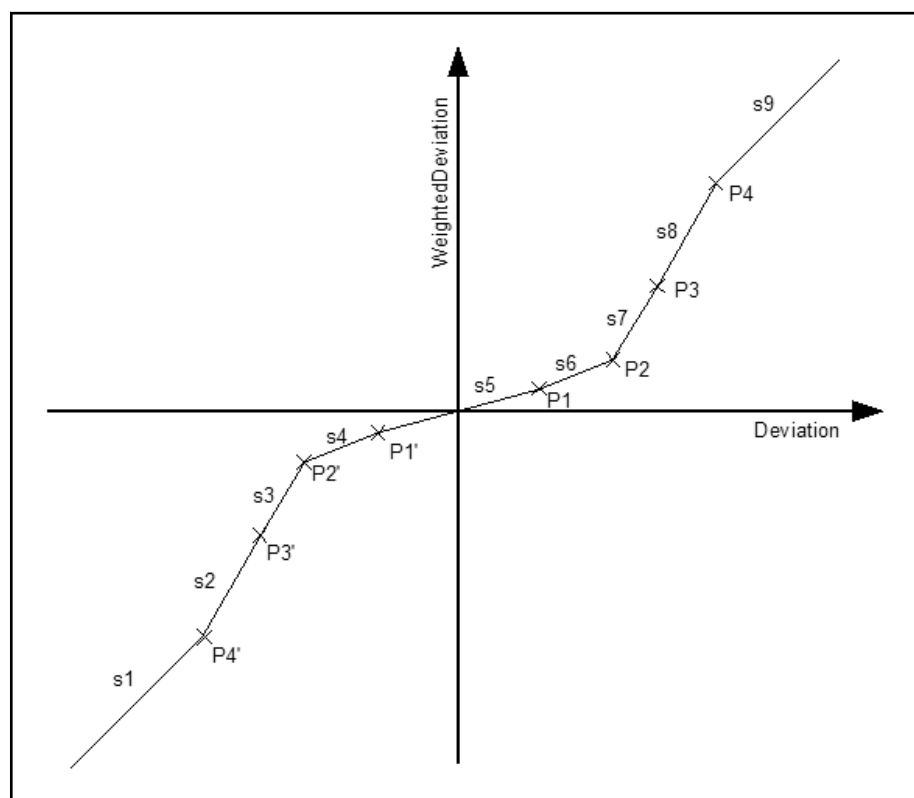


Abb. 7-22: Sollwertkurve

Die Punkte P1 bis P4 werden in einem Array aus IL_COORD_TYPE01 abgelegt. Werden weniger als vier Punkte benötigt, muss die X-Koordinate der nicht benötigten Punkte (vom letzten Punkt beginnend) auf 0 gesetzt werden. Die X-Werte müssen positiv sein und in positiver Richtung aufsteigen. Es ist erlaubt, denselben Punkt zweimal abzulegen.

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR, 16#0001	16#00000C05	16#00000001	X-Werte in "ResponseCurve" nicht aufsteigend
INPUT_INVALID_ERROR, 16#0001	16#00000C05	16#00000002	Mehr als einen Werte für eine X-Position in "ResponseCurve"
INPUT_RANGE_ERROR, 16#0006	16#00000C05	16#00000003	Negativer X-Wert in „ResponseCurve“

Tab. 7-36: Generierte Fehlernummern des Funktionsbausteins IL_DeviationResponseCurve

7.5.5 IL_DT1Type01

Kurzbeschreibung

Dieser Funktionsbaustein implementiert ein zeitdiskretes Differenzierglied mit Verzögerung erster Ordnung. Die Berechnung erfolgt kontinuierlich, d.h. bei jedem Aufruf des Funktionsbausteins. Der Funktionsbaustein sollte zyklisch aufgerufen werden, um eine möglichst gleichmäßige Berechnung zu ermöglichen. Die vom Funktionsbaustein gemessene Zykluszeit steht auch als Ausgang zur Verfügung.

Schnittstellenbeschreibung

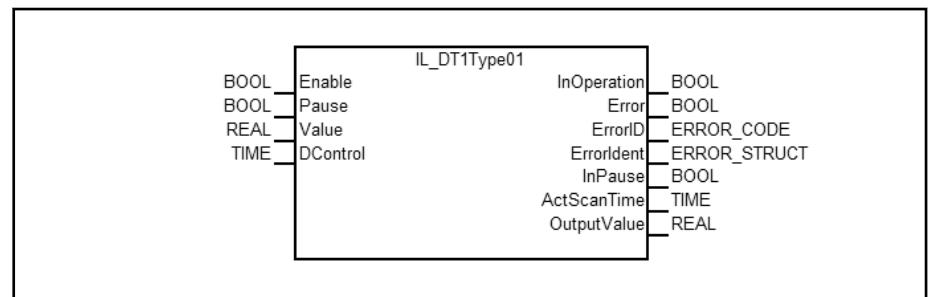


Abb. 7-23: Funktionsbaustein IL_DT1Type01

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Wenn TRUE, wird "OutputValue" auf den voreingestellten Wert von "StartValue" bzw. auf den berechneten Wert gesetzt.
	Pause	BOOL	Wenn TRUE, wird der Wert von "OutputValue" vom Regler nicht verändert.
	Value	REAL	Der aktuelle Istwert des Prozesses.
	DControl	TIME	Zeitkonstante Tn des DT1-Gliedes.
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, dann arbeitet das DT1-Glied und der Reglerausgang "OutputValue" ist gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes , Seite 379.
	InPause	BOOL	"Pause" ist aktiv und der Wert an "ControlValue" ist eingefroren.
	ActScanTime	TIME	Die vergangene Zeit seit der Funktionsbaustein das letzte Mal aufgerufen wurde.
	OutputValue	REAL	Der berechnete Reglerausgangswert

Tab. 7-37: Funktionsbaustein IL_DT1Type01

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Value	REAL	n. def.	n.def.	0.0	kontinuierlich
DControl	TIME	0s	n.def.	1s	kontinuierlich

Tab. 7-38: Min-, Max- und Default-Werte der Eingänge des IL_DT1Type01

Funktionsbeschreibung

Der Funktionsbaustein gibt am Ausgang "OutputValue" den berechneten Wert der implementierten DT1-Funktion aus. Die Berechnung erfolgt kontinuierlich bei gesetztem "Enable". Der Wert des "Value"-Eingangs wird bei einer steigenden Flanke am "Enable"-Eingang übernommen und für die interne Berechnung bereitgestellt. Alle Eingänge werden kontinuierlich behandelt.

Im Initialisierungsprozess des Bausteines werden die Eingangsvariablen "Value" sowie die Zeitkonstante "DControl" übernommen. "OutputValue" wird mit 0.0 initialisiert.

Pause

Der Eingang "Pause" wird nur ausgewertet, solange der "Enable"-Eingang aktiv ist. Liegt an "Pause" ein TRUE an, wird keine Berechnung durchgeführt und der Wert von "OutputValue" nicht verändert. Der interne Integrator ist ebenfalls eingefroren.

Der Ausgang "InPause" ist gesetzt, solange "Pause" aktiv ist.

Der Ausgang "ActScanTime" wird weiterhin aktualisiert.

Normalbetrieb

Die Sprungantwort eines DT1-Gliedes hat den Verlauf einer sich 0 annähernden Kurve.

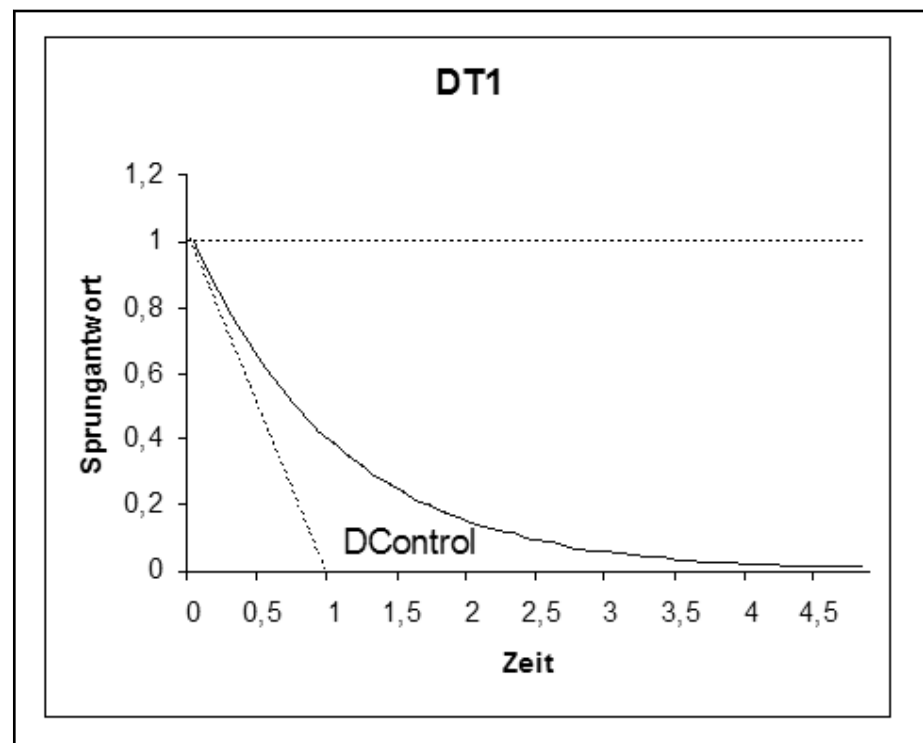


Abb. 7-24: Sprungantwort IL_DT1Type01

Die Übertragungsfunktion lautet:

$$G(s) = \frac{s}{1 + s \cdot DControl}$$

"DControl" gibt die Verzögerungszeitkonstante an. Ist "DControl" 0.0, so ist das DT1-Glied deaktiviert und der Ausgang "OutputValue" auf 0.0 gesetzt.

Fehlercodes Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR,16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.

Tab. 7-39: Generierte Fehlernummern des Funktionsbausteins IL_DT1Type01

7.5.6 IL_DT1Type02

Kurzbeschreibung

Dieser Funktionsbaustein implementiert ein eventgesteuertes Differenzierglied mit Verzögerung erster Ordnung. Die Berechnung erfolgt eventgesteuert, d.h. bei jeder Änderung am Step-Eingang. Der letzte bearbeitete Schritt steht als Ausgang zur Verfügung.

Schnittstellenbeschreibung

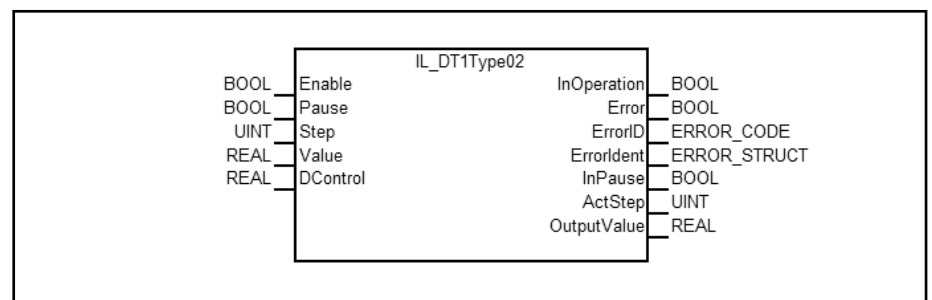


Abb. 7-25: Funktionsbaustein IL_DT1Type02

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Wenn TRUE, wird "OutputValue" auf den voreingestellten Wert von "StartValue" bzw. auf den berechneten Wert gesetzt.
	Pause	BOOL	Wenn TRUE, wird der Wert von "OutputValue" vom Regler nicht verändert.
	Step	UINT	Bearbeitungsschritt
	Value	REAL	Der aktuelle Istwert des Prozesses.
	DControl	REAL	Zeitkonstante Tn des DT1-Gliedes.
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, dann arbeitet das DT1-Glied und der Reglerausgang "OutputValue" ist gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes , Seite 381.
	InPause	BOOL	"Pause" ist aktiv und der Wert an "ControlValue" ist eingefroren.

	Name	Typ	Kommentar
	ActStep	UINT	Letzter bearbeiteter Schritt
	OutputValue	REAL	Der berechnete Reglerausgangswert

Tab. 7-40: Funktionsbaustein IL_DT1Type02

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Step	UINT	0	65335	0	kontinuierlich
Value	REAL	n. def.	n.def.	0.0	kontinuierlich
DControl	REAL	0.0	n.def.	1.0	kontinuierlich

Tab. 7-41: Min-, Max- und Default-Werte der Eingänge des IL_DT1Type02

Funktionsbeschreibung

Der Funktionsbaustein gibt am Ausgang "OutputValue" den berechneten Wert der implementierten DT1-Funktion aus. Die Berechnung erfolgt bei Änderung am "Step"-Eingang bei gesetztem "Enable". Die Schrittweite am "Step"-Eingang wird behandelt, d.h. es wird bei Bedarf mehr als ein Schritt berechnet. Der Wert des "Value"-Eingangs wird bei einer steigenden Flanke am "Enable"-Eingang übernommen und für die interne Berechnung bereitgestellt. Alle Eingänge werden kontinuierlich behandelt.

Im Initialisierungsprozess des Bausteines werden Eingangsvariablen "Value" sowie die Zeitkonstante "DControl" übernommen. "OutputValue" wird mit 0.0 initialisiert.

Pause

Der Eingang "Pause" wird nur ausgewertet, solange der "Enable"-Eingang aktiv ist. Liegt an "Pause" ein TRUE an, wird keine Berechnung durchgeführt und der Wert von "OutputValue" nicht verändert. Der interne Integrator ist ebenfalls eingefroren.

Der Ausgang "InPause" ist gesetzt, solange "Pause" aktiv ist.

Der Ausgang "ActStep" wird weiterhin aktualisiert.

Normalbetrieb

Die Sprungantwort eines DT1-Gliedes hat den Verlauf einer sich 0 annähernden Kurve.

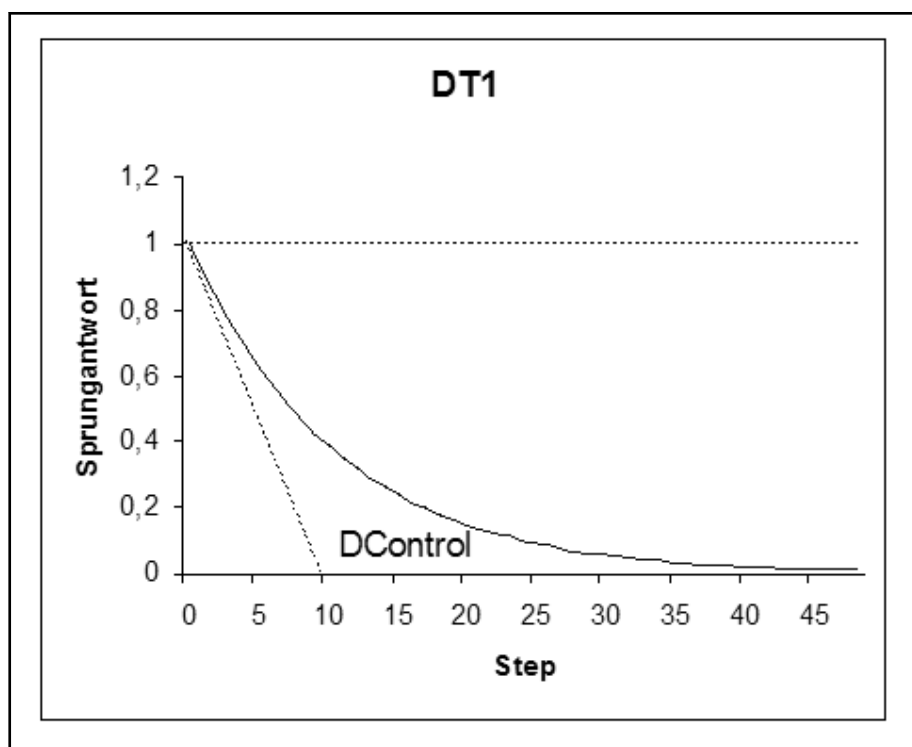


Abb. 7-26: Sprungantwort IL_DT1Type02

Die Übertragungsfunktion lautet:

$$G(s) = \frac{s}{1 + s \cdot DControl}$$

"DControl" gibt die Verzögerungszeitkonstante an. Ist "DControl" 0.0, so ist das DT1-Glied deaktiviert und der Ausgang "OutputValue" wird bei einer Änderung am "Step"-Eingang auf 0.0 gesetzt.

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR, 16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.
INPUT_RANGE_ERROR, 16#0006	16#00000C02	16#00000002	Der Eingang "DControl" ist ungültig

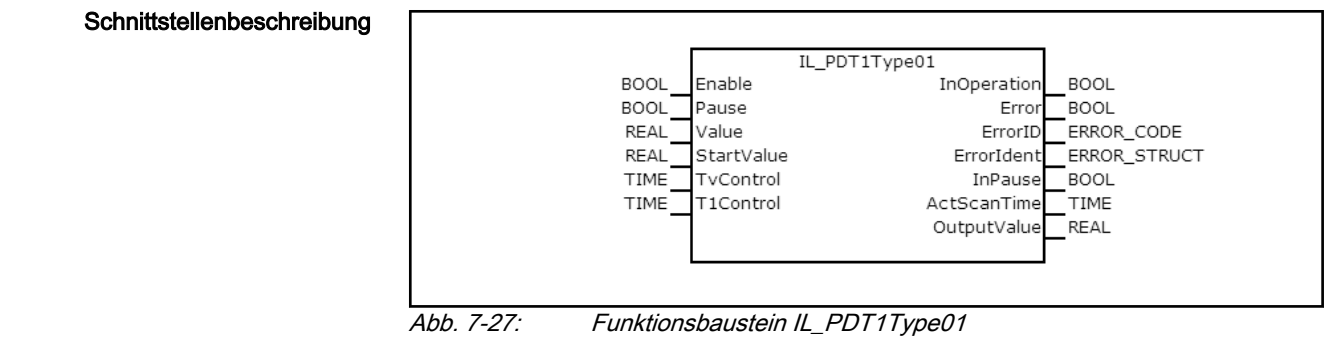
Tab. 7-42: Generierte Fehlernummern des Funktionsbausteins IL_DT1Type02

7.5.7 IL_PDT1Type01

Kurzbeschreibung

Der Funktionsbaustein IL_PDT1Type01 implementiert einen zeitdiskreten PDT1-Filter. Dieser digitale Filter lässt sich für das Anheben bzw. Absenken der Übertragungsfunktion ab einer bestimmten Frequenz einsetzen.

Die Berechnung erfolgt kontinuierlich, d.h. bei jedem Aufruf des Funktionsbausteins. Der Funktionsbaustein sollte zyklisch aufgerufen werden, um eine möglichst gleichmäßige Berechnung zu ermöglichen. Die vom Funktionsbaustein gemessene Zykluszeit steht auch als Ausgang zur Verfügung.



	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Wenn TRUE, wird "OutputValue" auf den mit "StartValue" berechneten Wert gesetzt.
	Pause	BOOL	aktiviert den Pause-Modus wenn TRUE
	Value	REAL	Der aktuelle Istwert des Prozesses.
	StartValue	REAL	Startwert, der bei steigender Flanke an "Enable" am Ausgang "OutputValue" angenommen wird.
	TvControl	TIME	Vorhaltzeitkonstante
	T1Control	TIME	Zeitkonstante
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, dann arbeitet der PT1-Glieds und der Reglerausgang "OutputValue" ist gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes, Seite 384.
	InPause	BOOL	"Pause" ist aktiv und der Wert an "ControlValue" ist eingefroren.
	ActScanTime	TIME	Die vergangene Zeit seit dem letzten Aufruf des Funktionsbausteins.
	OutputValue	REAL	Der berechnete Filterwert

Tab. 7-43: Funktionsbaustein IL_PDT1Type01

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Value	REAL	n. def.	n.def.	0.0	kontinuierlich
StartValue	REAL	n.def.	n.def.	0.0	steigende Flanke von Enable
TvControl	TIME	0s	n.def.	1ms	kontinuierlich
T1Control	TIME	0s	n.def.	1ms	kontinuierlich

Tab. 7-44: Min-, Max- und Default-Werte der Eingänge des IL_PDT1Type01

Funktionsbeschreibung

Der Funktionsbaustein IL_PDT1Type01 implementiert einen zeitdiskreten PDT1-Filter. Dieser digitale Filter lässt sich für das Anheben bzw. Absenken der Übertragungsfunktion ab einer bestimmten Frequenz einsetzen. Der Funktionsbaustein arbeitet zeitdiskret.

Das Verhalten hängt vom Verhältnis der beiden Zeitkonstanten zueinander ab:

Für $T_v > T_1$: Das Signal wird oberhalb von T_v und T_1 abgeschwächt.

Für $T_v < T_1$: Das Signal wird oberhalb von T_v und T_1 verstärkt.

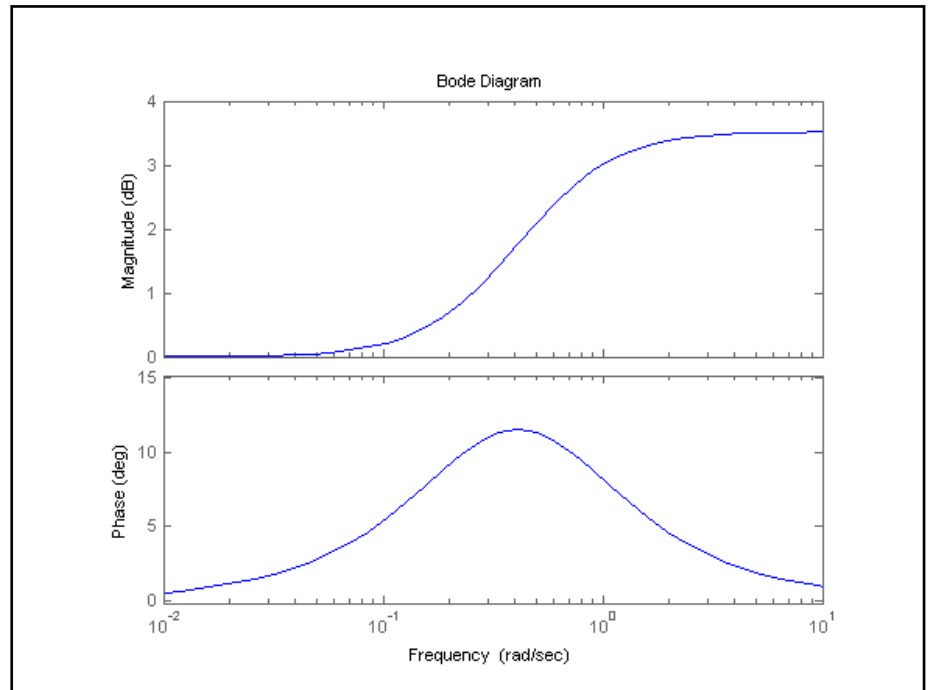


Abb. 7-28: Bode-Diagramm für $T_v > T_1$

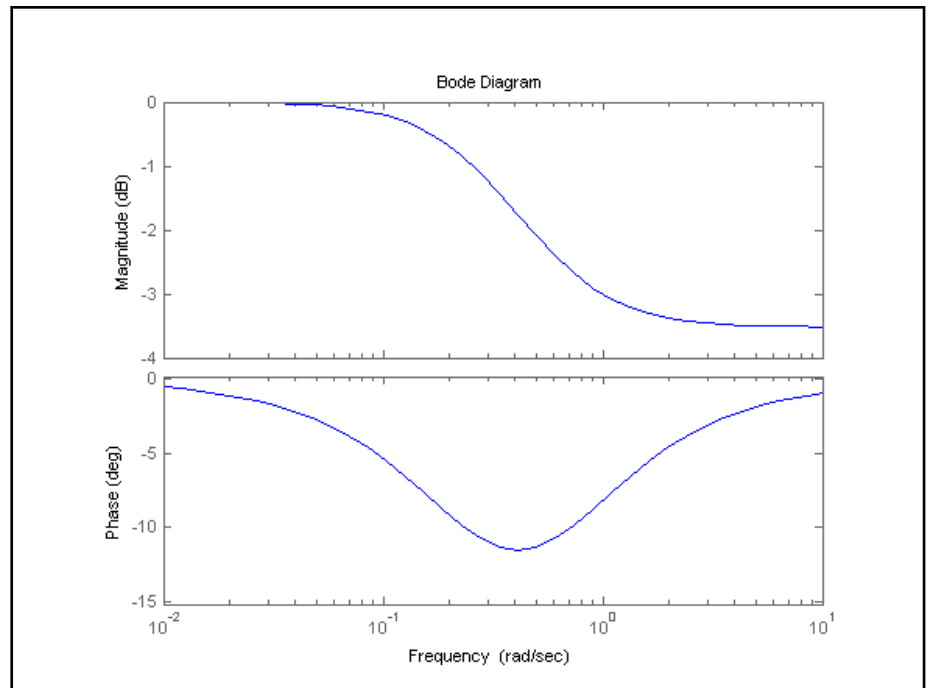


Abb. 7-29: Bode-Diagramm für $T_v < T_1$

Value, StartValue, OutputValue

Der Eingang "Value" enthält den Eingangswert des Filters. Dieser wird bei jedem Aufruf verarbeitet, wenn "InOperation" gesetzt ist. Bei steigender Flanke an "Enable" wird der Wert in "StartValue" in alle internen Register übernommen. Das Ergebnis wird am Ausgang "OutputValue" ausgegeben.

TvControl, T1Control

Diese beiden Eingänge bestimmen das Filterverhalten. "TvControl" ist die Vorhaltezeit. Das Verhältnis von Tv/T1 bestimmt die die Steilheit des Abfalls.

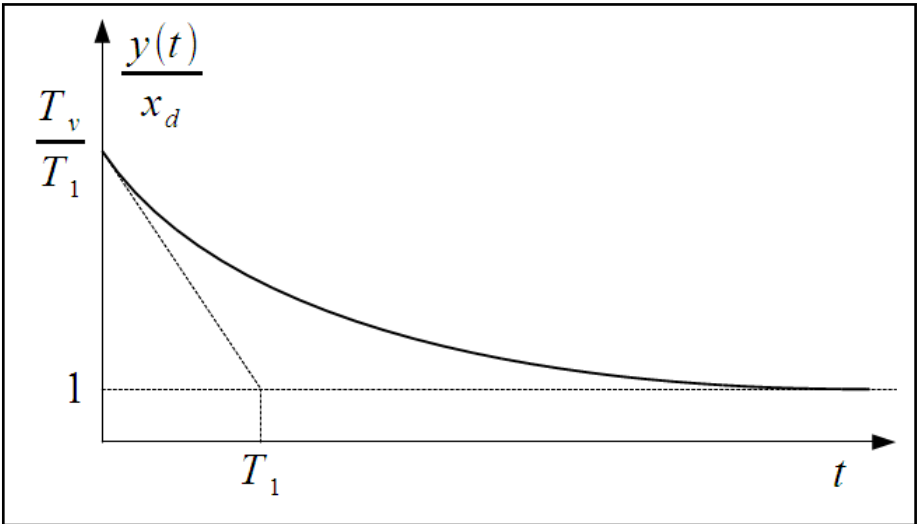


Abb. 7-30: Sprungantwort eines PDT1-Gliedes

Pause

Der Eingang "Pause" wird nur ausgewertet, solange der "Enable"-Eingang aktiv ist. Ist "Pause" TRUE, wird keine Berechnung durchgeführt und der Wert von "OutputValue" nicht verändert. Der interne Integrator ist ebenfalls eingefroren.

Der Ausgang "InPause" ist gesetzt, solange "Pause" aktiv ist.

Der Ausgang "ActScanTime" wird weiterhin aktualisiert.

Zeitverhalten Der Funktionsbaustein IL_PDT1Type01 arbeitet asynchron. Nach dem Start durch eine steigende Flanke am Eingang "Enable" liest er den Eingang "StartValue". Der Funktionsbaustein liest bei jedem Aufruf einen Wert vom Eingänge "Value", "TvControl" und "T1Control" und berechnet einen neuen Wert am Ausgang "OutputValue". Bei aktivem Filter ist der Ausgang "InOperation" gesetzt. Tritt ein Fehler auf, wird der Ausgang "Error" gesetzt.

Fehlercodes Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR, 16#0006	16#00000C02	16#00000006	Der Eingang "TvControl" ist ungültig.
INPUT_RANGE_ERROR, 16#0006	16#00000C02	16#00000007	Der Eingang "T1Control" ist ungültig.
STATE_MACHINE_ERROR, 16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.

Tab. 7-45: Generierte Fehlernummern des Funktionsbausteins IL_PDT1Type01

7.5.8 IL_PDT1Type02

Kurzbeschreibung

Der Funktionsbaustein IL_PDT1Type02 implementiert einen event-gesteuerten PDT1-Filter. Dieser digitale Filter lässt sich für das Anheben bzw. Absenken der Übertragungsfunktion ab einer bestimmten Frequenz einsetzen.

Die Berechnung erfolgt eventgesteuert, d.h. bei jeder Änderung am Step-Eingang. Der letzte bearbeitete Schritt steht als Ausgang zur Verfügung.

Schnittstellenbeschreibung

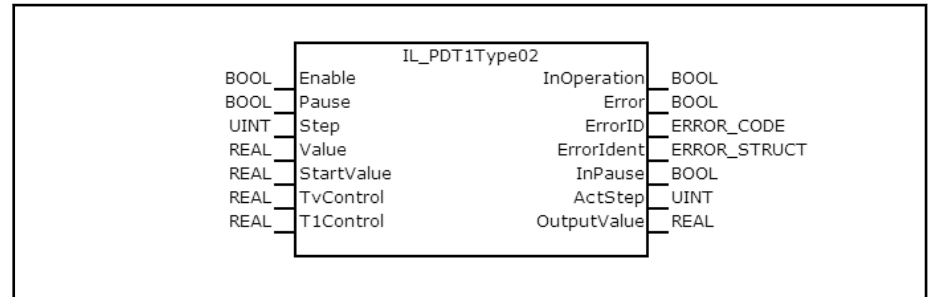


Abb. 7-31: Funktionsbaustein IL_PDT1Type02

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Wenn TRUE, wird "OutputValue" auf den mit "StartValue" berechneten Wert gesetzt.
	Pause	BOOL	aktiviert den Pause-Modus wenn TRUE
	Step	UINT	Bearbeitungsschritt
	Value	REAL	Der aktuelle Istwert des Prozesses.
	StartValue	REAL	Startwert, der bei steigender Flanke an "Enable" am Ausgang "OutputValue" angenommen wird.
	TvControl	TIME	Vorhaltzeitkonstante
	T1Control	TIME	Zeitkonstante
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, dann arbeitet der PT1-Glieds und der Reglerausgang "OutputValue" ist gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes , Seite 388.
	InPause	BOOL	"Pause" ist aktiv und der Wert an "ControlValue" ist eingefroren.
	ActStep	UINT	Schritt, auf den sich "OutputValue" bezieht.
	OutputValue	REAL	Der berechnete Filterwert

Tab. 7-46: Funktionsbaustein IL_PDT1Type02

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Step	UINT	0	65335	0	kontinuierlich
Value	REAL	n. def.	n.def.	0.0	kontinuierlich
StartValue	REAL	n.def.	n.def.	0.0	steigende Flanke von Enable
TvControl	TIME	>0s	n.def.	1.0	kontinuierlich
T1Control	TIME	>0s	n.def.	1.0	kontinuierlich

Tab. 7-47: Min-, Max- und Default-Werte der Eingänge des IL_PDT1Type02

Funktionsbeschreibung

Der Funktionsbaustein IL_PDT1Type02 implementiert einen event-gesteuerten PDT1-Filter. Dieser digitale Filter lässt sich für das Anheben bzw. Absenken der Übertragungsfunktion ab einer bestimmten Frequenz einsetzen. Der Funktionsbaustein arbeitet event-gesteuert, d.h. bei Änderung am Eingang „Step“ wird ein neuer Wert verarbeitet.

Das Verhalten hängt vom Verhältnis der beiden Zeitkonstanten zueinander ab:

Für $T_v > T_1$: Das Signal wird oberhalb von T_v und T_1 abgeschwächt.

Für $T_v < T_1$: Das Signal wird oberhalb von T_v und T_1 verstärkt.

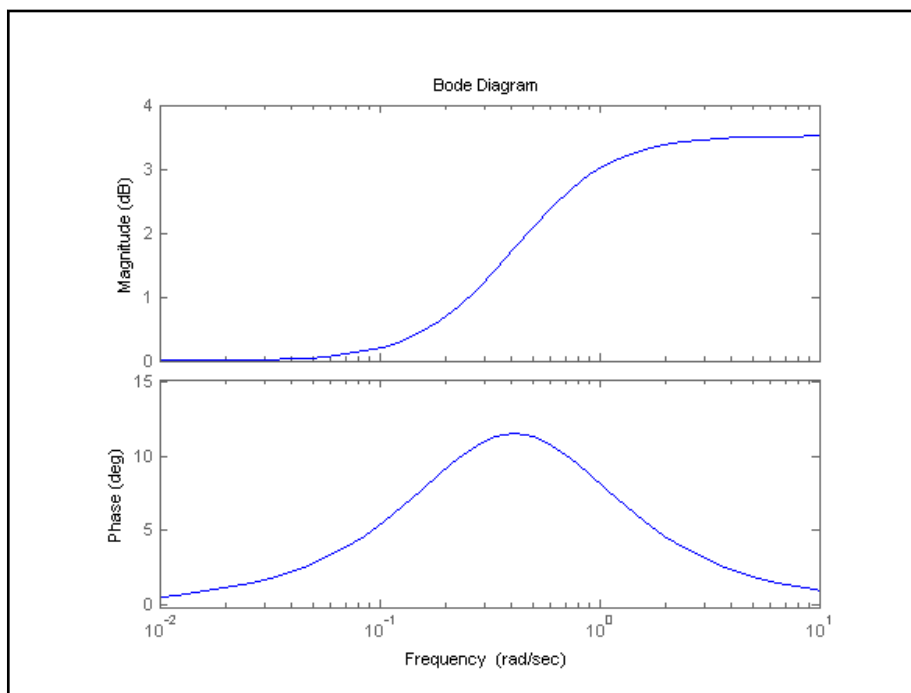
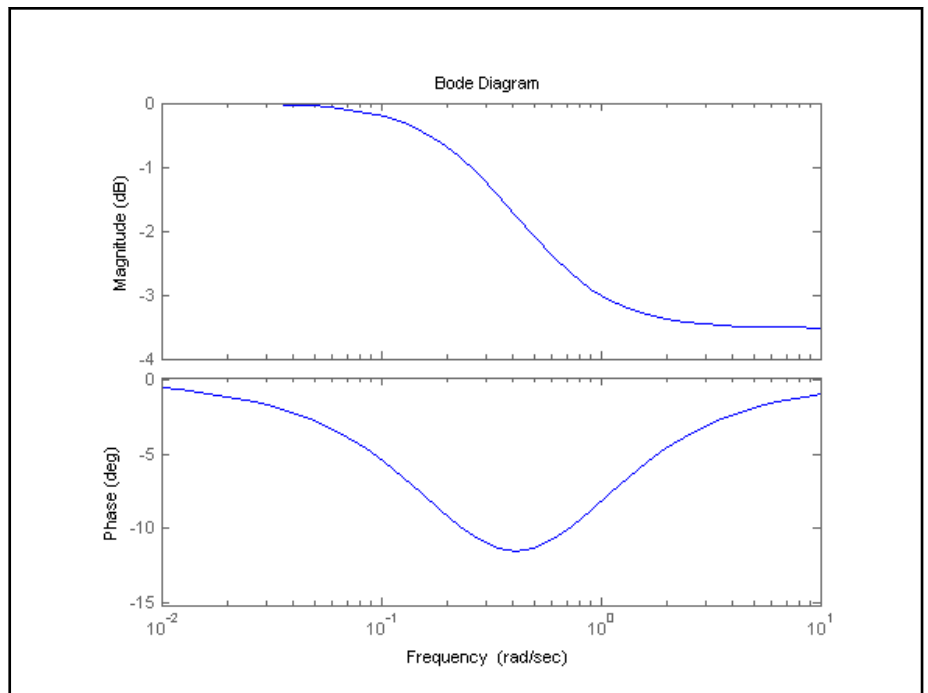


Abb. 7-32: Bode-Diagramm für $T_v > T_1$

Abb. 7-33: Bode-Diagramm für $T_v < T_1$ **Value, StartValue, OutputValue**

Der Eingang "Value" enthält den Eingangswert des Filters. Dieser wird bei jedem Aufruf verarbeitet, wenn "InOperation" gesetzt ist. Bei steigender Flanke an "Enable" wird der Wert in "StartValue" in alle internen Register übernommen. Das Ergebnis wird am Ausgang "OutputValue" ausgegeben.

TvControl, T1Control

Diese beiden Eingänge bestimmen das Filterverhalten. "TvControl" ist die Vorhaltezeit. Das Verhältnis von T_v/T_1 bestimmt die Steilheit des Abfalls.

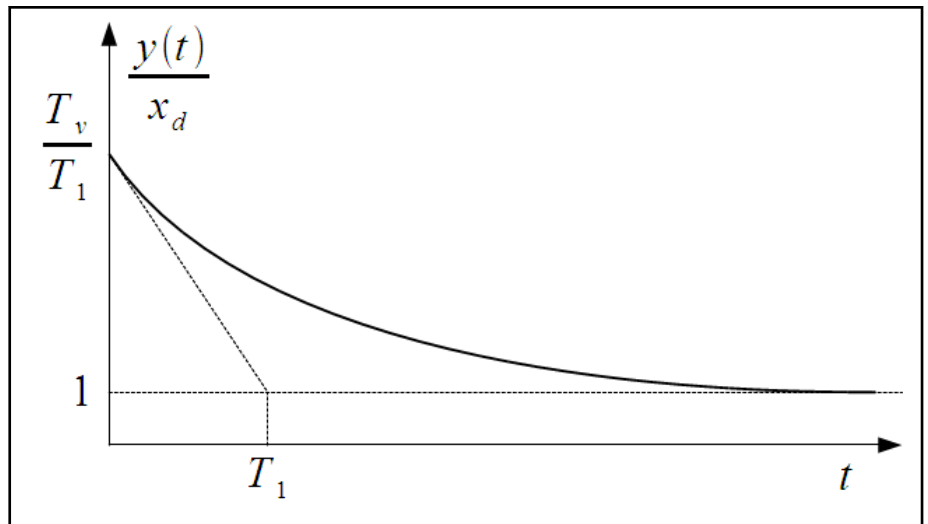


Abb. 7-34: Sprungantwort eines PDT1-Gliedes

Pause

Der Eingang "Pause" wird nur ausgewertet, solange der "Enable"-Eingang aktiv ist. Ist "Pause" TRUE, wird keine Berechnung durchgeführt und der Wert von "OutputValue" nicht verändert. Der interne Integrator ist ebenfalls eingefroren.

Der Ausgang "InPause" ist gesetzt, solange "Pause" aktiv ist.

Der Ausgang "ActStep" wird weiterhin aktualisiert.

Zeitverhalten Der Funktionsbaustein IL_PDT1Type02 arbeitet asynchron. Nach dem Start durch eine steigende Flanke am Eingang "Enable" liest er den Eingang "StartValue". Der Funktionsbaustein liest bei jedem Aufruf einen Wert vom Eingänge "Value", "TvControl" und "T1Control" und berechnet einen neuen Wert am Ausgang "OutputValue". Bei aktivem Filter ist der Ausgang "InOperation" gesetzt. Tritt ein Fehler auf, wird der Ausgang "Error" gesetzt.

Fehlercodes Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR, 16#0006	16#00000C02	16#00000006	Der Eingang "TvControl" ist ungültig.
INPUT_RANGE_ERROR, 16#0006	16#00000C02	16#00000007	Der Eingang "T1Control" ist ungültig.
STATE_MACHINE_ERROR, 16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.

Tab. 7-48: Generierte Fehlernummern des Funktionsbausteins IL_PDT1Type02

7.5.9 IL_PT1Type01

Kurzbeschreibung Dieser Funktionsbaustein implementiert ein zeitdiskretes Verzögerungsglied erster Ordnung. Die Berechnung erfolgt kontinuierlich, d.h. bei jedem Aufruf des Funktionsbausteins. Der Funktionsbaustein sollte zyklisch aufgerufen werden, um eine möglichst gleichmäßige Berechnung zu ermöglichen. Die vom Funktionsbaustein gemessene Zykluszeit steht auch als Ausgang zur Verfügung.

Schnittstellenbeschreibung

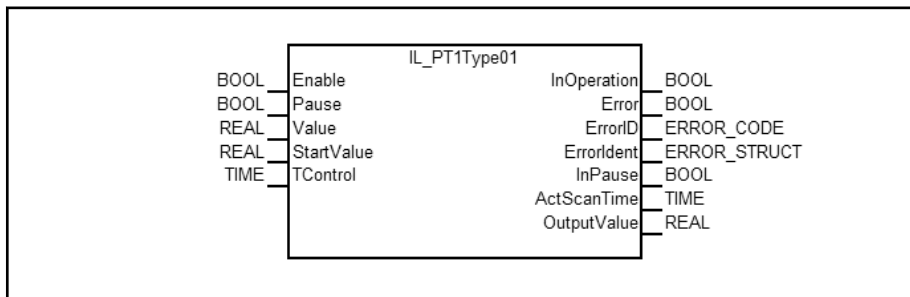


Abb. 7-35: Funktionsbaustein IL_PT1Type01

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Wenn TRUE, wird "OutputValue" auf den voreingestellten Wert von "StartValue" bzw. auf den berechneten Wert gesetzt.
	Pause	BOOL	Wenn TRUE, bleibt der Wert an "OutputValue" konstant.
	Value	REAL	Der aktuelle Istwert des Prozesses.
	StartValue	REAL	Startwert, der bei steigender Flanke an "Enable" am Ausgang "OutputValue" angenommen wird.
	TControl	TIME	Zeitkonstante Tn des PT1-Glieds.
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, dann arbeitet der PT1-Glieds und der Reglerausgang "OutputValue" ist gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.

	Name	Typ	Kommentar
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes , Seite 390.
	InPause	BOOL	"Pause" ist aktiv und der Wert an "ControlValue" ist eingefroren.
	ActScan-Time	TIME	Die vergangene Zeit seit dem letzten Aufruf des Funktionsbausteins.
	OutputValue	REAL	Der berechnete Ausgangswert

Tab. 7-49: Funktionsbaustein IL_PT1Type01

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Value	REAL	n. def.	n.def.	0.0	kontinuierlich
StartValue	REAL	n.def.	n.def.	0.0	steigende Flanke von Enable
TControl	TIME	0s	n.def.	1s	kontinuierlich

Tab. 7-50: Min-, Max- und Default-Werte der Eingänge des IL_PT1Type01

Funktionsbeschreibung

Der Funktionsbaustein gibt am Ausgang "OutputValue" den berechneten Wert der Verzögerungsfunktion erster Ordnung aus. Die Berechnung erfolgt kontinuierlich bei gesetztem "Enable"-Eingang. Die Werte des "StartValue"-Eingangs wird ausschließlich bei einer steigenden Flanke am "Enable"-Eingang übernommen. Er steht dann am Ausgang "OutputValue" an und dient als Startwert für den internen Integrator. Alle anderen Eingänge werden kontinuierlich behandelt.

Start-Wert

Dem Baustein kann über die Eingangsvariable "StartValue" optional ein Startwert der Funktion übergeben werden. Im Initialisierungsprozess des Bausteines wird dieser Wert zusammen mit den weiteren Eingangsvariablen wie „Value“ sowie der Zeitkonstanten "TControl" übernommen.

Pause

Der Eingang "Pause" wird nur ausgewertet, solange der "Enable"-Eingang aktiv ist. Liegt an "Pause" ein TRUE an, wird keine Berechnung durchgeführt und der Wert von "OutputValue" nicht verändert. Der interne Integrator ist ebenfalls eingefroren.

Der Ausgang "InPause" ist gesetzt, solange "Pause" aktiv ist.

Der Ausgang "ActScanTime" wird aktualisiert, solange "Pause" aktiv ist.

Normalbetrieb

Die Sprungantwort eines PT1-Gliedes hat den Verlauf einer sich dem Sollwert "Value" annähernden Kurve.

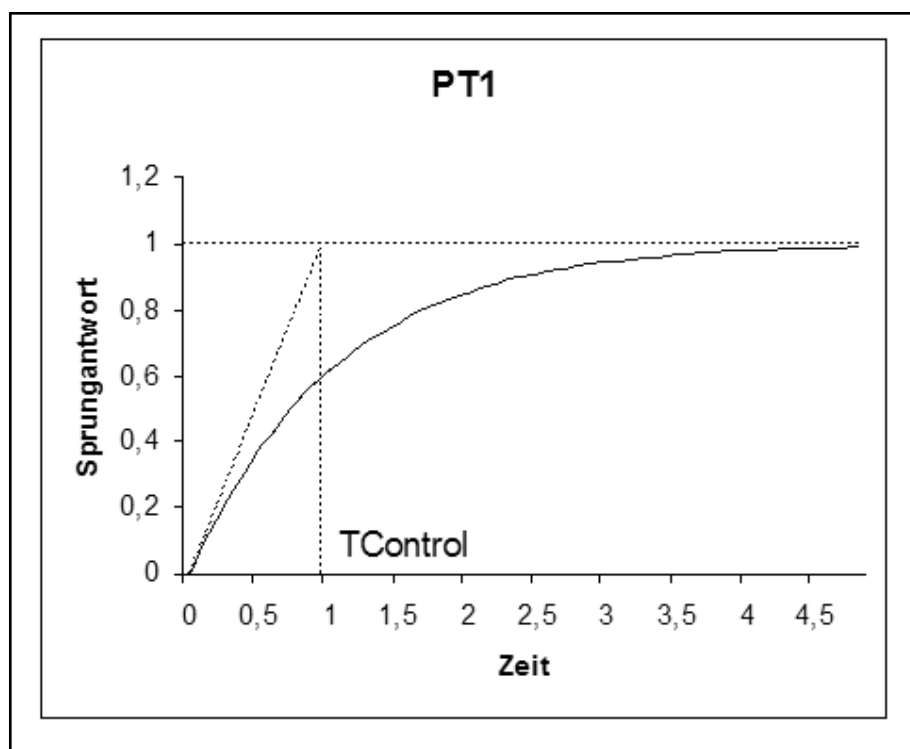


Abb. 7-36: Sprungantwort IL_PT1Type01

Die Übertragungsfunktion lautet:

$$G(s) = \frac{1}{1 + s \cdot TControl}$$

"TControl" gibt die Verzögerungszeitkonstante an. Ist "TControl" 0s, so ist das PT1-Glied deaktiviert und der Eingang "Value" wird direkt auf "OutputValue" kopiert.

Die Eingänge "TControl" und "Value" werden bei jedem Aufruf des Funktionsbausteins übernommen, solange "Enable" gesetzt ist.

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR,16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.
INPUT_RANGE_ERROR	16#00000C02	16#00000001	Der Eingang "TControl" ist ungültig

Tab. 7-51: Generierte Fehlernummern des Funktionsbausteins IL_PT1Type01

7.5.10 IL_PT1Type02

Kurzbeschreibung

Dieser Funktionsbaustein implementiert ein eventgesteuertes Verzögerungsglied erster Ordnung. Die Berechnung erfolgt eventgesteuert, d.h. bei jeder Änderung am Step-Eingang. Der letzte bearbeitete Schritt steht als Ausgang zur Verfügung.

Schnittstellenbeschreibung

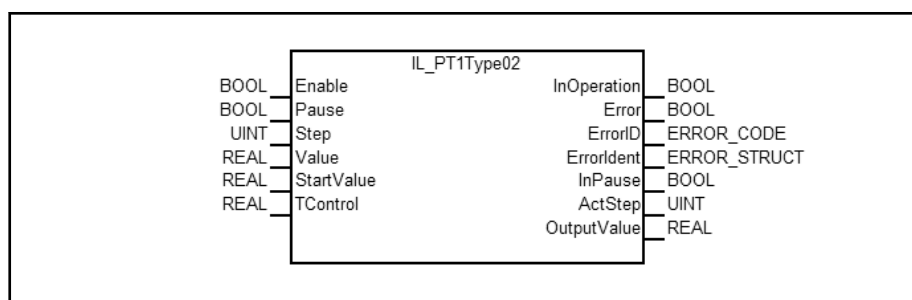


Abb. 7-37: Funktionsbaustein IL_PT1Type02

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Wenn TRUE, wird "OutputValue" auf den voreingestellten Wert von "StartValue" bzw. auf den berechneten Wert gesetzt.
	Pause	BOOL	Wenn TRUE, bleibt der Wert an "OutputValue" konstant.
	Step	UINT	Bearbeitungsschritt
	Value	REAL	Der aktuelle Istwert des Prozesses.
	StartValue	REAL	Startwert, der bei steigender Flanke an "Enable" am Ausgang "OutputValue" angenommen wird.
	TControl	REAL	Zeitkonstante Tn des PT1-Gliedes.
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, dann arbeitet das PT1-Glied und der Reglerausgang "OutputValue" ist gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes , Seite 393.
	InPause	BOOL	"Pause" ist aktiv und der Wert an "ControlValue" ist eingefroren.
	ActStep	UINT	Schritt, auf den sich "OutputValue" bezieht.
	OutputValue	REAL	Der berechnete Reglerausgangswert

Tab. 7-52: Funktionsbaustein IL_PT1Type02

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Step	UINT	0	65335	0	kontinuierlich
Value	REAL	n. def.	n.def.	0.0	kontinuierlich
StartValue	REAL	n.def.	n.def.	0.0	steigende Flanke von Enable
TControl	REAL	0.0	n.def.	1.0	kontinuierlich

Tab. 7-53: Min-, Max- und Default-Werte der Eingänge des IL_PT1Type02

Funktionsbeschreibung

Der Funktionsbaustein gibt am Ausgang "OutputValue" den berechneten Wert der implementierten PT1-Funktion aus. Die Berechnung erfolgt bei Än-

derung am "Step"-Eingang bei gesetztem "Enable". Die Schrittweite am "Step"-Eingang wird behandelt, d.h. es wird bei Bedarf mehr als ein Schritt berechnet. Der Wert des "StartValue"-Eingangs wird ausschließlich bei einer steigenden Flanke am "Enable"-Eingang übernommen. Er steht dann am Ausgang "OutputValue" an und dient als Startwert für den internen Integrator. Alle anderen Eingänge werden kontinuierlich behandelt.

Start-Wert

Dem Baustein kann über die Eingangsvariable "StartValue" optional ein Startwert der Funktion übergeben werden. Im Initialisierungsprozess des Bausteines wird dieser Wert zusammen mit den weiteren Eingangsvariablen wie "Value" sowie der Zeitkonstanten "TControl" übernommen.

Pause

Der Eingang "Pause" wird nur ausgewertet, solange der "Enable"-Eingang aktiv ist. Liegt an "Pause" ein TRUE an, wird keine Berechnung durchgeführt und der Wert von "OutputValue" nicht verändert. Der interne Integrator ist ebenfalls eingefroren.

Der Ausgang "InPause" ist gesetzt, solange "Pause" aktiv ist.

Der Ausgang "ActStep" wird weiterhin aktualisiert.

Normalbetrieb

Die Sprungantwort eines PT1-Gliedes hat den Verlauf einer sich dem Sollwert "Value" annähernden Kurve.

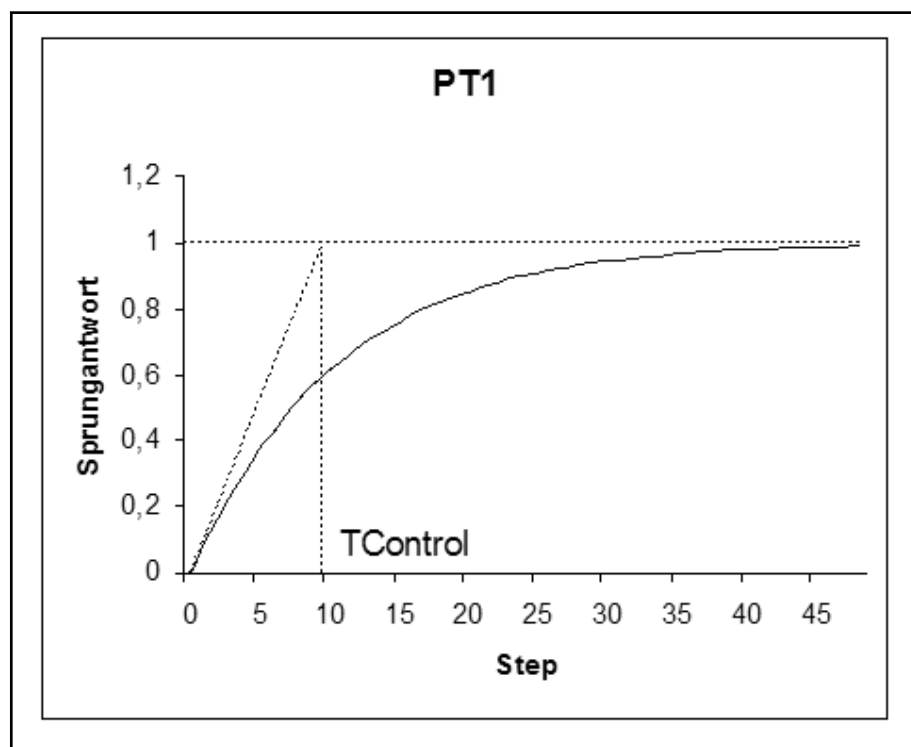


Abb. 7-38: Sprungantwort IL_PT1Type02

Die Übertragungsfunktion lautet:

$$G(s) = \frac{1}{1 + s \cdot TControl}$$

"TControl" gibt die Verzögerungszeitkonstante an. Ist "TControl" 0.0, so ist das PT1-Glied deaktiviert und der Eingang "Value" wird bei jeder Änderung am "Step"-Eingang direkt auf "OutputValue" kopiert.

Die Eingänge "TControl" und "Value" werden bei jedem Aufruf des Funktionsbausteins übernommen, solange "Enable" gesetzt ist.

Fehlercodes Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR,16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.
INPUT_RANGE_ERROR. 16#0006	16#00000C02	16#00000001	Der Eingang "TControl" ist ungültig

Tab. 7-54: Generierte Fehlernummern des Funktionsbausteins IL_PT1Type02

7.5.11 IL_TransportDelayType01

Kurzbeschreibung

Dieser Funktionsbaustein implementiert ein zeitdiskretes Totzeitglied mit variabler Totzeit. Die Totzeit kann kontinuierlich verändert werden. Die tatsächliche Totzeit wird fließend an den vorgegebenen Wert angepasst. Die Anpassung an die neue Totzeit erfolgt nach dem "variable transport delay"-Schema.

Der Totzeitbaustein sollte zyklisch aufgerufen werden. Da er als digitales Element implementiert ist, kann dies eine signifikante Auswirkung auf die Genauigkeit des Ausgangs haben. Der Funktionsbaustein misst die Zeit zwischen den Aufrufen (Abtastzeit) und führt die resultierende Totzeit fließend nach, wenn die Abtastzeit sich wesentlich ändert oder wenn die eingestellte Totzeit sich ändern. Die gemessene Abtastzeit steht auch als Ausgang zur Verfügung.

Schnittstellenbeschreibung

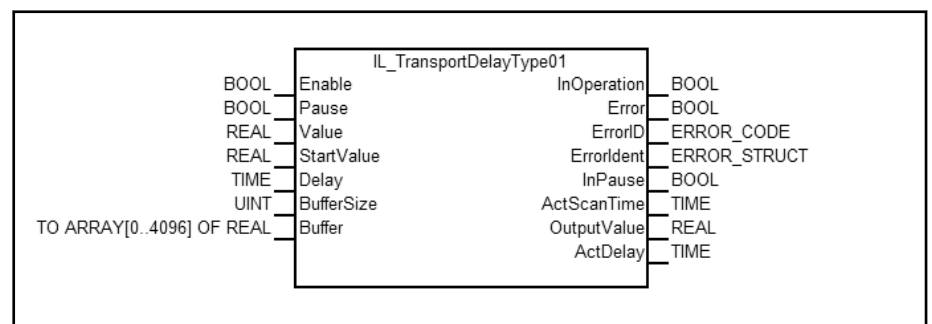


Abb. 7-39: Funktionsbaustein IL_TransportDelayType01

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Bei steigender Flanke wird "OutputValue" voreingestellten Wert von "StartValue" gesetzt. Wenn TRUE, ist der Baustein aktiv.
	Pause	BOOL	Wenn TRUE, wird der Wert von "OutputValue" nicht verändert.
	Value	REAL	Eingangswert, der um "Delay" verzögert wieder ausgegeben werden soll.
	StartValue	REAL	Ein voreingestellter Wert der bei steigender Flanke an "Enable" an "OutputValue" erscheint.
	Delay	TIME	Eingestellte Totzeit
	BufferSize	UINT	Größe des in "Buffer" angegebenen Puffers in Elementen

	Name	Typ	Kommentar
	Buffer	POINTER TO REAL	Puffer zur Zwischenspeicherung der Werte
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, dann arbeitet der Funktionsbaustein und die Ausgänge "OutputValue" und "ActDelay" sind gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes , Seite 395.
	InPause	BOOL	"Pause" ist aktiv und die Wert an "OutputValue" und "ActDelay" sind eingefroren.
	ActScan-Time	TIME	Die vergangene Zeit seit der Funktionsbaustein das letzte Mal aufgerufen wurde.
	OutputValue	REAL	Der um "ActDelay" verzögerte Wert
	ActDelay	TIME	Tatsächliche Verzögerung

Tab. 7-55: Funktionsbaustein IL_TransportDelayType01

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Value	REAL	n.def.	n.def.	0.0	kontinuierlich
StartValue	REAL	n.def.	n.def.	0.0	steigende Flanke von Enable
Delay	TIME	0s	n.def.	0s	kontinuierlich
BufferSize	UINT	0	4096	0	steigende Flanke von Enable
Buffer	POINTER TO REAL			0	steigende Flanke von Enable

Tab. 7-56: Min-, Max- und Default-Werte der Eingänge des IL_TransportDelayType01

Funktionsbeschreibung

Initialisierung

Bei steigender Flanke an "Enable" wird nach Prüfung von "BufferSize" und "Buffer" der Puffer mit dem Wert von "StartValue" gefüllt. "ActDelay" wird mit "Delay" initialisiert. "StartValue" erscheint sofort am Ausgang "OutputValue" und bleibt dort stehen, bis die in "Delay" angegebene Zeitspanne vergangen ist. Erst danach erscheinen die ersten Werte von "Value" am Ausgang "OutputValue".

Normalbetrieb

Bei aktivem "Enable"-Eingang arbeitet der Funktionsbaustein. Bei jedem Aufruf wird die Zeit zum vergangenen Aufruf gemessen und am Ausgang "ActScanTime" ausgegeben. "ActScanTime" wird außerdem für die Berechnung der Pufferverwaltung verwendet.

Der Funktionsbaustein speichert bei jedem Aufruf den Wert an "Value" und gibt diesen um "ActDelay" verzögert wieder an "OutputValue" aus. Sollte die Größe "BufferSize" des Puffers "Buffer" nicht ausreichen, um alle Werte für

die in "Delay" angegebene Zeit zwischenspeichern, so werden Werte ausgelassen. Auf die resultierende Totzeit hat dies keinen Einfluss.

Der Ausgang "ActDelay" gibt die derzeitige Verzögerung des Wertes an "OutputValue" an. Bei Änderung von "Delay" wird "ActDelay" fließend auf den Wert in "Delay" nachgeführt. Wird „Delay“ geändert, während der Baustein aktiv ist, so wird die Totzeit fließend nach dem Schema der variablen Transportverzögerung angepasst. Das Anpassen der Totzeit dauert maximal "BufferSize" * "ActScanTime".

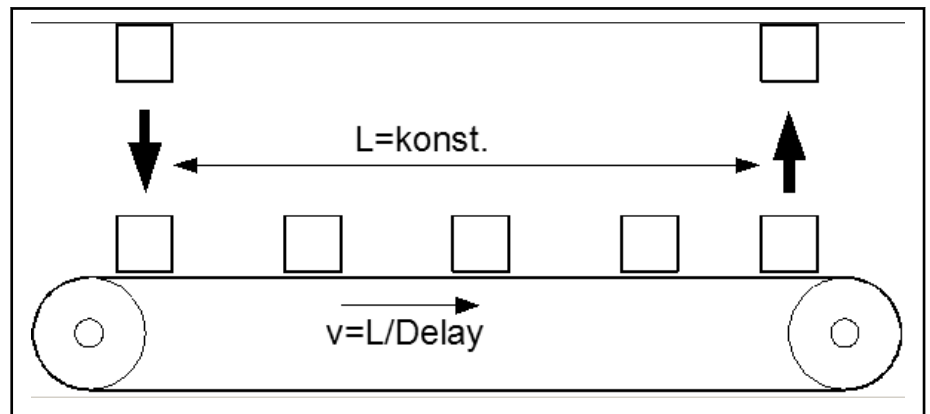


Abb. 7-40: Zur Veranschaulichung der Totzeitanpassung läuft ein Transportband mit variabler Geschwindigkeit. Die Pakete werden an einem festen Punkt auf das Band gelegt und nach einer festen Lauflänge vom Band genommen. Die Geschwindigkeit des Bandes ist so eingestellt, dass jedes Paket die Zeitspanne der Totzeit auf dem Band verweilt. Ändert sich die Totzeit, so wird die Bandgeschwindigkeit angepasst.

Pause

Der Eingang "Pause" wird nur ausgewertet, wenn der "Enable"-Eingang aktiv ist. Der Funktionsbaustein kann über "Pause" angehalten werden. In diesem Fall werden keine weiteren Werte gespeichert und "OutputValue" und "ActDelay" bleiben konstant.

Der Ausgang "InPause" ist gesetzt, solange "Pause" aktiv ist.

Der Ausgang "ActScanTime" wird weiterhin aktualisiert.

Konfiguration

Die Größe des in "Buffer" angelegten Puffers beeinflusst die Genauigkeit der Ausgangskurve. Es ist möglich, an "Buffer" einen kleineren Puffer als 4096 Werte anzulegen. Im Idealfall ist der Puffer größer, als die maximale benötigte Verzögerung geteilt durch die Aufrufzykluszeit des Funktionsbausteins. "BufferSize" kann auch kleiner sein, als die Verzögerung. In diesem Fall steht nicht bei jedem Zyklus ein neuer Wert am Ausgang "OutputValue" an. Die zeitliche Auflösung sinkt.

Notwendige Puffergröße:

"BufferSize" = max("Delay") / benötigte Auflösung

Hinreichende Puffergröße:

"BufferSize" = max("Delay") / "ActScanTime"

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR, 16#0001	16#00000C03	16#00000001	Der Eingang "Delay" ist ungültig
INPUT_RANGE_ERROR, 16#0006	16#00000C03	16#00000002	Die in "Delay" angegebene Verzögerung ist zu groß
INPUT_INVALID_ERROR, 16#0001	16#00000C03	16#00000003	Der Eingang "Buffer" ist nicht belegt
INPUT_RANGE_ERROR, 16#0006	16#00000C03	16#00000004	"BufferSize" ist zu klein

Tab. 7-57: Generierte Fehlernummern des Funktionsbausteins IL_TransportDelayType01

7.5.12 IL_TransportDelayType02

Kurzbeschreibung Dieser Funktionsbaustein implementiert ein eventgesteuertes Totzeitglied mit variabler Totzeit. Die Totzeit kann kontinuierlich verändert werden. Die tatsächliche Totzeit wird fließend an den vorgegebenen Wert angepasst. Die Anpassung an die neue Totzeit erfolgt nach dem "variable transport delay"-Schema.

Der Totzeitbaustein sollte zyklisch aufgerufen werden. Die Schrittweite des "Step"-Eingangs sollte möglichst konstant sein. Der letzte bearbeitete Schritt steht auch als Ausgang zur Verfügung.

Schnittstellenbeschreibung

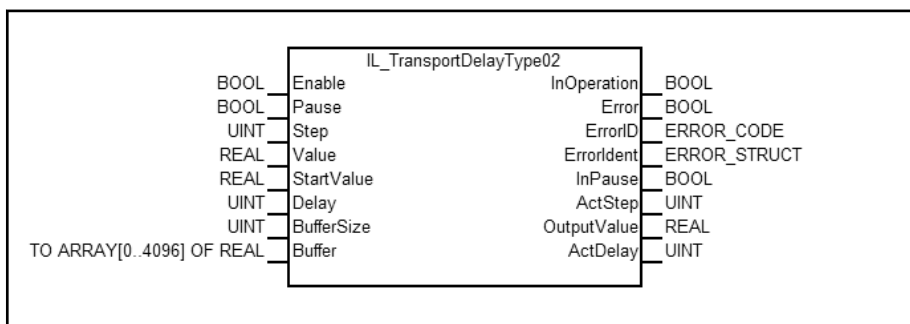


Abb. 7-41: Funktionsbaustein IL_TransportDelayType02

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Bei steigender Flanke wird "OutputValue" voreingestellten Wert von "StartValue" gesetzt. Wenn TRUE, ist der Baustein aktiv.
	Pause	BOOL	Wenn TRUE, wird der Wert von "OutputValue" nicht verändert.
	Step	UINT	Verarbeitungsschritt
	Value	REAL	Eingangswert, der um "Delay" verzögert wieder ausgegeben werden soll.
	StartValue	REAL	Ein voreingestellter Wert der bei steigender Flanke an "Enable" an "OutputValue" erscheint.
	Delay	UINT	Anzahl der zu verzögernden Schritte
	BufferSize	UINT	Größe des in "Buffer" angegebenen Puffers in Elementen
	Buffer	POINTER TO REAL	Puffer zur Zwischenspeicherung der Werte
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, dann arbeitet der Funktionsbaustein und die Ausgänge "OutputValue" und "ActDelay" sind gültig.

	Name	Typ	Kommentar
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes , Seite 398.
	InPause	BOOL	"Pause" ist aktiv und die Wert an "OutputValue" und "ActDelay" sind eingefroren.
	ActStep	UINT	Letzter bearbeiteter Schritt.
	OutputValue	REAL	Der um "ActDelay" verzögerte Wert
	ActDelay	UINT	Tatsächliche Verzögerung in Schritten

Tab. 7-58: Funktionsbaustein IL_TransportDelayType02

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Step	UINT	0	65335	0	kontinuierlich
Value	REAL	n.def.	n.def.	0.0	kontinuierlich
StartValue	REAL	n.def.	n.def.	0.0	steigende Flanke von Enable
Delay	UINT	0	65335	0	kontinuierlich
BufferSize	UINT	0	4096	0	steigende Flanke von Enable
Buffer	POINTER TO REAL			0	steigende Flanke von Enable

Tab. 7-59: Min-, Max- und Default-Werte der Eingänge des IL_TransportDelayType02

Funktionsbeschreibung Initialisierung

Bei steigender Flanke an "Enable" wird nach Prüfung von "BufferSize" und "Buffer" der Puffer mit dem Wert von "StartValue" gefüllt. "ActDelay" wird mit "Delay" initialisiert. "StartValue" erscheint sofort am Ausgang "OutputValue" und bleibt konstant, bis die in "Delay" angegebene Schrittzahl vergangen ist. Erst danach erscheinen die ersten Werte von "Value" am Ausgang "OutputValue".

Normalbetrieb

Bei aktivem "Enable"-Eingang arbeitet der Funktionsbaustein. Bei jedem Aufruf wird "ActStep" auf "Step" aktualisiert. Eine Bearbeitung findet nur statt, wenn sich der "Step"-Eingang ändert. Die Differenz zwischen den Schritten wird außerdem für die Pufferverwaltung verwendet.

Der Funktionsbaustein speichert bei jedem Aufruf den Wert an "Value" und gibt diesen um "ActDelay" Schritte verzögert wieder an "OutputValue" aus. Sollte die Größe "BufferSize" des Puffers "Buffer" nicht ausreichen, um alle Werte für die in "Delay" angegebene Schrittzahl zwischenspeichern, so werden Werte ausgelassen. Auf die resultierende Totzeit hat dies keinen Einfluss.

Der Ausgang "ActDelay" gibt die derzeitige Verzögerung des Wertes an "OutputValue" an. Bei Änderung von "Delay" wird die "ActDelay" fließend auf den Wert in "Delay" nachgeführt. Wird „Delay“ geändert, während der Baustein aktiv ist, so wird die Totzeit fließend nach dem Schema der variablen Transportverzögerung angepasst. Das Anpassen der Totzeit dauert maximal "BufferSize" Schritte.

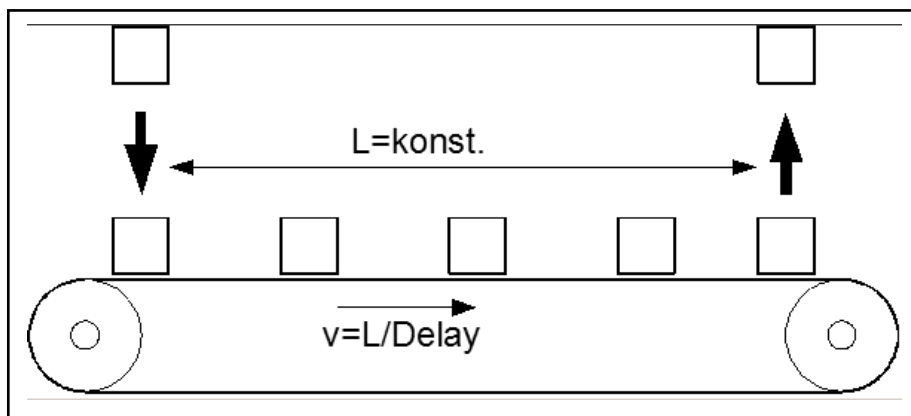


Abb. 7-42: Zur Veranschaulichung der Totzeitanpassung läuft ein Transportband mit variabler Geschwindigkeit. Die Pakete werden an einem festen Punkt auf das Band gelegt und nach einer festen Lauflänge vom Band genommen. Die Geschwindigkeit des Bandes ist so eingestellt, dass jedes Paket die Zeitspanne der Totzeit auf dem Band verweilt. Ändert sich die Totzeit, so wird die Bandgeschwindigkeit angepasst.

Pause

Der Eingang "Pause" wird nur ausgewertet, wenn der "Enable"-Eingang aktiv ist. Der Funktionsbaustein kann über "Pause" angehalten werden. In diesem Fall werden keine weiteren Werte gespeichert und "OutputValue" und "ActDelay" bleiben konstant.

Der Ausgang "InPause" ist gesetzt, solange "Pause" aktiv ist.

Der Ausgang "ActStep" wird weiterhin aktualisiert

Konfiguration

Die Größe des in "Buffer" angelegten Puffers beeinflusst die Genauigkeit der Ausgangskurve. Es ist möglich, an "Buffer" einen kleineren Puffer als 4096 Werte anzulegen. Im Idealfall ist der Puffer größer, als die maximale benötigte Verzögerung in Schritten. "BufferSize" kann auch kleiner sein, als die Verzögerung. In diesem Fall steht nicht bei jedem Zyklus ein neuer Wert am Ausgang "OutputValue" an. Die Auflösung, bezogen auf die Schritte, sinkt.

Notwendige Puffergröße:

"BufferSize" = max("Delay") / benötigte Auflösung

Hinreichende Puffergröße:

"BufferSize" = max("Delay")

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR, 16#0001	16#00000C03	16#00000001	Der Eingang "Delay" ist ungültig
INPUT_RANGE_ERROR, 16#0006	16#00000C03	16#00000002	Die in "Delay" angegebene Verzögerung ist zu groß

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR, 16#0001	16#00000C03	16#00000003	Der Eingang "Buffer" ist nicht belegt
INPUT_RANGE_ERROR, 16#0006	16#00000C03	16#00000004	"BufferSize" ist zu klein

Tab. 7-60: Generierte Fehlernummern des Funktionsbausteins IL_TransportDelayType02

7.5.13 IL_ValueLimit

Kurzbeschreibung

Dieser Funktionsbaustein begrenzt einen Wert auf den Bereich innerhalb der Schranken.

Schnittstellenbeschreibung

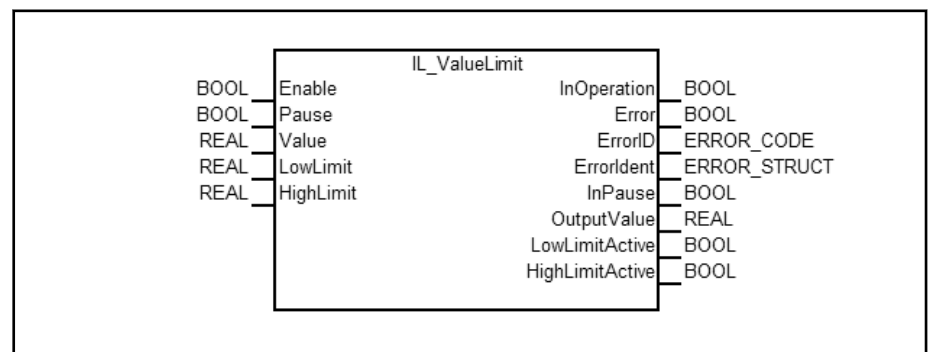


Abb. 7-43: Funktionsbaustein IL_ValueLimit

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Wenn TRUE, wird "Value" auf die Grenzen "LowWindow" und "HighWindow" hin geprüft und die Ausgänge "LowWindowActive" und "HighWindowActive" entsprechend gesetzt.
	Pause	BOOL	Wenn TRUE, bleiben "InWindow", "LowWindowActive" und "HighWindowActive" konstant.
	Value	REAL	Der aktuelle Istwert des Prozesses.
	LowLimit	REAL	Unterer Grenzwert
	HighLimit	REAL	Oberer Grenzwert
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, dann arbeitet der Funktionsbaustein und die Ausgänge "InWindow", "LowWindowActive" und "HighWindowActive" sind gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes, Seite 400 .
	InPause	BOOL	"Pause" ist aktiv und die Werte an "InWindow", "LowWindowActive" und "HighWindowActive" sind eingefroren.
	OutputValue	REAL	Auf den Bereich zwischen "LowLimit" und "HighLimit" begrenzter Wert.

	Name	Typ	Kommentar
	LowLimitActive	BOOL	Aktiv, wenn "Value" $\leq$ "LowLimit"
	HighLimitActive	BOOL	Aktiv, wenn "Value" $\geq$ "HighLimit"

Tab. 7-61: Funktionsbaustein IL_ValueLimit

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Value	REAL	n. def.	n.def.	0.0	kontinuierlich
LowLimit	REAL	n.def.	< HighLimit	-10.0	kontinuierlich
HighLimit	REAL	> LowLimit	n.def.	10.0	kontinuierlich

Tab. 7-62: Min-, Max- und Default-Werte der Eingänge des IL_ValueLimit

Funktionsbeschreibung

Der Funktionsbaustein IL_ValueLimit vergleicht den Wert des Eingangs „Value“ mit den in „LowLimit“ und „HighLimit“ angegebenen Grenzwerten und gibt den begrenzten Wert von "Value" an "OutputValue" aus. Die Überprüfung erfolgt kontinuierlich bei gesetztem "Enable"-Eingang. Alle Eingänge werden kontinuierlich behandelt.

LowLimitActive, HighLimitActive

Die Ausgänge "LowLimitActive" und "HighLimitActive" werden wie folgt gesetzt:

- "LowLimitActive" wird gesetzt, wenn "Value" kleiner gleich "LowLimit" ist.
- "HighLimitActive" wird gesetzt, wenn "Value" größer gleich "HighLimit" ist.

Pause, InPause

Der Eingang "Pause" wird nur ausgewertet, solange der "Enable"-Eingang aktiv ist. Ist "Pause" gesetzt, wird kein Vergleich durchgeführt und die Ausgänge "OutputValue", "LowLimitActive" und "HighLimitActive" nicht verändert.

Der Ausgang "InPause" ist gesetzt, solange "Pause" aktiv ist.

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR, 16#0001	16#00000C01	16#00000001	LowLimit ist größer als HighLimit.
STATE_MACHINE_ERROR, 16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.

Tab. 7-63: Generierte Fehlernummern des Funktionsbausteins IL_ValueLimit

7.5.14 IL_ValueWindow

Kurzbeschreibung

Dieser Funktionsbaustein prüft, ob ein Wert innerhalb der Schranken liegt und gibt das Ergebnis der Prüfung aus. Der Wert wird nicht verändert.

Schnittstellenbeschreibung

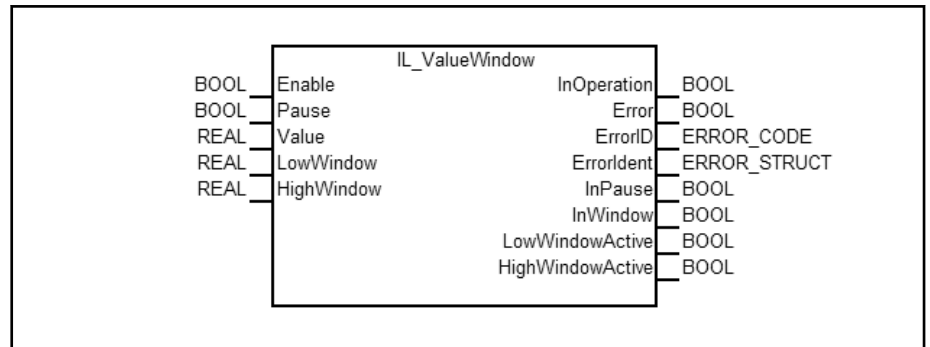


Abb. 7-44: Funktionsbaustein IL_ValueWindow

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Wenn TRUE, wird "Value" auf auf die Grenzen "LowWindow" und "HighWindow" hin geprüft und die Ausgänge "LowWindowActive" und "HighWindowActive" entsprechend gesetzt.
	Pause	BOOL	Wenn TRUE, bleiben "InWindow", "LowWindowActive" und "HighWindowActive" konstant.
	Value	REAL	Der aktuelle Istwert des Prozesses.
	LowWindow	REAL	Unterer Grenzwert
	HighWindow	REAL	Oberer Grenzwert
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, dann arbeitet der Funktionsbaustein und die Ausgänge "InWindow", "LowWindowActive" und "HighWindowActive" sind gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes, Seite 402 .
	InPause	BOOL	"Pause" ist aktiv und die Werte an "InWindow", "LowWindowActive" und "HighWindowActive" sind eingefroren.
	InWindow	BOOL	Aktiv, wenn "LowWindow" < "Value" < "HighWindow"
	LowWindowActive	BOOL	Aktiv, wenn "Value" ≤ "LowWindow"
	HighWindowActive	BOOL	Aktiv, wenn "Value" ≥ "HighWindow"

Tab. 7-64: Funktionsbaustein IL_ValueWindow

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Value	REAL	n. def.	n.def.	0.0	kontinuierlich
LowWindow	REAL	n.def.	< HighWindow	-10.0	kontinuierlich
HighWindow	REAL	> LowWindow	n.def.	10.0	kontinuierlich

Tab. 7-65: Min-, Max- und Default-Werte der Eingänge des IL_ValueWindow

Funktionsbeschreibung

Der Funktionsbaustein IL_ValueWindow vergleicht den Wert des Eingangs „Value“ mit den in „LowWindow“ und „HighWindow“ angegebenen Grenzwerten. Die Überprüfung erfolgt kontinuierlich bei gesetztem "Enable"-Eingang. Alle Eingänge werden kontinuierlich behandelt.

InWindow, LowWindowActive, HighWindowActive

Die Ausgänge "InWindow", "LowWindowActive" und "HighWindowActive" werden wie folgt gesetzt:

- "InWindow" wird gesetzt, wenn "Value" größer als "LowWindow" und kleiner als "HighWindow" ist.
- "LowWindowActive" wird gesetzt, wenn "Value" kleiner gleich "LowWindow" ist.
- "HighWindowActive" wird gesetzt, wenn "Value" größer gleich "HighWindow" ist.

Pause, InPause

Der Eingang "Pause" wird nur ausgewertet, solange der "Enable"-Eingang aktiv ist. Ist "Pause" gesetzt, wird kein Vergleich durchgeführt und die Ausgänge "InWindow", "LowWindowActive" und "HighWindowActive" nicht verändert.

Der Ausgang "InPause" ist gesetzt, solange "Pause" aktiv ist.

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR, 16#0001	16#00000C08	16#00000001	LowWindow ist größer als HighWindow.
STATE_MACHINE_ERROR, 16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.

Tab. 7-66: Generierte Fehlernummern des Funktionsbausteins IL_ValueWindow

7.6 Filter (Filter)

7.6.1 Einführung und Übersicht

Die RIL_LoopControl stellt die folgenden Filter-Funktionsbausteine zur Verfügung:

zeitdiskret	eventgesteuert	Beschreibung
IL_FIRType01, Seite 403	IL_FIRType02, Seite 405	Digitales FIR-Filter für Hoch-, Tief- und Bandpass-Verwendung
IL_IIRType01, Seite 407	IL_IIRType02, Seite 409	Digitales IIR-Filter für Hoch-, Tief- und Bandpass-Verwendung
IL_StateVariableType01, Seite 412	IL_StateVariableType02, Seite 414	IIR-Filter (Infinite Impulse Response) zweiter Ordnung für Hoch-, Tief- und Bandpass-Verwendung, sowie zum Einsatz als Bandsperre

Tab. 7-67: Filter der RIL_LoopControl

7.6.2 IL_FIRType01

Kurzbeschreibung

Der Funktionsbaustein IL_FIRType01 implementiert einen allgemeinen zeitdiskreten FIR-Filter. Dieser digitale Filter lässt sich für verschiedene Hoch-, Tief- und Bandpässe einsetzen. Der Funktionsbaustein arbeitet zeitdiskret.

Schnittstellenbeschreibung

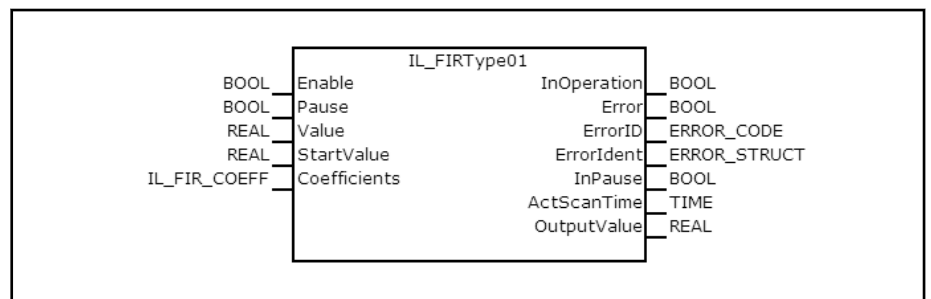


Abb. 7-45: Funktionsbaustein IL_FIRType01

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Wenn TRUE, wird "OutputValue" auf den mit "StartValue" berechneten Wert gesetzt.
	Pause	BOOL	aktiviert den Pause-Modus wenn TRUE
	Value	REAL	Der aktuelle Istwert des Prozesses.
	StartValue	REAL	Startwert, der bei steigender Flanke an "Enable" am Ausgang "OutputValue" angenommen wird.
	Coefficients	IL_FIR_COEFF	Struktur mit Filterkoeffizienten IL_FIR_COEFF, Seite 417
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, dann arbeitet der Filter und der Ausgang "OutputValue" ist gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlererkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes, Seite 405
	InPause	BOOL	"Pause" ist aktiv und der Wert an "OutputValue" ist eingefroren.

	Name	Typ	Kommentar
	ActScan-Time	TIME	Die vergangene Zeit seit dem letzten Aufruf des Funktionsbausteins.
	OutputValue	REAL	Der berechnete Filterwert

Tab. 7-68: Funktionsbaustein IL_FIRType01

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Value	REAL	n. def.	n.def.	0.0	kontinuierlich
StartValue	REAL	n.def.	n.def.	0.0	steigende Flanke von Enable
Coefficients	IL_FIR_COEFF				steigende Flanke von Enable

Tab. 7-69: Min-, Max- und Default-Werte der Eingänge des IL_FIRType01

Funktionsbeschreibung

Der Funktionsbaustein IL_FIRType01 implementiert einen allgemeinen zeitdiskreten FIR-Filter. Dieser digitale Filter lässt sich für verschiedene Hoch-, Tief- und Bandpässe einsetzen. Der Funktionsbaustein arbeitet zeitdiskret.

Value, StartValue, OutputValue

Der Eingang "Value" enthält den Eingangswert des Filters. Dieser wird bei jedem Aufruf verarbeitet, wenn "InOperation" gesetzt ist. Bei steigender Flanke an "Enable" wird der Wert in "StartValue" in alle internen Register übernommen. Wenn "Enable" gesetzt und "InPause" nicht gesetzt ist, wird das Ergebnis am Ausgang "OutputValue" ausgegeben.

Coefficients

Über den Eingang "Coefficients" wird das Verhalten des Filters definiert. Diese Datenstruktur besteht aus dem Eintrag "NumberOfCoeffs" für die Anzahl der Koeffizienten und dem Array "Coeffs" für die einzelnen Koeffizienten a_0 bis a_{15} .

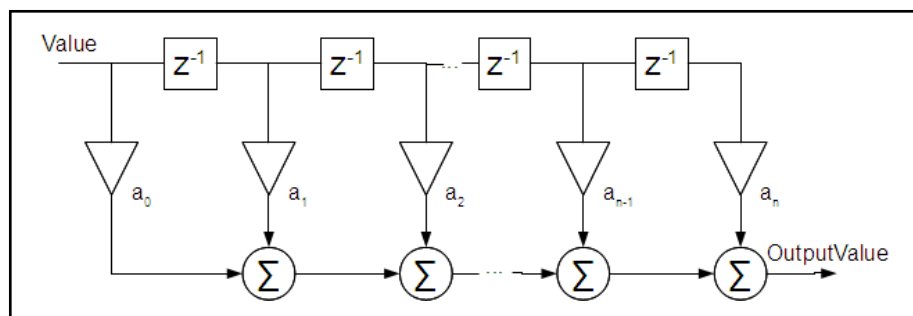


Abb. 7-46: Schema eines FIR-Filters

Pause, InPause

Der Eingang "Pause" wird nur ausgewertet, solange der "Enable"-Eingang aktiv ist. Ist "Pause" TRUE, wird keine Berechnung durchgeführt und der Wert von "OutputValue" nicht verändert. Die internen Register werden ebenfalls eingefroren.

Der Ausgang "InPause" ist gesetzt, solange "Pause" aktiv ist.

Der Ausgang "ActScanTime" wird weiterhin aktualisiert.

Zeitverhalten Der Funktionsbaustein IL_FIRType01 arbeitet asynchron. Nach dem Start durch eine steigende Flanke am Eingang "Enable" liest er den Eingang "Coefficients" und die internen Register werden mit dem "StartValue" geladen. Der Funktionsbaustein liest bei jedem Aufruf einen Wert vom Eingang "Value" und berechnet einen neuen Wert am Ausgang "OutputValue". Bei aktivem Filter ist der Ausgang "InOperation" gesetzt. Tritt ein Fehler auf, wird der Ausgang "Error" gesetzt.

Fehlercodes Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR, 16#0006	16#00000C0C	16#00000001	Der Eintrag "NumberOfCoeffs" des Eingangs "Coefficients" ist ungültig.
STATE_MACHINE_ERROR, 16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.

Tab. 7-70: Generierte Fehlernummern des Funktionsbausteins IL_FIRType01

7.6.3 IL_FIRType02

Kurzbeschreibung Der Funktionsbaustein IL_FIRType02 implementiert einen allgemeinen event-gesteuerten FIR-Filter. Dieser digitale Filter lässt sich für verschiedene Hoch-, Tief- und Bandpässe einsetzen. Dieser Funktionsbaustein arbeitet event-gesteuert, d.h. bei Änderung am Eingang "Step" wird ein neuer Wert verarbeitet.

Schnittstellenbeschreibung

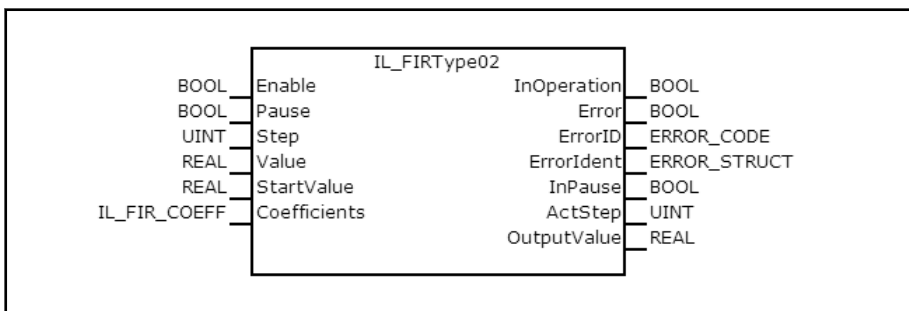


Abb. 7-47: Funktionsbaustein IL_FIRType02

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Wenn TRUE, wird "OutputValue" auf den mit "StartValue" berechneten Wert gesetzt.
	Pause	BOOL	aktiviert den Pause-Modus wenn TRUE
	Step	UINT	Bearbeitungsschritt, eine Änderung löst die Verarbeitung eines neuen Signalwerts aus
	Value	REAL	Der aktuelle Istwert des Prozesses.
	StartValue	REAL	Startwert, der bei steigender Flanke an "Enable" am Ausgang "OutputValue" angenommen wird.
	Coefficients	IL_FIR_COEFF	Struktur mit Filterkoeffizienten IL_FIR_COEFF, Seite 417
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, dann arbeitet der Filter und der Ausgang "OutputValue" ist gültig.

	Name	Typ	Kommentar
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes , Seite 407.
	InPause	BOOL	"Pause" ist aktiv und der Wert an "OutputValue" ist eingefroren.
	ActStep	UINT	Zuletzt bearbeiteter Schritt
	OutputValue	REAL	Der berechnete Filterwert

Tab. 7-71: Funktionsbaustein IL_FIRType02

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Step	UINT	0	65535	0	kontinuierlich
Value	REAL	n. def.	n.def.	0.0	kontinuierlich
StartValue	REAL	n.def.	n.def.	0.0	steigende Flanke von Enable
Coefficients	IL_FIR_COEFF				steigende Flanke von Enable

Tab. 7-72: Min-, Max- und Default-Werte der Eingänge des IL_FIRType02

Funktionsbeschreibung

Der Funktionsbaustein IL_FIRType02 implementiert einen allgemeinen event-gesteuerten FIR-Filter. Dieser digitale Filter lässt sich für verschiedene Hoch-, Tief- und Bandpässe einsetzen. Dieser Funktionsbaustein arbeitet event-gesteuert, d.h. bei Änderung am Eingang "Step" wird ein neuer Wert verarbeitet. Dabei ist es unerheblich, um welchen Betrag sich der Wert am "Step"-Eingang ändert, es wird jedesmal genau ein Filter-Rechenschritt (Iteration) durchgeführt.

Value, StartValue, OutputValue

Der Eingang "Value" enthält den Eingangswert des Filters. Dieser wird bei jedem Aufruf verarbeitet, wenn "InOperation" gesetzt ist. Bei steigender Flanke an "Enable" wird der Wert in "StartValue" in alle internen Register übernommen. Wenn "Enable" gesetzt und "InPause" nicht gesetzt ist, wird das Ergebnis am Ausgang "OutputValue" ausgegeben.

Coefficients

Über den Eingang "Coefficients" wird das Verhalten des Filters definiert. Diese Datenstruktur besteht aus dem Eintrag "NumberOfCoeffs" für die Anzahl der Koeffizienten und dem Array "Coeffs" für die einzelnen Koeffizienten a_0 bis a_{15} .

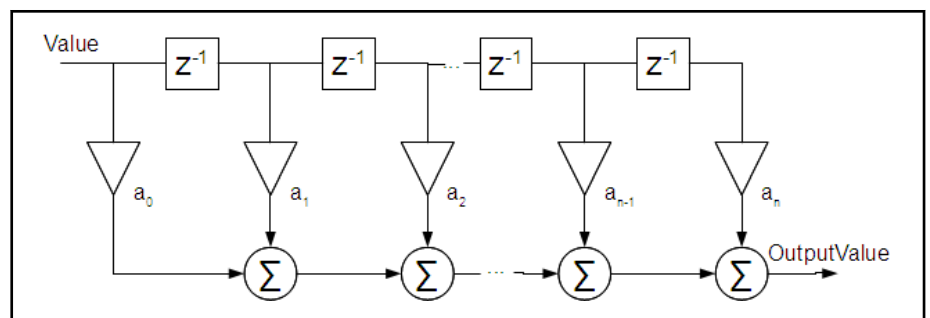


Abb. 7-48: Schema eines FIR-Filters

Pause, InPause

Der Eingang "Pause" wird nur ausgewertet, solange der "Enable"-Eingang aktiv ist. Ist "Pause" TRUE, wird keine Berechnung durchgeführt und der Wert von "OutputValue" nicht verändert. Die internen Register werden ebenfalls eingefroren.

Der Ausgang "InPause" ist gesetzt, solange "Pause" aktiv ist.

Der Ausgang "ActStep" wird weiterhin aktualisiert.

Zeitverhalten

Der Funktionsbaustein IL_FIRType02 arbeitet asynchron. Nach dem Start durch eine steigende Flanke am Eingang "Enable" liest er den Eingang "Coefficients". Der Funktionsbaustein liest bei jeder Änderung am Eingang "Step" einen Wert vom Eingang "Value" und berechnet einen neuen Wert am Ausgang "OutputValue". Bei aktiver Signalgenerierung ist der Ausgang "In-Operation" gesetzt. Tritt ein Fehler auf, wird der Ausgang "Error" gesetzt.

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR, 16#0006	16#00000C0C	16#00000001	Der Eintrag "NumberOfCoeffs" des Eingangs "Coefficients" ist ungültig.
STATE_MACHINE_ERROR, 16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.

Tab. 7-73: Generierte Fehlernummern des Funktionsbausteins IL_FIRType02

7.6.4 IL_IIRType01

Kurzbeschreibung

Der Funktionsbaustein IL_IIRType01 implementiert einen allgemeinen zeitdiskreten IIR-Filter (Infinite Impulse Response). Dieser digitale Filter lässt sich u.a. für Chebyshev-, Butterworth- und Bessel-Filter einsetzen. Der Funktionsbaustein arbeitet zeitdiskret.

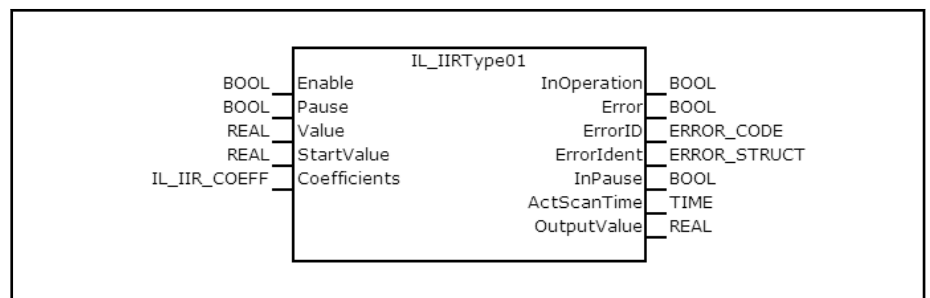
Schnittstellenbeschreibung

Abb. 7-49: Funktionsbaustein IL_IIRType01

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Wenn TRUE, wird "OutputValue" auf den mit "StartValue" berechneten Wert gesetzt.
	Pause	BOOL	aktiviert den Pause-Modus wenn TRUE
	Value	REAL	Der aktuelle Istwert des Prozesses.
	StartValue	REAL	Startwert, der bei steigender Flanke an "Enable" am Ausgang "OutputValue" angenommen wird.
	Coefficients	IL_IIR_COEFF	Struktur mit Filterkoeffizienten IL_IIR_COEFF, Seite 417.
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, dann arbeitet der Filter und der Ausgang "OutputValue" ist gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes, Seite 409.
	InPause	BOOL	"Pause" ist aktiv und der Wert an "OutputValue" ist eingefroren.
	ActScan-Time	TIME	Die vergangene Zeit seit dem letzten Aufruf des Funktionsbausteins.
	OutputValue	REAL	Der berechnete Filterwert

Tab. 7-74: Funktionsbaustein IL_IIRType01

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Value	REAL	n. def.	n.def.	0.0	kontinuierlich
StartValue	REAL	n.def.	n.def.	0.0	steigende Flanke von Enable
Coefficients	IL_IIR_COEFF				steigende Flanke von Enable

Tab. 7-75: Min-, Max- und Default-Werte der Eingänge des IL_IIRType01

Funktionsbeschreibung

Der Funktionsbaustein IL_IIRType01 implementiert einen allgemeinen zeitdiskreten IIR-Filter. Dieser digitale Filter lässt sich u.a. für Chebyshev-, Butterworth- und Bessel-Filter einsetzen. Der Funktionsbaustein arbeitet zeitdiskret.

Value, StartValue, OutputValue

Der Eingang "Value" enthält den Eingangswert des Filters. Dieser wird bei jedem Aufruf verarbeitet, wenn "InOperation" gesetzt ist. Bei steigender Flanke an "Enable" wird der Wert in "StartValue" in alle internen Register übernommen. Wenn "Enable" gesetzt und "InPause" nicht gesetzt ist, wird das Ergebnis am Ausgang "OutputValue" ausgegeben.

Coefficients

Über den Eingang "Coefficients" wird das Verhalten des Filters definiert. Diese Datenstruktur besteht aus dem Eintrag "NumberOfCoeffs" für die Anzahl

der Koeffizienten und den Arrays "FowardCoeffs" a_0 bis a_{15} und "Backward-Coeffs" b_1 bis b_{15} für die einzelnen Koeffizienten. Das Array "ForwardCoeffs" enthält einen Eintrag mehr als das Array "BackwardCoeffs".

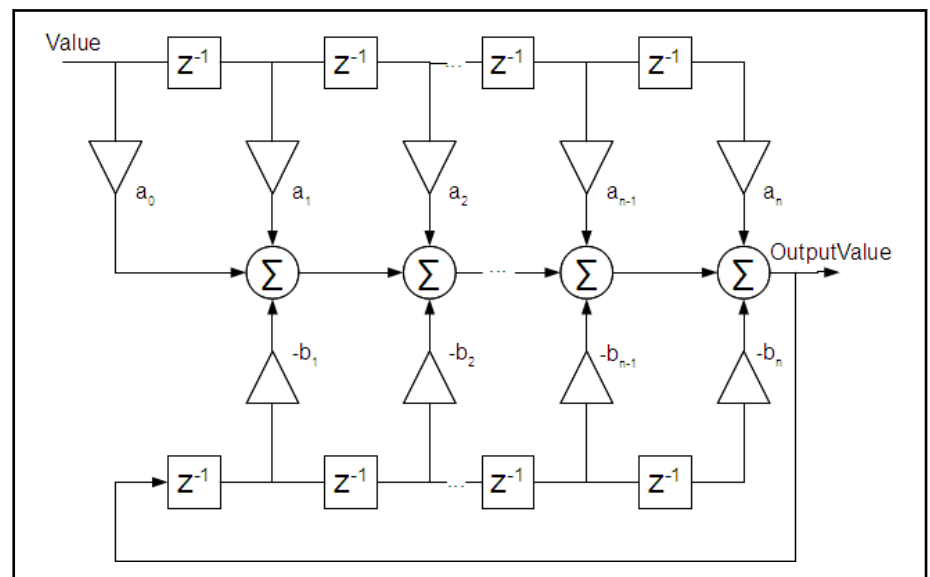


Abb. 7-50: Schema eines IIR-Filters

Pause, InPause

Der Eingang "Pause" wird nur ausgewertet, solange der "Enable"-Eingang aktiv ist. Ist "Pause" TRUE, wird keine Berechnung durchgeführt und der Wert von "OutputValue" nicht verändert. Die internen Register werden ebenfalls eingefroren.

Der Ausgang "InPause" ist gesetzt, solange "Pause" aktiv ist.

Der Ausgang "ActScanTime" wird weiterhin aktualisiert.

Zeitverhalten Der Funktionsbaustein IL_IIRType01 arbeitet asynchron. Nach dem Start durch eine steigende Flanke am Eingang "Enable" liest er den Eingang "Coefficients" und die internen Register werden mit dem "StartValue" geladen. Der Funktionsbaustein liest bei jedem Aufruf einen Wert vom Eingang "Value" und berechnet einen neuen Wert am Ausgang "OutputValue". Bei aktivem Filter ist der Ausgang "InOperation" gesetzt. Tritt ein Fehler auf, wird der Ausgang "Error" gesetzt.

Fehlercodes Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR, 16#0006	16#00000C0C	16#00000001	Der Eintrag "NumberOfCoeffs" des Eingangs "Coefficients" ist ungültig.
STATE_MACHINE_ERROR, 16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.

Tab. 7-76: Generierte Fehlernummern des Funktionsbausteins IL_IIRType01

7.6.5 IL_IIRType02

Kurzbeschreibung

Der Funktionsbaustein IL_IIRType02 implementiert einen allgemeinen event-gesteuerten IIR-Filter. Dieser digitale Filter lässt sich für verschiedene Hoch-, Tief- und Bandpässe einsetzen. Dieser Funktionsbaustein arbeitet event-ge-

steuert, d.h. bei Änderung am Eingang "Step" wird ein neuer Wert verarbeitet.

Schnittstellenbeschreibung

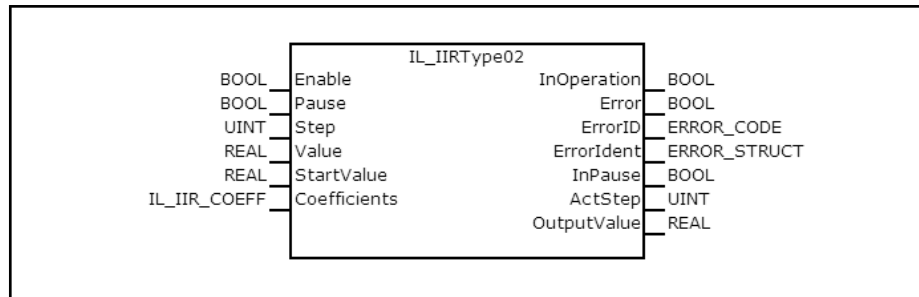


Abb. 7-51: Funktionsbaustein IL_IIRType02

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Wenn TRUE, wird "OutputValue" auf den mit "StartValue" berechneten Wert gesetzt.
	Pause	BOOL	aktiviert den Pause-Modus wenn TRUE
	Step	UINT	Bearbeitungsschritt, eine Änderung löst die Verarbeitung eines neuen Signalwerts aus
	Value	REAL	Der aktuelle Istwert des Prozesses.
	StartValue	REAL	Startwert, der bei steigender Flanke an "Enable" am Ausgang "OutputValue" angenommen wird.
	Coefficients	IL_IIR_COEFF	Struktur mit Filterkoeffizienten IL_IIR_COEFF, Seite 417.
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, dann arbeitet der Filter und der Ausgang "OutputValue" ist gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes, Seite 412.
	InPause	BOOL	"Pause" ist aktiv und der Wert an "OutputValue" ist eingefroren.
	ActStep	UINT	Zuletzt bearbeiteter Schritt
	OutputValue	REAL	Der berechnete Filterwert

Tab. 7-77: Funktionsbaustein IL_IIRType02

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Step	UINT			0	kontinuierlich
Value	REAL	n. def.	n.def.	0.0	kontinuierlich

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
StartValue	REAL	n.def.	n.def.	0.0	steigende Flanke von Enable
Coefficients	IL_IIR_COEFF				steigende Flanke von Enable

Tab. 7-78: Min-, Max- und Default-Werte der Eingänge des IL_IIRType02

Funktionsbeschreibung

Der Funktionsbaustein IL_IIRType02 implementiert einen allgemeinen event-gesteuerten IIR-Filter. Dieser digitale Filter lässt sich für verschiedene Hoch-, Tief- und Bandpässe einsetzen. Dieser Funktionsbaustein arbeitet event-gesteuert, d.h. bei Änderung am Eingang "Step" wird ein neuer Wert verarbeitet. Dabei ist es unerheblich, um welchen Betrag sich der Wert am "Step"-Eingang ändert, es wird jedesmal genau ein Filter-Rechenschritt (Iteration) durchgeführt.

Value, StartValue, OutputValue

Der Eingang "Value" enthält den Eingangswert des Filters. Dieser wird bei jedem Aufruf verarbeitet, wenn "InOperation" gesetzt ist. Bei steigender Flanke an "Enable" wird der Wert in "StartValue" in alle internen Register übernommen. Wenn "Enable" gesetzt und "InPause" nicht gesetzt ist, wird das Ergebnis am Ausgang "OutputValue" ausgegeben.

Coefficients

Über den Eingang "Coefficients" wird das Verhalten des Filters definiert. Diese Datenstruktur besteht aus dem Eintrag "NumberOfCoeffs" für die Anzahl der Koeffizienten und den Arrays "ForwardCoeffs" a_0 bis a_{15} und "BackwardCoeffs" b_1 bis b_{15} für die einzelnen Koeffizienten. Das Array "ForwardCoeffs" enthält einen Eintrag mehr als das Array "BackwardCoeffs".

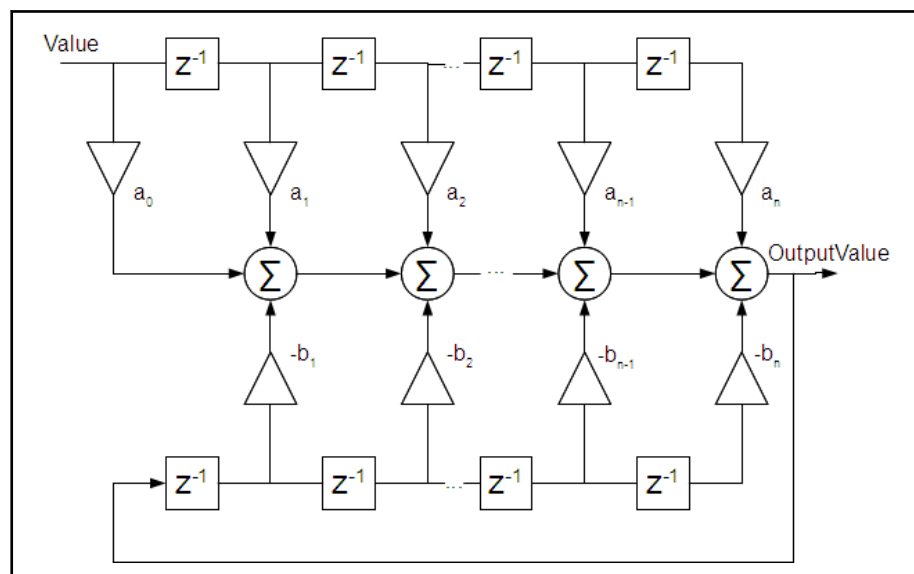


Abb. 7-52: Schema eines IIR-Filters

Pause, InPause

Der Eingang "Pause" wird nur ausgewertet, solange der "Enable"-Eingang aktiv ist. Ist "Pause" TRUE, wird keine Berechnung durchgeführt und der Wert von "OutputValue" nicht verändert. Die internen Register werden ebenfalls eingefroren.

Der Ausgang "InPause" ist gesetzt, solange "Pause" aktiv ist.

Der Ausgang "ActStep" wird weiterhin aktualisiert.

Zeitverhalten Der Funktionsbaustein IL_IIRType02 arbeitet asynchron. Nach dem Start durch eine steigende Flanke am Eingang "Enable" liest er den Eingang "Coefficients" und die internen Register werden mit dem "StartValue" geladen. Der Funktionsbaustein liest bei jedem Aufruf einen Wert vom Eingang "Value" und berechnet einen neuen Wert am Ausgang "OutputValue". Bei aktivem Filter ist der Ausgang "InOperation" gesetzt. Tritt ein Fehler auf, wird der Ausgang "Error" gesetzt.

Fehlercodes Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR, 16#0006	16#00000C0C	16#00000001	Der Eintrag "NumberOfCoeffs" des Eingangs "Coefficients" ist ungültig.
STATE_MACHINE_ERROR, 16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.

Tab. 7-79: Generierte Fehlernummern des Funktionsbausteins IL_IIRType02

7.6.6 IL_StateVariableType01

Kurzbeschreibung Der Funktionsbaustein IL_StateVariableType01 implementiert einen speziellen zeitdiskreten IIR-Filter (Infinite Impulse Response) zweiter Ordnung. Dieser digitale Filter lässt sich als Hochpass, Tiefpass, Bandpass oder Bandsperrfilter einsetzen. Der Funktionsbaustein arbeitet zeitdiskret.

Schnittstellenbeschreibung

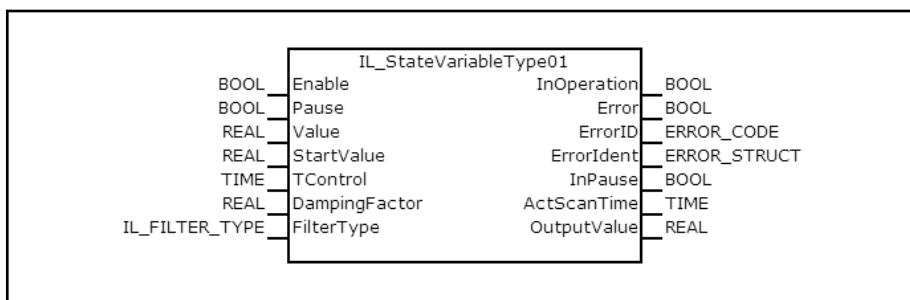


Abb. 7-53: Funktionsbaustein IL_StateVariableType01

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Wenn TRUE, wird "OutputValue" auf den mit "StartValue" berechneten Wert gesetzt.
	Pause	BOOL	aktiviert den Pause-Modus wenn TRUE
	Value	REAL	Der aktuelle Istwert des Prozesses.
	StartValue	REAL	Startwert, der bei steigender Flanke an "Enable" am Ausgang "OutputValue" angenommen wird.
	TControl	TIME	Zeitkonstante
	Damping-Factor	REAL	Dämpfungsfaktor
	FilterType	IL_FILTER_TYPE	Filterart IL_FILTER_TYPE , Seite 417.
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, dann arbeitet der Filter und der Ausgang "OutputValue" ist gültig.

	Name	Typ	Kommentar
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes, Seite 414 .
	InPause	BOOL	"Pause" ist aktiv und der Wert an "OutputValue" ist eingefroren.
	ActScan-Time	TIME	Die vergangene Zeit seit dem letzten Aufruf des Funktionsbausteins.
	OutputValue	REAL	Der berechnete Filterwert

Tab. 7-80: Funktionsbaustein IL_StateVariableType01

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Value	REAL	n. def.	n.def.	0.0	kontinuierlich
StartValue	REAL	n.def.	n.def.	0.0	steigende Flanke von Enable
TControl	TIME	>0s	n.def.	1ms	kontinuierlich
DampingFactor	REAL	0.0		1.0	kontinuierlich
FilterType	IL_FILTER_TYPE			IL_LOWPASS	steigende Flanke von Enable

Tab. 7-81: Min-, Max- und Default-Werte der Eingänge des IL_StateVariableType01

Funktionsbeschreibung

Der Funktionsbaustein IL_StateVariableType01 implementiert einen speziellen zeitdiskreten IIR-Filter (Infinite Impulse Response) zweiter Ordnung. Dieser digitale Filter lässt sich als Hochpass, Tiefpass, Bandpass oder Bandsperrfilter einsetzen. Der Funktionsbaustein arbeitet zeitdiskret.

Value, StartValue, OutputValue

Der Eingang „Value“ enthält den Eingangswert des Filters. Dieser wird bei jedem Aufruf verarbeitet, wenn „InOperation“ gesetzt ist. Bei steigender Flanke an „Enable“ wird der Wert in „StartValue“ in alle internen Register übernommen. Wenn „Enable“ gesetzt und „InPause“ nicht gesetzt ist, wird das Ergebnis am Ausgang „OutputValue“ ausgegeben.

TControl, DampingFactor, FilterType

Der Eingang „FilterType“ gibt die Art des zu berechnenden Filters an. Es stehen die folgenden Filterarten zur Verfügung: IL_LOWPASS, IL_HIGHPASS, IL_BANDPASS, IL_BANDREJECT.

Je nach Filterart werden die Eingänge „TControl“ und „DampingFactor“ interpretiert:

FilterType	1/TControl	DampingFactor
IL_LOWPASS	Grenzfrequenz oberhalb der Frequenzen unterdrückt werden	DampingFactor<1: Überschwingen, DampingFactor=1: kritische Dämpfung
IL_HIGHPASS	Grenzfrequenz unterhalb der Frequenzen unterdrückt werden	DampingFactor>1: kein Überschwingen Amplitude bei Grenzfrequenz=Dampingfactor ^{0.5}
IL_BANDPASS	Mittenfrequenz des Bandpasses	Amplitude bei Mittenfrequenz=Dampingfactor ^{0.5}
IL_BANDREJECT	Mittenfrequenz der Bandsperre	Grenzfrequenz=Mittenfrequenz*(1±Damping-Factor/2)

Tab. 7-82: Interpretation der Eingangswerte nach eingestellter Filterart

Pause

Der Eingang "Pause" wird nur ausgewertet, solange der "Enable"-Eingang aktiv ist. Ist "Pause" TRUE, wird keine Berechnung durchgeführt und der Wert von "OutputValue" nicht verändert. Der interne Integrator ist ebenfalls eingefroren.

Der Ausgang "InPause" ist gesetzt, solange "Pause" aktiv ist.

Der Ausgang "ActScanTime" wird weiterhin aktualisiert.

Zeitverhalten Der Funktionsbaustein IL_StateVariableType01 arbeitet asynchron. Nach dem Start durch eine steigende Flanke am Eingang "Enable" liest er den Eingang "StartValue". Der Funktionsbaustein liest bei jedem Aufruf einen Wert vom Eingang "Value", "TControl" und "DampingFactor" und berechnet einen neuen Wert am Ausgang "OutputValue". Bei aktivem Filter ist der Ausgang "InOperation" gesetzt. Tritt ein Fehler auf, wird der Ausgang "Error" gesetzt.

Fehlercodes Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR, 16#0006	16#00000C02	16#00000001	Der Eingang "TControl" ist ungültig.
INPUT_RANGE_ERROR, 16#0006	16#00000C02	16#00000008	Der Eingang "DampingFactor" ist ungültig.
INPUT_RANGE_ERROR, 16#0006	16#00000C02	16#00000002	Der im Eingang "FilterType" angegebene Wert ist ungültig
STATE_MACHINE_ERROR, 16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.

Tab. 7-83: Generierte Fehlernummern des Funktionsbausteins IL_StateVariableType01

7.6.7 IL_StateVariableType02

Kurzbeschreibung Der Funktionsbaustein IL_StateVariableType02 implementiert einen speziellen event-gesteuerten IIR-Filter (Infinite Impulse Response) zweiter Ordnung. Dieser digitale Filter lässt sich als Hochpass, Tiefpass, Bandpass oder Bandsperre einsetzen. Dieser Funktionsbaustein arbeitet event-gesteuert, d.h. bei Änderung am Eingang "Step" wird ein neuer Wert verarbeitet.

Schnittstellenbeschreibung

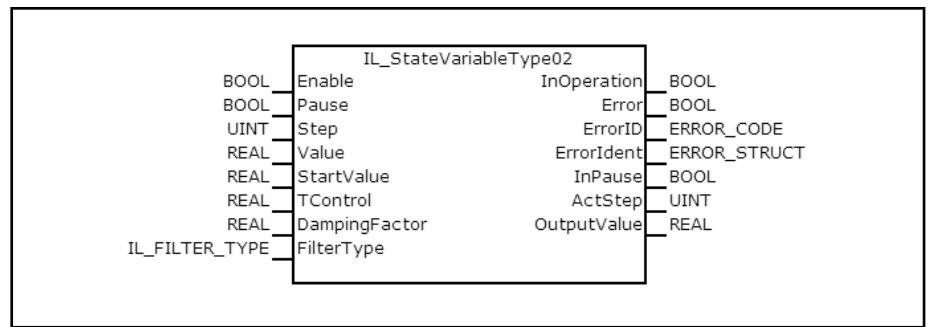


Abb. 7-54: Funktionsbaustein IL_StateVariableType02

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Wenn TRUE, wird "OutputValue" auf den mit "StartValue" berechneten Wert gesetzt.
	Pause	BOOL	aktiviert den Pause-Modus wenn TRUE
	Value	REAL	Der aktuelle Istwert des Prozesses.
	StartValue	REAL	Startwert, der bei steigender Flanke an "Enable" am Ausgang "OutputValue" angenommen wird.
	TControl	REAL	Zeitkonstante
	Damping-Factor	REAL	Dämpfungsfaktor
	FilterType	IL_FILTER_TYPE	Filterart IL_FILTER_TYPE , Seite 417.
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, dann arbeitet der Filter und der Ausgang "OutputValue" ist gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes , Seite 416.
	InPause	BOOL	"Pause" ist aktiv und der Wert an "OutputValue" ist eingefroren.
	ActStep	UINT	Schritt, auf den sich "OutputValue" bezieht.
	OutputValue	REAL	Der berechnete Filterwert

Tab. 7-84: Funktionsbaustein IL_StateVariableType02

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Value	REAL	n. def.	n.def.	0.0	kontinuierlich
StartValue	REAL	n.def.	n.def.	0.0	steigende Flanke von Enable
TControl	REAL	>0.0	n.def.	1	kontinuierlich

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
DampingFactor	REAL	0.0		1.0	kontinuierlich
FilterType	IL_FILTER_TYPE			IL_LOWPASS	steigende Flanke von Enable

Tab. 7-85: Min-, Max- und Default-Werte der Eingänge des IL_StateVariable-Type02

Funktionsbeschreibung

Der Funktionsbaustein IL_StateVariableType02 implementiert einen speziellen event-gesteuerten IIR-Filter (Infinite Impulse Response) zweiter Ordnung. Dieser digitale Filter lässt sich als Hochpass, Tiefpass, Bandpass oder Bandsperrfilter einsetzen. Dieser Funktionsbaustein arbeitet event-gesteuert, d.h. bei Änderung am Eingang "Step" wird ein neuer Wert verarbeitet.

Value, StartValue, OutputValue

Der Eingang „Value“ enthält den Eingangswert des Filters. Dieser wird bei jedem Aufruf verarbeitet, wenn „InOperation“ gesetzt ist. Bei steigender Flanke an „Enable“ wird der Wert in „StartValue“ in alle internen Register übernommen. Wenn „Enable“ gesetzt und „InPause“ nicht gesetzt ist, wird das Ergebnis am Ausgang „OutputValue“ ausgegeben.

TControl, DampingFactor, FilterType

Der Eingang „FilterType“ gibt die Art des zu berechnenden Filters an. Es stehen die folgenden Filterarten zur Verfügung: IL_LOWPASS, IL_HIGHPASS, IL_BANDPASS, IL_BANDREJECT.

Je nach Filterart werden die Eingänge „TControl“ und „DampingFactor“ interpretiert:

FilterType	1/TControl	DampingFactor
IL_LOWPASS	Grenzfrequenz oberhalb der Frequenzen unterdrückt werden	DampingFactor<1: Überspringen DampingFactor=1: kritische Dämpfung DampingFactor>1: kein Überspringen
IL_HIGHPASS	Grenzfrequenz unterhalb der Frequenzen unterdrückt werden	
IL_BANDPASS	Mittenfrequenz des Bandpasses	Bandbreite = $2 \cdot \text{DampingFactor} / \text{TControl}$
IL_BANDREJECT	Mittenfrequenz der Bandsperre	

Tab. 7-86: Interpretation der Eingangswerte nach eingestellter Filterart

Pause

Der Eingang "Pause" wird nur ausgewertet, solange der "Enable"-Eingang aktiv ist. Ist "Pause" TRUE, wird keine Berechnung durchgeführt und der Wert von "OutputValue" nicht verändert. Der interne Integrator ist ebenfalls eingefroren.

Der Ausgang "InPause" ist gesetzt, solange "Pause" aktiv ist.

Der Ausgang "ActStep" wird weiterhin aktualisiert.

Zeitverhalten

Der Funktionsbaustein IL_StateVariableType02 arbeitet asynchron. Nach dem Start durch eine steigende Flanke am Eingang "Enable" liest er den Eingang "StartValue". Der Funktionsbaustein liest bei jeder Änderung am Eingang "Step" einen Wert vom Eingang "Value", "TControl" und "DampingFactor" und berechnet einen neuen Wert am Ausgang "OutputValue". Bei aktivem Filter ist der Ausgang "InOperation" gesetzt. Tritt ein Fehler auf, wird der Ausgang "Error" gesetzt.

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR, 16#0006	16#00000C02	16#00000001	Der Eingang "TControl" ist ungültig.
INPUT_RANGE_ERROR, 16#0006	16#00000C02	16#00000008	Der Eingang "DampingFactor" ist ungültig.
INPUT_RANGE_ERROR, 16#0006	16#00000C02	16#00000002	Der im Eingang "FilterType" angegebene Wert ist ungültig
STATE_MACHINE_ERROR, 16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.

Tab. 7-87: Generierte Fehlernummern des Funktionsbausteins IL_StateVariableType02

7.6.8 IL_FILTER_TYPE

Der Aufzählungstyp IL_FILTER_TYPE definiert mehrere Filtergrundtypen.

Element	Wert	Beschreibung
IL_LOWPASS	1	Tiefpass
IL_HIGHPASS	2	Hochpass
IL_BANDPASS	3	Bandpass
IL_BANDREJECT	4	Bandsperre

Tab. 7-88: Elemente des Aufzählungstyps IL_FILTER_TYPE

Anwendung in

- [IL_StateVariableType01, Seite 412](#),
- [IL_StateVariableType02, Seite 414](#)

7.6.9 IL_FIR_COEFF

Der Datentyp IF_FIR_COEFF wird verwendet, um die Filterkoeffizienten einzustellen.

Strukturelement	Typ	Min-Wert	Max-Wert	Default	Beschreibung
Coeffs	ARRAY[0..15] OF LREAL			16(0.0)	Filter-Koeffizienten
NumberOfOeffs	UINT	1	15	1	Anzahl der Filterkoeffizienten

Tab. 7-89: Datenstruktur IL_FIR_COEFF

Anwendung in

- [IL_FIRType01, Seite 403](#),
- [IL_FIRType02, Seite 405](#).

7.6.10 IL_IIR_COEFF

Der Datentyp IF_IIR_COEFF wird verwendet, um die Filterkoeffizienten einzustellen.

Strukturelement	Typ	Min-Wert	Max-Wert	Default	Beschreibung
ForwardCoeffs	ARRAY[0..15] OF LREAL			16(0.0)	Filter-Koeffizienten des Vorwärtszweiges
BackwardCoeffs	ARRAY[1..15] OF LREAL			15(0.0)	Filter-Koeffizienten des Rückkopplungszweiges
NumberOfOeffs	UINT	1	15	1	Anzahl der Filterkoeffizienten

Tab. 7-90: Datenstruktur IL_IIR_COEFF

Anwendung in

- [IL_IIRType01, Seite 407,](#)
- [IL_IIRType02, Seite 409.](#)

7.7 Signalgeneratoren (Generators)

7.7.1 Einführung und Übersicht

Dieser Abschnitt enthält die Signalgeneratoren der RIL_LoopControl. Generatoren sind Signalquellen.

Die RIL_LoopControl stellt die folgenden Signalgeneratoren zur Verfügung:

zeitdiskret	eventgesteuert	Beschreibung
Generators		
IL_NoiseGeneratorType01, Seite 419		Pseudozufallszahl und Rauschgenerator
IL_NoiseGeneratorType02, Seite 420		Rauschgenerator für weißes Rauschen
IL_PRBSGeneratorType01, Seite 422	IL_PRBSGeneratorType02, Seite 424	Generator für pseudozufällige Binärzahlen
Generators/SignalGenerators		
IL_AngleGeneratorType01, Seite 426	IL_AngleGeneratorType02, Seite 428	Winkelgenerator
IL_PhaseController, Seite 429		Phasenregler
IL_SignalGeneratorType01, Seite 431	IL_SignalGeneratorType02, Seite 435	Signalgenerator
IL_AmplitudeControllerType01, Seite 439	IL_AmplitudeControllerType02, Seite 442	Amplitudenregler

Tab. 7-91: Signalgeneratoren der RIL_LoopControl

Die Signalgenerator-Funktionsbausteine [IL_NoiseGeneratorType01](#), [IL_NoiseGeneratorType02](#) und [IL_PRBSGeneratorType0x](#) sind eigenständig und können direkt verwendet werden. Sie haben einen "OutputValue"-Ausgang, aber keinen "Value"-Eingang.

Die Funktionsbausteine [IL_AngleGeneratorType0x](#), [IL_PhaseController](#), [IL_SignalGeneratorType0x](#) und [IL_AmplitudeControllerType0x](#) gehören zusammen und können wie folgt verschaltet werden:

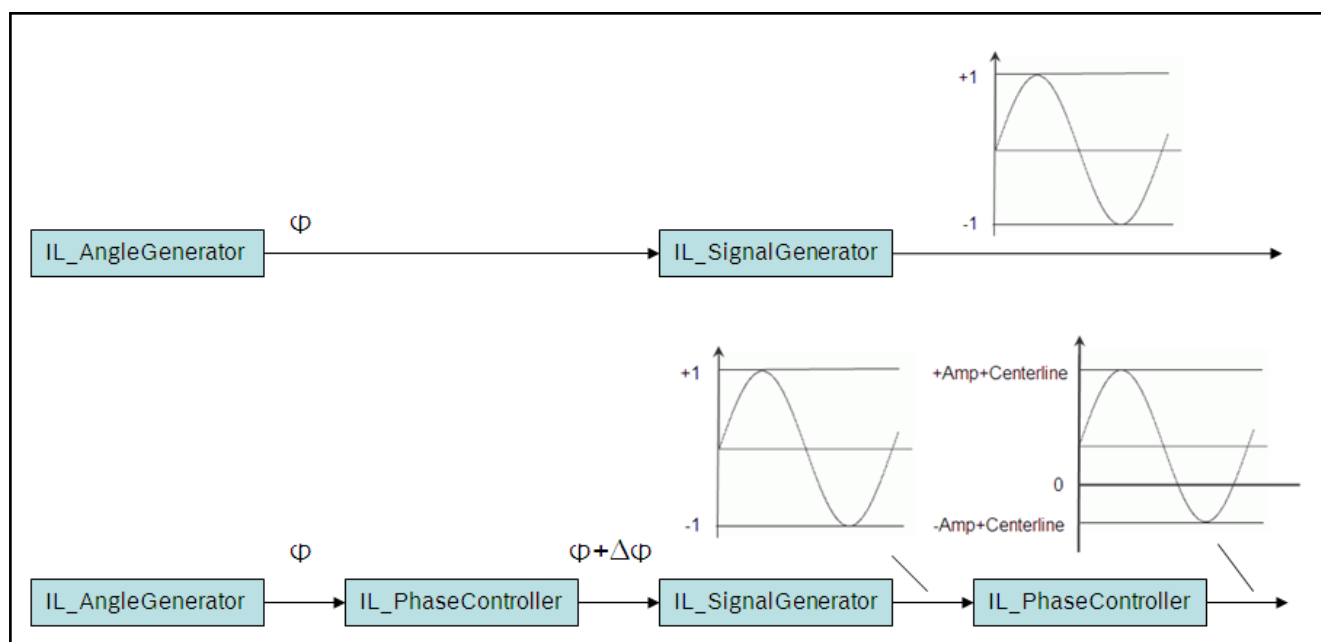


Abb. 7-55: Schaltungsbeispiele für Signalgenerator-Funktionsbausteine

7.7.2 IL_NoiseGeneratorType01

Kurzbeschreibung

Dieser Funktionsbaustein generiert ein gleichverteiltes Rauschsignal mit wählbarer Amplitude auf Basis einer Pseudo-Zufallszahl.

Schnittstellenbeschreibung

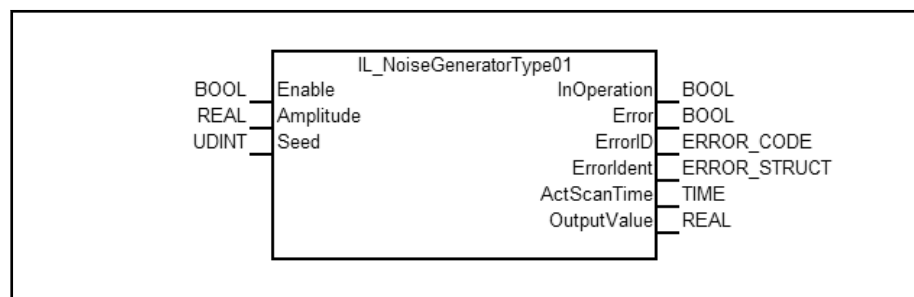


Abb. 7-56: Funktionsbaustein IL_NoiseGeneratorType01

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins
	Amplitude	REAL	Amplitude des Rauschsignals
	Seed	UDINT	Startwert, 0 für CPU-Tick
VAR_OUTPUT	InOperation	BOOL	Der Baustein arbeitet, die Ausgänge sind gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes , Seite 420.
	ActScanTime	TIME	Gemessene Aufrufzykluszeit.
	OutputValue	REAL	Generierte Pseudozufallszahl.

Tab. 7-92: Funktionsbaustein IL_NoiseGeneratorType01

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Amplitude	REAL	0.0	n.def.	1.0	kontinuierlich
Seed	UDINT	0	$2^{32}-1$	0	steigende Flanke an Enable

Tab. 7-93: Min-, Max- und Default-Werte der Eingänge des IL_NoiseGeneratorType01

Funktionsbeschreibung

Der Funktionsbaustein IL_NoiseGeneratorType01 erzeugt ein gleichverteiltes Rauschsignal auf Basis von Pseudo-Zufallszahlen. Bei steigender Flanke an "Enable" wird der Wert am "Seed"-Eingang gelesen. Ist "Seed" 0, so wird der aktuelle CPU-Counter zur Initialisierung verwendet. Alle anderen Werte an "Seed" werden direkt als Startwert verwendet. Wenn "InOperation" gesetzt ist, erzeugt der Baustein bei jedem Aufruf einen neuen Wert zwischen 0.0 und "Amplitude" an "OutputValue". Dieser Baustein sollte zyklisch aufgerufen werden. Der gemessene Aufrufzyklus wird in "ActScanTime" ausgegeben. "OutputValue" enthält das erzeugte Rauschsignal.

Die erzeugten Zufallszahlen liegen gleichverteilt im Bereich zwischen 0 und "Amplitude".

Die Abarbeitung erfolgt kontinuierlich bei gesetztem "Enable"-Eingang.

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR, 16#0001	16#00000C06	16#00000001	"Amplitude" ist kleiner als 0.0
STATE_MACHINE_ERROR, 16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.

Tab. 7-94: Generierte Fehlernummern des Funktionsbausteins IL_NoiseGeneratorType01

7.7.3 IL_NoiseGeneratorType02

Kurzbeschreibung

Der Funktionsbaustein IL_NoiseGeneratorType02 erzeugt ein in der Amplitude gaußverteiltes Rauschsignal um einen vorgegebenen Mittelwert. Dieses Signal ist ein weißes Rauschen, d.h. alle Frequenzen sind in gleicher Amplitude enthalten. Zudem ist es nahezu gleichspannungsfrei. Dieser Funktionsbaustein arbeitet zeitdiskret.

Schnittstellenbeschreibung

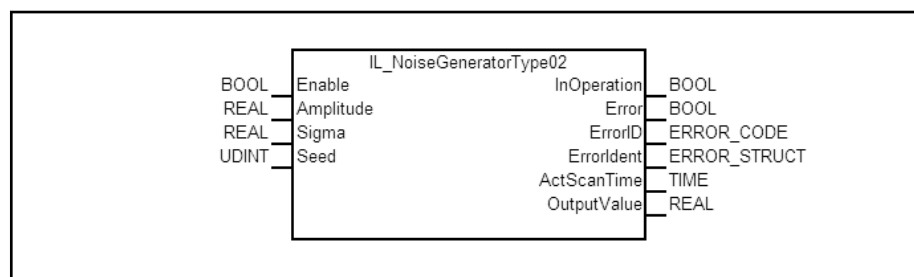


Abb. 7-57: Funktionsbaustein IL_NoiseGeneratorType02

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins
	Amplitude	REAL	Mittlere Signalamplitude
	Sigma	REAL	Standardabweichung des Signals
	Seed	UDINT	Startwert, 0 für CPU-Tick
VAR_OUTPUT	InOperation	BOOL	Der Baustein arbeitet, die Ausgänge sind gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes , Seite 422.
	ActScan-Time	TIME	Gemessene Aufrufzykluszeit.
	OutputValue	REAL	Generiertes Rauschsignal.

Tab. 7-95: Funktionsbaustein IL_NoiseGeneratorType02

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Amplitude	REAL	0.0	n.def.	1.0	steigende Flanke an Enable
Sigma	REAL	>0.0		1.0	steigende Flanke an Enable
Seed	UDINT	0	2 ³² -1	0	steigende Flanke an Enable

Tab. 7-96: Min-, Max- und Default-Werte der Eingänge des IL_NoiseGeneratorType02

Funktionsbeschreibung

Der Funktionsbaustein IL_NoiseGeneratorType02 erzeugt ein in der Amplitude gaußverteiltes Rauschsignal. Dieses Signal ist ein weißes Rauschen, d.h. alle Frequenzen sind in gleicher Amplitude enthalten. Zudem ist es gleichspannungsfrei. Dieser Funktionsbaustein arbeitet zeitdiskret.

Amplitude, Sigma:

Der Eingang "Amplitude" definiert die mittlere Amplitude, auch als μ oder Erwartungswert bezeichnet, des erzeugten Signals. Das generierte Signal schwankt um diesen Wert. Der Eingang "Sigma" gibt die Standardabweichung, auch σ genannt, an.

Es ergibt sich die folgende Werteverteilung:

68,3 % im Intervall $\mu \pm \sigma$

95,4 % im Intervall $\mu \pm 2\sigma$

99,7 % im Intervall $\mu \pm 3\sigma$

Seed:

Dieser Eingang wird als Startwert für den internen Zufallsgenerator verwendet. In der Regel kann er weggelassen werden.

Zeitverhalten:

Der Funktionsbaustein IL_NoiseGeneratorType02 arbeitet asynchron. Nach dem Start durch eine steigende Flanke am Eingang "Enable" liest er den Ein-

gänge "Amplitude", "Sigma" und "Seed". Der Funktionsbaustein generiert bei jedem Aufruf einen neuen Wert am Ausgang "OutputValue". Bei aktiver Signalgenerierung ist der Ausgang "InOperation" gesetzt. Tritt ein Fehler auf, wird der Ausgang "Error" gesetzt.

Fehlercodes Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR, 16#0001	16#00000C0B	16#00000001	Der Eingang "Sigma" ist kleiner gleich 0.0.
STATE_MACHINE_ERROR,16#0005	16#00000006	16#0000FFFF	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.

Tab. 7-97: Generierte Fehlernummern des Funktionsbausteins IL_NoiseGeneratorType02

7.7.4 IL_PRBSGeneratorType01

Kurzbeschreibung Der Funktionsbaustein IL_PRBSGeneratorType01 erzeugt ein binäres Pseudo-Rausch Signal. Dieses Signal ist ein näherungsweise weißes Rauschen, d.h. alle Frequenzen sind in gleicher Amplitude enthalten. Zudem ist es annähernd gleichspannungsfrei. Dieser Funktionsbaustein arbeitet zeitdiskret.

Schnittstellenbeschreibung

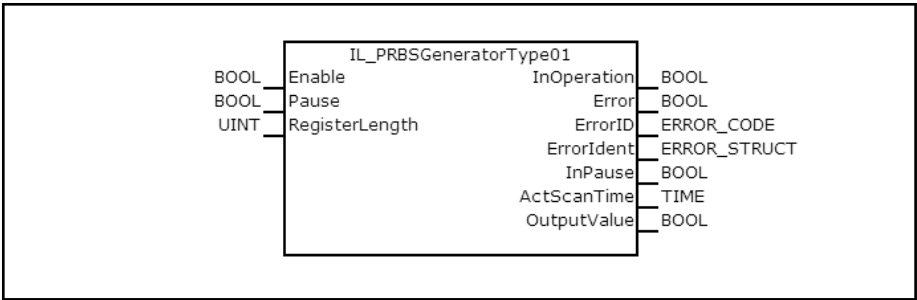


Abb. 7-58: Funktionsbaustein IL_PRBSGeneratorType01

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins
	Pause	BOOL	Unterbricht die Generierung des Signals
	Register- Length	UINT	Länge des internen Schieberegisters
VAR_OUTPUT	InOperation	BOOL	Der Baustein arbeitet, die Ausgänge sind gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes, Seite 424.
	InPause	BOOL	Abarbeitung unterbrochen
	ActScan- Time	TIME	Gemessene Aufrufzykluszeit.
	OutputValue	BOOL	Generierte Pseudozufallszahl.

Tab. 7-98: Funktionsbaustein IL_PRBSGeneratorType01

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
RegisterLength	UINT	4	32	4	steigende Flanke an Enable

Tab. 7-99: Min-, Max- und Default-Werte der Eingänge des IL_PRBSGeneratorType01

Funktionsbeschreibung

Der Funktionsbaustein IL_PRBSGeneratorType01 erzeugt ein binäres Pseudo-Rausch Signal. Dieses Signal ist ein näherungsweise weißes Rauschen, d.h. alle Frequenzen sind in gleicher Amplitude enthalten. Zudem ist es annähernd gleichspannungsfrei. Dieser Funktionsbaustein arbeitet zeitdiskret.

Das generierte Signal wiederholt sich nach N Zyklen. Die Anzahl der Zyklen ist abhängig von der Registertiefe.

RegisterLength:

"RegisterLength" stellt die Tiefe des internen Schieberegisters ein, dass für die Signalgenerierung verwendet wird. Die Registertiefe hat Einfluss auf die Güte des erzeugten Signals. Eine größere Registertiefe erzeugt einen längeren Zyklus, bevor sich das Signal wiederholt.

Die folgende Tabelle zeigt die Signalperiode in Abhängigkeit von "RegisterLength":

RegisterLength	Periode	RegisterLength	Periode
		17	131071
2	3	18	262143
3	7	19	524287
4	15	20	1048575
5	31	21	2097151
6	63	22	4194303
7	127	23	8388607
8	255	24	16777215
9	511	25	33554431
10	1023	26	67108863
11	2047	27	134217727
12	4095	28	268435455
13	8191	29	536870911
14	16383	30	1073741823
15	32767	31	2147483647
16	65535	32	4294967295

"RegisterLength" darf nur die in der Tabelle aufgeführten Werte enthalten.

Pause, InPause:

Die Abarbeitung des Funktionsbausteins kann durch den Eingang "Pause" vorübergehend unterbrochen werden. Ist "Pause" aktiv, wird der Ausgang

"InPause" gesetzt. In diesem Modus werden keine neuen Signalwerte am Ausgang "OutputValue" generiert.

Zeitverhalten:

Der Funktionsbaustein IL_PRBSGeneratorType01 arbeitet asynchron. Nach dem Start durch eine steigende Flanke am Eingang "Enable" liest er den Eingang "RegisterLength". Der Funktionsbaustein generiert bei jedem Aufruf einen neuen Wert am Ausgang "OutputValue". Bei aktiver Signalgenerierung ist der Ausgang "InOperation" gesetzt. Tritt ein Fehler auf, wird der Ausgang "Error" gesetzt.

Fehlercodes Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR, 16#0001	16#00000C0A	16#00000001	Der Eingang "RegisterLength" ist ungültig.
STATE_MACHINE_ERROR, 16#0005	16#00000006	16#0000FFFF	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.

Tab. 7-100: Generierte Fehlernummern des Funktionsbausteins IL_PRBSGeneratorType01

7.7.5 IL_PRBSGeneratorType02

Kurzbeschreibung Der Funktionsbaustein IL_PRBSGeneratorType02 erzeugt ein binäres Pseudo-Rausch Signal. Dieses Signal ist ein näherungsweise weißes Rauschen, d.h. alle Frequenzen sind in gleicher Amplitude enthalten. Zudem ist es annähernd gleichspannungsfrei. Dieser Funktionsbaustein arbeitet event-gesteuert, d.h. bei Änderung am Eingang "Step" wird ein neuer Wert generiert.

Schnittstellenbeschreibung

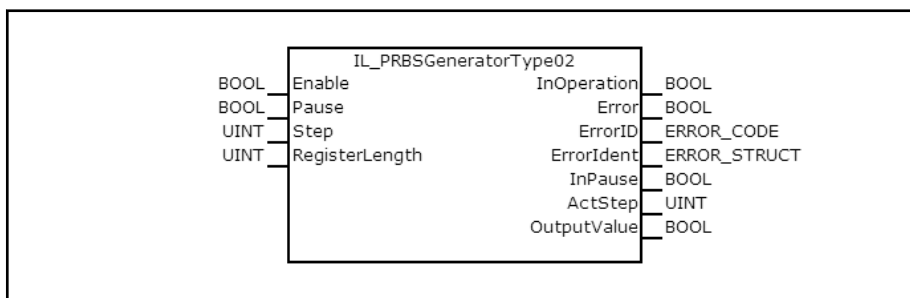


Abb. 7-59: Funktionsbaustein IL_PRBSGeneratorType02

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins
	Pause	BOOL	Unterbricht die Generierung des Signals
	Step	UINT	Bearbeitungsschritt, eine Änderung löst einen neuen Signalwert aus
	Register- Length	UINT	Länge des internen Schieberegisters
VAR_OUTPUT	InOperation	BOOL	Der Baustein arbeitet, die Ausgänge sind gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes , Seite 426.

	Name	Typ	Kommentar
	InPause	BOOL	Abarbeitung unterbrochen
	ActStep	UINT	Zuletzt bearbeiteter Schritt
	OutputValue	BOOL	Generiertes Pseudo-Rausch Signal

Tab. 7-101: Funktionsbaustein IL_PRBSGeneratorType02

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich, wenn Enable gesetzt
Step	UINT	0	65535	0	kontinuierlich, wenn Enable gesetzt
RegisterLength	UINT	4	32	4	steigende Flanke an Enable

Tab. 7-102: Min-, Max- und Default-Werte der Eingänge des IL_PRBSGenerator-Type02

Funktionsbeschreibung

Der Funktionsbaustein IL_PRBSGeneratorType02 erzeugt ein binäres Pseudo-Rausch Signal. Dieses Signal ist ein näherungsweise weißes Rauschen, d.h. alle Frequenzen sind in gleicher Amplitude enthalten. Zudem ist es annähernd gleichspannungsfrei. Dieser Funktionsbaustein arbeitet event-gesteuert, d.h. bei Änderung am Eingang "Step" wird ein neuer Wert generiert.

Das generierte Signal wiederholt sich nach N Zyklen. Die Anzahl der Zyklen ist abhängig von der Registertiefe.

RegisterLength:

"RegisterLength" stellt die Tiefe des internen Schieberegisters ein, dass für die Signalgenerierung verwendet wird. Die Registertiefe hat Einfluss auf die Güte des erzeugten Signals. Eine größere Registertiefe erzeugt einen längeren Zyklus, bevor sich das Signal wiederholt.

Die folgende Tabelle zeigt die Signalperiode in Abhängigkeit von "RegisterLength":

RegisterLength	Periode	RegisterLength	Periode
		17	131071
2	3	18	262143
3	7	19	524287
4	15	20	1048575
5	31	21	2097151
6	63	22	4194303
7	127	23	8388607
8	255	24	16777215
9	511	25	33554431
10	1023	26	67108863

RegisterLength	Periode		RegisterLength	Periode
11	2047		27	134217727
12	4095		28	268435455
13	8191		29	536870911
14	16383		30	1073741823
15	32767		31	2147483647
16	65535		32	4294967295

"RegisterLength" darf nur die in der Tabelle aufgeführten Werte enthalten.

Pause, InPause:

Die Abarbeitung des Funktionsbausteins kann durch den Eingang "Pause" vorübergehend unterbrochen werden. Ist "Pause" aktiv, wird der Ausgang "InPause" gesetzt. In diesem Modus werden keine neuen Signalwerte am Ausgang "OutputValue" generiert.

Zeitverhalten:

Der Funktionsbaustein IL_PRBSGeneratorType02 arbeitet asynchron. Nach dem Start durch eine steigende Flanke am Eingang "Enable" liest er den Eingang "RegisterLength". Der Funktionsbaustein generiert bei jeder Änderung am Eingang "Step" einen neuen Wert am Ausgang "OutputValue". Bei aktiver Signalgenerierung ist der Ausgang "InOperation" gesetzt. Tritt ein Fehler auf, wird der Ausgang "Error" gesetzt.

Fehlercodes Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR, 16#0001	16#00000C0A	16#00000001	Der Eingang "RegisterLength" ist ungültig.
STATE_MACHINE_ERROR, 16#0005	16#00000006	16#0000FFFF	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.

Tab. 7-103: Generierte Fehlernummern des Funktionsbausteins IL_PRBSGeneratorType02

7.7.6 IL_AngleGeneratorType01

Kurzbeschreibung Dieser Funktionsbaustein generiert anhand der vorgegebenen Frequenz einen Winkel zwischen 0.0 und 360.0 Grad.

Schnittstellenbeschreibung

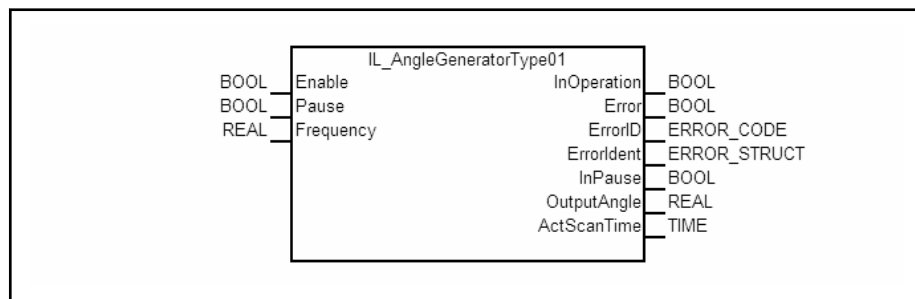


Abb. 7-60: Funktionsbaustein IL_AngleGeneratorType01

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins
	Pause	BOOL	Unterbrechung der Bearbeitung
	Frequency	REAL	Frequenz in Hz
VAR_OUTPUT	InOperation	BOOL	Der Funktionsbaustein arbeitet, die Ausgänge sind gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes , Seite 427.
	InPause	BOOL	Unterbrechung der Bearbeitung aktiv
	OutputAngle	REAL	Generierter Winkel
	ActScan-Time	TIME	Gemessene Aufrufzykluszeit.

Tab. 7-104: Funktionsbaustein IL_AngleGeneratorType01

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Frequency	REAL	0.0	1/(2*Zykluszeit)	1.0	kontinuierlich

Tab. 7-105: Min-, Max- und Default-Werte der Eingänge des IL_AngleGeneratorType01

Funktionsbeschreibung

Der Funktionsbaustein IL_AngleGeneratorType01 erzeugt einen Winkel zwischen 0 und 360 Grad mit der am Eingang "Frequency" in Hz angegebenen Frequenz. Ist "InOperation" gesetzt, erzeugt der Baustein bei jedem Aufruf einen neuen Wert zwischen 0.0 und 360.0 an "OutputAngle". Dieser Baustein sollte zyklisch aufgerufen werden. Der gemessene Aufrufzyklus wird in "ActScanTime" ausgegeben.

Die erzeugten Winkelwerte liegen im Bereich zwischen 0 und 360 Grad.

Die Abarbeitung erfolgt kontinuierlich bei gesetztem "Enable"-Eingang. Die Bearbeitung kann mit dem "Pause"-Eingang temporär unterbrochen werden.

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR, 16#0006	16#00000C0D	16#00000001	"Frequency" ist kleiner als 0.0 oder größer als 1/(2*ActScanTime)
STATE_MACHINE_ERROR, 16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.

Tab. 7-106: Generierte Fehlernummern des Funktionsbausteins IL_AngleGeneratorType01

7.7.7 IL_AngleGeneratorType02

Kurzbeschreibung Dieser Funktionsbaustein generiert anhand der vorgegebenen Frequenz einen Winkel zwischen 0.0 und 360.0 Grad.

Schnittstellenbeschreibung

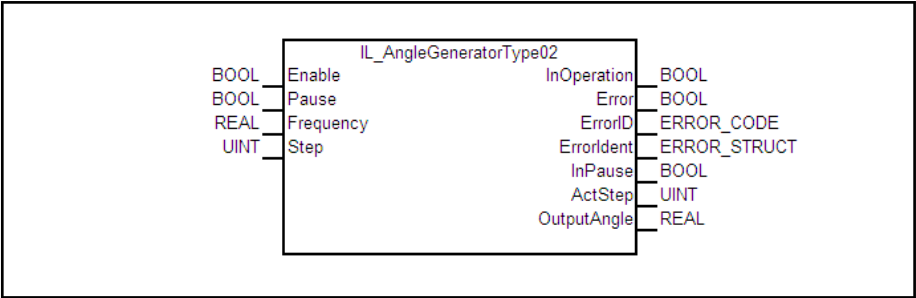


Abb. 7-61: Funktionsbaustein IL_AngleGeneratorType02

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins
	Pause	BOOL	Unterbrechung der Bearbeitung
	Frequency	REAL	Frequenz in 1/Schritte
	Step	UINT	Bearbeitungsschritt
VAR_OUTPUT	InOperation	BOOL	Der Funktionsbaustein arbeitet, die Ausgänge sind gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes, Seite 429.
	InPause	BOOL	Unterbrechung der Bearbeitung aktiv
	ActStep	UINT	Gemessene Aufrufzykluszeit.
	OutputAngle	REAL	Generierter Winkel

Tab. 7-107: Funktionsbaustein IL_AngleGeneratorType02

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Frequency	REAL	0.0	0.5	0.1	Änderung an "Step"
Step	BOOL	0	65535	0	kontinuierlich

Tab. 7-108: Min-, Max- und Default-Werte der Eingänge des IL_AngleGeneratorType02

Funktionsbeschreibung Der Funktionsbaustein IL_AngleGeneratorType01 erzeugt einen Winkel zwischen 0 und 360 Grad mit der am Eingang "Frequency" in 1/Step angegebenen Frequenz. Ist "InOperation" gesetzt, erzeugt der Baustein bei jedem Aufruf einen neuen Wert zwischen 0.0 und 360.0 an "OutputAngle". Dieser Baustein sollte zyklisch aufgerufen werden. Die am Eingang "Frequency" angegebene Frequenz bezieht sich auf die Schritte am Eingang "Step". Sie muss

daher kleiner 0.5 sein. Eine Berechnung wird nur ausgeführt, wenn sich der Eingang "Step" ändert.

Die erzeugten Winkelwerte liegen im Bereich zwischen 0 und 360 Grad.

Die Abarbeitung erfolgt durch den Eingang "Step" gesteuert bei gesetztem "Enable"-Eingang. Die Bearbeitung kann mit dem "Pause"-Eingang temporär unterbrochen werden.

Fehlercodes Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR, 16#0006	16#00000C0D	16#00000001	"Frequency" ist kleiner als 0.0 oder größer als 0.5
STATE_MACHINE_ERROR, 16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.

Tab. 7-109: Generierte Fehlernummern des Funktionsbausteins IL_AngleGeneratorType02

7.7.8 IL_PhaseController

Kurzbeschreibung Dieser Funktionsbaustein korrigiert die Phase eines Winkels über eine Regelschleife.

Schnittstellenbeschreibung

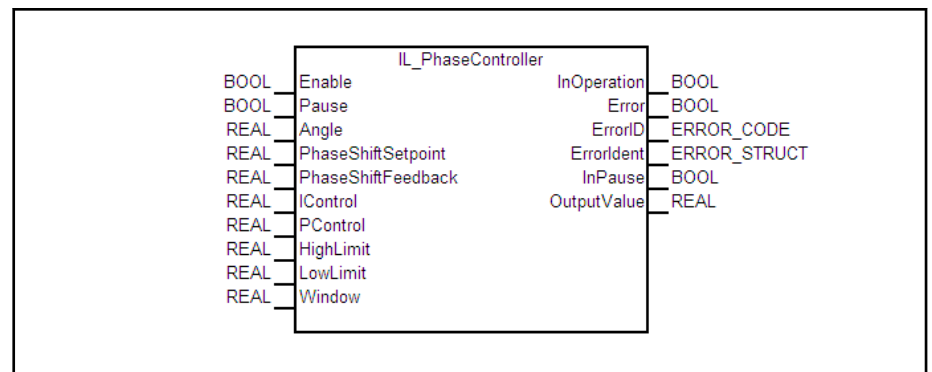


Abb. 7-62: Funktionsbaustein IL_PhaseController

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins
	Pause	BOOL	Unterbrechung der Bearbeitung
	Angle	REAL	Eingangswinkel
	PhaseShift-Setpoint	REAL	Führungsgröße für Phasenregelung
	PhaseShift-Feedback	REAL	Rückkopplungswert der Regelschleife
	IControl	REAL	I-Anteil des Reglers
	PControl	REAL	P-Anteil des Reglers
	HighLimit	REAL	obere Grenze der Korrektur in Grad
	LowLimit	REAL	untere Grenze der Korrektur in Grad
	Window	REAL	Regelfenster

	Name	Typ	Kommentar
VAR_OUTPUT	InOperation	BOOL	Der Funktionsbaustein arbeitet, die Ausgänge sind gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes , Seite 431.
	InPause	BOOL	Unterbrechung der Bearbeitung aktiv
	OutputValue	REAL	Geregelter Winkel

Tab. 7-110: Funktionsbaustein IL_PhaseController

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Angle	REAL	0.0	360.0	0.0	kontinuierlich
PhaseShiftSet-point	REAL			0.0	kontinuierlich
PhaseShiftFeedback	REAL			0.0	kontinuierlich
IControl	REAL	0.0		0.0	kontinuierlich
PControl	REAL	0.0		0.0	kontinuierlich
HighLimit	REAL	0.0	180.0	10.0	kontinuierlich
LowLimit	REAL	-180.0	0.0	-10.0	kontinuierlich
Window	REAL	0.0		5.0	kontinuierlich

Tab. 7-111: Min-, Max- und Default-Werte der Eingänge des IL_PhaseController

versteckte Ausgänge

Um die Komplexität des Funktionsbausteins zu verringern, wurden nicht alle internen Werte auf Ausgänge herausgeführt. Diese Werte lassen sich jedoch direkt am Funktionsbaustein lesen. Ein schreibender Zugriff ist nicht möglich.

Variable	Datentyp	Beschreibung
ActAngle	REAL	Berechnete Zielnulllinie
IpartCorrection	REAL	Korrekturwert des PI-Reglers
PhaseShift	REAL	Errechnete Phasenverschiebung

Tab. 7-112: Über Properties lesbare interne Werte des Funktionsbausteins

Funktionsbeschreibung

Der Funktionsbaustein IL_PhaseController verschiebt einen Eingangswinkel "Angle" um einen Korrekturwert und gibt ihn am Ausgang "OutputAngle" aus. Der Korrekturwert wird durch einen PI-Regler der 1x pro Zyklus im Nulldurchgang bestimmt, wenn die Differenz zwischen "PhaseShiftSetPoint" und "PhaseShiftFeedback" größer als "Window" ist. Der Korrekturwert wird auf einen Wert zwischen "LowLimit" und "HighLimit" begrenzt.

Die Regeleinstellungen "PControl" und "IControl" beziehen sich auf einen Zyklus. Ein Zyklus beginnt, wenn der Eingang "Angle" von 360 auf 0 Grad springt.

Die erzeugten Winkelwerte liegen im Bereich zwischen 0 und 360 Grad.

Die Abarbeitung erfolgt kontinuierlich bei gesetztem "Enable"-Eingang.

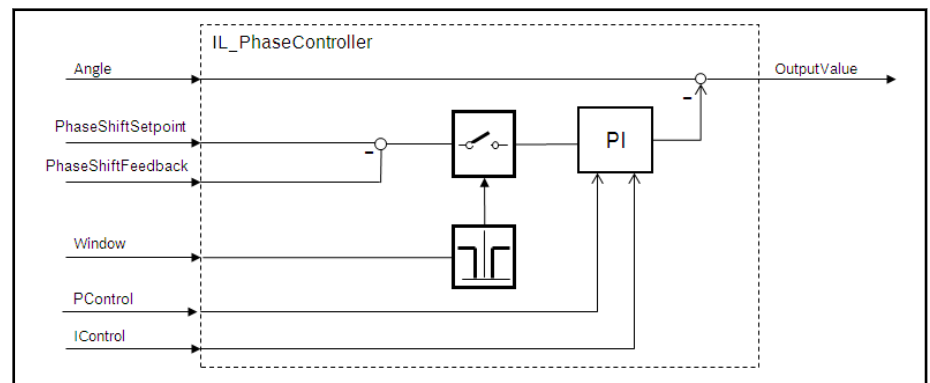


Abb. 7-63: Signalfluss des Funktionsbausteins IL_PhaseController

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR, 16#0001	16#00000C0E	16#00000001	"Angle" ist kleiner als 0 oder größer als 360
INPUT_RANGE_ERROR, 16#0001	16#00000C0E	16#00000002	"PhaseShiftSetpoint" ist kleiner als -180 oder größer als 180
INPUT_RANGE_ERROR, 16#0001	16#00000C0E	16#00000003	"PhaseShiftFeedback" ist kleiner als -180 oder größer als 180
INPUT_RANGE_ERROR, 16#0001	16#00000C0E	16#00000004	"IControl" ist negativ
INPUT_RANGE_ERROR, 16#0001	16#00000C0E	16#00000005	"PControl" ist negativ
INPUT_RANGE_ERROR, 16#0001	16#00000C0E	16#00000006	"HighLimit" ist kleiner als "LowLimit"
INPUT_RANGE_ERROR, 16#0001	16#00000C0E	16#00000007	"Window" ist kleiner als 0
STATE_MACHINE_ERROR, 16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.

Tab. 7-113: Generierte Fehlernummern des Funktionsbausteins IL_PhaseController

7.7.9 IL_SignalGeneratorType01

Kurzbeschreibung

Dieser Funktionsbaustein generiert verschiedene Signalformen anhand eines Eingangswinkels.

Schnittstellenbeschreibung

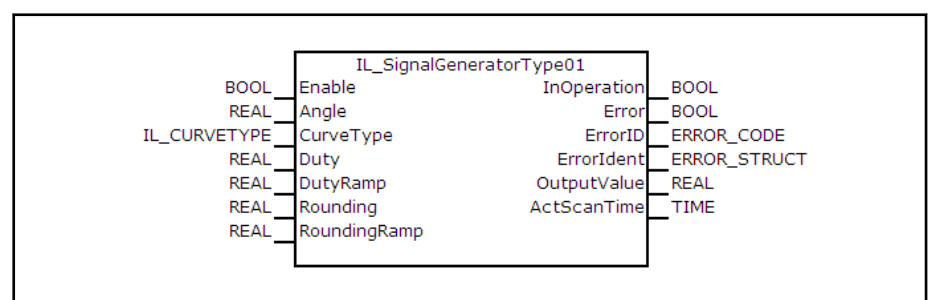


Abb. 7-64: Funktionsbaustein IL_SignalGeneratorType01

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins
	Angle	REAL	Winkel
	CurveType	IL_CURVETYPE	Generierte Signalform IL_CURVETYPE, Seite 446
	Duty	REAL	Tastverhältnis in % (50=Symmetrisch) Erlaubter Wertebereich: - Sinusoidal: 25-75 - Double_Sine: 10-90 - Triangular: 10-90
	DutyRamp	REAL	Änderung des Tastverhältnisses in %/ms
	Rounding	REAL	Rundung des Signals in % (nur bei Signalform Dreieck aktiv)
	RoundingRamp	REAL	Änderung der Rundung in %/ms
VAR_OUTPUT	InOperation	BOOL	Der Funktionsbaustein arbeitet, die Ausgänge sind gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes, Seite 435 .
	OutputValue	REAL	Generierte Signalform
	ActScan-Time	TIME	gemessene Zykluszeit

Tab. 7-114: Funktionsbaustein IL_SignalGeneratorType01

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Angle	REAL	0.0	360.0	0.0	kontinuierlich
CurveType	IL_CURVETYPE			SINUSOIDAL	kontinuierlich
Duty	REAL	10.0	90.0	50.0	kontinuierlich
DutyRamp	REAL	0.0	100.0	0.0001	kontinuierlich
Rounding	REAL	0.0	100.0	0.0	kontinuierlich
RoundingRamp	REAL	0.0	100.0	0.0001	kontinuierlich

Tab. 7-115: Min-, Max- und Default-Werte der Eingänge des IL_SignalGeneratorType01

Funktionsbeschreibung

Der Funktionsbaustein IL_SignalGeneratorType01 erzeugt die am Eingang "CurveType" eingestellte Signalform auf Basis des am Eingang "Angle" angelegten Winkels. Das Tastverhältnis kann über den Eingang "Duty" eingestellt werden. Wenn "InOperation" gesetzt ist, erzeugt der Baustein bei jedem Aufruf einen neuen Wert zwischen -1.0 und 1.0 am Ausgang "OutputValue".

Die Abarbeitung erfolgt kontinuierlich bei gesetztem "Enable"-Eingang.

CurveType

Der Eingang CurveType bestimmt die generierte Signalform. Die folgenden Signalformen sind implementiert:

- SINUSOIDAL: Sinusähnliche Funktion mit möglichst langen gleichmäßig steigenden bzw. fallenden Abschnitten

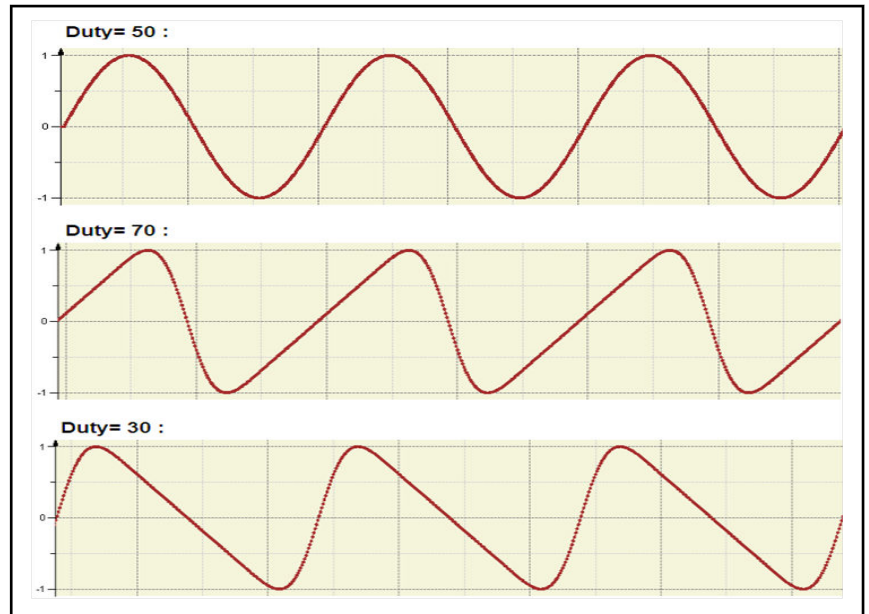


Abb. 7-65: Signalform Sinusoidal

- DOUBLE_SINE: Doppelsinus

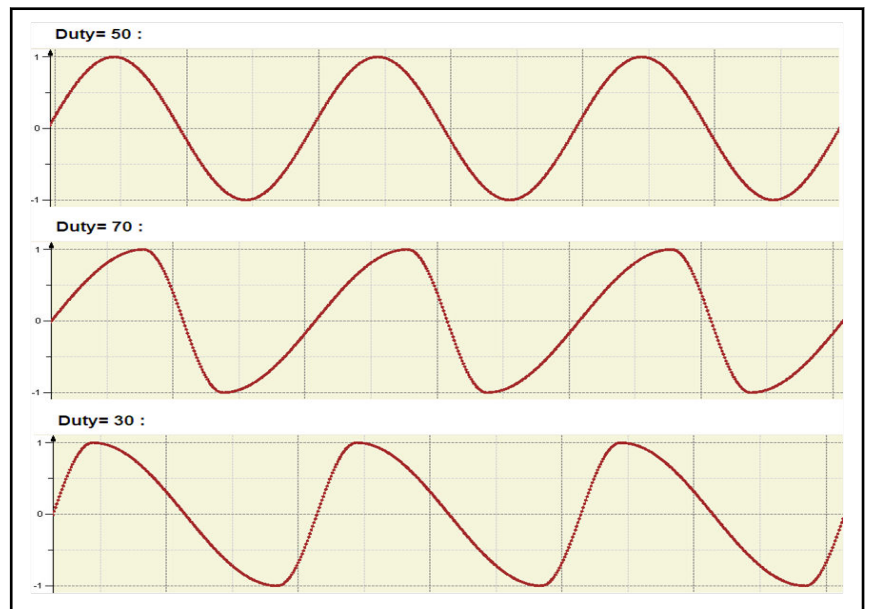
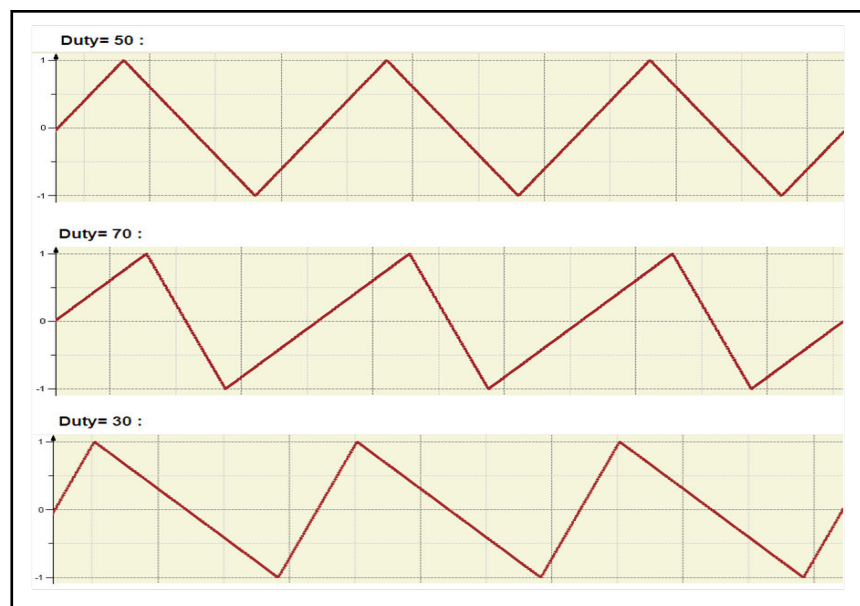


Abb. 7-66: Signalform Doppelsinus

- TRIANGULAR: Dreieck-Signal

Abb. 7-67: *Signalform Dreieck***Duty, DutyRamp**

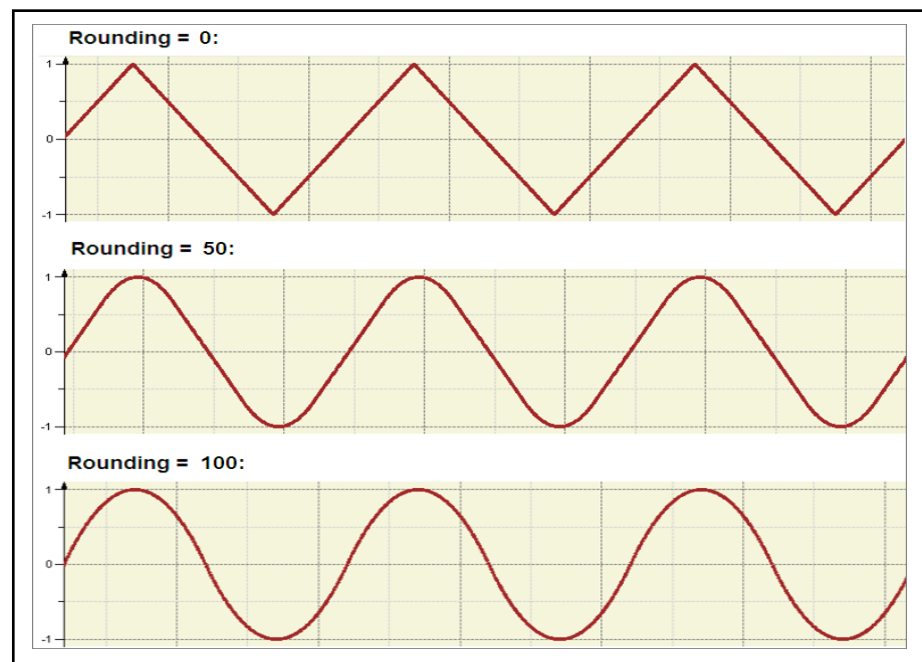
Der Eingang "Duty" bestimmt das Tastverhältnis des Signals. Im Prinzip wird die 180° Marke verschoben. Bei einem Tastverhältnis von 50% ist das Signal symmetrisch. Je nach Signalform ist "Duty" begrenzt.

Der Eingang "DutyRamp" bestimmt die Geschwindigkeit der Veränderung des Tastverhältnisses in %/ms, wenn der Wert am Eingang "Duty" verändert wird.

Rounding, RoundingRamp

Der Eingang Rounding ist nur bei der Signalform "TRIANGULAR" aktiv. Er gibt die Rundung des Dreiecks in % an. 0 bedeutet keine Rundung.

Der Eingang "RoundingRamp" bestimmt die Geschwindigkeit der Veränderung der Rundung in %/ms, wenn der Eingang "Rounding" verändert wird.

Abb. 7-68: *Auswirkungen des Eingangs "Rounding" auf die Signalform*

Fehlercodes Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR, 16#0006	16#00000C0F	16#00000001	"Angle" ist kleiner als 0.0 oder größer als 360.0
INPUT_RANGE_ERROR, 16#0006	16#00000C0F	16#00000002	"CurveType" ist ungültig
INPUT_RANGE_ERROR, 16#0006	16#00000C0F	16#00000003	"Duty" ist außerhalb des erlaubten Bereichs
INPUT_RANGE_ERROR, 16#0006	16#00000C0F	16#00000004	"DutyRamp" ist kleiner als 0.0001
INPUT_RANGE_ERROR, 16#0006	16#00000C0F	16#00000005	"Rounding" ist negativ oder größer als 100
INPUT_RANGE_ERROR, 16#0006	16#00000C0F	16#00000006	"RoundingRamp" ist kleiner als 0.0001
STATE_MACHINE_ERROR, 16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.

Tab. 7-116: Generierte Fehlernummern des Funktionsbausteins IL_SignalGeneratorType01

7.7.10 IL_SignalGeneratorType02

Kurzbeschreibung Dieser Funktionsbaustein generiert verschiedene Signalformen anhand eines Eingangswinkels.

Schnittstellenbeschreibung

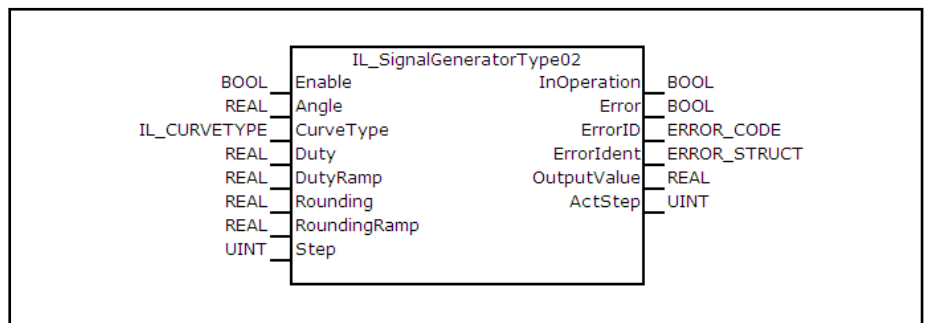


Abb. 7-69: Funktionsbaustein IL_SignalGeneratorType02

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins
	Angle	REAL	Winkel
	CurveType	IL_CURVETYPE	Generierte Signalform IL_CURVETYPE , Seite 446
	Duty	REAL	Tastverhältnis in % (50=Symmetrisch) - Sinusoidal: 25-75 - Double_Sine: 10-90 - Triangular: 10-90
	DutyRamp	REAL	Änderung des Tastverhältnisses in %/Schritt
	Rounding	REAL	Rundung des Signals in % (nur bei Signalform Dreieck aktiv)
	RoundingRamp	REAL	Änderung der Rundung in %/Schritt
	Step	UINT	Bearbeitungsschritt

	Name	Typ	Kommentar
VAR_OUTPUT	InOperation	BOOL	Der Funktionsbaustein arbeitet, die Ausgänge sind gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes , Seite 439.
	OutputValue	REAL	Generierte Signalform
	ActStep	UINT	zuletzt ausgeführter Bearbeitungsschritt

Tab. 7-117: Funktionsbaustein IL_SignalGeneratorType02

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Angle	REAL	0.0	360.0	0.0	Änderung an "Step"
CurveType	IL_CURVETYPE			SINUSOIDAL	Änderung an "Step"
Duty	REAL	10.0	90.0	50.0	Änderung an "Step"
DutyRamp	REAL	0.0	100.0	0.0001	Änderung an "Step"
Rounding	REAL	0.0	100.0	0.0	Änderung an "Step"
RoundingRamp	REAL	0.0	100.0	0.0001	Änderung an "Step"

Tab. 7-118: Min-, Max- und Default-Werte der Eingänge des IL_SignalGeneratorType02

Funktionsbeschreibung

Der Funktionsbaustein IL_SignalGeneratorType02 erzeugt die am Eingang "CurveType" eingestellte Signalform auf Basis des am Eingang "Angle" angelegten Winkels. Das Tastverhältnis kann über den Eingang "Duty" eingestellt werden. Wenn "InOperation" gesetzt ist, erzeugt der Baustein bei jeder Änderung an Step einen neuen Wert zwischen -1.0 und 1.0 am Ausgang "OutputValue".

Die Abarbeitung erfolgt kontinuierlich bei gesetztem "Enable"-Eingang.

CurveType

Der Eingang CurveType bestimmt die generierte Signalform. Die folgenden Signalformen sind implementiert:

- SINUSOIDAL: Sinusähnliche Funktion mit möglichst langen gleichmäßig steigenden bzw. fallenden Abschnitten

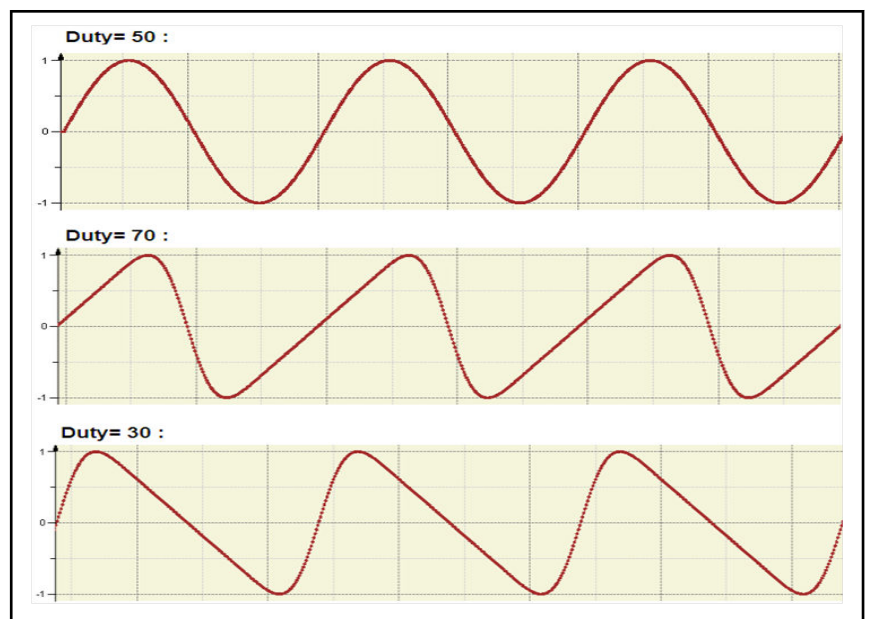


Abb. 7-70: *Signalform Sinusoidal*

- DOUBLE_SINE: Doppelsinus

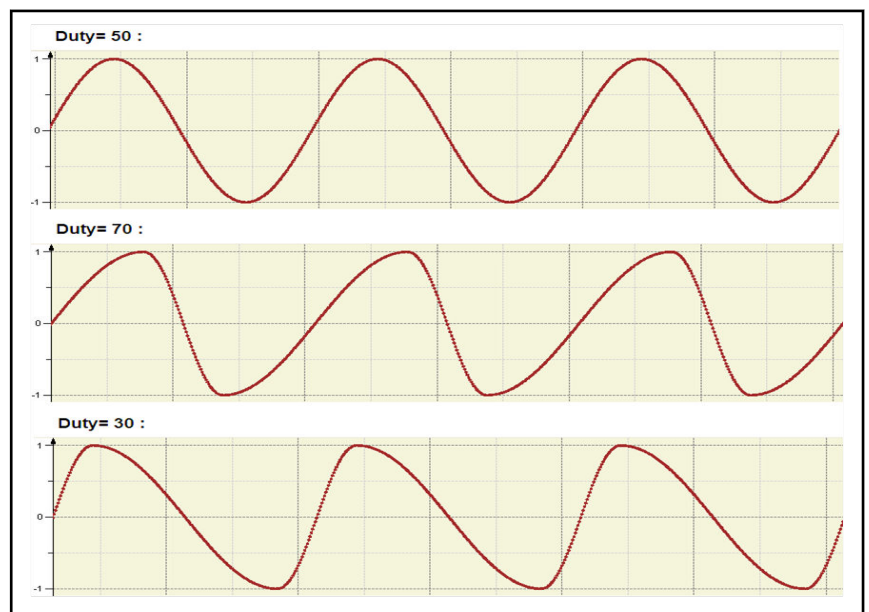


Abb. 7-71: *Signalform Doppelsinus*

- TRIANGULAR: Dreieck-Signal

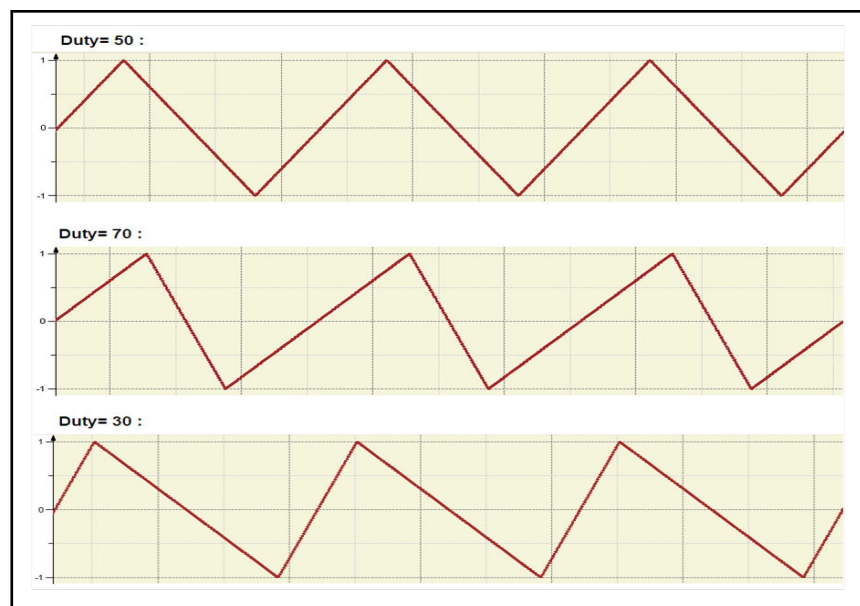


Abb. 7-72: Signalform Dreieck

Duty, DutyRamp

Der Eingang "Duty" bestimmt das Tastverhältnis des Signals. Im Prinzip wird die 180° Marke verschoben. Bei einem Tastverhältnis von 50% ist das Signal Symmetrisch. Je nach Signalform ist "Duty" begrenzt.

Der Eingang "DutyRamp" bestimmt die Geschwindigkeit der Veränderung des Tastverhältnisses in %/ms, wenn der Wert am Eingang "Duty" verändert wird.

Rounding, RoundingRamp

Der Eingang Rounding ist nur bei der Signalform "TRIANGULAR" aktiv. Er gibt die Rundung des Dreiecks in % an. 0 bedeutet keine Rundung.

Der Eingang "RoundingRamp" bestimmt die Geschwindigkeit der Veränderung der Rundung in %/ms, wenn der Eingang "Rounding" verändert wird.

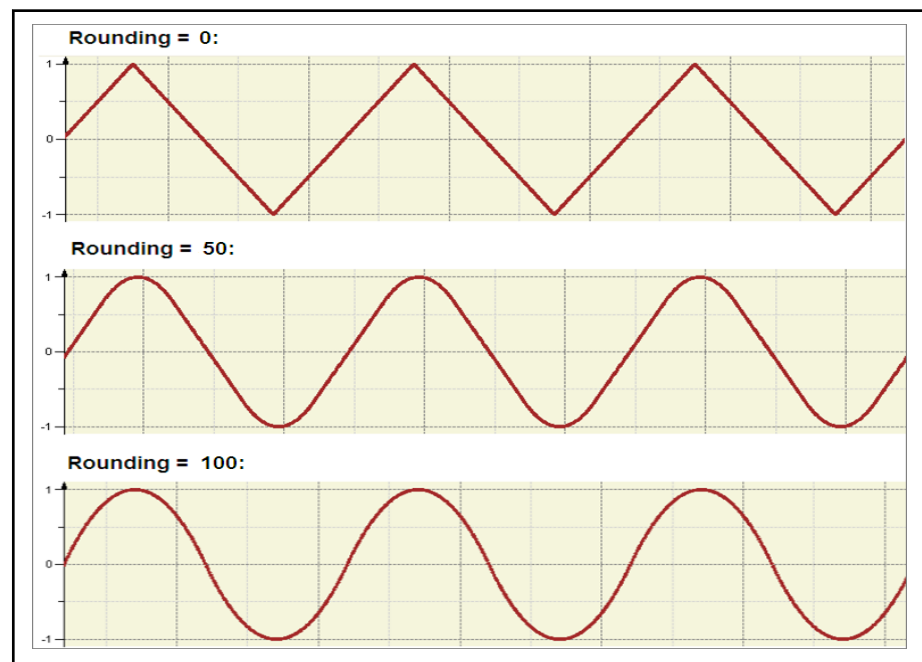


Abb. 7-73: Auswirkungen des Eingangs "Rounding" auf die Signalform

Fehlercodes Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR, 16#0006	16#00000C0F	16#00000001	"Angle" ist kleiner als 0.0 oder größer als 360.0
INPUT_RANGE_ERROR, 16#0006	16#00000C0F	16#00000002	"CurveType" ist ungültig
INPUT_RANGE_ERROR, 16#0006	16#00000C0F	16#00000003	"Duty" ist außerhalb des erlaubten Bereichs
INPUT_RANGE_ERROR, 16#0006	16#00000C0F	16#00000004	"DutyRamp" ist kleiner als 0.0001
INPUT_RANGE_ERROR, 16#0006	16#00000C0F	16#00000005	"Rounding" ist negativ oder größer als 100
INPUT_RANGE_ERROR, 16#0006	16#00000C0F	16#00000006	"RoundingRamp" ist kleiner als 0.0001
STATE_MACHINE_ERROR, 16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.

Tab. 7-119: Generierte Fehlernummern des Funktionsbausteins IL_SignalGeneratorType02

7.7.11 IL_AmplitudeControllerType01

Kurzbeschreibung Dieser Funktionsbaustein regelt die Amplitude und die Nulllinie eines Signals.

Schnittstellenbeschreibung

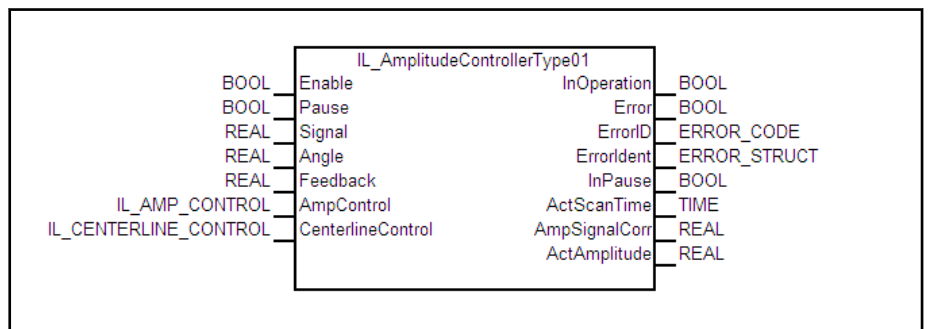


Abb. 7-74: Funktionsbaustein IL_AmplitudeControllerType01

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins
	Pause	BOOL	Unterbrechung der Bearbeitung
	Signal	REAL	Eingangssignal
	Angle	REAL	Eingangswinkel
	Feedback	REAL	Rückkopplungswert der Regelschleife
	AmpControl	IL_AMP_CONTROL	Einstellungen des Amplitudenreglers IL_AMP_CONTROL, Seite 445
	Centerline-Control	IL_CENTERLINE_CONTROL	Einstellungen des Nulllinienreglers IL_CENTERLINE_CONTROL, Seite 445

	Name	Typ	Kommentar
VAR_OUTPUT	InOperation	BOOL	Der Funktionsbaustein arbeitet, die Ausgänge sind gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes , Seite 441.
	InPause	BOOL	Unterbrechung der Bearbeitung aktiv
	ActScan-Time	TIME	Gemessene Zykluszeit
	AmpSignal-Corr	REAL	Amplitudengeregeltes Signal
	ActAmplitude	REAL	letzte gemessene Amplitude

Tab. 7-120: Funktionsbaustein IL_AmplitudeControllerType01

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Signal	REAL	-1.0	1.0	0.0	kontinuierlich
Angle	REAL	0.0	360.0	0.0	kontinuierlich
Feedback	REAL			0.0	kontinuierlich
AmpControl	IL_AMP_CONTROLLER				kontinuierlich
CenterlineControl	IL_CENTERLINE_CONTROL				kontinuierlich

Tab. 7-121: Min-, Max- und Default-Werte der Eingänge des IL_AmplitudeControllerType01

versteckte Ausgänge

Um die Komplexität des Funktionsbausteins zu verringern, wurden nicht alle internen Werte auf Ausgänge herausgeführt. Diese Werte lassen sich jedoch direkt am Funktionsbaustein lesen. Ein schreibender Zugriff ist nicht möglich.

Variable	Datentyp	Beschreibung
ActCenterline	REAL	Berechnete Zielnulllinie
ActPhaseShift	REAL	Zuletzt bestimmte Abweichung der Phase
AmpCorr	REAL	Korrekturwert der Amplitude
AmpSignal	REAL	Verstärktes Signal ohne Nulllinienkorrektur
CenterlineCorr	REAL	Geregelte Nulllinie
Strobe	BOOL	Neuer Zyklus erkannt

Tab. 7-122: Über Properties lesbare interne Werte des Funktionsbausteins

Funktionsbeschreibung

Der Funktionsbaustein IL_AmplitudeControllerType01 regelt die Amplitude und die Nulllinie des am Eingang "Signal" angelegten Signals.

AmpControl

Die Amplitudenkorrektur erfolgt durch einen PI-Regler der 1x pro Zyklus im Nulldurchgang regelt wenn die Differenz zwischen "AmpControl.Target" und der gemessenen Amplitude des Eingangs "Signal" größer als "AmpControl.Window" ist. Der Korrekturwert wird auf einen Wert zwischen "AmpControl.LowLimit" und "AmpControl.HighLimit" begrenzt.

Die Regeleinstellungen "AmpControl.PControl" und "AmpControl.IControl" beziehen sich auf einen Zyklus. Ein Zyklus beginnt, wenn der Eingang "Angle" von 360 auf 0 Grad springt.

CenterlineControl

Die Nulllinienkorrektur erfolgt durch einen PI-Regler der 1x pro Zyklus im Nulldurchgang regelt, wenn die Differenz zwischen "CenterlineControl.Target" und dem gemessenen Mittelwert des Eingangs "Signal" größer als "CenterlineControl.Window" ist. Der Korrekturwert wird auf einen Wert zwischen "CenterlineControl.LowLimit" und "CenterlineControl.HighLimit" begrenzt.

Die Regeleinstellungen "CenterlineControl.PControl" und "CenterlineControl.IControl" beziehen sich auf einen Zyklus. Ein Zyklus beginnt, wenn der Eingang "Angle" von 360 auf 0 Grad springt.

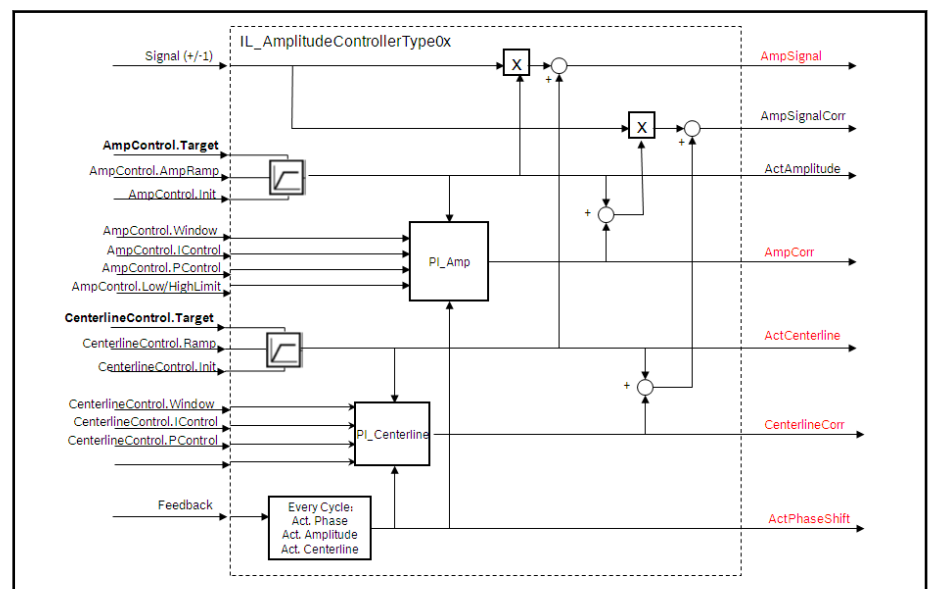


Abb. 7-75: Signalfluss des Funktionsbausteins IL_PhaseController

Die Abarbeitung erfolgt kontinuierlich bei gesetztem "Enable"-Eingang.

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR, 16#0001	16#00000C0D	16#00000001	"Signal" ist kleiner als -1.0 oder größer als 1.0
INPUT_RANGE_ERROR, 16#0001	16#00000C0D	16#00000002	"Angle" ist kleiner als 0 oder größer als 360
INPUT_RANGE_ERROR, 16#0001	16#00000C0D	16#00000003	"AmpControl.Target" ist negativ
INPUT_RANGE_ERROR, 16#0001	16#00000C0D	16#00000004	"AmpControl.Ramp" ist kleiner als 0.0001
INPUT_RANGE_ERROR, 16#0001	16#00000C0D	16#00000005	"AmpControl.Init" ist negativ
INPUT_RANGE_ERROR, 16#0001	16#00000C0D	16#00000006	"AmpControl.Window" ist negativ

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR, 16#0001	16#00000C0D	16#00000007	"AmpControl.IControl" ist negativ
INPUT_RANGE_ERROR, 16#0001	16#00000C0D	16#00000008	"AmpControl.PControl" ist negativ
INPUT_RANGE_ERROR, 16#0001	16#00000C0D	16#00000009	"AmpControl.LowLimit" ist größer als "AmpControl.HighLimit"
INPUT_RANGE_ERROR, 16#0001	16#00000C0D	16#0000000A	"CenterlineControl.Ramp" ist kleiner als 0.0001
INPUT_RANGE_ERROR, 16#0001	16#00000C0D	16#0000000B	"CenterlineControl.Window" ist negativ
INPUT_RANGE_ERROR, 16#0001	16#00000C0D	16#0000000C	"CenterlineControl.IControl" ist negativ
INPUT_RANGE_ERROR, 16#0001	16#00000C0D	16#0000000D	"CenterlineControl" ist negativ
INPUT_RANGE_ERROR, 16#0001	16#00000C0D	16#0000000E	"CenterlineControl.LowLimit" ist größer als "CenterlineControl.HighLimit"
STATE_MACHINE_ERROR, 16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.

Tab. 7-123: Generierte Fehlernummern des Funktionsbausteins IL_AmplitudeControllerType01

7.7.12 IL_AmplitudeControllerType02

Kurzbeschreibung Dieser Funktionsbaustein regelt die Amplitude und die Nulllinie eines Signals.

Schnittstellenbeschreibung

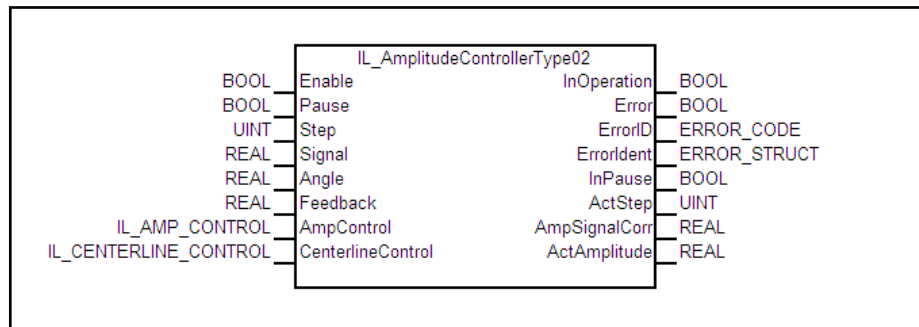


Abb. 7-76: Funktionsbaustein IL_AmplitudeControllerType02

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins
	Pause	BOOL	Unterbrechung der Bearbeitung
	Step	UINT	Bearbeitungsschritt
	Signal	REAL	Eingangssignal
	Angle	REAL	Eingangswinkel
	Feedback	REAL	Rückkopplungswert der Regelschleife
	AmpControl	IL_AMP_CONTROL	Einstellungen des Amplitudenreglers IL_AMP_CONTROL, Seite 445
	Centerline-Control	IL_CENTERLINE_CONTROL	Einstellungen des Nulllinienreglers IL_CENTERLINE_CONTROL, Seite 445

	Name	Typ	Kommentar
VAR_OUTPUT	InOperation	BOOL	Der Funktionsbaustein arbeitet, die Ausgänge sind gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes, Seite 444
	InPause	BOOL	Unterbrechung der Bearbeitung aktiv
	ActStep	UINT	Zuletzt bearbeiteter Schritt
	AmpSignal-Corr	REAL	Amplitudengeregeltes Signal
	ActAmplitude	REAL	Letzte gemessene Amplitude

Tab. 7-124: Funktionsbaustein IL_AmplitudeControllerType02

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Signal	REAL	-1.0	1.0	0.0	Änderung an "Step"
Angle	REAL	0.0	360.0	0.0	Änderung an "Step"
Feedback	REAL			0.0	Änderung an "Step"
AmpControl	IL_AMP_CONTROLLER				Änderung an "Step"
CenterlineControl	IL_CENTERLINE_CONTROL				Änderung an "Step"

Tab. 7-125: Min-, Max- und Default-Werte der Eingänge des IL_AmplitudeControllerType02

versteckte Ausgänge

Um die Komplexität des Funktionsbausteins zu verringern, wurden nicht alle internen Werte auf Ausgänge herausgeführt. Diese Werte lassen sich jedoch direkt am Funktionsbaustein lesen. Ein schreibender Zugriff ist nicht möglich.

Variable	Datentyp	Beschreibung
ActCenterline	REAL	Berechnete Ziellnullinie
ActPhaseShift	REAL	zuletzt bestimmte Abweichung der Phase
AmpCorr	REAL	Korrekturwert der Amplitude
AmpSignal	REAL	Verstärktes Signal ohne Nulllinienkorrektur
CenterlineCorr	REAL	geregelter Nulllinie
Strobe	BOOL	Neuer Zyklus erkannt

Tab. 7-126: Über Properties lesbare interne Werte des Funktionsbausteins

Funktionsbeschreibung

Der Funktionsbaustein IL_AmplitudeControllerType02 regelt die Amplitude und die Nulllinie des am Eingang "Signal" angelegten Signals.

AmpControl

Die Amplitudenkorrektur erfolgt durch einen PI-Regler der 1x pro Zyklus im Nulldurchgang regelt wenn die Differenz zwischen "AmpControl.Target" und der gemessenen Amplitude des Eingangs "Signal" größer als "AmpControl.Window" ist. Der Korrekturwert wird auf einen Wert zwischen "AmpControl.LowLimit" und "AmpControl.HighLimit" begrenzt.

Die Regeleinstellungen "AmpControl.PControl" und "AmpControl.IControl" beziehen sich auf einen Zyklus. Ein Zyklus beginnt, wenn der Eingang "Angle" von 360 auf 0 Grad springt.

CenterlineControl

Die Nulllinienkorrektur erfolgt durch einen PI-Regler der 1x pro Zyklus im Nulldurchgang regelt wenn die Differenz zwischen "CenterlineControl.Target" und dem gemessenen Mittelwert des Eingangs "Signal" größer als "CenterlineControl.Window" ist. Der Korrekturwert wird auf einen Wert zwischen "CenterlineControl.LowLimit" und "CenterlineControl.HighLimit" begrenzt.

Die Regeleinstellungen "CenterlineControl.PControl" und "CenterlineControl.IControl" beziehen sich auf einen Zyklus. Ein Zyklus beginnt, wenn der Eingang "Angle" von 360 auf 0 Grad springt.

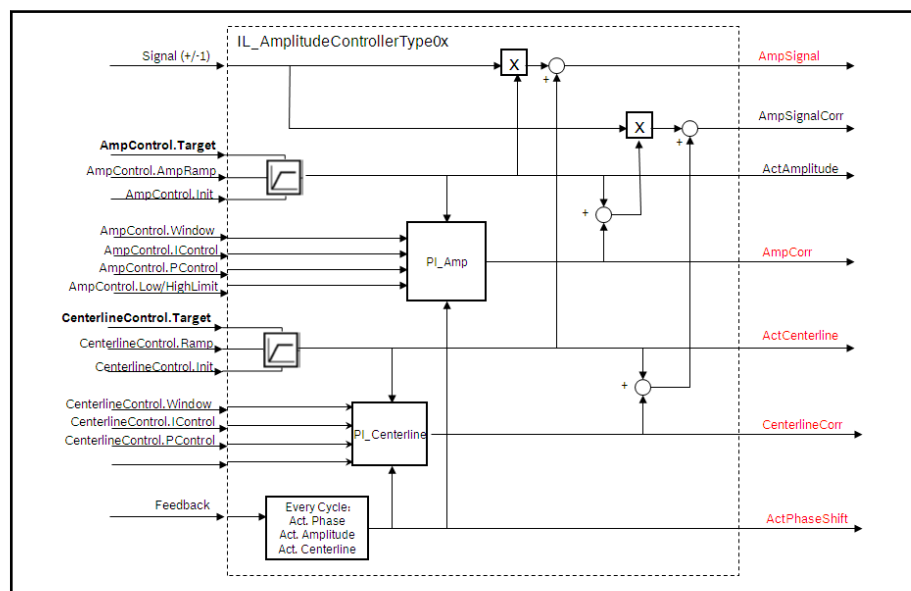


Abb. 7-77: Signalfuss des Funktionsbausteins IL_PhaseController

Die Abarbeitung erfolgt bei Änderung am Eingang "Step" bei gesetztem "Enable"-Eingang.

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR, 16#0001	16#00000C0D	16#00000001	"Signal" ist kleiner als -1.0 oder größer als 1.0
INPUT_RANGE_ERROR, 16#0001	16#00000C0D	16#00000002	"Angle" ist kleiner als 0 oder größer als 360
INPUT_RANGE_ERROR, 16#0001	16#00000C0D	16#00000003	"AmpControl.Target" ist negativ
INPUT_RANGE_ERROR, 16#0001	16#00000C0D	16#00000004	"AmpControl.Ramp" ist kleiner als 0.0001
INPUT_RANGE_ERROR, 16#0001	16#00000C0D	16#00000005	"AmpControl.Init" ist negativ
INPUT_RANGE_ERROR, 16#0001	16#00000C0D	16#00000006	"AmpControl.Window" ist negativ

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR, 16#0001	16#00000C0D	16#00000007	"AmpControl.IControl" ist negativ
INPUT_RANGE_ERROR, 16#0001	16#00000C0D	16#00000008	"AmpControl.PControl" ist negativ
INPUT_RANGE_ERROR, 16#0001	16#00000C0D	16#00000009	"AmpControl.LowLimit" ist größer als "AmpControl.HighLimit"
INPUT_RANGE_ERROR, 16#0001	16#00000C0D	16#0000000A	"CenterlineControl.Ramp" ist kleiner als 0.0001
INPUT_RANGE_ERROR, 16#0001	16#00000C0D	16#0000000B	"CenterlineControl.Window" ist negativ
INPUT_RANGE_ERROR, 16#0001	16#00000C0D	16#0000000C	"CenterlineControl.IControl" ist negativ
INPUT_RANGE_ERROR, 16#0001	16#00000C0D	16#0000000D	"CenterlineControl" ist negativ
INPUT_RANGE_ERROR, 16#0001	16#00000C0D	16#0000000E	"CenterlineControl.LowLimit" ist größer als "CenterlineControl.HighLimit"
STATE_MACHINE_ERROR, 16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.

Tab. 7-127: Generierte Fehlernummern des Funktionsbausteins IL_Amplitude-ControllerType02

7.7.13 IL_AMP_CONTROL

Kurzbeschreibung Die Struktur IL_AMP_CONTROL enthält die für die Amplitudenkorrektur erforderlichen Informationen.

Die Amplitudenkorrektur erfolgt durch einen PI-Regler der 1x pro Zyklus im Nulldurchgang regelt wenn die Differenz zwischen "AmpControl.Target" und der gemessenen Amplitude des Eingangs "Signal" größer als "AmpControl.Window" ist. Der Korrekturwert wird auf einen Wert zwischen "AmpControl.LowLimit" und "AmpControl.HighLimit" begrenzt.

Deklaration der Struktur IL_AMP_CONTROL

```

TYPE IL_AMP_CONTROL:
STRUCT
  Target:    REAL:= 0; // target Amplitude
  Ramp:     REAL:= 0.0001; // Ramp for change amplitude value /s
  Init:     REAL:= 0; // Start Amplitude
  Window:   REAL:= 0; // Window amplitudecorrection (outside this window correction)
  IControl: REAL:= 0; // Integration time Tn for the integration fraction of the PI controller
  PControl: REAL:= 0; // Proportional value for the gaining fraction of the PI controller
  HighLimit: REAL:= 1; // Upper limit of amplitudecorrection
  LowLimit: REAL:= 0; // Lower limit of amplitudecorrection
END STRUCT
END_TYPE

```

Verwendung in:

- [IL_AmplitudeControllerType01, Seite 439](#)
- [IL_AmplitudeControllerType02, Seite 442](#)

7.7.14 IL_CENTERLINE_CONTROL

Kurzbeschreibung Die Struktur IL_CENTERLINE_CONTROL enthält die für die Nulllinienkorrektur erforderlichen Informationen.

Die Nulllinienkorrektur erfolgt durch einen PI-Regler der 1x pro Zyklus im Nulldurchgang regelt, wenn die Differenz zwischen "CenterlineControl.Target" und dem gemessenen Mittelwert des Eingangs "Signal" größer als "CenterlineControl.Window" ist. Der Korrekturwert wird auf einen Wert zwischen "CenterlineControl.LowLimit" und "CenterlineControl.HighLimit" begrenzt.

Die Regeleinstellungen "CenterlineControl.PControl" und "CenterlineControl.IControl" beziehen sich auf einen Zyklus. Ein Zyklus beginnt, wenn der Eingang "Angle" von 360 auf 0 Grad springt.

Deklaration der Struktur IL_CENTERLINE_CONTROL

```

TYPE IL_CENTERLINE_CONTROL:
STRUCT
  Target:    REAL:= 0; // Centerline amplitude
  Ramp:     REAL:= 0.0001; // Ramp for change centerline value /s
  Init:     REAL:= 0; // Start centerline
  Window:   REAL:= 0; // Window centerlinecorrection (outside this window correction
  IControl: REAL:= 0; // Integration time Tn for the integration fraction of the cont
  PControl: REAL:= 0; // Proportional value for the gaining fraction of the controller
  HighLimit: REAL:= 1; // Upper limit of centerlinecorrection
  LowLimit: REAL:= 0; // Lower limit of centerlinecorrection
END_STRUCT
END_TYPE

```

Verwendung in:

- [IL_AmplitudeControllerType01, Seite 439](#)
- [IL_AmplitudeControllerType02, Seite 442](#)

7.7.15 IL_CURVETYPE

Kurzbeschreibung Der Aufzählungstyp IL_CURVETYPE enthält die möglichen Kurvenformen der Signalgeneratoren.

Deklaration der Struktur IL_CURVETYPE

```

TYPE IL_CURVETYPE :
(
  SINUSOIDAL := 1,
  DOUBLE_SIN := 2,
  TRIANGULAR := 3,
);
END_TYPE

```

SINUSOIDAL: Sinusähnliche Funktion mit möglichst langen gleichmäßig steigenden bzw. fallenden Abschnitten

DOUBLE_SINE: Doppelsinus

TRIANGULAR: Dreieck-Signal

Verwendung in:

- [IL_SignalGeneratorType01, Seite 431](#)
- [IL_SignalGeneratorType02, Seite 435](#)

7.8 Statistikbausteine (Statistics)

7.8.1 Einführung und Überblick

Dieser Abschnitt enthält die Statistikbausteine der RIL_LoopControl. Statistikbausteine stellen werten ein Datensignal aus und generieren statistische Werte.

Die RIL_LoopControl stellt die folgenden Statistikbausteine zur Verfügung:

zeitdiskret	eventgesteuert	Beschreibung
IL_HistogramType01, Seite 447	IL_HistogramType02, Seite 449	sortiert Werte in Kategorien
IL_StatisticType01, Seite 451	IL_StatisticType02, Seite 453	ermittelt gleitenden Mittel-, Min- und Max-Wert

zeitdiskret	eventgesteuert	Beschreibung
IL_SuppressionType01, Seite 455	IL_SuppressionType02, Seite 460	unterdrückt Messwerte, die eine exzessive Abweichung zum Mittelwert der Messwerte besitzen (1 Schranke)
IL_SuppressionType03, Seite 465	IL_SuppressionType04, Seite 470	unterdrückt Messwerte, die eine exzessive Abweichung zum Mittelwert der Messwerte besitzen (2 Schranken)

Tab. 7-128: Statistikbausteine der RIL_LoopControl

7.8.2 IL_HistogramType01

Kurzbeschreibung

Dieser Funktionsbaustein zählt das Auftreten von Werten in vorgegebenen gleich großen Kategorien. Dieser Baustein arbeitet kontinuierlich.

Schnittstellenbeschreibung

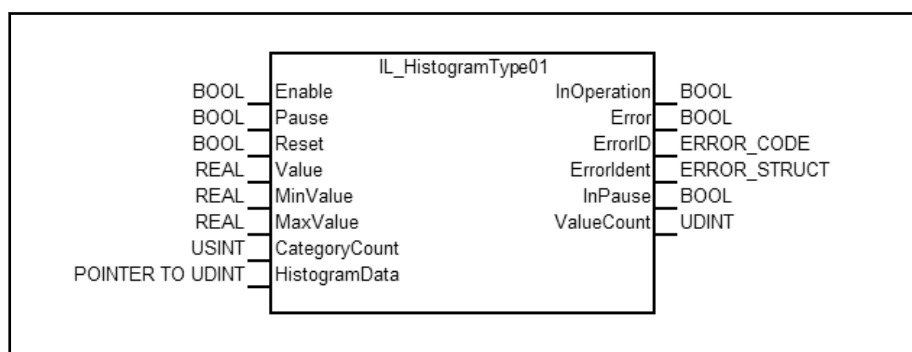


Abb. 7-78: Funktionsbaustein IL_HistogramType01

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins
	Pause	BOOL	Vorübergehende Aussetzung der Abarbeitung. Die Ausgänge werden eingefroren.
	Reset	BOOL	Rücksetzen der Werte in "HistogramData" und "ValueCount" bei steigender Flanke
	Value	REAL	In Kategorien einzuteilender Wert
	MinValue	REAL	Untere Grenze der Klasseneinteilung
	MaxValue	REAL	Obere Grenze der Klasseneinteilung
	Category-Count	USINT	Anzahl der Kategorien, in die "Value" aufgeteilt werden soll.
	Histogram-Data	POINTER TO UDINT	Adresse eines Arrays mit den Werten der Histogrammkategorien. Das Array muss mindestens "CategoryCount" Elemente haben.
VAR_OUTPUT	InOperation	BOOL	Der Baustein arbeitet, die Ausgänge sind gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes, Seite 448 .

	Name	Typ	Kommentar
	InPause	BOOL	Die Auswertung wurde angehalten. Die Werte in "HistogramData" und "ValueCount" ändern sich nicht.
	ValueCount	UDINT	Anzahl der bereits in "HistogramData" einsortierten Werte.

Tab. 7-129: Funktionsbaustein IL_HistogramType01

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Reset	BOOL			FALSE	kontinuierlich
Value	REAL	n. def.	n.def.	0.0	kontinuierlich
MinValue	REAL	n.def.	< MaxValue	0.0	steigende Flanke an Enable
MaxValue	REAL	> MinValue	n.def.	1.0	steigende Flanke an Enable
CategoryCount	USINT	1	255	10	steigende Flanke an Enable
HistogramData	POINTER TO UDINT	0	n.def.	0	steigende Flanke an Enable

Tab. 7-130: Min-, Max- und Default-Werte der Eingänge des IL_Histogram-Type01

Funktionsbeschreibung

Der Funktionsbaustein IL_HistogramType01 sortiert die Werte an "Value" zwischen "MinValue" und "MaxValue" in eine von "CategoryCount" Kategorien ein. Jede Kategorie ist gleich groß und deckt ein spezifisches nicht überlappendes Intervall ab. Die Intervallbreite ist durch $(\text{"MaxValue"} - \text{"MinValue"}) / \text{"CategoryCount"}$ eindeutig bestimmt. Jede Kategorie zählt die in sie fallenden Werte von "Value". Die Anzahl der insgesamt bereits aufgenommenen Werte wird in "ValueCount" ausgegeben.

"HistogramData" zeigt auf ein Array von UDINTs. Das Array muss mindestens "CategoryCount" Elemente vom Typ UDINT haben. Jedes Element des Arrays bildet eine Kategorie und zählt das auftreten von "Value" innerhalb der Kategorie.

Die Abarbeitung erfolgt kontinuierlich bei gesetztem "Enable"-Eingang.

Pause, InPause

Ist "Pause" gesetzt, so wird der aktuelle Zustand eingefroren. "InPause" wird gesetzt und es findet keine weitere Auswertung von "Value" statt.

Reset

Bei steigender Flanke an "Reset" werden alle Elemente des Arrays an "HistogramData" auf 0 gesetzt und der Ausgang "ValueCount" gelöscht.

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR, 16#0001	16#00000C07	16#00000001	"MinValue" ist größer als "MaxValue".
INPUT_RANGE_ERROR, 16#0006	16#00000C07	16#00000002	"CategoryCount" ist 0

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR, 16#0001	16#00000C07	16#00000003	"HistogramData" ist 0
STATE_MACHINE_ERROR, 16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.

Tab. 7-131: Generierte Fehlernummern des Funktionsbausteins IL_HistogramType01

7.8.3 IL_HistogramType02

Kurzbeschreibung

Dieser Funktionsbaustein zählt das Auftreten von Werten in vorgegebenen gleich großen Kategorien. Dieser Baustein arbeitet nur bei Änderung am Eingang "Step".

Schnittstellenbeschreibung

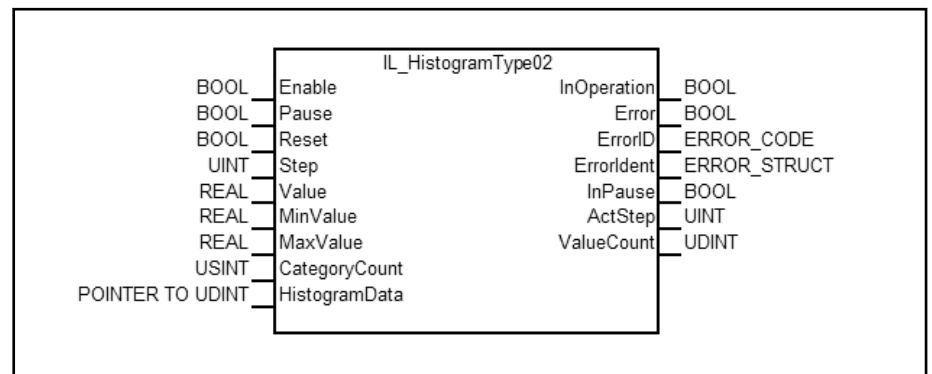


Abb. 7-79: Funktionsbaustein IL_HistogramType02

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins
	Pause	BOOL	Vorübergehende Aussetzung der Abarbeitung. Die Ausgänge werden eingefroren.
	Reset	BOOL	Rücksetzen der Werte in "HistogramData" und "ValueCount" bei steigender Flanke
	Step	UINT	Aktueller Bearbeitungsschritt, ein neuer Wert an "Value" wird nur bei Änderung an "Step" bearbeitet
	Value	REAL	In Kategorien einzuteilender Wert
	MinValue	REAL	Untere Grenze der Klasseneinteilung
	MaxValue	REAL	Obere Grenze der Klasseneinteilung
	Category-Count	USINT	Anzahl der Kategorien, in die "Value" aufgeteilt werden soll.
	Histogram-Data	POINTER TO UDINT	Adresse eines Arrays mit den Werten der Histogrammkategorien. Das Array muss mindestens "CategoryCount" Elemente haben.
VAR_OUTPUT	InOperation	BOOL	Der Baustein arbeitet, die Ausgänge sind gültig.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist.
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler, siehe Fehlercodes , Seite 450.

	Name	Typ	Kommentar
	InPause	BOOL	Die Auswertung wurde angehalten. Die Werte in "HistogramData" und "ValueCount" ändern sich nicht.
	ActStep	UINT	Letzter bearbeiteter Schritt
	ValueCount	UDINT	Anzahl der bereits in "HistogramData" einsortierten Werte.

Tab. 7-132: Funktionsbaustein IL_HistogramType02

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Reset	BOOL			FALSE	kontinuierlich
Step	UINT	0	65335	0	kontinuierlich
Value	REAL	n. def.	n.def.	0.0	bei Änderung an Step
MinValue	REAL	n.def.	< MaxValue	0.0	steigende Flanke an Enable
MaxValue	REAL	> MinValue	n.def.	1.0	steigende Flanke an Enable
CategoryCount	USINT	1	255	10	steigende Flanke an Enable
HistogramData	POINTER TO UDINT	0	n.def.	0	steigende Flanke an Enable

Tab. 7-133: Min-, Max- und Default-Werte der Eingänge des IL_HistogramType02

Funktionsbeschreibung

Der Funktionsbaustein IL_HistogramType02 sortiert die Werte an "Value" zwischen "MinValue" und "MaxValue" in eine von "CategoryCount" Kategorien ein. Jede Kategorie ist gleich groß und deckt ein spezifisches nicht überlappendes Intervall ab. Die Intervallbreite ist durch $(\text{"MaxValue"} - \text{"MinValue"}) / \text{"CategoryCount"}$ eindeutig bestimmt. Jede Kategorie zählt die in sie fallenden Werte von "Value". Die Anzahl der insgesamt bereits aufgenommenen Werte wird in "ValueCount" ausgegeben. Die Auswertung von "Value" erfolgt nur, wenn sich der Eingang "Step" geändert hat.

"HistogramData" zeigt auf ein Array von UDINTs. Das Array muss mindestens "CategoryCount" Elemente vom Typ UDINT haben. Jedes Element des Arrays bildet eine Kategorie und zählt das auftreten von "Value" innerhalb der Kategorie.

Die Abarbeitung erfolgt nur bei Änderung des Wertes am Eingang "Step."

Pause, InPause

Ist "Pause" gesetzt, so wird der aktuelle Zustand eingefroren. "InPause" wird gesetzt und es findet keine weitere Auswertung von "Value" statt.

Reset

Bei steigender Flanke an "Reset" werden alle Elemente des Arrays an "HistogramData" auf 0 gesetzt und der Ausgang "ValueCount" gelöscht.

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR, 16#0001	16#00000C07	16#00000001	"MinValue" ist größer als "MaxValue".
INPUT_RANGE_ERROR, 16#0006	16#00000C07	16#00000002	"CategoryCount" ist 0
INPUT_INVALID_ERROR, 16#0001	16#00000C07	16#00000003	"HistogramData" ist 0
STATE_MACHINE_ERROR, 16#0005	16#00000006	16#00000000	Laufzeitfehler im Funktionsbaustein: Ungültiger Zustand der Zustandsmaschine.

Tab. 7-134: Generierte Fehlernummern des Funktionsbausteins IL_Histogram-Type02

7.8.4 IL_StatisticType01

Kurzbeschreibung Der FB ermittelt zeitdiskret statistische Kenndaten von Messwerten bzw. Regelabweichungen.

Ermittelte Kenndaten:

- Mittelwert (exponentiell gleitender Mittelwert)
- Varianz (exponentiell gleitende Varianz)
- Min-/Max-Werte (exponentiell abfallend, d.h. Werte altern)

Schnittstellenbeschreibung

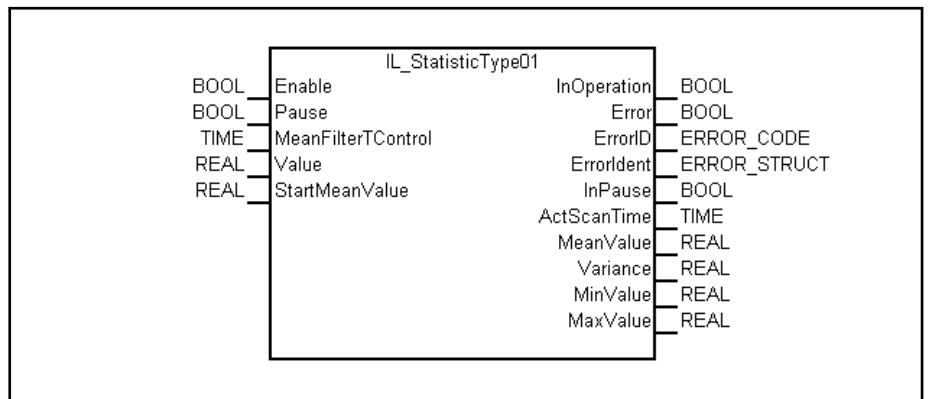


Abb. 7-80: Funktionsbaustein IL_StatisticType01

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Berechnung der Daten in jedem Zyklus solange Enable = TRUE.
	Pause	BOOL	Die Daten der Ausgänge werden nicht verändert, solange Pause = TRUE.
	MeanFilterTControl	TIME	Die Konstante "Tn" der exponentiell gleitenden Berechnung.
	Value	REAL	Eingangswert, dessen statistischen Kenngrößen ermittelt werden.
	StartMeanValue	REAL	Startwert, der bei steigender Flanke an "Enable" am Ausgang "MeanValue" übernommen wird.
VAR_OUTPUT	InOperation	BOOL	Bearbeitung läuft, Ausgänge sind gültig.
	Error	BOOL	Fehler (siehe ErrorID und ErrorIdent)
	ErrorID	ERROR_CODE	Fehlerbeschreibung

	Name	Typ	Kommentar
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerbeschreibung, siehe Fehlercodes , Seite 453.
	InPause	BOOL	Zeigt an, dass der Eingang "Pause" aktiv ist und sich die Werte in "MeanValue", "Variance", "MinValue", "MaxValue" nicht ändern.
	ActScanTime	TIME	Die Zeitdifferenz seit dem der Funktionsbaustein das letzte Mal aufgerufen wurde.
	MeanValue	REAL	Mittelwert von "Value" (exponentiell gleitende Berechnung)
	Variance	REAL	Varianz von "Value" (exponentiell gleitende Berechnung)
	MinValue	REAL	Minimalwert von "Value" (exponentiell gleitende Ermittlung)
	MaxValue	REAL	Maximalwert von "Value" (exponentiell gleitende Ermittlung)

Tab. 7-135: Funktionsbaustein IL_StatisticType01

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
MeanFilterT-Control	TIME	(2 x ActScan-Time)	n. def.	100ms	kontinuierlich
Value	REAL	n. def.	n. def.	0.0	kontinuierlich
StartMeanValue	REAL	n. def.	n. def.	0.0	steigende Flanke an Enable

Tab. 7-136: Min-, Max- und Default-Werte der Eingänge des IL_StatisticType01

Funktionsbeschreibung

Der Funktionsbaustein ermittelt statistische Kenngrößen des Eingangswertes "Value". Die Kenngrößen können z.B. dazu verwendet werden, um bei Regelvorgängen den Mittelwert, die Varianz und Min-/Max-Werte darzustellen.

"MeanValue" und "Variance"

Die Berechnung berücksichtigt länger zurückliegende Eingangswerte in geringerem Maße als neuere Eingangswerte (sog. exponentiell gleitende Berechnung). Hierdurch wird erreicht, dass die Kennwerte nicht komplett alle Messwerte seit Aktivierung des Funktionsbausteins beinhalten, sondern auch bei Veränderung der Kenngrößen des Eingangs recht gut berücksichtigen. Zum Beispiel wird bei einer sprunghaften Änderung des Mittelwertes der Eingangsgröße "Value" auf einen neuen Mittelwert dieser relativ schnell am Ausgang "MeanValue" dargestellt, ohne dass ältere Eingangswerte noch lange den neuen Mittelwert verfälschen. In gleicher Weise erfolgt die Berechnung der Varianz.

Mittels des Eingangs "MeanFilterTControl" ist die Mittelung beeinflussbar. Größere Werte berücksichtigen ältere Daten in einem stärkeren Maße zur Bestimmung der statistischen Kenngrößen. Je größer der Wert, umso träger reagieren die Ausgänge auf Änderungen der Kenngrößen des Eingangssignals und umso konstanter sind die ermittelten Kennwerte.

"MinValue" und "MaxValue"

Die Ausgänge zeigen die Tendenz der minimal und maximal aufgetretenen Werte an. Wie auch beim Mittelwert und der Varianz werden nicht alle Eingangswerte seit Aktivierung des Funktionsbausteins ausgewertet, sondern die Min- und Max-Werte werden gleitend ermittelt. D.h., dass ein aufgetretener Minimal- bzw. Maximalwert altert und durch neuere (evtl. kleinere) Werte überschrieben wird. Die Funktion ist vergleichbar mit einem Schleppzeiger, der mit einem neuen höheren Eingangswert sofort mitgenommen wird, aber dann mit einer großen Zeitkonstante wieder absinkt.

Der Eingang "MeanFilterTControl" gibt auch hier die Trägheit des Ausgangs in Bezug auf das Absinken der Werte "MinValue" und "MaxValue" an.

Fehlercodes Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINNE_ERROR(16#0005)	16#0006	16#0000	Ungültiger Zustand des Funktionsbausteins
INPUT_RANGE_ERROR	16#0C02	16#0005	Wert von "MeanFilterTControl" < (2 x "ActScanTime")

Tab. 7-137: Generierte Fehlernummern des Funktionsbausteins IL_StatisticType01

7.8.5 IL_StatisticType02

Kurzbeschreibung Der FB ermittelt eventgesteuert statistische Kenndaten von Messwerten bzw. Regelabweichungen.

Ermittelte Kenndaten:

- Mittelwert (exponentiell gleitender Mittelwert)
- Varianz (exponentiell gleitende Varianz)
- Min-/Max-Werte (exponentiell abfallend, d.h. Werte altern)

Schnittstellenbeschreibung

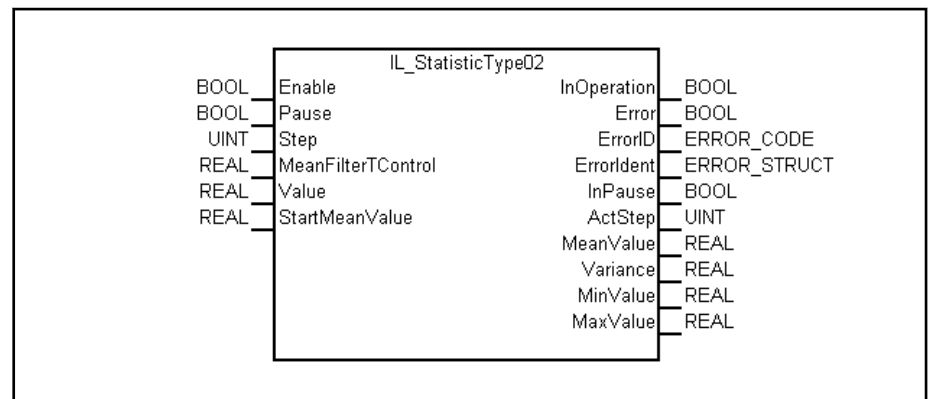


Abb. 7-81: Funktionsbaustein IL_StatisticType02

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Berechnung der Daten bei jeder Änderung von "Step", solange Enable = TRUE
	Pause	BOOL	Die Daten von OutputValue werden nicht verändert, solange "Pause" = TRUE. Wird währenddessen der "Step"-Eingang inkrementiert, wird der aktuell anliegende Schritt am Ausgang "ActStep" ausgegeben.
	Step	UINT	aktueller Berechnungsschritt

	Name	Typ	Kommentar
	MeanFilterTControl	REAL	Die Konstante "Tn" der exponentiell gleitenden Berechnung bezogen auf "Step".
	Value	REAL	Eingangswert, dessen statistischen Kenngrößen ermittelt werden.
	StartMeanValue	REAL	Startwert, der bei steigender Flanke an "Enable" am Ausgang "MeanValue" übernommen wird.
VAR_OUTPUT	InOperation	BOOL	Fehlerfreie Bearbeitung, Ausgangsdaten sind gültig .
	Error	BOOL	Fehler (siehe ErrorID und ErrorIdent)
	ErrorID	ERROR_CODE	Fehlerbeschreibung
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerbeschreibung, siehe Fehlercodes, Seite 455 .
	InPause	BOOL	Zeigt an, dass der Eingang "Pause" aktiv ist und sich die Werte in "MeanValue", "Variance", "MinValue" und "MaxValue" nicht ändern.
	ActStep	UINT	"Step"-Wert, auf den sich "OutputValue" bezieht. "ActStep" wird bei steigender Flanke von "Enable" mit dem Wert von "Step" initialisiert
	MeanValue	REAL	Mittelwert von "Value" (exponentiell gleitende Berechnung)
	Variance	REAL	Varianz von "Value" (exponentiell gleitende Berechnung)
	MinValue	REAL	Minimalwert von "Value" (exponentiell gleitende Ermittlung)
	MaxValue	REAL	Maximalwert von "Value" (exponentiell gleitende Ermittlung)

Tab. 7-138: Funktionsbaustein IL_StatisticType02

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Step	UINT	0	16#FFFF	0	kontinuierlich
MeanFilterT-Control	REAL	2.0	n.def.	100.0	kontinuierlich
Value	REAL	n.def.	n.def.	0.0	kontinuierlich
StartMeanValue	REAL	n.def.	n.def.	0.0	steigende Flanke an Enable

Tab. 7-139: Min-, Max- und Default-Werte der Eingänge des IL_StatisticType02

Funktionsbeschreibung

Der Funktionsbaustein ermittelt statistische Kenngrößen des Eingangswertes "Value". Die Kenngrößen können z.B. dazu verwendet werden, um bei Regelvorgängen den Mittelwert, die Varianz und Min-/Max-Werte darzustellen.

"MeanValue" und "Variance"

Die Berechnung berücksichtigt länger zurückliegende Eingangswerte in geringerem Maße als neuere Eingangswerte (sog. exponentiell gleitende Berechnung). Hierdurch wird erreicht, dass die Kennwerte nicht komplett alle

Messwerte seit Aktivierung des Funktionsbausteins beinhalten, sondern auch bei Veränderung der Kenngrößen des Eingangs recht gut berücksichtigen. Zum Beispiel wird bei einer sprunghaften Änderung des Mittelwertes der Eingangsgröße "Value" auf einen neuen Mittelwert dieser relativ schnell am Ausgang "MeanValue" dargestellt, ohne dass ältere Eingangswerte noch lange den neuen Mittelwert verfälschen. In gleicher Weise erfolgt die Berechnung der Varianz.

Mittels des Eingangs "MeanFilterTControl" ist die Mittelung beeinflussbar. Größere Werte berücksichtigen ältere Daten in einem stärkeren Maße zur Bestimmung der statistischen Kenngrößen. Je größer der Wert, umso träger reagieren die Ausgänge auf Änderungen der Kenngrößen des Eingangssignals und umso konstanter sind die ermittelten Kennwerte.

"MinValue" und "MaxValue"

Die Ausgänge zeigen die Tendenz der minimal und maximal aufgetretenen Werte an. Wie auch beim Mittelwert und der Varianz werden nicht alle Eingangswerte seit Aktivierung des Funktionsbausteins ausgewertet, sondern die Min- und Max-Werte werden gleitend ermittelt. D.h., dass ein aufgetretener Minimal- bzw. Maximalwert altert und wird durch neuere (evtl. kleinere) Werte überschrieben. Die Funktion ist vergleichbar mit einem Schleppzeiger, der mit einem neuen höheren Eingangswert sofort mitgenommen wird, aber dann mit einer großen Zeitkonstante wieder absinkt.

Der Eingang "MeanFilterTControl" gibt auch hier die Trägheit des Ausgangs in Bezug auf das Absinken der Werte "MinValue" und "MaxValue" an.

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR(16#0005)	16#0006	16#0000	Ungültiger Zustand des Funktionsbausteins
INPUT_RANGE_ERROR	16#0C02	16#0005	Wert von "MeanFilterTControl" muss größer gleich 2.0 sein.

Tab. 7-140: Generierte Fehlernummern des Funktionsbausteins IL_StatisticType02

7.8.6 IL_SuppressionType01

Kurzbeschreibung

Dieser zeitdiskrete Funktionsbaustein unterdrückt Messwerte, die eine exzessive Abweichung zum Mittelwert der Messwerte besitzen. Die Messwerte, die außerhalb der parametrierbaren Schwellwerten liegen, werden nicht weitergegeben, sondern stattdessen wird der gefilterte Wert ausgegeben.

Schnittstellenbeschreibung

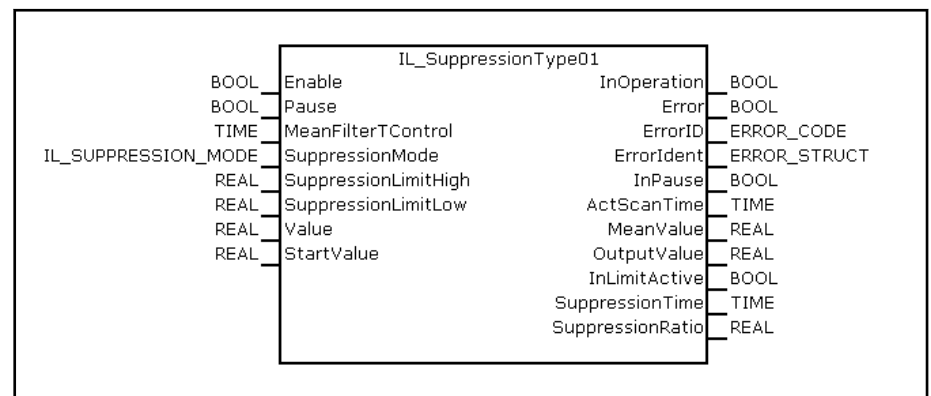


Abb. 7-82: Funktionsbaustein IL_SuppressionType01

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Berechnung der Daten in jedem Zyklus solange "Enable" = TRUE.
	Pause	BOOL	Die Daten der Ausgänge werden nicht verändert, solange "Pause" = TRUE.
	MeanFilterTControl	TIME	Die Konstante "Tn" des (PT1-) Mittelwertfilters.
	SuppressionMode	IL_SUPPRESSION_MODE	Beschreibt die Funktion der Eingänge "SuppressionLimitLow" und "SuppressionLimitHigh". IL_SUPPRESSION_MODE, Seite 512
	SuppressionLimitHigh	REAL	Oberer Wert des Unterdrückungsfensters (wird zum Mittelwert addiert).
	SuppressionLimitLow	REAL	Unterer Wert des Unterdrückungsfensters (wird vom Mittelwert subtrahiert).
	Value	REAL	Der aktuelle Istwert des Prozesses.
	StartValue	REAL	Startwert, der bei steigender Flanke an "Enable" am Ausgang "OutputValue" angenommen wird.
VAR_OUTPUT	InOperation	BOOL	Fehlerfreie Bearbeitung, Ausgangsdaten sind gültig .
	Error	BOOL	Fehler (siehe ErrorID und ErrorIdent)
	ErrorID	ERROR_CODE	Fehlerbeschreibung
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerbeschreibung, siehe Fehlercodes, Seite 459 .
	InPause	BOOL	Zeigt an, dass der Eingang „Pause“ aktiv ist und sich die Werte der Ausgänge nicht ändern.
	ActScanTime	TIME	Die Zeitdifferenz seit dem der Funktionsbaustein das letzte Mal aufgerufen wurde.
	MeanValue	REAL	Gefilterter Mittelwert, gegen den "Value" verglichen wird.
	OutputValue	REAL	Der berechnete Ausgangswert.
	InLimitActive	BOOL	Zeigt an, dass das Unterdrückungsfenster aktiv ist.
	SuppressionTime	TIME	Zeitähler, wie lange Messwerte unterdrückt wurden. Der Wert wird bei steigender Flanke an "Enable" gelöscht (T#0s) und zählt dann bei aktiver Unterdrückung hoch. Sobald wieder ein Eingangswert vorhanden ist, der nicht unterdrückt wird, wird der Ausgang wieder gelöscht. Bei einem Überlauf fällt der Zähler automatisch auf "0" zurück.
	SuppressionRatio	REAL	Prozentualer Anteil aller unterdrückten Werte (0-100%).

Tab. 7-141: Funktionsbaustein IL_SuppressionType01

IL_SUPPRESSION_MODE ist ein Aufzählungstyp und beschreibt die Funktion der Eingänge "SuppressionLimitLow" und "SuppressionLimitHigh" mit folgender Belegung:

Element	Beschreibung
IL_SUPPRESSION_PERCENTAGE	Werte sind prozentual zum gemittelten Eingangswert
IL_SUPPRESSION_ABSOLUTE_VALUE	Werte sind absolute Werte zum gemittelten Eingangswert

Tab. 7-142: Aufbau des Aufzählungstyps IL_SUPPRESSION_MODE

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
MeanFilterTControl	TIME	0.0	n. def.	100ms	kontinuierlich
SuppressionMode	IL_SUPPRESSIO- N_MODE			IL_SUPPRESSIO- N_PERCENTAGE	steigende Flanke an Enable
SuppressionLimitHigh	REAL	0.0	n. def.	20.0	kontinuierlich
SuppressionLimitLow	REAL	0.0	n. def.	20.0	kontinuierlich
Value	REAL	n. def.	n. def.	0.0	kontinuierlich
StartValue	REAL	n. def.	n. def.	0.0	steigende Flanke an Enable

Tab. 7-143: Min-, Max- und Default-Werte der Eingänge des IL_Suppression-Type01

Funktionsbeschreibung

Bei Regelvorgängen ist es oft notwendig, Messwerte, die offensichtlich stark von dem erwarteten Wert abweichen, auszufiltern (unterdrücken). Hierfür ist im FB ein Unterdrückungsfenster implementiert, der mit dem Mittelwert des zu filternden Wertes mitläuft und ein Toleranzfenster um diesen Mittelwert bildet. Jeder Messwert "Value" wird dabei mit dem gefilterten Messwert verglichen (Filterung der Messwerte mit einem PT1-Filter). Befindet sich der jeweilige Messwert außerhalb eines um den Mittelwert gebildeten Unterdrückungsfensters (Toleranzfenster, wird durch "SuppressionLimitLow" bzw. "SuppressionLimitHigh" definiert), so wird er nicht an den Bausteinoutput "OutputValue" ausgegeben, sondern unterdrückt; d.h. am Ausgang "OutputValue" wird nicht "Value", sondern der Mittelwert ausgegeben.

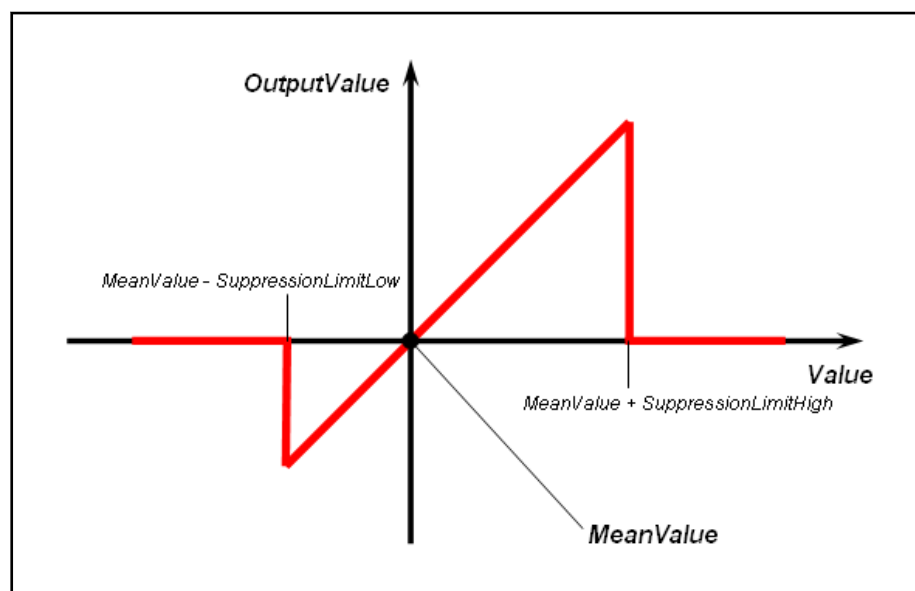


Abb. 7-83: Prinzip des Unterdrückungsfensters

Interne Struktur des FBs

Der FB besteht aus einem Mittelwertfilter und einem Block, der die Unterdrückung der Messwerte durchführt. Die Mittelwertfilterung wird durch ein PT1-Glied durchgeführt. Die Filterzeitkonstante sollte dabei groß gegenüber der Prozesszeitkonstante gewählt werden, damit eine Mittelwertfilterung korrekt

wirken kann. Der gefilterte Mittelwert kann über den Ausgang "MeanValue" diagnostiziert werden.

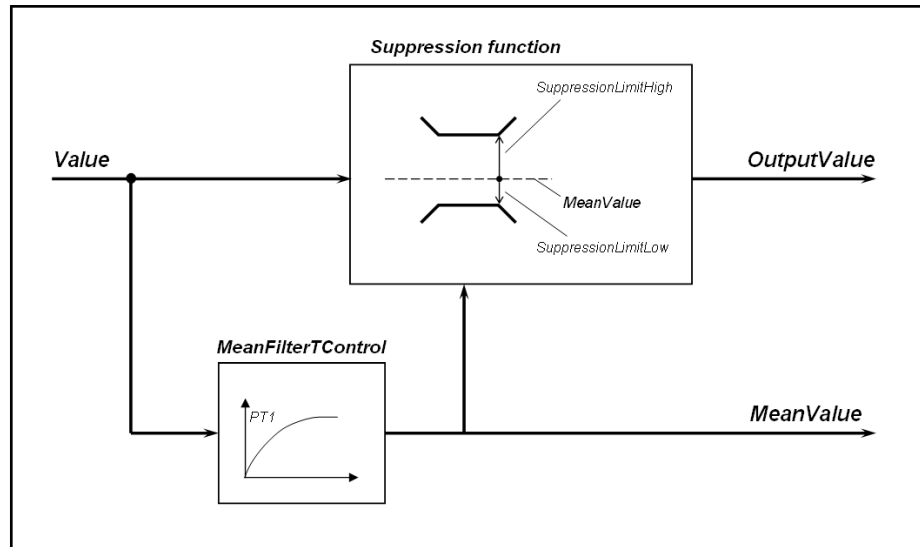


Abb. 7-84: Interne Struktur IL_SuppressionType01

Ausfensterungs-Modi

Bei der Ausfensterung sind die obere ("SuppressionLimitHigh") und untere Schwelle ("SuppressionLimitLow") getrennt konfigurierbar. Es stehen zwei Modi zur Verfügung:

- **Prozentual auf den Mittelwert bezogen (IL_SUPPRESSION_PERCENTAGE):** die Fensterbreite wird durch einen Prozentsatz des Mittelwertes vorgegeben, d.h. "SuppressionLimitHigh" gibt einen Prozentanteil (vom Absolutwert des Mittelwertes) an, der zum Mittelwert addiert wird, um die obere Schwelle zu erhalten und "SuppressionLimitLow" gibt einen Prozentanteil (vom Absolutwert des Mittelwertes) an, der vom Mittelwert subtrahiert wird, für die untere Schwelle. Es ist zu berücksichtigen, dass die Größe des Fensters sehr klein werden kann, wenn der Betrag des Mittelwertes sehr klein wird.
- **Absolutes Fenster auf den Mittelwert bezogen (IL_SUPPRESSION_ABSOLUTE_VALUE):** die Werte der Eingänge "SuppressionLimitLow" und "SuppressionLimitHigh" geben die Größe des Fensters um den Mittelwert direkt vor.

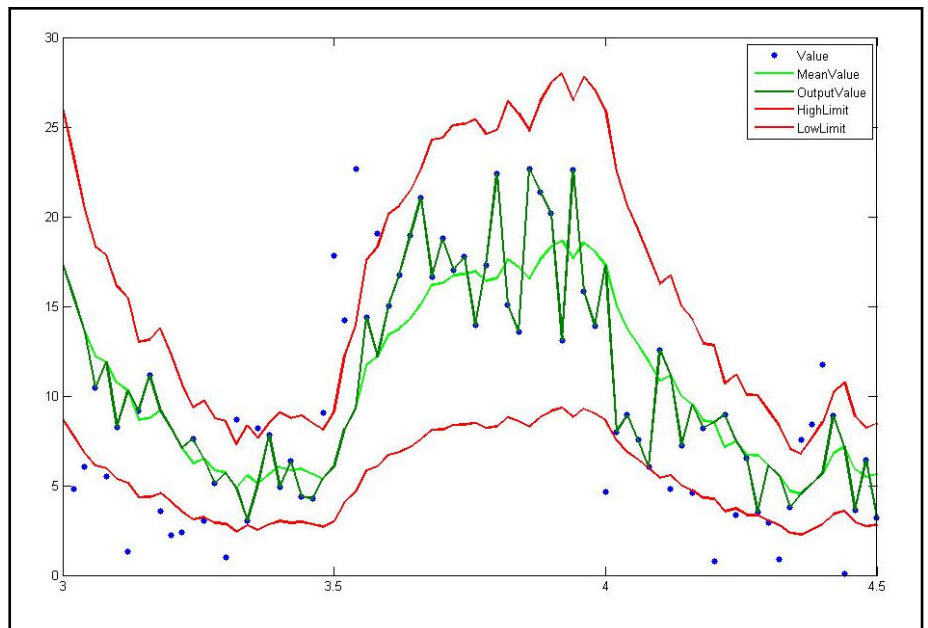


Abb. 7-85: Beispiel einer Ausfensterung "prozentual auf den Mittelwert bezogen"

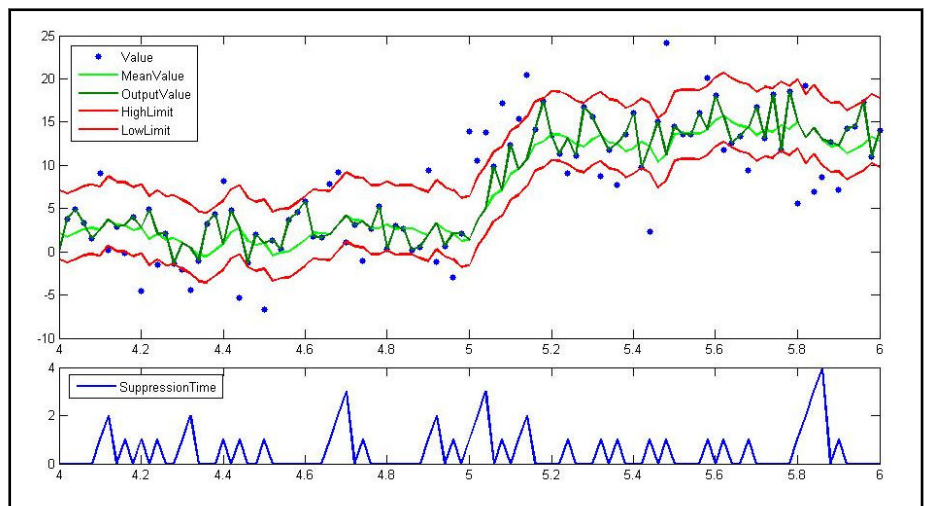


Abb. 7-86: Beispiel einer Ausfensterung „mit absolutem Fenster um den Mittelwert“

Diagnose der Ausfensterung

Zur Diagnose kann die Funktion der Ausfensterung mittels der Ausgänge "SuppressionTime" und "SuppressionRatio" beobachtet werden. "SuppressionTime" ist dabei ein Zeitzähler wie lange Messwerte unterdrückt wurden, d.h. er zählt bei aktiver Unterdrückung hoch und wird auf "0" zurückgesetzt, wenn wieder ein Eingangswert vorhanden ist, der nicht unterdrückt wird. Er wird ebenfalls auf "0" gesetzt bei einer steigenden Flanke an "Enable", sowie bei einem Überlauf. "SuppressionRatio" ist der prozentuale Zeitanteil aller unterdrückten Werte.

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR (16#0005)	16#0006	16#0000	Ungültiger Zustand des Funktionsbausteins
INPUT_INVALID_ERROR	16#0C09	16#0002	Wert von "SuppressionLimitHigh" ist kleiner als 0.0
INPUT_INVALID_ERROR	16#0C09	16#0003	Wert von "SuppressionLimitLow" ist kleiner als 0.0

Tab. 7-144: Generierte Fehlernummern des Funktionsbausteins IL_SuppressionType01

7.8.7 IL_SuppressionType02

Kurzbeschreibung

Dieser eventgesteuerte Funktionsbaustein unterdrückt Messwerte, die eine exzessive Abweichung zum Mittelwert der Messwerte besitzen. Die Messwerte, die außerhalb der parametrierbaren Schwellwerten liegen, werden nicht weitergegeben, sondern stattdessen wird der gefilterte Wert ausgegeben.

Schnittstellenbeschreibung

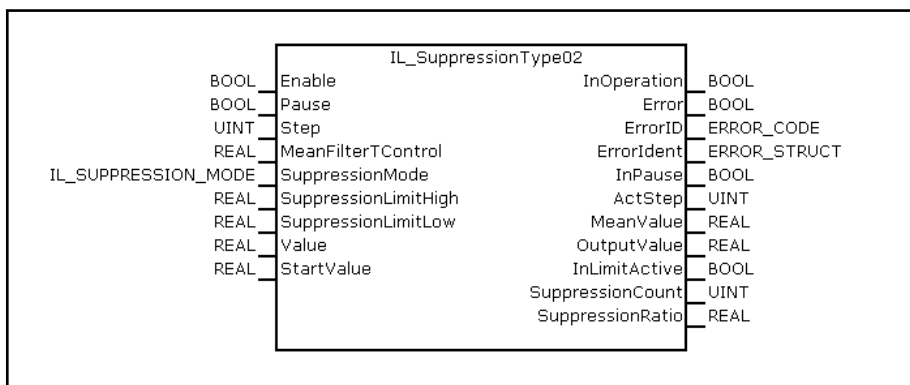


Abb. 7-87: Funktionsbaustein IL_SuppressionType02

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Berechnung der Daten in jedem Zyklus (Step) solange "Enable"=TRUE.
	Pause	BOOL	Die Daten der Ausgänge werden nicht verändert, solange "Pause" = TRUE. Wird währenddessen der "Step"-Eingang inkrementiert, wird der aktuell anliegende Schritt am Ausgang "ActSteps" ausgegeben.
	Step	UINT	Aktueller Berechnungsschritt .
	MeanFilterTControl	REAL	Die Konstante "Tn" des Mittelwertfilter-Gliedes bezogen auf "Step".
	SuppressionMode	IL_SUPPRESSION_MODE	Beschreibt die Funktion der Eingänge "SuppressionLimitLow" und "SuppressionLimitHigh". IL_SUPPRESSION_MODE, Seite 512
	SuppressionLimitHigh	REAL	Oberer Wert des Unterdrückungsfensters (wird zum Mittelwert addiert).
	SuppressionLimitLow	REAL	Unterer Wert des Unterdrückungsfensters (wird vom Mittelwert subtrahiert).
	Value	REAL	Der aktuelle Istwert des Prozesses.
	StartValue	REAL	Startwert, der bei steigender Flanke an "Enable" am Ausgang "OutputValue" angenommen wird.

	Name	Typ	Kommentar
VAR_OUTPUT	InOperation	BOOL	Fehlerfreie Bearbeitung, Ausgangsdaten sind gültig .
	Error	BOOL	Fehler (siehe ErrorID und ErrorIdent)
	ErrorID	ERROR_CODE	Fehlerbeschreibung
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerbeschreibung, siehe Fehlercodes , Seite 465.
	InPause	BOOL	Zeigt an, dass der Eingang „Pause“ aktiv ist und sich die Werte der Ausgänge nicht ändern.
	ActStep	UINT	Step Wert, auf den sich "OutputValue" bezieht. "Act-Step" wird bei steigender Flanke von "Enable" mit dem Wert von "Step" initialisiert.
	MeanValue	REAL	Gefilterter Mittelwert, gegen den "Value" verglichen wird.
	OutputValue	REAL	Der berechnete Ausgangswert.
	InLimitActive	BOOL	Zeigt an, dass das Unterdrückungsfenster aktiv ist.
	SuppressionCount	UINT	Zähler, der die Anzahl der unterdrückten Werte zählt. Der Wert wird bei steigender Flanke an "Enable" gelöscht und zählt dann bei aktiver Unterdrückung hoch. Sobald wieder ein Eingangswert vorhanden ist, der nicht unterdrückt wird, wird der Ausgang wieder gelöscht. Bei einem Überlauf (16xFFFF) fällt der Zähler automatisch auf "0" zurück.
	SuppressionRatio	REAL	Prozentualer Anteil aller unterdrückten Werte (0-100%).

Tab. 7-145: Funktionsbaustein IL_SuppressionType02

IL_SUPPRESSION_MODE ist ein Aufzählungstyp und beschreibt die Funktion der Eingänge "SuppressionLimitLow" und "SuppressionLimitHigh" mit folgender Belegung:

Element	Beschreibung
IL_SUPPRESSION_PERCENTAGE	Werte sind prozentual zum gemittelten Eingangswert
IL_SUPPRESSION_ABSOLUTE_VALUE	Werte sind absolute Werte zum gemittelten Eingangswert

Tab. 7-146: Aufbau des Aufzählungstyps IL_SUPPRESSION_MODE

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Step	UINT	0	16#FFFF	0	kontinuierlich
MeanFilterTControl	TIME	0.0	n. def.	100ms	kontinuierlich
SuppressionMode	IL_SUPPRESSION_MODE			IL_SUPPRESSION_PERCENTAGE	steigende Flanke an Enable
SuppressionLimitHigh	REAL	0.0	n. def.	20.0	kontinuierlich
SuppressionLimitLow	REAL	0.0	n. def.	20.0	kontinuierlich

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Value	REAL	n. def.	n. def.	0.0	kontinuierlich
StartValue	REAL	n. def.	n. def.	0.0	steigende Flanke an Enable

Tab. 7-147: Min-, Max- und Default-Werte der Eingänge des IL_Suppression-Type02

Funktionsbeschreibung

Bei Regelvorgängen ist es oft notwendig, Messwerte, die offensichtlich stark von dem erwarteten Wert abweichen, auszufiltern (unterdrücken). Hierfür ist im FB ein Unterdrückungsfenster implementiert, der mit dem Mittelwert des zu filternden Wertes mitläuft und ein Toleranzfenster um diesen Mittelwert bildet. Jeder Messwert "Value" wird dabei mit dem gefilterten Messwert verglichen (Filterung der Messwerte mit einem PT1-Filter). Befindet sich der jeweilige Messwert außerhalb eines um den Mittelwert gebildeten Unterdrückungsfensters (Toleranzfenster, wird durch "SuppressionLimitLow" bzw. "SuppressionLimitHigh" definiert), so wird er nicht an den Bausteinoutput "OutputValue" ausgegeben, sondern unterdrückt; d.h. am Ausgang "OutputValue" wird nicht "Value", sondern der Mittelwert ausgegeben.

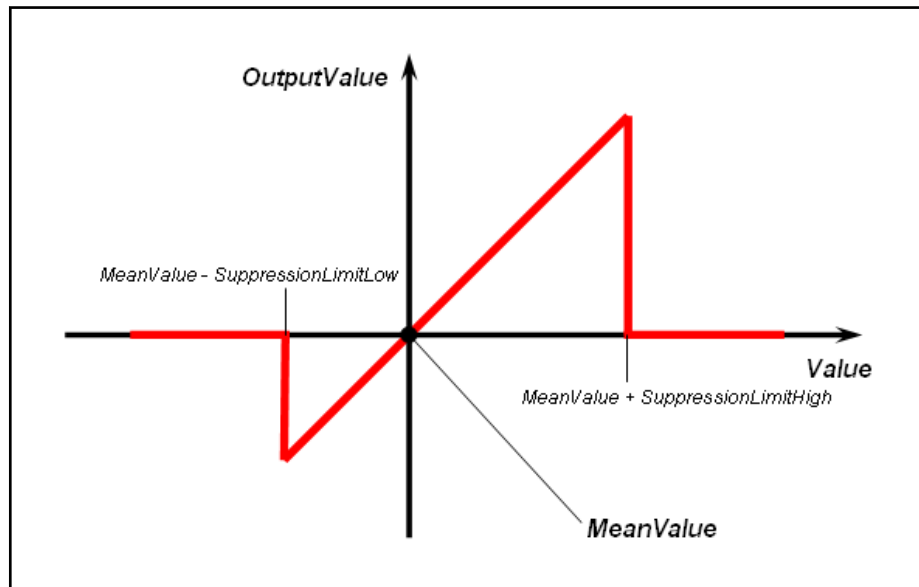


Abb. 7-88: Prinzip des Unterdrückungsfensters

Interne Struktur des FBs

Der FB besteht aus einem Mittelwertfilter und einem Block, der die Unterdrückung der Messwerte durchführt. Die Mittelwertfilterung wird durch ein PT1-Glied durchgeführt. Die Filterzeitkonstante sollte dabei groß gegenüber der Prozesszeitkonstante gewählt werden, damit eine Mittelwertfilterung korrekt wirken kann. Der gefilterte Mittelwert kann über den Ausgang "MeanValue" diagnostiziert werden.

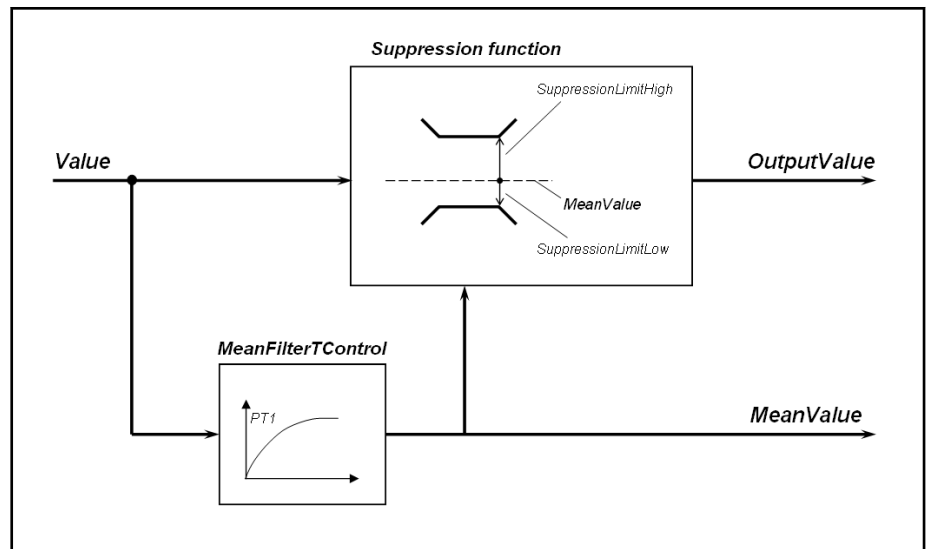


Abb. 7-89: Interne Struktur IL_SuppressionType02

Ausfensterungs-Modi

Bei der Ausfensterung sind die obere ("SuppressionLimitHigh") und untere Schwelle ("SuppressionLimitLow") getrennt konfigurierbar. Es stehen zwei Modi zur Verfügung:

- **Prozentual auf den Mittelwert bezogen (IL_SUPPRESSION_PERCENTAGE):** die Fensterbreite wird durch einen Prozentsatz des Mittelwertes vorgegeben, d.h. "SuppressionLimitHigh" gibt einen Prozentanteil (vom Absolutwert des Mittelwertes) an, der zum Mittelwert addiert wird, um die obere Schwelle zu erhalten und "SuppressionLimitLow" gibt einen Prozentanteil (vom Absolutwert des Mittelwertes) an, der vom Mittelwert subtrahiert wird, für die untere Schwelle. Es ist zu berücksichtigen, dass die Größe des Fensters sehr klein werden kann, wenn der Betrag des Mittelwertes sehr klein wird.
- **Absolutes Fenster auf den Mittelwert bezogen (IL_SUPPRESSION_ABSOLUTE_VALUE):** die Werte der Eingänge "SuppressionLimitLow" und "SuppressionLimitHigh" geben die Größe des Fensters um den Mittelwert direkt vor.

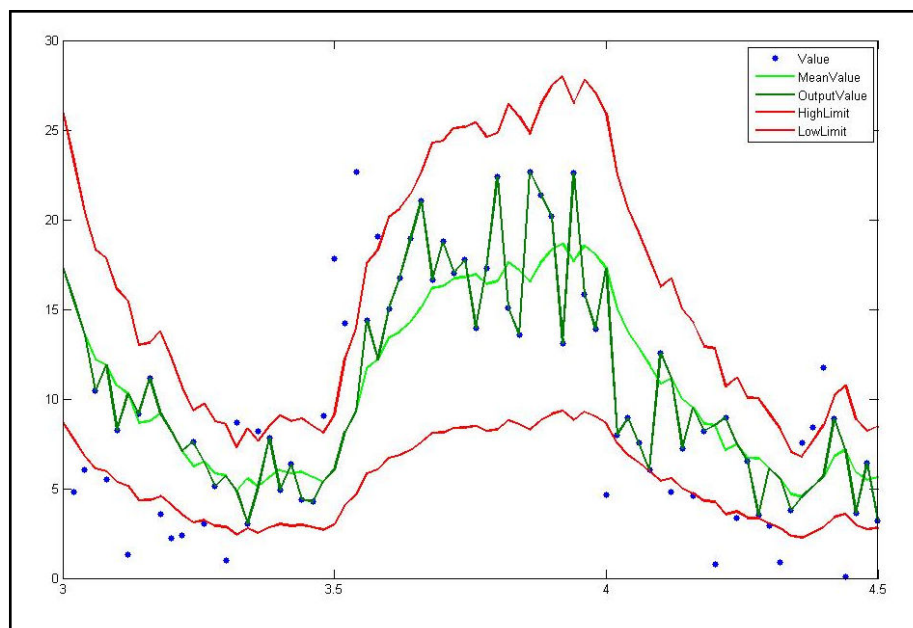


Abb. 7-90: Beispiel einer Ausfensterung "prozentual auf den Mittelwert bezogen"

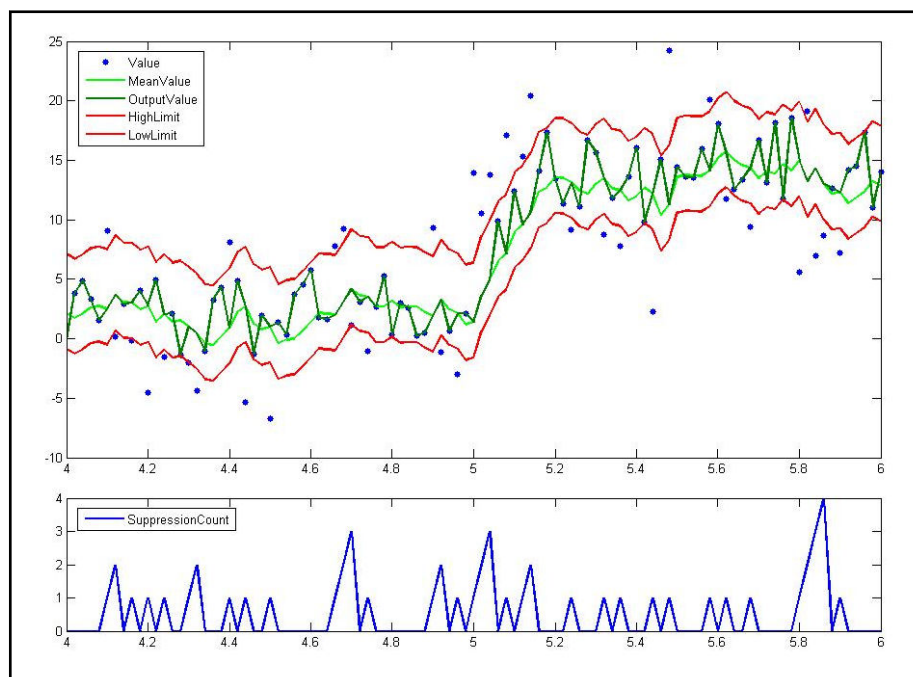


Abb. 7-91: Beispiel einer Ausfensterung „mit absolutem Fenster um den Mittelwert“

Diagnose der Ausfensterung

Zur Diagnose kann die Funktion der Ausfensterung mittels der Ausgänge "SuppressionCount" und "SuppressionRatio" beobachtet werden. "SuppressionCount" ist dabei ein Zähler wie lange Messwerte unterdrückt wurden, d.h. er zählt bei aktiver Unterdrückung hoch und wird auf "0" zurückgesetzt, wenn wieder ein Eingangswert vorhanden ist, der nicht unterdrückt wird. Er wird ebenfalls auf "0" gesetzt bei einer steigenden Flanke an "Enable", sowie bei einem Überlauf. "SuppressionRatio" ist der prozentuale Zeitanteil aller unterdrückten Werte.

Fehlercodes Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR (16#0005)	16#0006	16#0000	Ungültiger Zustand des Funktionsbausteins
INPUT_INVALID_ERROR	16#0C09	16#0001	Wert von "MeanFilterTControl" ist kleiner als 0.0
INPUT_INVALID_ERROR	16#0C09	16#0002	Wert von "SuppressionLimitHigh" ist kleiner als 0.0
INPUT_INVALID_ERROR	16#0C09	16#0003	Wert von "SuppressionLimitLow" ist kleiner als 0.0

Tab. 7-148: Generierte Fehlernummern des Funktionsbausteins IL_SuppressionType02

7.8.8 IL_SuppressionType03

Kurzbeschreibung

Dieser zeitdiskrete Funktionsbaustein unterdrückt Messwerte, die eine exzessive Abweichung zum Mittelwert der Messwerte besitzen. Die Messwerte, die außerhalb der parametrierbaren Schwellwerten liegen, werden nicht weitergegeben, sondern stattdessen wird der gefilterte Wert ausgegeben. Der IL_SuppressionType03 besitzt im Vergleich zum Type01 funktionelle Erweiterungen, u.a. zwei umschaltbare Grenzen für das Unterdrückungsfenster, sowie eine zusätzliche Unterdrückungsfunktion.

Schnittstellenbeschreibung

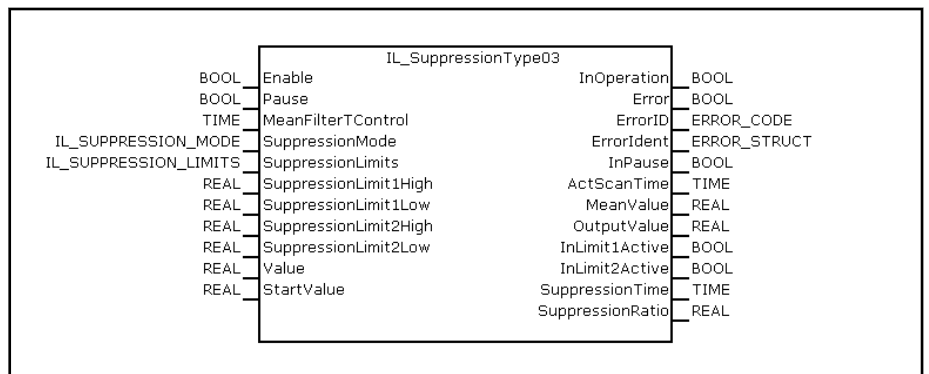


Abb. 7-92: Funktionsbaustein IL_SuppressionType03

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Berechnung der Daten in jedem Zyklus solange "Enable" = TRUE.
	Pause	BOOL	Die Daten der Ausgänge werden nicht verändert, solange "Pause" = TRUE.
	MeanFilterTControl	TIME	Die Konstante "Tn" des (PT1-) Mittelwertfilters.
	SuppressionMode	IL_SUPPRESSION_MODE	Beschreibt die Funktion der Eingänge "SuppressionLimitLow" und "SuppressionLimitHigh": - IL_SUPPRESSION_PERCENTAGE - IL_SUPPRESSION_ABSOLUTE_VALUE - IL_SUPPRESSION_STATISTICAL IL_SUPPRESSION_MODE, Seite 512

	Name	Typ	Kommentar
	SuppressionLimits	IL_SUPPRESSIO- N_LIMITS	Auswahl des aktiven Unterdrückungsfensters: • IL_SUPPRESSION_OFF • IL_SUPPRESSION_LIMIT_1 • IL_SUPPRESSION_LIMIT_2 IL_SUPPRESSION_LIMITS, Seite 511
	SuppressionLimit1High	REAL	Unterer Wert des Unterdrückungsfensters #1.
	SuppressionLimit1Low	REAL	Oberer Wert des Unterdrückungsfensters #1.
	SuppressionLimit2High	REAL	Unterer Wert des Unterdrückungsfensters #2.
	SuppressionLimit2Low	REAL	Oberer Wert des Unterdrückungsfensters #2.
	Value	REAL	Der aktuelle Istwert des Prozesses.
	StartValue	REAL	Startwert, der bei steigender Flanke an "Enable" am Ausgang "OutputValue" angenommen wird.
VAR_OUTPUT	InOperation	BOOL	Fehlerfreie Bearbeitung, Ausgangsdaten sind gültig .
	Error	BOOL	Fehler (siehe ErrorID und ErrorIdent)
	ErrorID	ERROR_CODE	Fehlerbeschreibung
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerbeschreibung, siehe Fehlercodes, Seite 470 .
	InPause	BOOL	Zeigt an, dass der Eingang „Pause“ aktiv ist und sich die Werte der Ausgänge nicht ändern.
	ActScanTime	TIME	Die Zeitdifferenz seit dem der Funktionsbaustein das letzte Mal aufgerufen wurde.
	MeanValue	REAL	Gefilterter Mittelwert, gegen den "Value" verglichen wird.
	OutputValue	REAL	Der berechnete Ausgangswert.
	InLimit1Active	BOOL	Zeigt an, dass Unterdrückungsfenster #1 aktiv ist.
	InLimit2Active	BOOL	Zeigt an, dass Unterdrückungsfenster #2 aktiv ist.
	SuppressionTime	TIME	Zeitzähler, wie lange Messwerte unterdrückt wurden. Der Wert wird bei steigender Flanke an "Enable" gelöscht (T#0s) und zählt dann bei aktiver Unterdrückung hoch. Sobald wieder ein Eingangswert vorhanden ist, der nicht unterdrückt wird, wird der Ausgang wieder gelöscht. Bei einem Überlauf fällt der Zähler automatisch auf "0" zurück.
	SuppressionRatio	REAL	Prozentualer Anteil aller unterdrückten Werte (0-100%).

Tab. 7-149: Funktionsbaustein IL_SuppressionType03

IL_SUPPRESSION_MODE ist ein Aufzählungstyp und beschreibt die Funktion der Eingänge "SuppressionLimitLow1/2" und "SuppressionLimitHigh1/2" mit folgender Belegung:

Element	Beschreibung
IL_SUPPRESSION_PERCENTAGE	Werte sind prozentual zum gemittelten Eingangswert
IL_SUPPRESSION_ABSOLUTE_VALUE	Werte sind absolute Werte zum gemittelten Eingangswert
IL_SUPPRESSION_STATISTICAL	Werte sind statistisch zur Standardabweichung σ des Eingangswertes (z.B. $0.5 \cdot \sigma$, $1.0 \cdot \sigma$, $4.0 \cdot \sigma$,...)

Tab. 7-150: Aufbau des Aufzählungstyps IL_SUPPRESSION_MODE

IL_SUPPRESSION_LIMITS ist ein Aufzählungstyp und aktiviert die Unterdrückungsfenster mit folgender Belegung:

Element	Beschreibung
IL_SUPPRESSION_OFF	Unterdrückung ist deaktiviert, alle Werte kommen unverändert an den Ausgang.
IL_SUPPRESSION_LIMIT_1	Unterdrückungsfenster #1 aktiv ("SuppressionLimit1Low" und "SuppressionLimit1High")
IL_SUPPRESSION_LIMIT_2	Unterdrückungsfenster #2 aktiv ("SuppressionLimit2Low" und "SuppressionLimit2High")

Tab. 7-151: Aufbau des Aufzählungstyps IL_SUPPRESSION_LIMITS

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
MeanFilterTControl	TIME	0.0	n. def.	100ms	kontinuierlich
SuppressionMode	IL_SUPPRESSION_MODE			IL_SUPPRESSION_PERCENTAGE	steigende Flanke an Enable
SuppressionLimit	IL_SUPPRESSION_LIMITS			IL_SUPPRESSION_LIMIT_1	kontinuierlich
SuppressionLimit1High	REAL	0.0	n. def.	5.0	kontinuierlich
SuppressionLimit1Low	REAL	0.0	n. def.	5.0	kontinuierlich
SuppressionLimit2High	REAL	0.0	n. def.	20.0	kontinuierlich
SuppressionLimit2Low	REAL	0.0	n. def.	20.0	kontinuierlich
Value	REAL	n. def.	n. def.	0.0	kontinuierlich
StartValue	REAL	n. def.	n. def.	0.0	steigende Flanke an Enable

Tab. 7-152: Min-, Max- und Default-Werte der Eingänge des IL_Suppression-Type03

Funktionsbeschreibung

Bei Regelvorgängen ist es oft notwendig, Messwerte, die offensichtlich stark von dem erwarteten Wert abweichen, auszufiltern (unterdrücken). Hierfür ist im FB ein Unterdrückungsfenster implementiert, der mit dem Mittelwert des zu filternden Wertes mitläuft und ein Toleranzfenster um diesen Mittelwert bildet. Jeder Messwert "Value" wird dabei mit dem gefilterten Messwert verglichen (Filterung der Messwerte mit einem PT1-Filter). Befindet sich der jeweilige Messwert außerhalb eines um den Mittelwert gebildeten Unterdrückungsfensters (Toleranzfenster, wird durch "SuppressionLimit1/2Low" bzw. "SuppressionLimit1/2High" definiert), so wird er nicht an den Baustein-Ausgang "OutputValue" ausgegeben, sondern unterdrückt; d.h. am Ausgang "OutputValue" wird nicht "Value", sondern der Mittelwert ausgegeben.

Die Funktion besitzt zwei unterschiedliche Fenster, zwischen denen umgeschaltet werden kann, zusätzlich kann die Unterdrückungsfunktion deaktiviert werden. Dies kann dazu verwendet werden, um im Betrieb bei großen Abweichungen das Unterdrückungsfenster kurzzeitig zu vergrößern (z.B. bei Einschwingvorgängen).

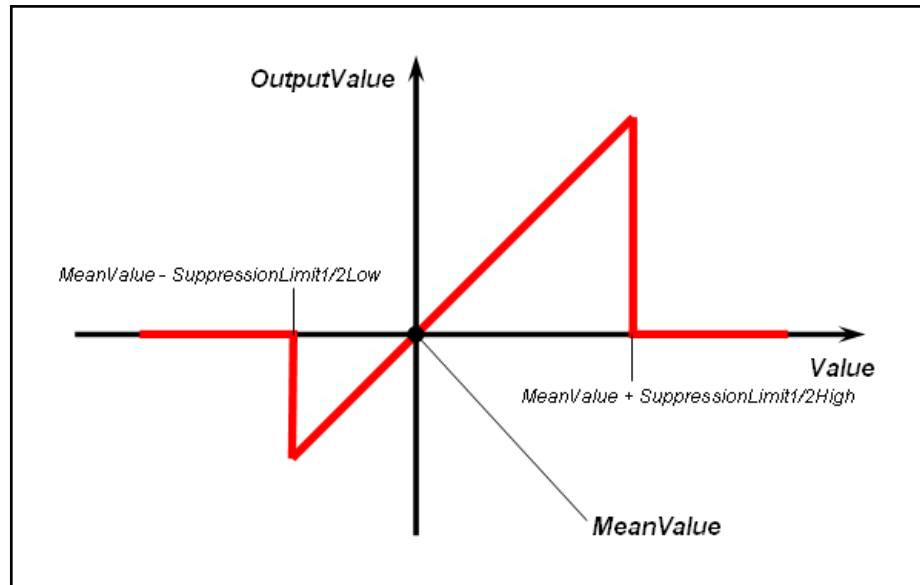


Abb. 7-93: Prinzip des Unterdrückungsfensters

Interne Struktur des FBs

Der FB besteht aus einem Mittelwertfilter und einem Block, der die Unterdrückung der Messwerte durchführt. Die Mittelwertfilterung wird durch ein PT1-Glied durchgeführt. Die Filterzeitkonstante sollte dabei groß gegenüber der Prozesszeitkonstante gewählt werden, damit eine Mittelwertfilterung korrekt wirken kann. Der gefilterte Mittelwert kann über den Ausgang "MeanValue" diagnostiziert werden.

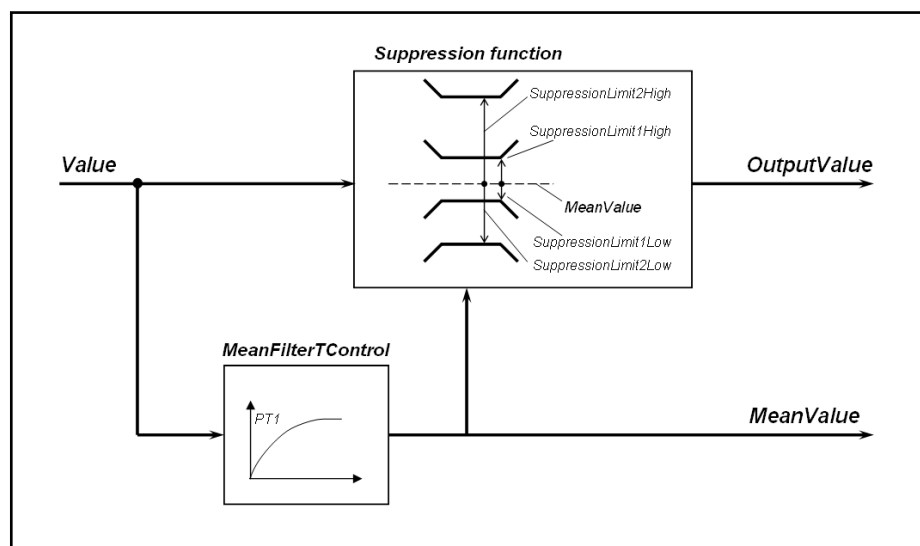


Abb. 7-94: Interne Struktur IL_SuppressionType03

Ausfensterungs-Modi

Bei der Ausfensterung sind die obere ("SuppressionLimit1/2High") und untere Schwelle ("SuppressionLimit1/2Low") getrennt konfigurierbar. Es stehen drei Modi zur Verfügung:

- **Prozentual auf den Mittelwert bezogen (IL_SUPPRESSION_PERCENTAGE):** die Fensterbreite wird durch einen Prozentsatz des Mittelwertes vorgegeben, d.h. "SuppressionLimitHigh" gibt einen Prozentanteil (vom Absolutwert des Mittelwertes) an, der zum Mittelwert addiert wird, um die obere Schwelle zu erhalten und "SuppressionLimitLow" gibt einen Prozentanteil (vom Absolutwert des Mittelwertes) an, der vom Mittelwert subtrahiert wird, für die untere Schwelle. Es ist zu berücksichtigen, dass die Größe des Fensters sehr klein werden kann, wenn der Betrag des Mittelwertes sehr klein wird.
- **Absolutes Fenster auf den Mittelwert bezogen (IL_SUPPRESSION_ABSOLUTE_VALUE):** die Werte der Eingänge "SuppressionLimitLow" und "SuppressionLimitHigh" geben die Größe des Fensters um den Mittelwert direkt vor.
- **Auf die statistischen Gegebenheiten des Eingangssignals bezogen (IL_SUPPRESSION_STATISTICAL):** die Fensterbreite wird auf die Standardabweichung σ des Messsignals bezogen, d.h. die Standardabweichung σ des Messsignals wird mit den Werten der Eingänge "SuppressionLimit1/2Low" und "SuppressionLimit1/2High" multipliziert und ergibt die Größe des Fensters. (Zur Ermittlung der Standardabweichung wird eine exponentiell gleitende Berechnung mit "MeanFilterTControl" als Zeitkonstante verwendet.)

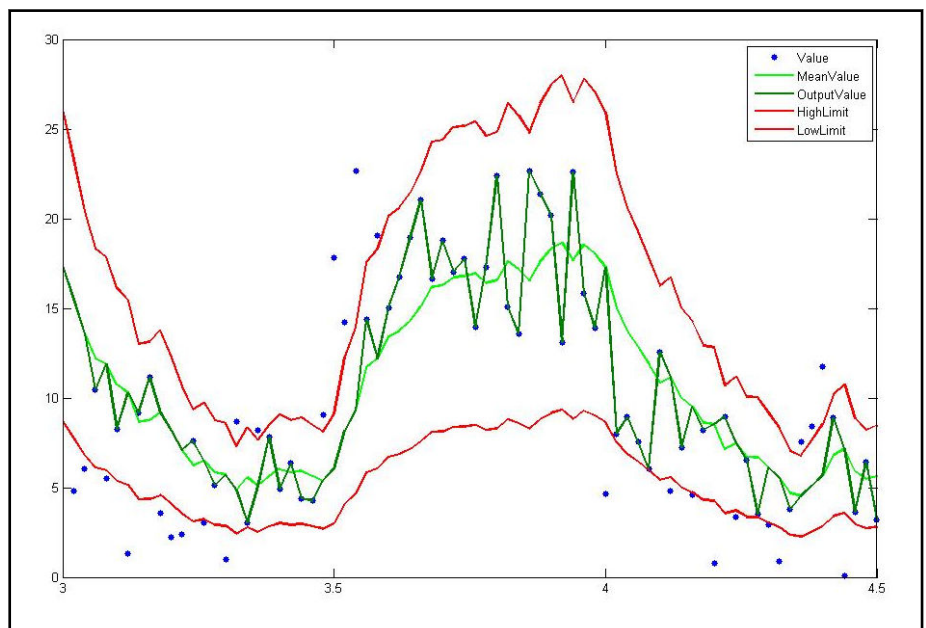


Abb. 7-95: Beispiel einer Ausfensterung "prozentual auf den Mittelwert bezogen"

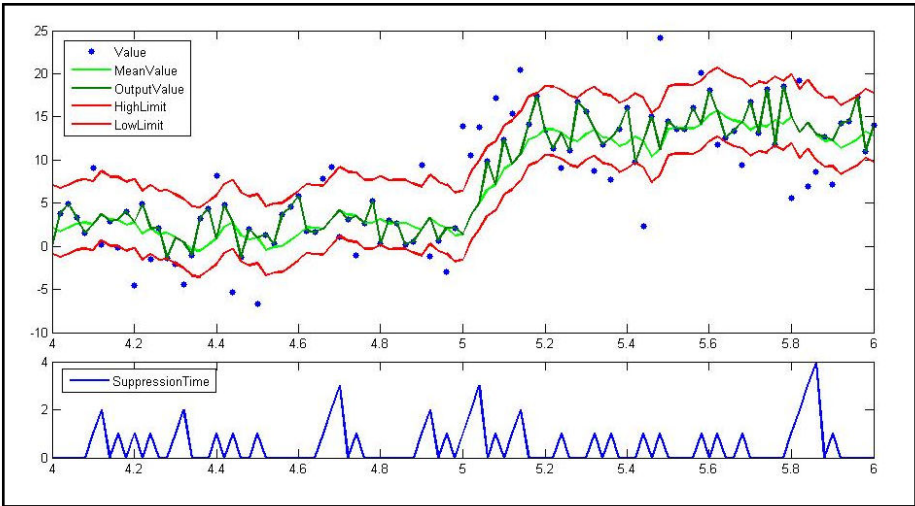


Abb. 7-96: *Beispiel einer Ausfensterung „mit absolutem Fenster um den Mittelwert“*

Diagnose der Ausfensterung

Zur Diagnose kann die Funktion der Ausfensterung mittels der Ausgänge "SuppressionTime" und "SuppressionRatio" beobachtet werden. "SuppressionTime" ist dabei ein Zeitzähler wie lange Messwerte unterdrückt wurden, d.h. er zählt bei aktiver Unterdrückung hoch und wird auf "0" zurückgesetzt, wenn wieder ein Eingangswert vorhanden ist, der nicht unterdrückt wird. Er wird ebenfalls auf "0" gesetzt bei einer steigenden Flanke an "Enable", sowie bei einem Überlauf. "SuppressionRatio" ist der prozentuale Zeitanteil aller unterdrückten Werte.

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR (16#0005)	16#0006	16#0000	Ungültiger Zustand des Funktionsbausteins
INPUT_INVALID_ERROR	16#0C09	16#0002	Wert von "SuppressionLimit1/2High" ist kleiner als 0.0
INPUT_INVALID_ERROR	16#0C09	16#0003	Wert von "SuppressionLimit1/2Low" ist kleiner als 0.0
INPUT_INVALID_ERROR	16#0C09	16#0004	Wert von "MeanFilterTControl" ist kleiner als (2 x "ActScanTime"), nur im Modus "Statistical" relevant

Tab. 7-153: *Generierte Fehlernummern des Funktionsbausteins IL_SuppressionType03*

7.8.9 **IL_SuppressionType04**

Kurzbeschreibung

Dieser eventgesteuerte Funktionsbaustein unterdrückt Messwerte, die eine exzessive Abweichung zum Mittelwert der Messwerte besitzen. Die Messwerte, die außerhalb der parametrierbaren Schwellwerten liegen, werden nicht weitergegeben, sondern stattdessen wird der gefilterte Wert ausgegeben. Der IL_SuppressionType04 besitzt im Vergleich zum Type02 funktionelle Erweiterungen, u.a. zwei umschaltbare Grenzen für das Unterdrückungsfenster, sowie eine zusätzliche Unterdrückungsfunktion.

Schnittstellenbeschreibung

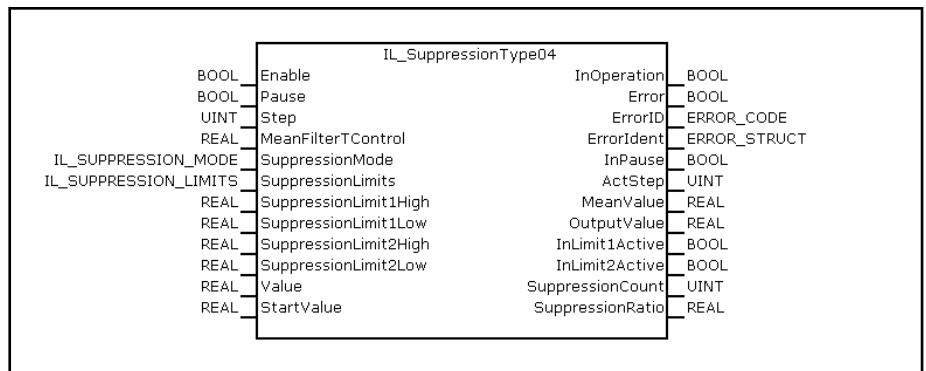


Abb. 7-97: Funktionsbaustein IL_SuppressionType04

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Berechnung der Daten in jedem Zyklus (Step) solange "Enable"=TRUE.
	Pause	BOOL	Die Daten der Ausgänge werden nicht verändert, solange "Pause" = TRUE. Wird währenddessen der "Step"-Eingang inkrementiert, wird der aktuell anliegende Schritt am Ausgang "ActSteps" ausgegeben.
	Step	UINT	Aktueller Berechnungsschritt .
	MeanFilterTControl	REAL	Die Konstante "Tn" des Mittelwertfilter-Gliedes bezogen auf "Step".
	SuppressionMode	IL_SUPPRESSION_MODE	Beschreibt die Funktion der Eingänge "SuppressionLimitLow" und "SuppressionLimitHigh": - IL_SUPPRESSION_PERCENTAGE - IL_SUPPRESSION_ABSOLUTE_VALUE - IL_SUPPRESSION_STATISTICAL IL_SUPPRESSION_MODE, Seite 512
	SuppressionLimits	IL_SUPPRESSION_LIMITS	Auswahl des aktiven Unterdrückungsfensters: - IL_SUPPRESSION_OFF - IL_SUPPRESSION_LIMIT_1 - IL_SUPPRESSION_LIMIT_2 IL_SUPPRESSION_LIMITS, Seite 511
	SuppressionLimit1High	REAL	Unterer Wert des Unterdrückungsfensters #1.
	SuppressionLimit1Low	REAL	Oberer Wert des Unterdrückungsfensters #1.
	SuppressionLimit2High	REAL	Unterer Wert des Unterdrückungsfensters #2.
	SuppressionLimit2Low	REAL	Oberer Wert des Unterdrückungsfensters #2.
	Value	REAL	Der aktuelle Istwert des Prozesses.
	StartValue	REAL	Startwert, der bei steigender Flanke an "Enable" am Ausgang "OutputValue" angenommen wird.
VAR_OUTPUT	InOperation	BOOL	Fehlerfreie Bearbeitung, Ausgangsdaten sind gültig .
	Error	BOOL	Fehler (siehe ErrorID und ErrorIdent)
	ErrorID	ERROR_CODE	Fehlerbeschreibung
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerbeschreibung, siehe Fehlercodes, Seite 476 .

	Name	Typ	Kommentar
	InPause	BOOL	Zeigt an, dass der Eingang „Pause“ aktiv ist und sich die Werte der Ausgänge nicht ändern.
	ActStep	UINT	Step Wert, auf den sich "OutputValue" bezieht. "Act-Step" wird bei steigender Flanke von "Enable" mit dem Wert von "Step" initialisiert.
	MeanValue	REAL	Gefilterter Mittelwert, gegen den "Value" verglichen wird.
	OutputValue	REAL	Der berechnete Ausgangswert.
	InLimit1Active	BOOL	Zeigt an, dass Unterdrückungsfenster #1 aktiv ist.
	InLimit2Active	BOOL	Zeigt an, dass Unterdrückungsfenster #2 aktiv ist.
	SuppressionCount	UINT	Zähler, der die Anzahl der unterdrückten Werte zählt. Der Wert wird bei steigender Flanke an "Enable" gelöscht und zählt dann bei aktiver Unterdrückung hoch. Sobald wieder ein Eingangswert vorhanden ist, der nicht unterdrückt wird, wird der Ausgang wieder gelöscht. Bei einem Überlauf (16xFFFF) fällt der Zähler automatisch auf "0" zurück.
	SuppressionRatio	REAL	Prozentualer Anteil aller unterdrückten Werte (0-100%).

Tab. 7-154: Funktionsbaustein IL_SuppressionType04

IL_SUPPRESSION_MODE ist ein Aufzählungstyp und beschreibt die Funktion der Eingänge "SuppressionLimitLow1/2" und "SuppressionLimitHigh1/2" mit folgender Belegung:

Element	Beschreibung
IL_SUPPRESSION_PERCENTAGE	Werte sind prozentual zum gemittelten Eingangswert
IL_SUPPRESSION_ABSOLUTE_VALUE	Werte sind absolute Werte zum gemittelten Eingangswert
IL_SUPPRESSION_STATISTICAL	Werte sind statistisch zur Standardabweichung σ des Eingangswertes (z.B. $0.5 \cdot \sigma$, $1.0 \cdot \sigma$, $4.0 \cdot \sigma$,...)

Tab. 7-155: Aufbau des Aufzählungstyps IL_SUPPRESSION_MODE

IL_SUPPRESSION_LIMITS ist ein Aufzählungstyp und aktiviert die Unterdrückungsfenster mit folgender Belegung:

Element	Beschreibung
IL_SUPPRESSION_OFF	Unterdrückung ist deaktiviert, alle Werte kommen unverändert an den Ausgang.
IL_SUPPRESSION_LIMIT_1	Unterdrückungsfenster #1 aktiv ("SuppressionLimit1Low" und "SuppressionLimit1High")
IL_SUPPRESSION_LIMIT_2	Unterdrückungsfenster #2 aktiv ("SuppressionLimit2Low" und "SuppressionLimit2High")

Tab. 7-156: Aufbau des Aufzählungstyps IL_SUPPRESSION_LIMITS

Min-, Max- und Default-Werte der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
Pause	BOOL			FALSE	kontinuierlich
Step	UINT	0	16#FFFF	0	kontinuierlich
MeanFilterTControl	TIME	0.0	n. def.	100ms	kontinuierlich
SuppressionMode	IL_SUPPRESS- ION_MODE			IL_SUPPRESS- ION_PERCENTAGE	steigende Flanke an Enable
SuppressionLimid	IL_SUPPRESS- ION_LIMITS			IL_SUPPRESSION_LI- MIT_1	kontinuierlich
SuppressionLimit1High	REAL	0.0	n. def.	5.0	kontinuierlich
SuppressionLimit1Low	REAL	0.0	n. def.	5.0	kontinuierlich
SuppressionLimit2High	REAL	0.0	n. def.	20.0	kontinuierlich
SuppressionLimit2Low	REAL	0.0	n. def.	20.0	kontinuierlich
Value	REAL	n. def.	n. def.	0.0	kontinuierlich
StartValue	REAL	n. def.	n. def.	0.0	steigende Flanke an Enable

Tab. 7-157: Min-, Max- und Default-Werte der Eingänge des IL_Suppression-Type04

Funktionsbeschreibung

Bei Regelvorgängen ist es oft notwendig, Messwerte, die offensichtlich stark von dem erwarteten Wert abweichen, auszufiltern (unterdrücken). Hierfür ist im FB ein Unterdrückungsfenster implementiert, der mit dem Mittelwert des zu filternden Wertes mitläuft und ein Toleranzfenster um diesen Mittelwert bildet. Jeder Messwert "Value" wird dabei mit dem gefilterten Messwert verglichen (Filterung der Messwerte mit einem PT1-Filter). Befindet sich der jeweilige Messwert außerhalb eines um den Mittelwert gebildeten Unterdrückungsfensters (Toleranzfenster, wird durch "SuppressionLimit1/2Low" bzw. "SuppressionLimit1/2High" definiert), so wird er nicht an den Bausteinoutput "OutputValue" ausgegeben, sondern unterdrückt; d.h. am Ausgang "OutputValue" wird nicht "Value", sondern der Mittelwert ausgegeben.

Die Funktion besitzt zwei unterschiedliche Fenster, zwischen denen umgeschaltet werden kann, zusätzlich kann die Unterdrückungsfunktion deaktiviert werden. Dies kann dazu verwendet werden, um im Betrieb bei großen Abweichungen das Unterdrückungsfenster kurzzeitig zu vergrößern (z.B. bei Einschwingvorgängen).

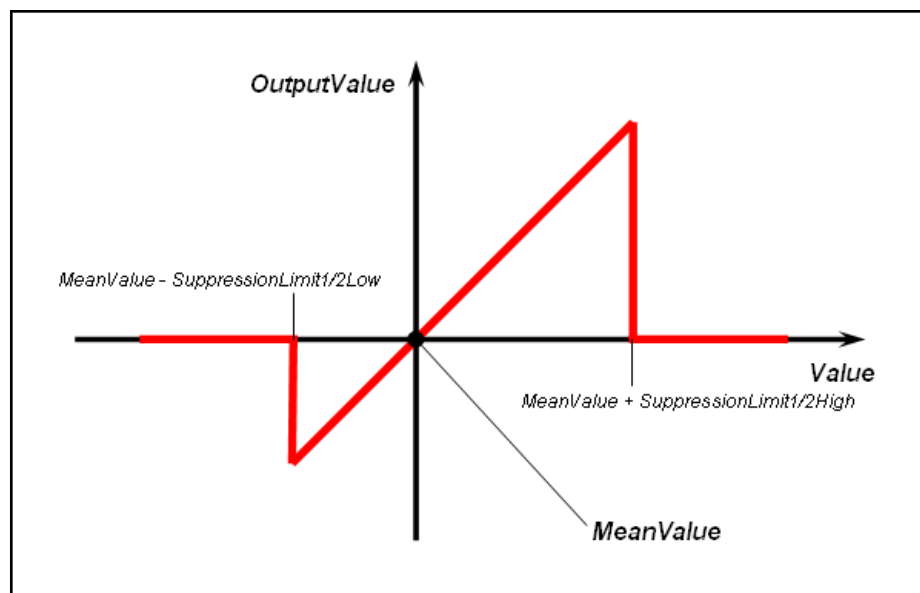


Abb. 7-98: Prinzip des Unterdrückungsfensters

Interne Struktur des FBs

Der FB besteht aus einem Mittelwertfilter und einem Block, der die Unterdrückung der Messwerte durchführt. Die Mittelwertfilterung wird durch ein PT1-Glied durchgeführt. Die Filterzeitkonstante sollte dabei groß gegenüber der Prozesszeitkonstante gewählt werden, damit eine Mittelwertfilterung korrekt wirken kann. Der gefilterte Mittelwert kann über den Ausgang "MeanValue" diagnostiziert werden.

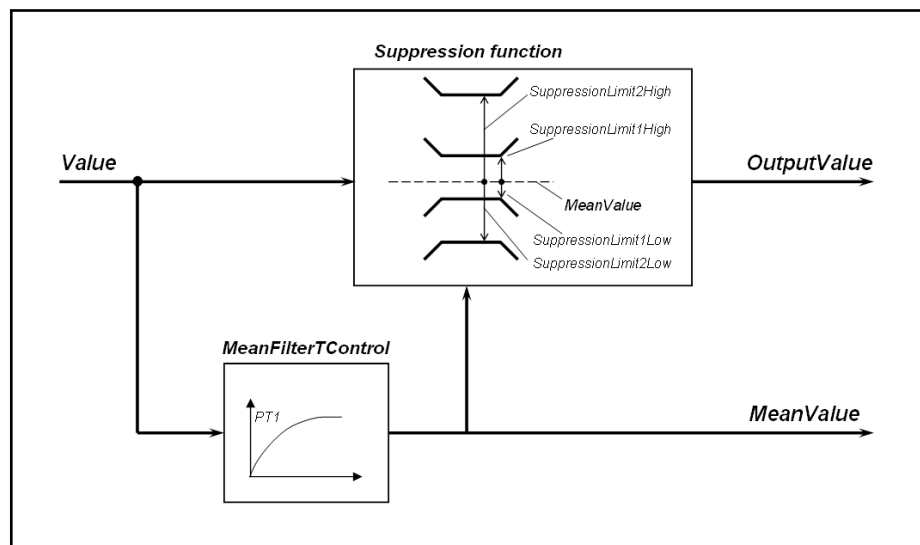


Abb. 7-99: Interne Struktur IL_SuppressionType02

Ausfensterungs-Modi

Bei der Ausfensterung sind die obere ("SuppressionLimit1/2High") und untere Schwelle ("SuppressionLimit1/2Low") getrennt konfigurierbar. Es stehen drei Modi zur Verfügung:

- **Prozentual auf den Mittelwert bezogen (IL_SUPPRESSION_PERCENTAGE):** die Fensterbreite wird durch einen Prozentsatz des Mittelwertes vorgegeben, d.h. "SuppressionLimitHigh" gibt einen Prozentanteil (vom Absolutwert des Mittelwertes) an, der zum Mittelwert addiert wird, um die obere Schwelle zu erhalten und "SuppressionLimitLow" gibt einen

Prozentanteil (vom Absolutwert des Mittelwertes) an, der vom Mittelwert subtrahiert wird, für die untere Schwelle. Es ist zu berücksichtigen, dass die Größe des Fensters sehr klein werden kann, wenn der Betrag des Mittelwertes sehr klein wird.

- **Absolutes Fenster auf den Mittelwert bezogen (IL_SUPPRESSION_ABSOLUTE_VALUE):** die Werte der Eingänge "SuppressionLimitLow" und "SuppressionLimitHigh" geben die Größe des Fensters um den Mittelwert direkt vor.
- **Auf die statistischen Gegebenheiten des Eingangssignals bezogen (IL_SUPPRESSION_STATISTICAL):** die Fensterbreite wird auf die Standardabweichung σ des Messsignals bezogen, d.h. die Standardabweichung σ des Messsignals wird mit den Werten der Eingänge "SuppressionLimit1/2Low" und "SuppressionLimit1/2High" multipliziert und ergibt die Größe des Fensters. (Zur Ermittlung der Standardabweichung wird eine exponentiell gleitende Berechnung mit "MeanFilterTControl" als Zeitkonstante verwendet.)

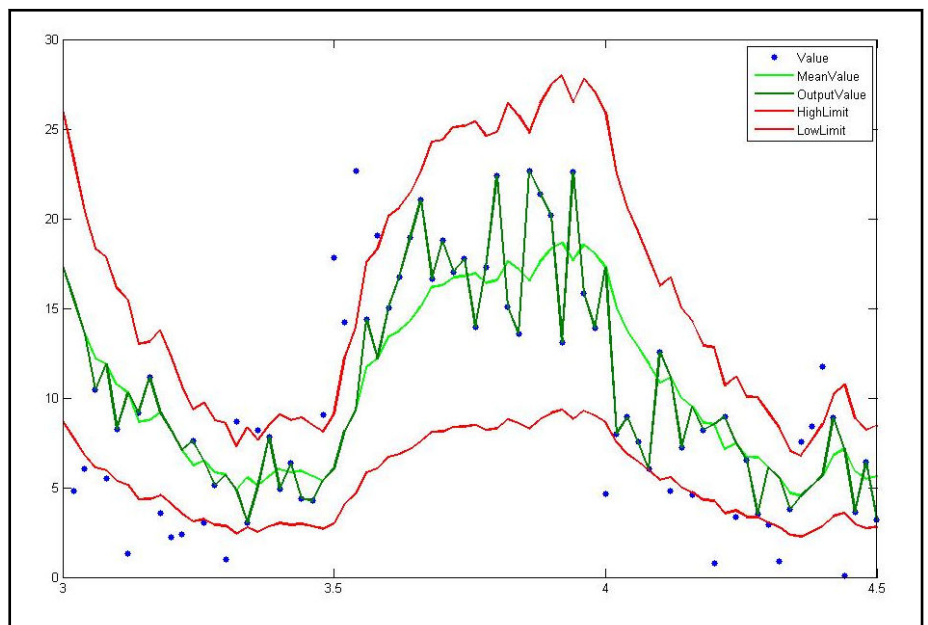


Abb. 7-100: Beispiel einer Ausfensterung "prozentual auf den Mittelwert bezogen"

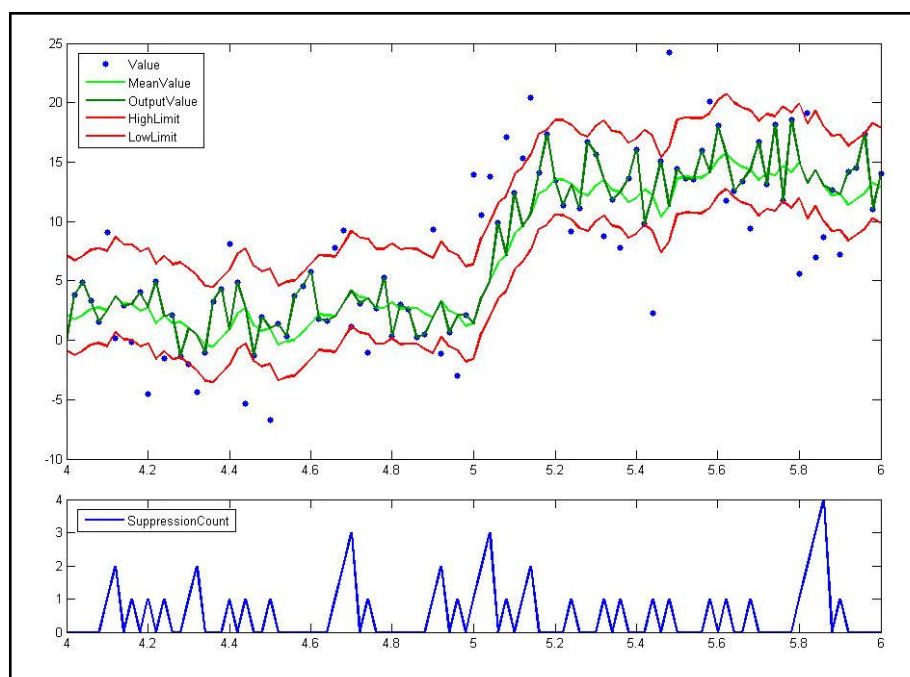


Abb. 7-101: Beispiel einer Ausfensterung „mit absolutem Fenster um den Mittelwert“

Diagnose der Ausfensterung

Zur Diagnose kann die Funktion der Ausfensterung mittels der Ausgänge "SuppressionCount" und "SuppressionRatio" beobachtet werden. "SuppressionCount" ist dabei ein Zähler wie lange Messwerte unterdrückt wurden, d.h. er zählt bei aktiver Unterdrückung hoch und wird auf "0" zurückgesetzt, wenn wieder ein Eingangswert vorhanden ist, der nicht unterdrückt wird. Er wird ebenfalls auf "0" gesetzt bei einer steigenden Flanke an "Enable", sowie bei einem Überlauf. "SuppressionRatio" ist der prozentuale Zeitanteil aller unterdrückten Werte.

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**, 16#0170. Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR (16#0005)	16#0006	16#0000	Ungültiger Zustand des Funktionsbausteins
INPUT_INVALID_ERROR	16#0C09	16#0001	Wert von "MeanFilterTControl" ist kleiner als 0.0
INPUT_INVALID_ERROR	16#0C09	16#0002	Wert von "SuppressionLimit1/2High" ist kleiner als 0.0
INPUT_INVALID_ERROR	16#0C09	16#0003	Wert von "SuppressionLimit1/2Low" ist kleiner als 0.0
INPUT_INVALID_ERROR	16#0C09	16#0004	Wert von "MeanFilterTControl" muss doppelt so groß sein wie die Differenz von zwei (unterschiedlichen) aufeinander folgenden "Step"-Werten (nur im Modus "Statistical")

Tab. 7-158: Generierte Fehlernummern des Funktionsbausteins IL_Suppression-Type04

7.9 Temperaturregler (TemperatureController)

7.9.1 Einführung und Überblick



Die nachstehenden Funktionsbausteine und Datentypen befinden sich in der Version 13 in der ML_TechBase-Bibliothek.

Sie wurden in der Version 14 in die RIL_LoopControl-Bibliothek verlegt.

Mit den nachstehend genannten Funktionsbausteinen können Temperaturrecken geregelt werden.

Anwendungsgebiete und Funktionen / Funktionsbausteine von RIL_LoopControl

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine	Beschreibung
Temperaturregler (TemperatureController)	
IL_TempControlType01, Seite 477	Regler für Temperaturrecke mit automatischer offline Reglereinstellung
IL_TempControlType02, Seite 495	Regler für Temperaturrecke mit automatischer online Reglereinstellung

Tab. 7-159: Funktionsbausteine zur Regelung von Temperaturrecken

7.9.2 IL_TempControlType01

Kurzbeschreibung

Der Funktionsbaustein IL_TempControlType01 dient zur Regelung von Temperaturen. Er kann die Reglerparameter entweder selbst ermitteln oder mit Vorgaben arbeiten. Komfortfunktionen wie Überwachung, Pause, Handbetrieb können zusätzlich aktiviert werden.



Der Funktionsbaustein "IL_TempControlType01 (ML_TechBase-Bibliothek)" wurde mit Version 14VRS in die "RIL_LoopControl-Bibliothek" übernommen.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek
MLC 15VRS	RIL_LoopControl

Tab. 7-160: Referenztablette Funktionsbaustein IL_TempControlType01

Schnittstellenbeschreibung

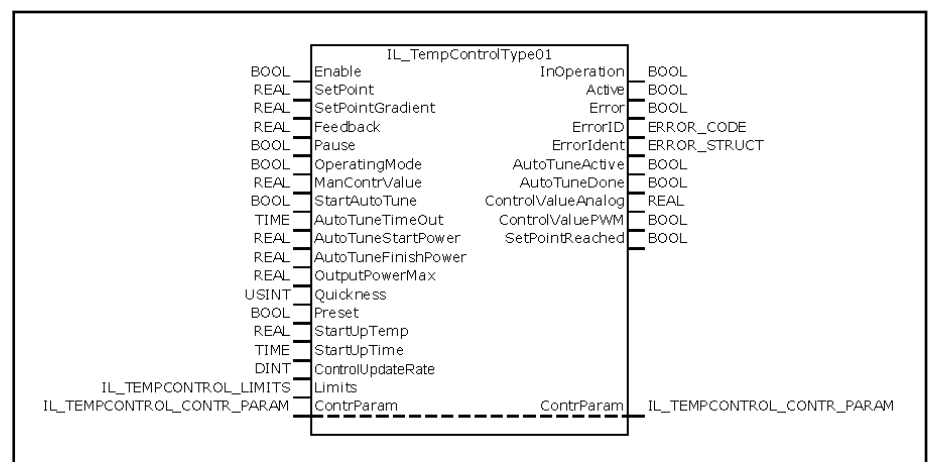


Abb. 7-102: Funktionsbaustein IL_TempControlType01

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Der Funktionsbaustein ist aktiv, wenn dieser Eingang "TRUE" ist. Der FB wird mit einer fallenden Flanke auf seine Ausgangswerte zurückgesetzt. Der Eingang wird in jedem Zyklus gelesen
	SetPoint	REAL	Solltemperatur Einheit in [u]* Der Eingang wird in jedem Zyklus gelesen
	SetPointGradient	REAL	Maximale Änderung des Parameters "SetPoint" in [u/s]. Der Eingang wird in jedem Zyklus gelesen. Voreingestellter Wert ist "0 [u/s]" (nicht aktiv)
	Feedback	REAL	Eingang für die Isttemperatur Einheit in [u] Der Eingang wird in jedem Zyklus gelesen
	Pause	BOOL	Dieser Eingang "friert" Stellgröße und Reglerberechnung ein. Der Eingang wird in jedem Zyklus gelesen. Voreingestellter Wert ist "FALSE" (nicht aktiv)
	OperatingMode	BOOL	Setzt den Reglermodus von Automatik auf Handbetrieb: AUTO:=FALSE, MANUAL:=TRUE. Voreingestellter Wert ist "FALSE" (AUTO). Der Eingang wird in jedem Zyklus gelesen
	ManContrValue	REAL	Gibt die Stellgröße in [%] bei Handbetrieb an, wird aber nur aktiv gelesen, wenn "OperatingMode:=1". Der Eingang wird in jedem Zyklus gelesen. Voreingestellter Wert ist "0%" (keine Stellgröße)
	StartAutoTune	BOOL	Eine steigende Flanke startet den Identifikationsprozess. Die PID-Reglerparameter werden im Anschluss berechnet, jedoch nicht aktiv bis eine fallende Flanke ansteht. Der Eingang wird in jedem Zyklus gelesen. Voreingestellter Wert ist "FALSE" (nicht aktiv)
	AutoTuneTimeOut	TIME	Watchdog, der die Zeitüberwachung während der Identifikation vornimmt. Voreingestellter Wert ist " $10000 \cdot T_A$ ". Der Eingang wird bei einer steigenden Flanke von "StartAutoTune" gelesen

E/A-Typ	Name	Datentyp	Beschreibung
	AutoTuneStartPower	REAL	Stellgröße in [%], die die untere Grenze des Identifikationsprozesses darstellt. Voreingestellter Wert ist "0%". Der Eingang wird bei einer steigenden Flanke von "StartAutoTune" gelesen
	AutoTuneFinishPower	REAL	Stellgröße in [%], die die obere Grenze des Identifikationsprozesses darstellt. Voreingestellter Wert ist "50%". Der Eingang wird bei einer steigenden Flanke von "StartAutoTune" gelesen
	OutputPowerMax	REAL	Maximal ausgegebene Stellgröße in [%]. Voreingestellter Wert ist "100%". Der Eingang wird in jedem Zyklus gelesen
	Quickness	USINT	Schnelligkeit des Reglers Voreingestellter Wert ist "3" (ganz leichtes Überschwingen). Der Eingang kann im Bereich von "1" (schnelle Regelung, starkes Überschwingen) bis "5" (langsame Regelung, kein Überschwingen) gewählt werden. Der Eingang wird bei steigender Flanke von "StartAutoTune" gelesen
	Preset	BOOL	Eine steigende Flanke übernimmt die Reglerparameter der "ContrParam"-Struktur. Voreingestellter Wert ist "FALSE" (nicht aktiv). Der Eingang wird in jedem Zyklus gelesen und wirkt nur wenn "Enable" = TRUE
	StartUpTemp	REAL	Anfahrtemperatur in [u]. Der Eingang wird bei steigender Flanke von "Enable" gelesen. Voreingestellter Wert ist "0u" (nicht aktiv). Wird nur aktiv, wenn Eingang "StartUpTime" mit einem Wert ungleich "0" belegt wird
	StartUpTime	TIME	Anfahrzeit, d.h., Zeit, in der die Anfahrtemperatur ("StartUpTemp") gehalten wird bis in die "normale" Regelung übergegangen wird. Voreingestellter Wert ist "T#0s" (nicht aktiv). Der Eingang wird bei steigender Flanke von "Enable" gelesen
	ControlUpdateRate	DINT	Dieser Faktor gibt an, um wieviel langsamer der Regler regelt, im Vergleich zur Task in der sich der Baustein befindet. Aus Multiplikation der SPS-Zykluszeit (= Task Intervallzeit) und "ControlUpdateRate" ergibt sich die Abtastzeit

E/A-Typ	Name	Datentyp	Beschreibung
	Limits	IL_TEMPCONTROL_LIMITS	Struktur, die die Alarmschranken in [u] (Einheiten) enthält. IL_TEMPCONTROL_LIMITS, Seite 508
VAR_OUTPUT	InOperation	BOOL	Die Temperaturregelung ist bei gesetztem Ausgang aktiv
	Active	BOOL	Dieser Ausgang wird gesetzt, wenn "Enable" gesetzt und der Regler aktiv ist
	Error	BOOL	Ein Fehler ist während des Betriebes aufgetreten. Die Stellgröße wird auf "0%" gesetzt
	ErrorID	ERROR_CODE	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang eine grobe Klassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bei gesetztem Error-Ausgang enthält dieser Ausgang detaillierte Informationen über den Fehler Fehlerbehandlung, Seite 493
	AutoTuneActive	BOOL	Der Identifikationsprozess ist aktiv
	AutoTuneDone	BOOL	Der Identifikationsprozess wurde erfolgreich beendet
	ControlValueAnalog	REAL	Analoger Wert der Stellgröße in %
	ControlValuePWM	BOOL	Digitaler, pulswidenmodulierter Stellgrößenwert, entweder "0" oder "1"
	SetPointReached	BOOL	Ein Statusausgang der anzeigt, ob sich der Istwert innerhalb der Schranken ("Setpoint" + "Limits.UpperAt-Setpoint") $\geq$ Istwert $\geq$ (Setpoint - "Limits.LowerAt-Setpoint") befindet
VAR_IN_OUT	ContrParam	IL_TEMPCONTROL_CONTR_PARAM	Diese Struktur enthält die aktuellen Parameter K_P , K_I und K_D des Reglers sowie die Totzeit T_t IL_TEMPCONTROL_CONTR_PARAM, Seite 506

*

Die Einheit [u] bedeutet „unit“. Dies kann eine beliebige Einheit wie z.B. [°C] oder [°F] sein. Es ist darauf zu achten, dass die gewählte Einheit einheitlich verwendet wird.

+

Die Abtastzeit T_A ist die effektive Abtastzeit des Temperaturreglers und setzt sich zusammen aus $T_A = \text{Zykluszeit der SPS-Task in der der Baustein aufgerufen wird, multipliziert mit ControlUpdateRate}$

Tab. 7-161: Schnittstellenvariablen Funktionsbaustein IL_TempControlType01

Zeitdiagramm

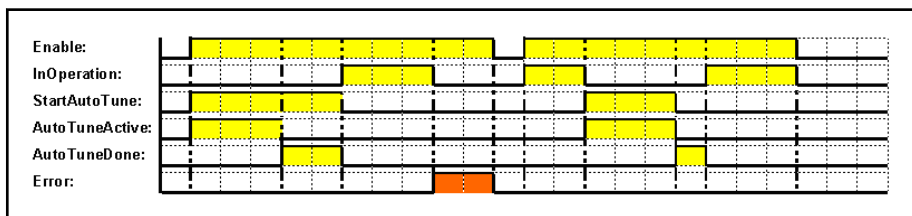


Abb. 7-103: Zeitdiagramm

Min- Max- und Default-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	Zyklisch
SetPoint	REAL	N. Def.	N. Def.	0.0	Zyklisch
SetPointGradient	REAL	0.0	N. Def.	0.0	Zyklisch
Feedback	REAL	N. Def.	N. Def.	0.0	Zyklisch
Pause	BOOL	FALSE	TRUE	FALSE	Zyklisch
OperatingMode	BOOL	FALSE	TRUE	FALSE	Zyklisch
ManContrValue	REAL	0.0	100.0	0.0	Zyklisch
StartAutoTune	BOOL	FALSE	TRUE	FALSE	Zyklisch
AutoTuneTimeOut	TIME	T#0s	"10000 • T _A ".	"10000 • T _A ". (wird automatisch eingestellt wenn T#0s eingestellt wird)	Bei steigender Flanke an „StartAutoTune“
AutoTuneStartPower	REAL	0.0	95.0	0.0	Bei steigender Flanke an „StartAutoTune“
AutoTuneFinishPower	REAL	5.0	100.0	50.0	Bei steigender Flanke an „StartAutoTune“
OutputPowerMax	REAL	>0.0	100.0	100.0	Zyklisch
Quickness	USINT	1	5	3	Bei steigender Flanke an „StartAutoTune“
Preset	BOOL	FALSE	TRUE	FALSE	Zyklisch
StartUpTemp	REAL	N. Def.	N. Def.	0.0	Bei steigender Flanke an „Enable“
StartUpTime	TIME	T#0s	N. Def.	T#0s	Bei steigender Flanke an „Enable“
ControlUpdateRate	DINT	1	1000	100	Bei steigender Flanke an „Enable“
Limits	IL_TEMPCONTROL_LIMITS	Siehe Datenstruktur	Siehe Datenstruktur	Siehe Datenstruktur	Zyklisch
ContrParam	IL_TEMPCONTROL_CONTR_PARAM	Siehe Datenstruktur	Siehe Datenstruktur	Siehe Datenstruktur	Daten werden durch Autotune Prozess geschrieben bei steigender Flanke von "AutoTuneDone". Daten werden bei steigender Flanke an „Preset“ übernommen.

Tab. 7-162: Min- Max- und Default-Werte sowie Übernahmeverhalten der Eingänge des Funktionsbausteins IL_TempControlType01

Anwendung

Der Baustein kann für das System MLC mit Hilfe von Inline Temperatur-Modulen (IL TEMP 2 RTD) betrieben werden. Generell kann jeder beliebige Wert als Temperaturwert benutzt werden, da der Temperaturregler nicht direkt auf Hardware zugreift. Damit ist es auch möglich, Temperaturwerte über z.B. Feldbusverbindungen zu erhalten.

Identifikation Die automatische Identifikation, auch "AutoTune" genannt, dient der Findung der Parameter der Regelstrecke, um daraus die Parameter des Reglers festzulegen. Es werden Modellparameter berechnet, anschließend wird das Modell mit der realen Regelstrecke verglichen, um Fehlparametrierungen der Reglerparameter zu verhindern.

Dazu wird ein Sollwertsprung auf die Regelstrecke aufgeschaltet. Bevor der Sprung aufgeschaltet wird, wird sichergestellt, dass sich die Regelstrecke in Ruhe befindet. Dazu wird eine Wartezeit gestartet. Voreingestellter Zeitwert für diese ist 5000 multipliziert mit der Wurzel aus der Abtastzeit " T_A " in [ms]. Vor Ablauf der Wartezeit darf der Istwert die Toleranzgrenzen der definierten Toleranzschranken nicht überschreiten, ansonsten wird die Wartezeit neu gestartet und wieder gewartet bis diese abgelaufen ist. Die Grenzschränke für das Temperaturfenster in dem der Istwert für die Zeit der Wartezeit liegen muss, wird wie folgt berechnet:

$$|\text{IdentTemp} - \text{IstTemp}| \leq \sqrt{\frac{|\text{IdentTemp}|}{10}} + 0,5$$

Die Variable "IdentTemp" ist dabei der Wert um den das Toleranzfenster für die Isttemperatur aufgespannt wird. "IdentTemp" wird zu Beginn mit dem Istwert ("IstTemp") initialisiert. Das Warten auf den stationären Endwert nach dem Sprung erfolgt in gleicher Weise.

Beispiel:

Abschätzung der Dauer einer Identifikation

Angenommen, der Funktionsbaustein wird in einer SPS-Task mit 10ms Zykluszeit betrieben. Der Eingang "ControlUpdateRate" ist mit 100 beschaltet. Was ist in diesem Fall die minimale Dauer für den Autotune Prozess?

Aus der SPS-Task Zykluszeit und der "ControlUpdateRate" kann durch Multiplikation die Abtastzeit errechnet werden: " T_A " = SPS-Task Zykluszeit * "ControlUpdateRate" = 10ms * 100 = 1000ms

Damit lässt sich nach obiger Formel die Wartezeit berechnen: Wartezeit = 5000 * SQRT(" T_A ") = 158114ms oder 2min und 38s. Da für die Sprungantwort zweimal diese Zeit gewartet werden muss, ergibt sich eine gesamte Wartezeit von mindestens 5 min und 16s. Dazu kommt dann noch der eigentliche Zeitbedarf der Strecke um in den stationären Zustand zu kommen.

Die Güte des ermittelten Modells wird abgeschätzt. Wird dabei eine zu große Abweichung festgestellt, gibt der FB den Fehler

"16#1603" (CALCULATION_ERROR)

mit der Kennung "16#0008" (Prozessmodell ungültig) aus.

Einstellung der Reglerparameter

Quickness = 1...5;

1 = sehr schnelle Regelung mit großem Überschwingen;

5 = sehr langsame Regelung ohne Überschwingen;

Voreingestellter Wert ist Quickness = 3.

Die errechneten Reglerparameter werden in die dafür vorgesehene Struktur "IL_TEMPCONTROL_CONTR_PARAM" eingetragen.

Dabei entspricht der Eintrag:

KP dem proportionalem Verstärkungsfaktor

KI der Integrationskonstante

KD der Differentiationskonstante und

Tt der Totzeit

Werden diese Parameter durch den Anwender vorgegeben, sollte für die Totzeit zumindest ein Schätzwert eingetragen werden um den Fehler

"Fehler im Regelkreis, Sensor möglicherweise verrutscht/zerstört"

(Additional1 = 16#1602; Additional2 = 16#0006)

zu vermeiden.

Regelalgorithmus

Es ist ein additiver PIDT₁-Algorithmus mit Anti-Wind-Up des I-Anteils und PT₁-Filterung des D-Anteils implementiert. Die additive Form ermöglicht das Abschalten einzelner Anteile des Reglers, so dass bei Bedarf beispielsweise auf die Struktur eines PI-Reglers umgeschaltet werden kann, indem der D-Anteil zu "Null" parametrisiert wird.

Die Begrenzung des I-Anteils erfolgt durch einen Anti-Wind-Up-Begrenzer, der intern auf einen Wert von 90% der Stellgröße voreingestellt ist.

Die Sättigungsgrenze des Reglers bildet der maximale Ausgangswert (VAR_IN OutputPowerMax) des Reglers. Voreingestellt ist ein Wert von 100%.

Reglerstruktur

Dargestellt ist im Folgenden die PIDT₁-Reglerstruktur mit Anti-Wind-Up-Begrenzung und Begrenzung der maximalen Stellgröße. Die additive Struktur ermöglicht ein selektives Abschalten einzelner Regleranteile. Somit können auch reine P-, PI- oder PDT₁-Regler parametrisiert werden. Diese Abschaltung der einzelnen Anteile muss jedoch vom Anwender selbst durchgeführt werden. Dazu müssen in die Struktur "contrParam" die gewünschten Werte eingestellt werden. Mit einer steigenden Flanke am Eingang "Preset" werden diese neuen Parameter dann im Regler aktiv geschaltet. Standardmäßig arbeitet der Regler mit der kompletten PIDT₁-Struktur.

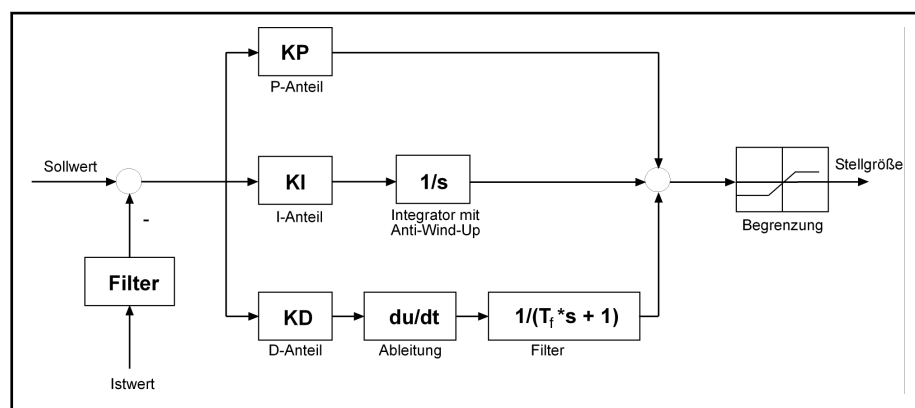


Abb. 7-104: Reglerstruktur

Der Istwert der Temperatur wird sowohl bei der Identifikation als auch beim Regeln mit einem PT₁-Filter geglättet um stochastische Störungen zu vermindern. Die Zeitkonstante des Filters ist fest auf das 20-fache der Zykluszeit eingestellt.

Regelkreisüberwachung

Der Funktionsbaustein verfügt über eine Regelkreisüberwachung. Bei Ausgabe einer Stellgröße von "100%" (volle Leistung) muss sich die Temperatur in Richtung des in K_p eingestellten Vorzeichens ändern. Ansonsten wird ein Timer aktiv, der bei Ablauf einen Fehler ausgibt. Der Timer wird zurückgesetzt, sobald sich die Temperatur wieder in die richtige Richtung bewegt. Der Timer wird folgendermaßen mit Werten belegt:

Bei vorhandenem I-Anteil wird ein Wert von $(2000 / K_I) + \text{Totzeit}$ verwendet. Wird kein I-Anteil verwendet, wird lediglich der Wert $2000 \cdot \text{Abtastzeit}$ vorein-

Verwendung des Funktionsbausteins

gestellt. Auch wenn die Stellgröße wieder kleiner "100%" wird, wird der Timer zurückgesetzt. Es ist wichtig, die Totzeit (Tt) in der Struktur für die Reglerparameter mit anzugeben, wenn die Reglerparameter durch den "Preset"-Eingang vorgegeben werden. Bei großem K_I und großer Totzeit könnte sonst die Überwachung ansprechen. Durch diese Überwachung können Fehler am Sensor (z.B. Verrutschen) oder Fehler am Stellglied erkannt werden.

Zusätzlich verfügt der FB noch über eine Gradientenüberwachung des Istwertes, der aber nur bei sehr großen Abweichungen anschlägt. Solche können z.B. vorliegen, wenn der Sensor einen Fehlercode ausgibt und dieser Fehlercode vom Anwender nicht abgefangen/ausgewertet wird. Der Gradient ist fest auf 10 u/ms eingestellt. Es wird eine einmalige Überschreitung toleriert, ehe ein Fehler generiert wird.

Der Funktionsbaustein sollte in einer zyklischen Task betrieben werden. Der FB erkennt die Taskintervallzeit (= Zykluszeit) automatisch und passt den Regelalgorithmus daran an. Der Regler arbeitet mit der Abtastzeit, die sich aus der Taskintervallzeit multipliziert mit der Eingangsvariable "ControlUpdateRate" ergibt. Wenn "ControlUpdateRate" = 1, entspricht die Abtastzeit somit der Taskintervallzeit. Die Abtastzeit findet Verwendung bei der Identifizierung der Strecke, sowie bei eingeschaltetem Regler. Alle anderen Funktionen des Bausteins (Istwerte lesen, Gradientenüberwachung, PWM Ausgabe, Manuelle Regelung) arbeiten mit der Taskintervallzeit. Dadurch ist es möglich, den Regelungs-/Identifizierungsprozess zeitlich von anderen Funktionen des Bausteins zu entkoppeln. Dies ist besonders wichtig, um die Ausgabe der Stellgröße mittels PWM vom Regelungs-/Identifizierungsprozess abzukoppeln.

Die Taskintervallzeit bildet die Grundlage für die Pulsweitenmodulation (PWM), d.h. die Taskintervallzeit ist die kleinste mögliche Zeitscheibe, die bei der PWM auf "An" bzw. "Aus" gesetzt werden kann. Eine Periode der PWM dauert 20-mal länger als die Taskintervallzeit. Damit kann die Stellgröße zwischen 0% und 100% in 5% Schritten variiert werden.

Beispiel:

Die Taskintervallzeit wird für die zyklische Task auf 20 ms parametrier. Das heißt, dass alle 400 ms ein neuer Wert für die Stellgröße an die PWM übergeben werden kann. Mögliche Ein-/Ausschaltzeiten für die PWM sind somit:

Stellgröße [%]	Einschaltdauer [ms]	Ausschaltdauerzeit [ms]
0	0	400
5	20	380
10	40	360
...		
50	200	200
...		
95	380	20
100	400	0

Tab. 7-163: *Beispiel Ein-/Ausschaltzeiten*

Die Wahl der Taskintervallzeit und "ControlUpdateRate" muss stets in Abhängigkeit von Stellglied und Regelstrecke getroffen werden. Zu kleine Taskintervallzeiten verschleissen das Stellglied sehr schnell (Bsp. Relaismodule),

zu große Taskintervallzeiten/Abtastzeiten vermindern die Genauigkeit der Regelung.

Weiterhin ist eine Überwachung der Abtastzeit vorhanden, die bei Über-/Unterschreiten der Abtastzeit um mehr als 20% einen Fehler erzeugt und den Ausgang "Error" auf TRUE setzt.

Empfehlung für die Wahl der Taskintervallzeit

Die Taskintervallzeit/Zykluszeit sollte so eingestellt werden, dass die PWM keine sichtbare Temperaturänderungen innerhalb einer PWM-Periode von $20 \cdot \text{Taskintervallzeit}$ hervorruft. Das heißt, wenn bei einer Stellgröße von z.B. 20% die Temperatur der Strecke während des Heizens deutlich zunimmt und während dem ausgeschalteten Zustand der Heizung die Temperatur wieder abnimmt, ist die Taskintervallzeit zu groß gewählt. Umgekehrt ist es auch nicht sinnvoll die Taskintervallzeit zu klein zu wählen, da das Stellglied dann erhöhten Belastungen ausgesetzt ist (z.B. Relaismodul muss sehr oft schalten).

Empfehlung für die Wahl der Abtastzeit

Eine sinnvolle Taskintervallzeit muss durch Tests ermittelt werden.

Zunächst sollte die Wahl der Taskintervallzeit, wie im vorgehenden Absatz beschrieben, gewählt werden. Dann kann man mittels des Eingangs "ControlUpdateRate" die Abtastzeit auf den gewünschten Wert festlegen.

Eine Empfehlung für die Abtastzeit ist:

- 10ms für schnelle/dynamische Strecken (Strecken Zeitkonstante von 200ms bis 2s)
- 100ms für mittlere Strecken (Strecken Zeitkonstante von 2s bis 20s)
- 1s bis 30s für langsame/träge Strecken (Strecken Zeitkonstante von 20s bis 100min)

Die Abtastzeit sollte etwa 20- bis 200-mal schneller sein als die "dominierende Zeitkonstante" der Regelstrecke. Die "dominierende Zeitkonstante" ist in etwa die Zeit, die eine Regelstrecke benötigt um 63% der Änderung eines aufgeschalteten Sprunges zu durchlaufen. Prinzipiell ist die Abtastzeit jedoch vom Anwender komplett frei wählbar.

Bei zu kleiner Wahl der Abtastzeit kann bei der Identifikation der Fehler "Zeitüberlauf bei Identifikation (Watchdog)" (Additional1 = 16#1605; Additional2 = 16#0009) auftreten. In diesem Fall ist eine größere Abtastzeit zu wählen, da bei Ablauf der Zeitüberwachung die Regelstrecke noch nicht eingeschwungen war.

Bei zu großer Wahl der Abtastzeit kann bei der Identifikation der Fehler "Identifizierte Streckenparameter ungültig, Streckenselbstidentifikation fehlgeschlagen" (Additional1 = 16#1603; Additional2 = 16#0007) auftreten. In diesem Fall ist eine kleinere Abtastzeit zu wählen, da das Streckenverhalten in zu großen zeitlichen Abständen gemessen wurde.

Empfehlung für die Identifikation

Für die Identifikation sind einige Regeln notwendig um gute Ergebnisse zu erzielen. Diese sollen hier näher erläutert werden.

Temperaturstrecken sind in der Regel nichtlinear. Bei kleinen Temperaturen muss weniger Leistung für eine bestimmte Temperaturerhöhung aufgebracht werden als bei größerer Temperatur. Ein Regler benötigt bei niedrigen Temperaturen andere Einstellungen als bei hohen Temperaturen.

Deshalb ist es besonders wichtig, die Identifikation auch in dem Temperaturbereich durchzuführen, in dem der Regler später arbeiten soll. Dabei hat man zwei Möglichkeiten.

1. Bei sehr genauer Regelung in einem kleinen Temperaturbereich sollte zunächst die im Handbetrieb benötigte Leistung so vorgegeben werden, dass sich die gewünschte Temperatur ergibt. Anschließend kann man

dann eine Identifikation durchführen, bei der der Wert für "AutoTune-StartPower" etwas niedriger als der ermittelte Wert liegt und der Wert für "AutoTuneFinishPower" etwas höher als der ermittelte Wert liegt. Hier werden Reglerparameter ermittelt, die genau für diesen Temperaturbereich bestimmt sind. Außerhalb des Temperaturbereiches wird die Regelung dagegen ungenauer sein.

2. Bei einem breiten Regelungsbereich sollte im Handbetrieb jeweils die Leistung für die niedrigste und höchste Temperatur ermittelt werden. Diese dienen dann als Werte für "AutoTuneStartPower" und "AutoTuneFinishPower". Die Regelung wird dabei über den gesamten Temperaturbereich gemittelt. Das bedeutet in der Regel bei kleinen Temperaturen eine etwas dynamischere Regelung und bei höheren Temperaturen eine etwas gedämpftere Regelung.

Beschreibung der einzelnen Funktionen

Für die im Folgenden vorgestellten Anwendungsbeispiele sind im Programm des Anwenders folgende Variablen im Deklarationsteil für die Belegung des Funktionsbausteines zu vereinbaren.

Programm:

```
VAR
fb_TempControl:IL_TempControlType01;      (* Instance of the function blocks *)
strContrParam: IL_TEMPCONTROL_CONTR_PARAM; (* Structure containing the controller parameters *)
strLimits: IL_TEMPCONTROL_LIMITS;         (* Alarm/status limits *)
rActualTemp:REAL;                         (* Actual temperature*)
rActualPower:REAL;                        (* Control variable, analog *)
bActualPower:BOOL;                        (* Control variable, digital (PWM) *)
bAutoTuneDone:BOOL;                       (* Identification state *)
bSetPointReached:BOOL;                    (* Actual value is within the command value window *)
bError:BOOL;                              (* State of the error output *)
iState:INT;                               (* Value in SFC *)
END_VAR
```

In den nachfolgenden Beispielen wurde der Übersicht halber auf eine Fehlerbetrachtung verzichtet.

Einfache Regelung mit Reglerparametervorgabe

Der Funktionsbaustein wird mit den gewünschten Reglerparametern in der Struktur "strContrParam" parametrisiert. Mit einer steigenden Flanke am Eingang "Preset" können neue Reglerparameter auch während der Regelung parametrisiert werden. Der Sollwert ("Setpoint") muss vorgegeben, der aktuelle Istwert ("Feedback") muss zurückgeführt werden. Die Stellgröße kann entweder am Analogausgang ("ControlValueAnalog") oder am Digitalausgang ("ControlValuePWM") abgeholt werden.

Programm:

```
* Setting the controller parameters *)
strContrParam.KP := 1.5;
strContrParam.KI := 2;
strContrParam.KD := 0.6;
strContrParam.Tt := T#1s;

(* Starting the function block at simultaneous transfer *)
(* of the controller parameters *)
fb_TempControl( Enable:=TRUE,
                Setpoint:=50,
                Feedback:=rActualTemp,
                Preset:=TRUE,
                ContrParam:=strContrParam,
                ControlValueAnalog=>rActualPower,
                ControlValuePWM=>bActualPower);
```

Damit werden die Reglerparameter einmalig übernommen und der Regler arbeitet mit den vorgegebenen Werten.

Einfache Identifikation einer Regelstrecke mit anschließendem Übergang in eine Regelung

Der Funktionsbaustein wird zuerst in den Identifikations-/AutoTune-Modus geschaltet. Nach erfolgreichem Abschluss der Identifikation wird die Rege-

lung gestartet, indem am Eingang "StartAutoTune" eine fallende Flanke generiert wird.

Die berechneten Reglerparameter werden in der Struktur "strContrParam" abgespeichert und stehen somit dem Anwender zur Verfügung. Diese Reglerparameter können somit vom Anwender in den Retainbereich der Steuerung gesichert werden (siehe Beispiel ["Retainsicherung der Reglerparameter" auf Seite 491](#)). Damit ist bei einem erneuten Anlauf der Regelung keine Identifikation nötig, es kann sofort die einfache Regelung mit Parametervorgabe gestartet werden.

Mit dem optionalen Eingang "Quickness" wurde eine etwas schnellere Einstellung des Reglers vorgegeben (schnelleres Anregeln, dafür aber höhere Neigung zum Überspringen).

Die Identifikation wird mit einem Sprung von 0% ("AutoTuneStartPower") auf 10% ("AutoTuneFinishPower") Ausgangsleistung vorgenommen

Programm:

```

CASE iState OF
1: (* Identification *)
    fb_TempControl( Enable:=TRUE,
                    StartAutoTune := TRUE
                    Feedback:=rActualTemp,
                    AutoTuneStartPower:= 0,
                    AutoTuneFinishPower:= 10,
                    Quickness:=2,
                    ContrParam:=strContrParam,
                    ControlValueAnalog=>rActualPower,
                    ControlValuePWM=>bActualPower,
                    AutoTuneDone=>bAutoTuneDone);
    IF (bAutoTuneDone = TRUE) THEN
        iState := 2;
    END_IF

2: (* Transition to closed-loop control *)
    fb_TempControl( Enable:=TRUE,
                    StartAutoTune := FALSE,
                    Setpoint:=50,
                    Feedback:=rActualTemp,
                    ContrParam:=strContrParam,
                    ControlValueAnalog=>rActualPower,
                    ControlValuePWM=>bActualPower);

```

Identifikation der Regelstrecke im Arbeitspunkt

Während einer Prozessregelung kann eine erneute Identifikation im Arbeitspunkt gewünscht sein. Ausgehend vom vorgehenden Beispiel "Einfache Identifikation einer Regelstrecke mit anschließendem Übergang in eine Regelung" wird hier nun beschrieben, wie für diesen Fall vorzugehen ist. Mit einem Setzen des Eingangs "AutoTuneStart" wird die Identifikation gestartet, die Anfangs- und Endparameter der Stellgrößenparameter werden in "AutoTuneStartPower" und "AutoTuneFinishPower" übergeben. Die neuen Reglerparameter stehen dem Anwender nach erfolgreicher Identifikation wieder in der Struktur "strContrParam" zur Verfügung.

Im Beispiel wird der Arbeitsbereich, ausgehend von dem aktuellen Wert der Stellgröße (in %) berechnet. Voraussetzung ist natürlich, dass sich die Strecke im eingeschwungenen (stationären) Zustand befindet. Der Arbeitsbereich wird von "Stellgröße-10%" bis "Stellgröße+10%" im Beispiel gewählt. Die Identifikation im Arbeitspunkt startet mit Wechsel in den State "3".

Programm:

```

CASE iState OF
2: (* Closed-loop control *)
    fb_TempControl( Enable:=TRUE,
                    StartAutoTune := FALSE,
                    Setpoint:=50,
                    Feedback:=rActualTemp,
                    ContrParam:=strContrParam,

```

```

ControlValueAnalog=>rActualPower);
If (bIdent = TRUE) THEN
    iState := 3;
End_IF;

3: (* Identification in the operating point *)
fb_TempControl( Enable:=TRUE,
                StartAutoTune := TRUE,
                AutoTuneStartPower:=rActualPower-10,
                AutoTuneFinishPower:=rActualPower+10,
                Feedback:=rActualTemp,
                ContrParam:=strContrParam,
                ControlValueAnalog=>rActualPower,
                AutoTuneDone=>bAutoTuneDone);

IF (bAutoTuneDone = TRUE) THEN
    iState := 2;
END_IF

```

Definition einer Anlauframpe

Erfordert ein Prozess vor der eigentlichen Temperaturregelung das Verhalten an einer bestimmten Temperatur für eine gewisse Zeit (Anlauframpe), so ist diese Funktion über die Eingänge "StartUpTemp" und "StartUpTime" verfügbar.

Der Regler regelt zuerst bis zur Anlauftemperatur. Wenn diese erreicht wird, läuft die "StartUpTime" ab und der Regler hält die Anlauftemperatur konstant. Nach Ablauf der "StartUpTime" ist die Anlauframpe abgeschlossen und der Regler verhält sich in gewohnter Weise. Die Anlauframpe wird nach jedem Wechsel von "Enable" auf "TRUE" einmalig ausgeführt.

Im Beispiel wird zuerst auf eine Temperatur von "100" geregelt, dann 5 Minuten und 30 Sekunden gewartet und danach erst auf die gewünschte Temperatur von "150" geregelt (siehe [Abb. 7-106 "Verwendung eines Sollwertgradienten und Anlauframpe" auf Seite 491](#)).

Programm:

```

fb_TempControl( Enable:=TRUE,
                Setpoint:=150,
                Feedback:=rActualTemp,
                Preset:=TRUE,
                StartUpTemp:=100,
                StartUpTime:=t#5m30s,
                ContrParam:=strContrParam,
                ControlValueAnalog=>rActualPower);

```

Definition von Alarm- und Statusschranken

Zur Prozessüberwachung können Alarm- und Statusschranken angelegt werden.

Die "Limits"-Struktur beinhaltet die Werte "UpperAlarm" und "LowerAlarm". Bei Verletzung der Alarmschranken wird ein Fehler generiert. Die Überwachung wird erst aktiv, wenn sich die aktuelle Prozesstemperatur im Bereich zwischen "UpperAlarm" und "LowerAlarm" befindet, um während des Prozessanlaufs keine Fehler zu erzeugen. Bei den Werten handelt es sich um absolute Werte.

Die "Limits"-Struktur beinhaltet außerdem die Werte "UpperAtSetPoint" und "LowerAtSetPoint". Der Ausgang "SetPointReached" wird gesetzt sobald die Bedingung $(\text{Setpoint} - \text{Limits.LowerAtSetPoint}) \leq \text{Istwert} \leq (\text{Setpoint} + \text{Limits.UpperAtSetPoint})$ erfüllt ist. Die Werte sind relative Werte die sich auf den "Setpoint" beziehen. Somit kann der "Setpoint" geändert werden, ohne "Limits.LowerAtSetPoint" und "Limits.UpperAtSetPoint" ändern zu müssen. Anhand dieser Statusanzeige kann man erkennen, ob die gewünschte Temperatur bereits erreicht wurde.

Programm:

```

strLimits.UpperAlarm:=250;
strLimits.LowerAlarm:=70;

```



```

strLimits.UpperAtSetpoint:=20;
strLimits.LowerAtSetPoint:=20;

fb_TempControl(    Enable:=TRUE,
                   Setpoint:=150,
                   Feedback:=rActualTemp,
                   Limits:=strLimits,
                   ContrParam:=strContrParam,
                   ControlValueAnalog=>rActualPower,
                   SetPointReached=>bSetPointReached);

```

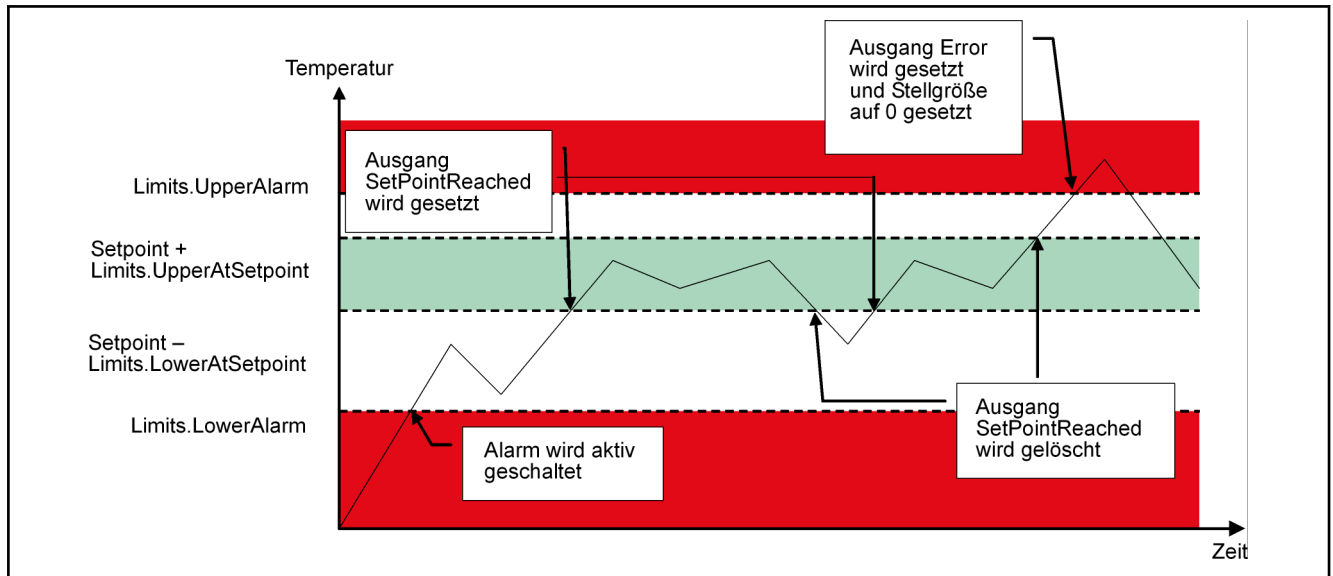


Abb. 7-105: Wirkungsweise der Alarm- und Statusschranken

Anhalten der Regelung (PAUSE-Modus)

Ein Anhalten der Regelung kann über den Eingang "Pause" realisiert werden. Dabei wird die Stellgröße konstant gehalten und es wird der Regler mit all seinen Werten "eingefroren". Nach dem Rücksetzen des Eingangs "Pause" läuft die Regelung weiter.

Im Beispiel wird in Zustand "4" die Temperatur auf "150" geregelt. Mit Umschalten auf Zustand 5 wird die Regelung "eingefroren" und keine neuen Stellgrößen ausgegeben. Mit Rückkehr auf Zustand "4" läuft die Regelung weiter, neue Stellgrößen werden wieder ausgegeben. Es kann dabei zu einem Sprung kommen, wenn sich zwischenzeitlich der Istwert stark verändert hat.

Programm:

```

CASE iState OF
4:
fb_TempControl(    Enable:=TRUE,
                   Pause:=FALSE,
                   Setpoint:=150,
                   Feedback:=rActualTemp,
                   Preset:=TRUE,
                   ContrParam:=strContrParam,
                   Limits:=strLimits,
                   ControlValueAnalog=>rActualPower);

5:
fb_TempControl(    Enable:=TRUE,
                   Pause:=TRUE,
                   Setpoint:=150,
                   Feedback:=rActualTemp,
                   Preset:=TRUE,
                   ContrParam:=strContrParam,
                   Limits:=strLimits,
                   ControlValueAnalog=>rActualPower);

```

Analoger/Digitaler Stellgrößenausgang

Der Funktionsbaustein verfügt über zwei Stellgrößenausgänge, einen Analog- ("ControlValueAnalog") an welchem REAL-Werte zwischen "0%" (keine

Leistung) und "100%" (volle Leistung) ausgegeben werden und einen digitalen ("ControlValuePWM") an welchem die BOOL-Werte "FALSE" (keine Leistung) oder "TRUE" (volle Leistung) ausgegeben werden. Der digitale Ausgang kann bei Benutzung von Relaismodulen zur Pulsung der Leistung genutzt werden. Die Pulsung erfolgt über eine Pulsweitenmodulation, deren Zykluszeit 20-mal langsamer ist als die zyklische Task in der der FB läuft. Somit können Stellgrößen mit einer Auflösung von „0%“ bis „100%“ in „5%“ Schritten moduliert werden.

Der analoge Ausgang kann für eine Analogausgabe oder zur Überwachung der aktuellen Stellgröße verwendet werden.

Programm:

```
fb_TempControl(  Enable:=TRUE,
                  Pause:=FALSE,
                  Setpoint:=150,
                  Feedback:=rActualTemp,
                  Preset:=TRUE,
                  ContrParam:=strContrParam,
                  Limits:=strLimits,
                  ControlValueAnalog=>rActualPower
                  ControlValuePWM=>bActualPower);
```

```
fb_TempControl( Enable:=TRUE, Pause:=FALSE, Setpoint:=150, Feedback:=rActualTemp, Preset:=TRUE, ContrParam:=strContrParam, Limits:=strLimits, ControlValueAnalog=>rActualPower ControlValuePWM=>bActualPower);
```

Vorgabe eines Sollwertgradienten

Der Funktionsbaustein bietet die Möglichkeit den Anstieg des Sollwertes zu begrenzen (zu verrampen). Dies vermindert die Überschwingweite und schützt die Anlage. Dabei wird bei jeder neuen Vorgabe des Eingangs "Set-Point" der Sollwert intern rampenförmig (nicht sprungförmig) solange verändert, bis die gewünschte Sollgröße erreicht ist. Die Eingabe von "SetPoint-Gradient" erfolgt in [u/s].

Im Beispiel wird die Solltemperatur auf "150" gefahren mit einem Anstieg von "0,5 je Sekunde".

Programm:

```
fb_TempControl(  Enable:=TRUE,
                  Pause:=FALSE,
                  Setpoint:=150,
                  Setpointgradient:=0.5,
                  Feedback:=rActualTemp,
                  Preset:=TRUE,
                  ContrParam:=strContrParam,
                  Limits:=strLimits,
                  ControlValueAnalog=>rActualPower);
```

Eine Verrampung mit Sollwertgradient und zusätzlicher Anlauframpe hätte dann den, wie folgt gezeigten, zeitlichen Verlauf von Sollwert und Istwert. Hier wurde auf eine Solltemperatur ("Setpoint:=200") von "200°C" gefahren mit einem Gradienten ("Setpointgradient:=1") von "1°C/s". Die Anlauftemperatur ("StartUpTemp:=100") beträgt "100°C", die Anlaufzeit ("StartUp-Time:=T#50s") "50s".

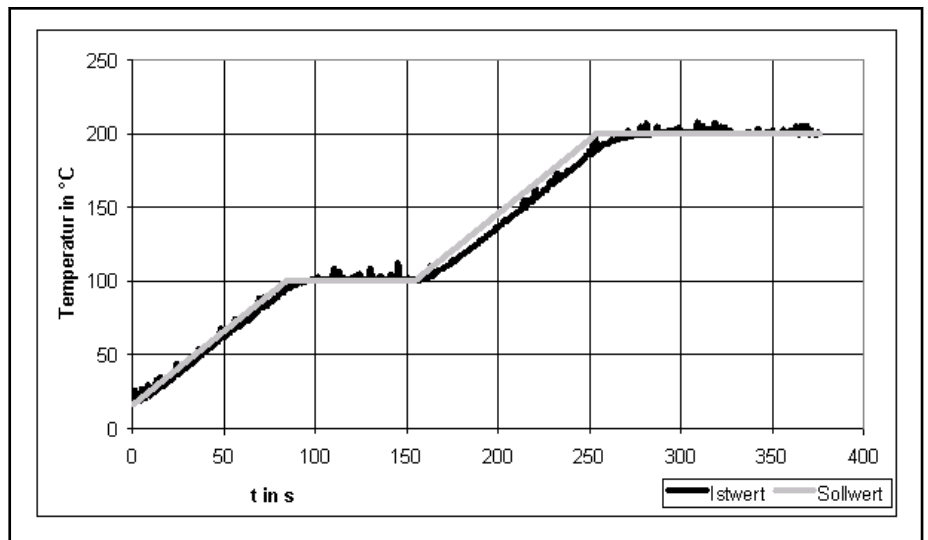


Abb. 7-106: Verwendung eines Sollwertgradienten und Anlauframpe

Vorgabe einer manuellen Stellgröße

Um die Temperatur manuell zu steuern, lässt sich eine Handstellgröße festlegen. Diese stellt gleichzeitig die Stellgröße am Ausgang des Funktionsbausteins dar. Eine Handstellgröße kann benötigt werden, um manuelle Testfunktionen aufzuzeichnen oder bei Sensorfehler in einen Notbetrieb übergehen zu können.

Im Beispiel wird der FB manuell mit einer Stellgröße von "50%" angesteuert.

Programm:

```
b_TempControl( Enable:=TRUE,
               Pause:=FALSE,
               OperatingMode:=TRUE,
               ManContrValue:=50,
               ContrParam:=strContrParam,
               Limits:=strLimits,
               ControlValueAnalog=>rActualPower);
```

Retainsicherung der Reglerparameter

Um die Reglerparameter spannungsausfallsicher zu speichern, bietet es sich an, diese in den Retainbereich zu sichern. Somit kann beim Wiedereinschalten des Reglers die Identifikation eingespart und sofort mit der Regelung begonnen werden.

Dazu muss vom Anwender die Struktur "strContrParam" als Retainvariablen ("VAR RETAIN") angelegt werden, die dann die Werte der einzelnen Reglerparameter beinhaltet. Gleichzeitig muss ein Mechanismus geschaffen werden, um beim Anlaufen zu unterscheiden, ob schon einmal eine Identifikation durchgeführt wurde.

Im Header müssen zusätzlich Retainvariablen angelegt werden.

Programm:

```
VAR RETAIN
    strContrParam: STRUCT_CONTR_PARAM;
    bIdentification: BOOL := FALSE;
END_VAR

If (bIdentification = FALSE) THEN
    fb_TempControl( Enable:=TRUE,
                   StartAutoTune := TRUE,
                   Feedback:=rActualTemp,
                   Quickness:=2,
                   ContrParam:=strContrParam,
                   ControlValueAnalog=>rActualPower,
                   ControlValuePWM=>bActualPower,
                   AutoTuneDone=>bAutoTuneDone);
    IF (bAutoTuneDone = TRUE) THEN
        bIdentification:= TRUE;
    END IF;
END IF;
```

```

    END_IF
ELSE
    fb_TempControl( Enable:=TRUE,
                    StartAutoTune := FALSE
                    Preset:=TRUE,
                    Setpoint:=150,
                    Feedback:=rActualTemp,
                    ContrParam:=strContrParam,
                    Limits:=strLimits,
                    ControlValueAnalog=>rActualPower);

END_IF;

```

Die Werte des Reglers werden im obigen Beispiel in jedem Zyklus in die Retainvariablen gesichert. Nach einem Abschalten/Netzausfall prüft der Code anhand der booleschen Variable "bldentification" ob bereits eine Identifizierung stattfand. Ist dies nicht der Fall, wird eine Identifizierung durchgeführt. Ansonsten kann der Regler mit den gesicherten (alten) Parametern neu gestartet werden. Der Eingang "Preset" muss dabei mit einer steigenden Flanke belegt werden.

Fehlerbehandlung mittels Ersatzwertstrategie

Im Fehlerfall wird vom Funktionsbaustein standardmäßig die Stellgröße "0%" (keine Leistung) ausgegeben. Dies wird aus Gründen der Anlagensicherheit getan. Wird vom Anwender eine Steuerung auf eine bestimmte Leistung gewünscht, um beispielsweise ein starkes Abkühlen einer Anlage zu verhindern, empfiehlt es sich, die im folgenden Beispiel gezeigte Ersatzwertstrategie zu verwenden.

Im Zustand "6" wird mit "bError" auf Fehler geprüft. Falls diese Bedingung "TRUE" wird, wird der Funktionsbaustein zurückgesetzt ("Enable:=FALSE") und in Zustand "7" übergegangen.

Im Zustand "7" wird der Funktionsbaustein mit einer steigenden Flanke am Eingang "Enable" neu gestartet und im manuellen Modus (siehe "[Vorgabe einer manuellen Stellgröße](#)" auf Seite 491) betrieben.

Im Beispiel wird eine Stellgröße von "50%" ausgegeben.

Nähere Informationen zum Fehler werden an den Ausgängen "ErrorID" und "ErrorIdent" ausgegeben, die jedoch in diesem Beispiel nicht ausgewertet wurden.

Programm:

```

CASE iState OF
6:
    fb_TempControl( Enable:=TRUE,
                    Setpoint:=50,
                    Feedback:=rActualTemp,
                    Preset:=TRUE,
                    ContrParam:=strContrParam,
                    Limits:=strLimits,
                    ControlValueAnalog=>rActualPower,
                    Error=>bError);

    IF (bError = TRUE) THEN
        fb_TempControl( Enable:=FALSE,
                        ContrParam:=strContrParam,
                        Limits:=strLimits);

        iState := 7;
    END_IF

7:
    fb_TempControl( Enable:=TRUE,
                    Pause:=FALSE,
                    OperatingMode:=TRUE,
                    ManContrValue:=50,
                    ContrParam:=strContrParam,
                    Limits:=strLimits,
                    ControlValueAnalog=>rActualPower);

```

Anzumerken ist hierbei noch, dass der Stellgrößenausgang im obigen Beispiel für einen Zyklus auf 0 gesetzt wird (wenn der Fehler erkannt und Enable:=FALSE gesetzt wird). Im Zustand "7" wird dann der manuelle Wert als Stellgröße verwendet.

Fehlerbehandlung Der Funktionsbaustein erzeugt in Additional1/Additional2 für die Tabelle **F_RELATED_TABLE**, 16#0170, folgende Fehlermeldungen:

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#1600	Aktueller State	Fehler in der Zustandsmaschine, ungültiger Zustand
INPUT_RANGE_ERROR	16#1601	16#0001	Eingabewerte für maximalen Ausgabewert ("OutputPowerMax", "AutoTuneStartPower" oder "AutoTuneFinishPower") außerhalb des gültigen Bereichs. Wert muss größer gleich 0[%] bis max. 100[%] sein
		16#1602	Eingabewerte für Alarmschranken ungültig (Bedingung: Limits.UpperAlarm > Limits.LowerAlarm)
		16#1603	Eingabewert für manuelle Stellgröße ungültig. Wert muss im Bereich von 0[%] und 100[%] liegen
		16#0004	Reglerparameter ungültig
		16#000A	Start- und Endwert für Identifikation sind zu nah beieinander. Die Mindestdifferenz ist 5[%]
		16#000B	Der Eingang "SetpointGradient" ist ungültig. Wert muss größer 0 sein
		16#000C	Eingabewerte für Statusschranken ungültig (Bedingung: (Setpoint + Limits.UpperAtSetPoint) > (Setpoint - Limits.LowerAtSetPoint))
		16#000D	Zeit für Identifikation ist zu groß gewählt ("AutoTuneTimeOut")
		16#000E	Wert für "Quickness" ist nicht innerhalb gültiger Grenzen (1..5)
		16#000F	Wert für "ControlUpdateRate" ist nicht innerhalb gültiger Grenzen (1..1000)
SYSTEM_ERROR	16#1602	Abtastzeit	Jittern der Abtastzeit > 20[%]. Fehler wird erst bei 2 aufeinander folgenden Zykluszeitüberschreitungen aktiv
		16#0005	Gradient des Istwertes zu groß, möglicherweise Sensorfehler (Diagnose)
		16#0006	Fehler im Regelkreis, Sensor möglicherweise verrutscht/zerstört
CALCULATION_ERROR	16#1603	16#0007	Identifizierte Streckenparameter ungültig, Streckenselbstidentifikation fehlgeschlagen
		16#0008	Das identifizierte Prozessmodell ist zu schlecht/ ungültig

ErrorID	Additional1	Additional2	Beschreibung
ACCESS_ERROR	16#1604	Feldnummer des Array	Fehlerhafter Zugriff auf interne Variable
OTHER_ERROR	16#1605	16#0009	Zeitüberlauf bei Identifikation (Watchdog)
DEVICE_ERROR	16#1606	Temperatur	Temperaturfehlerschranke überschritten, Temperatur außerhalb der Fehlerschranken

Tab. 7-164: Fehlercodes Funktionsbaustein IL_TempControlType01

Hinweise zu bestimmten Fehlern

- Fehler "Gradient des Istwertes zu groß"**
 (Additional1 = 16#1602; Additional2 = 16#0005):
 Hier wird der Gradient des Istwertes überwacht. Ist die Differenz zwischen zwei aufeinander folgenden Istwerten größer als 10u/ms, wird dieser Fehler generiert. Dies kann bei Einsatz eines Inline-Temperaturmoduls der Fall sein, denn dort werden Fehlerdiagnosen als Messwert mit Werten größer gleich 32768 ausgegeben (Bit 15 gesetzt). Wird anwenderseitig diese Diagnose nicht ausgewertet, kann es zu dieser Fehlermeldung kommen.
- Fehler "Fehler im Regelkreis, Sensor möglicherweise verrutscht/zerstört"**
 (Additional1 = 16#1602; Additional2 = 16#0006):
 Hier wird überwacht, ob bei maximaler Ausgangsleistung der Istwert sich dem Sollwert nähert. Dazu wird eine Zeitüberwachung gestartet, die bei aktivem I-Anteil $2000 / K_i + T_t$ und ohne I-Anteil $2000 \cdot \text{Abtastzeit} + T_t$ beträgt.
 Bewegt sich der Istwert innerhalb dieser Zeit nicht in Richtung Sollwert, wird der Fehler generiert. Es besteht die Möglichkeit, dass der Sensor defekt/verrutscht ist, oder die Heizung/Kühlung nicht funktioniert. Auch kann der Fehler auftreten, wenn bei Vorgabe der Reglerparameter über die Struktur "IL_TEMPCONTROL_CONTR_PARAM" keine Totzeit (T_t) angegeben wird. In diesem Fall sollte für die Totzeit ein geschätzter Wert angegeben und die Reglerparameter neu gesetzt werden. Bei der Identifikation der Strecke erfolgt dies automatisch.
- Fehler "Identifizierte Streckenparameter ungültig, Streckenselbstidentifikation fehlgeschlagen"**
 (Additional1 = 16#1603; Additional2 = 16#0007):
 Die Streckenparameter ergaben ungültige Werte. Dies kann sich durch falsche Wahl der Abtastzeit oder fehlerhaften Istwert (Sensor nicht angeschlossen) ergeben (siehe auch ["Empfehlung für die Wahl der Abtastzeit" auf Seite 485](#))
- Fehler "Das identifizierte Prozessmodell ist zu schlecht/ungültig"**
 (Additional1 = 16#1603; Additional2 = 16#0008):
 Bei der Streckenidentifizierung wird das errechnete Modell mit den gemessenen Werten verglichen. Der Vergleich ergab keine zufriedenstellende Übereinstimmung. Das Modell weicht um mehr als 20% von den Messwerten ab. Unter Umständen kann die Abtastzeit falsch gewählt sein, oder die Strecke weist kein aperiodisches Verhalten auf. Auch Störgrößen, die während der Identifikation auftreten, können dies verursachen

- Fehler "Zeitüberlauf bei Identifikation (Watchdog)"

(Additional1 = 16#1605; Additional2 = 16#0009):

Bei der Streckenidentifizierung wird gewartet bis der Istwert eingeschwungen ist, dann erfolgt ein Sollwertsprung und es wird erneut gewartet bis sich der Istwert eingeschwungen hat. Dauert dies zu lange, wird der Fehler generiert. Die Ursache liegt entweder in einem stark schwankenden Istwert, oder in einer falsch gewählten Abtastzeit (siehe auch nächster Absatz)

- Fehler "Zeit für Identifikation ist zu groß gewählt ('AutoTuneTimeOut')"

(Additional1 = 16#1601; Additional2 = 16#000D):

Bei der Streckenidentifizierung kann der Wert für die Zeitüberwachung eingestellt werden. Dieser Wert kann maximal so groß sein wie 10000 * Abtastzeit. In diesem Fall entweder die Zeitüberwachung auf 0 setzen (maximal mögliche Zeit der Identifizierung bei gegebener Abtastzeit), oder die Abtastzeit entsprechend erhöhen

7.9.3 IL_TempControlType02

Kurzbeschreibung

Der Funktionsbaustein IL_TempControlType02 dient, wie bereits der Funktionsbaustein [IL_TempControlType01, Seite 477](#), zur Regelung von Temperaturen. Er kann die Reglerparameter entweder selbst ermitteln oder mit Vorgaben arbeiten.

Die Vorzüge dieser zweiten Variante sind:

- die verkürzte Identifikationszeit (StartAutoTune)
- die geänderte Reglerparametrierung (Preset)
- die geänderte Reglerfeinjustierung (DampingFactor)
- die Ein-/Ausgabe der Streckenparameter (ModelParam)
- die individuelle Reglerauslegung (ControlMode)



Der Funktionsbaustein "IL_TempControlType02 (ML_TechBase-Bibliothek)" wurde mit Version 14VRS in die "RIL_LoopControl-Bibliothek" übernommen.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek
MLC 15VRS	RIL_LoopControl

Tab. 7-165: Referenztablette Funktionsbaustein IL_TempControlType02

Schnittstellenbeschreibung

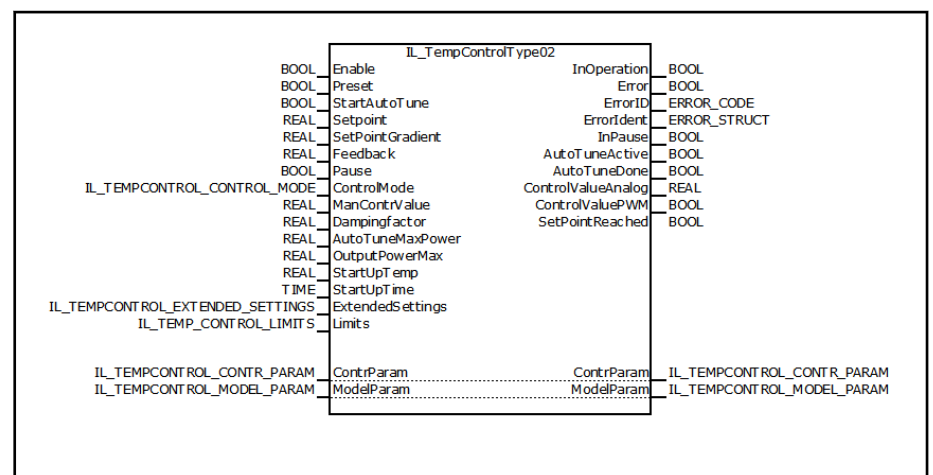


Abb. 7-107: Funktionsbaustein IL_TempControlType02

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Der Funktionsbaustein ist aktiv, wenn dieser Eingang "TRUE" ist. Die FB-Ausgänge werden mit einer fallenden Flanke auf ihre Defaultwerte zurückgesetzt
	Preset	BOOL	Eine steigende Flanke an diesem Eingang übernimmt die Regler- oder Streckenparameter der Strukturen "ContrParam" und/oder "ModelParam"
	StartAutoTune	BOOL	Eine steigende Flanke an diesem Eingang startet die Selbsteinstellung. Wird dieser Eingang gleichzeitig mit "Enable" gesetzt, ist keine Vorgabe von Reglerparametern nötig
	SetPoint	REAL	Solltemperatur. Einheit [u]*
	SetPointGradient	REAL	Maximale Änderung des Parameters "Setpoint". Einheit [u/s]
	Feedback	REAL	Isttemperatur. Einheit [u]
	Pause	BOOL	Dieser Eingang "friert" Stellgröße und Reglerberechnung ein
	ControlMode	IL_TEMPCONTROL_CONTROL_MODE	Auswahl des Regelkreisverhaltens. Steuerung im offenen Regelkreis oder Regelung optimiert auf Führungs- oder Störverhalten im geschlossenen Regelkreis IL_TEMPCONTROL_CONTROL_MODE, Seite 506
	ManContrValue	REAL	Dieser Eingang gibt die Stellgröße bei manueller Steuerung vor. Einheit [%]
	DampingFactor	REAL	Dieser Eingang wirkt auf das Regelkreisverhalten. Es kann stufenlos zwischen einem langsamen aperiodischen und einem schnellen oszillierenden Verhalten gewählt werden. Einheit [-]
	AutoTuneMaxPower	REAL	Stellgrößenbegrenzung zu Beginn der Selbsteinstellung. Einheit [%]
	OutputPowerMax	REAL	Begrenzung des Reglerausgangs. Ist eine Stellsignalbegrenzung in der Strecke kleiner dieser Begrenzung des Reglerausgangs vorhanden, so kann es zu Einschränkungen in der Anti-Windup-Funktion des Reglers kommen. Einheit [%]
	StartUpTemp	REAL	Anfahrtemperatur. Wird nur aktiv, wenn der Eingang "StartUpTime" mit einem Wert ungleich 0 belegt wird. Einheit [u]

E/A-Typ	Name	Datentyp	Beschreibung
	StartUpTime	TIME	Anfahrzeit, d.h., Zeit, in der die Anfahrtemperatur ("StartUpTemp") gehalten wird bis in die "normale" Regelung übergegangen wird. Voreingestellter Wert ist $T_{#0s}$ (nicht aktiv)
	ExtendedSettings	IL_TEMPCONTROL_EXTENDED_SETTINGS	Struktur, die erweiterte Einstellungsmöglichkeiten der Selbsteinstellung beinhaltet. Die Standardeinstellungen sind für übliche Strecken ausreichend IL_TEMPCONTROL_EXTENDED_SETTINGS, Seite 507
	Limits	IL_TEMPCONTROL_LIMITS	Struktur, die die Alarmschranken in [u] (Einheiten) enthält. IL_TEMPCONTROL_LIMITS, Seite 508 Einheit [u]
VAR_OUTPUT	InOperation	BOOL	Dieser Ausgang wird gesetzt wenn "Enable" gesetzt ist und der Baustein aktiv ist
	Error	BOOL	Ein Fehler ist während des Betriebes aufgetreten. Die Stellgröße wird auf 0 [%] gesetzt
	ErrorID	ERROR_CODE	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang eine grobe Klassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang detaillierte Informationen über den Fehler Fehlerbehandlung, Seite 504
	InPause	BOOL	Der Ausgang "ControlValueAnalog" ist "eingefroren" und der Ausgang "ControlValuePWM" ist weiter mit diesem Wert aktiv
	AutoTuneActive	BOOL	Die Selbsteinstellung ist aktiv
	AutoTuneDone	BOOL	Die Selbsteinstellung wurde erfolgreich beendet
	ControlValueAnalog	REAL	Analoger Wert der Stellgröße. Einheit [%]
	ControlValuePWM	BOOL	Digitaler, pulsweitenmodulierter Stellgrößenwert. Einheit [-]
	SetPointReached	BOOL	Statusausgang ob sich der Istwert innerhalb der Schranken ($\text{Setpoint} + \text{Limits.UpperAtSetpoint}$) $\geq$ Istwert $\geq$ ($\text{Setpoint} - \text{Limits.LowerAtSetpoint}$) befindet

E/A-Typ	Name	Datentyp	Beschreibung
VAR_IN_OUT	ContrParam	IL_TEMPCONTROL_CONTR_PARAM	Diese Struktur enthält die aktuellen Parameter K_P , K_I und K_D des Reglers sowie die Totzeit T_t IL_TEMPCONTROL_CONTR_PARAM, Seite 506
	ModelParam	IL_TEMPCONTROL_MODEL_PARAM	Diese Struktur enthält die Parameter K_S und T_S der Regelstrecke IL_TEMPCONTROL_MODEL_PARAM, Seite 509

*

Die Einheit [u] bedeutet "unit". Dies kann eine beliebige Einheit wie z.B. [°C] oder [°F] sein. Es ist darauf zu achten, dass die gewählte Einheit einheitlich verwendet wird.

Tab. 7-166: Schnittstellenvariablen Funktionsbaustein IL_TempControlType02

Min- Max- und Default-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	Zyklisch
Preset	BOOL	FALSE	TRUE	FALSE	Zyklisch
StartAutoTune	BOOL	FALSE	TRUE	FALSE	Zyklisch
SetPoint	REAL	N. Def.	N. Def.	0	Zyklisch
SetPointGradient	REAL	0	N. Def.	0	Zyklisch
Feedback	REAL	N. Def.	N. Def.	0	Zyklisch
Pause	BOOL	FALSE	TRUE	FALSE	Zyklisch
ControlMode	IL_TEMPCONTROL_CONTROL_MODE	CONTROL_TEMP_MODE_MANUAL	CONTROL_TEMP_MODE_ONLY_DISTURBANCE	CONTROL_TEMP_MODE_ONLY_DISTURBANCE	Zyklisch
ManContrValue	REAL	0	100	0	Zyklisch
DampingFactor	REAL	20	200	100	Zyklisch
AutoTuneMaxPower	REAL	>0	100	80	Bei steigender Flanke an "StartAutoTune"
OutputPowerMax	REAL	>0	100	100	Zyklisch
StartUpTemp	REAL	N. Def.	N. Def.	0	Bei steigender Flanke an "Enable"
StartUpTime	TIME	T#0s	N. Def.	T#0s	Bei steigender Flanke an "Enable"
ExtendedSettings	IL_TEMPCONTROL_EXTENDED_SETTINGS	Siehe Datenstruktur	Siehe Datenstruktur	Siehe Datenstruktur	Siehe Datenstruktur
Limits	IL_TEMPCONTROL_LIMITS	Siehe Datenstruktur	Siehe Datenstruktur	Siehe Datenstruktur	Siehe Datenstruktur

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
ContrParam	IL_TEMPCONT- ROL_ CONTR_PARAM	Siehe Datenstruktur	Siehe Datenstruktur	Siehe Datenstruktur	Siehe Datenstruktur
ModelParam	IL_TEMPCONT- ROL_ MODEL_PARAM	Siehe Datenstruktur	Siehe Datenstruktur	Siehe Datenstruktur	Siehe Datenstruktur

Tab. 7-167: Min- Max- und Default-Werte sowie Übernahmeverhalten der Eingänge des Funktionsbausteins IL_TempControlType02

Funktionsbeschreibung

Der Funktionsbaustein kann beim System MLC betrieben werden. Die zu regelnde Ist-Temperatur kann hierbei mit Hilfe von Inline Temperatur-Modulen (IL TEMP 2 RTD) ermittelt oder über eine anderweitige Quelle (z.B. Feldbusverbindung) bereitgestellt werden.

Bis auf die Funktionsweise des Auto-Tunings "StartAutoTune", der Reglerparametrierung "Preset", der Reglerfeinjustierung "DampingFactor", der In-Output-Struktur "ModelParam" und der Möglichkeit der Reglerauslegung "ControlMode" ist der Funktionsbaustein IL_TempControlType02 identisch zu dem Funktionsbaustein [IL_TempControlType01, Seite 477](#).

Der interne PID-Regler kann entweder von Anwender selbst als P-, PI-, PD- oder PID-Regler parametrierbar werden, oder der Funktionsbaustein kann automatisiert "StartAutoTune"-Reglerparameter durch eine Selbsteinstellung ermitteln.

Regler-/Streckenparameter

Der Identifikationsprozess des selbsteinstellenden Temperaturregler-Funktionsbausteins IL_TempControlType02 wurde im direkten Vergleich zu dem Funktionsbaustein IL_TempControlType01 so modifiziert, dass nun ab Start der Selbsteinstellung der Regelkreis geschlossen ist, und die Reglerparameter kontinuierlich den aktuell identifizierten Streckenparametern nachgeführt werden.

Diese Funktionalität wird erreicht, indem der Eingang "StartAutoTune" mit einem Reglerauslegungsverfahren "ControlMode" gesetzt wird.

Bei bekannten Regler- oder Streckenparametern kann auf eine Selbsteinstellung verzichtet werden. In diesem Fall werden die Werte über die Eingänge "ContrParam" und/oder "ModelParam" vorgegeben und mit einer steigenden Flanke am Eingang "Preset" übernommen.

Der Istwert der Temperatur "Feedback" wird im Funktionsbaustein zur Verminderung von Störungen kontinuierlich mittels eines PT_1 -Filters geglättet. Die Zeitkonstante des Filters beträgt hierbei fest das 20-fache der Zykluszeit ($T_{Filt}=20 \cdot T_{Task}$).

Infolge dieser und weiterer sekundärer prozesstechnischer Verzugszeiten können Temperaturregelstrecken erfahrungsgemäß als nicht schwingungsfähige PT_2 -Strecken mit einer dominanten Ausgleichzeitkonstante T_s und einer sekundären Verzugszeit T_t angenommen werden.

$$\frac{T_{ref}(s)}{u_{reg}(s)} \approx \frac{K_s}{(1+s \cdot T_s) \cdot (1+s \cdot T_t)}$$

Nachfolgend ist eine beispielhafte Sprungantwort einer Temperaturregelstrecke unter Annahme der Streckenparameter $K_s=1^\circ\text{C}/\%$, $T_s=10\text{s}$ und $T_t=1\text{s}$ dargestellt.

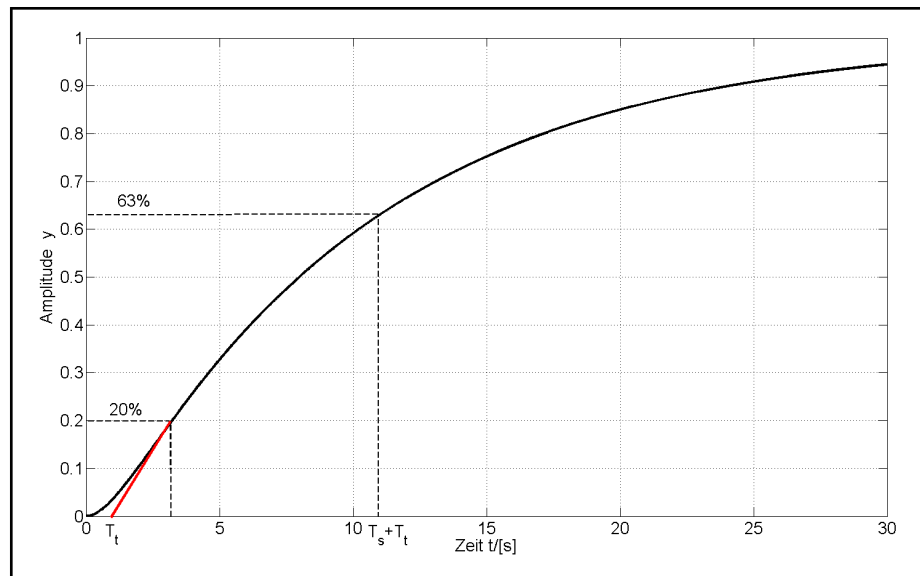


Abb. 7-108: Sprungantwort einer nicht schwingungsfähigen PT_2 -Strecke

Bei einer Selbsteinstellung wird direkt mit einer steigenden Flanke am Eingang "StartAutoTune" der Regelkreis mit einem P-Regler geschlossen um eine Streckenreaktion hervorzurufen. Der initiale, proportionale Reglerparameter K_P wird hierbei entweder aus der Struktur "ContrParam" entnommen, aus den anliegenden Streckenparametern (K_s, T_s, T_t) berechnet oder mit einem positiven intern hinterlegten Wert vorbelegt. Während sich der Istwert in Richtung Sollwert bewegt, wird das Streckenverhalten kontinuierlich identifiziert und die Reglereinstellungen entsprechend den gewonnenen Streckeninformationen nachgeführt. Nach abgeschlossener Identifikation wird das AutoTuning automatisch beendet und die ermittelten Regler- und Streckenparameter in die Strukturen "ModelParam" (K_s, T_s) und "ContrParam" (K_P, K_I, K_D, T_t) übernommen.

Kühlung Im Falle einer Kühlung muss der Eingang K_P mit einem negativen Wert, zur Erzeugung einer ersten Systemanregung, vorbelegt werden. Infolge der vorhandenen Stellsignalbegrenzung, sowie der vorhandenen Regleradaption, kann bei einer Erstidentifikation im Zweifelsfall $K_P = -10000$ vorgegeben werden.

Stellsignalbegrenzung Bereits nach einer Istwertänderung von 20% der Sollwertänderung sind ausreichend Informationen über das dynamische Streckenverhalten vorhanden um Reglerparameter berechnen zu können. Demzufolge wird zu diesem Zeitpunkt für eine möglichst schnelle Anregelzeit die Stellsignalbegrenzung "AutoTuneMaxPower" des automatisch eingestellten PI-Reglers auf den maximalen Reglerausgang "OutputPowerMax" erhöht.

Preset Reglerparameter Der interne PID-Regler kann durch zu Null setzen einzelner Regleranteile auch als P-Regler, PI-Regler, PD-Regler oder PID-Regler verwendet werden. Die Reglerparametrierung als rein integrierender I-Regler wird hingegen, in Hinblick auf das zu erwartende Regelstreckenverhalten, nicht unterstützt.

$$K_P = \text{ContrParam}.K_P, \quad K_I = \text{ContrParam}.K_I, \quad K_D = \text{ContrParam}.K_D$$

Abb. 7-109: Formale Abhängigkeit der Reglerparameter bei einem "Preset"

Preset Streckenparameter Neben der Reglerparametrierung mittels vorgegebener Reglerparameter ist ebenfalls eine Parametrierung mittels vorgegebener Streckenparameter möglich. Sind zum Zeitpunkt der steigenden Flanke am "Preset"-Eingang sämtliche Reglerparameter K_P , K_I und K_D der Struktur "ContrParam" Null, so wird

intern geprüft ob sämtliche Streckenparameter K_s , T_s und T_t in den Strukturen "ModelParam" und "ContrParam" anliegen. Nach einer Plausibilitätsprüfung dieser Streckenparameter wird anschließend auf Grundlage dieser ein PI-Regler ausgelegt. Vorteilhaft an dieser Reglerparametrierung ist die Möglichkeit den so eingestellten Regler zwischen Stör- und Führungsverhalten umschalten, sowie den Regler mittels des "DampingFactor" nachjustieren zu können.

$$K_p = f(\text{ModelParam}.K_s, \text{ModelParam}.T_s, \text{ModelParam}.T_t, \text{DampingFactor}, \text{ControlMode})$$

$$K_I = f(\text{ModelParam}.K_s, \text{ModelParam}.T_s, \text{ModelParam}.T_t, \text{DampingFactor}, \text{ControlMode})$$

Abb. 7-110: Formale Abhängigkeit der Reglerparameter bei einem "Preset" mit Streckenparametern



Für eine Sicherung bewährter Strecken- und/oder Reglerparameter bietet es sich an, diese in den Retainbereich zu sichern.

Hierzu können die Strukturen "ContrParam" und/oder "ModelParam" welche die Regler- und/oder Streckenparameter beinhalten als Retainvariablen (VAR RETAIN) angelegt werden.

AutoTuneMaxPower

Infolge einer erfahrungsgemäß vorhandenen dominanten Zeitkonstanten T_s und einer sekundären Verzugszeit T_t ergibt sich ohne Stellsignalbegrenzung bis zu einem Zeitpunkt $t \sim T_s/3$ eine Sprungantwort, welche ebenfalls durch ein IT_1 -System beschrieben werden kann (siehe Abb. 7-108 "Sprungantwort einer nicht schwingungsfähigen PT_2 -Strecke" auf Seite 500). Um in diesem frühen Zeitraum der Selbstjustierung (bis 20% der Istwertänderung) bereits Reglerparameter bestimmen zu können, kann es je nach Verhältnis der Streckenzeitkonstanten nötig sein, hier das Reglerausgangssignal zusätzlich zu begrenzen ("AutoTuneMaxPower"). Somit kann erreicht werden, dass bei einem geringen Quotienten der Zeitkonstanten $T_s/T_t < 10$ bis zu diesem Zeitpunkt (20% der Istwertänderung) der Einschwingvorgang der Verzugszeit T_t dennoch abgeschlossen ist, und somit genügend Informationen für eine Reglerauslegung vorhanden sind.

DampingFactor

Mit Hilfe dieses Eingangs ist es möglich die automatisch "StartAutoTune" oder manuell "Preset" mit Vorgabe der Streckenparameter K_s , T_s , T_t) ermittelten Reglerparameter nachzujustieren. Über die Grenzen eines schnellen und langsamen Regelkreisverhaltens bei idealen Streckenparametern können hierdurch ebenfalls sich ändernde bzw. abweichende Parameter berücksichtigt werden. Hierzu wurde die lineare Interpolation der Reglerparameter zwischen einem oszillierenden und aperiodischen Einschwingverhalten mittels einer Extrapolation erweitert.

Hierbei entspricht ein "DampingFactor" von 100 bei idealen Streckenparametern exakt dem aperiodischen Grenzfall, und ein "DampingFactor" von 120 einem oszillierenden Einschwingen mit ca. 20% Überspringen. Entsprechend kann das Regelkreisverhalten bis zu einem "DampingFactor" von 20 verlangsamt, bzw. bis zu einem "DampingFactor" von 200 beschleunigt werden. Liegen vom realen Verhalten abweichende Streckenparameter oder Systemverhalten vor, so kann des Weiteren über diesen Eingang eine Korrektur des gewünschten dynamischen Regelkreisverhaltens erzielt werden.

StartUpTemp / StartUpTime

Erfordert ein Prozess vor der eigentlichen Temperaturregelung das Verharren an einer bestimmten Temperatur für eine gewisse Zeit (Anlauframpe), so ist diese Funktion über die Eingänge "StartUpTemp" und "StartUpTime" verfügbar.

Der Regler regelt zuerst bis zur Anlauftemperatur "StartUpTemp". Wenn diese erreicht wird, läuft die "StartUpTime" ab und der Regler hält die Anlauf-

- temperatur konstant. Nach Ablauf der "StartUpTime" ist die Anlauframpe abgeschlossen und der Regler verhält sich in gewohnter Weise. Die Anlauframpe wird mit jeder steigenden Flanke am "Enable"-Eingang einmalig ausgeführt.
- Limits** Zur Prozessüberwachung können Alarm- und Statusschranken angelegt werden.
- Die "Limits"-Struktur beinhaltet die Werte "UpperAlarm" und "LowerAlarm". Bei Verletzung der Alarmschranken wird ein Fehler generiert. Die Überwachung wird erst aktiv, wenn sich die aktuelle Prozesstemperatur im Bereich zwischen "UpperAlarm" und "LowerAlarm" befindet, um während des Prozessanlaufs keine Fehler zu erzeugen. Bei den Werten handelt es sich um absolute Werte.
- Die "Limits"-Struktur beinhaltet außerdem die Werte "UpperAtSetPoint" und "LowerAtSetPoint". Der Ausgang "SetPointReached" wird gesetzt, sobald die Bedingung $(\text{Setpoint} - \text{Limits.LowerAtSetPoint}) \leq \text{Feedback} \leq (\text{Setpoint} + \text{Limits.UpperAtSetPoint})$ erfüllt ist.
- Die Werte sind relative Werte die sich auf den "Setpoint" beziehen. Somit kann der "Setpoint" geändert werden, ohne "Limits.LowerAtSetPoint" und "Limits.UpperAtSetPoint" ändern zu müssen. Anhand dieser Statusanzeige kann man erkennen, ob die gewünschte Temperatur bereits erreicht wurde.
- Pause** Ein Anhalten der Regelung kann über den Eingang "Pause" realisiert werden. Dabei wird die Stellgröße konstant gehalten und es wird der Regler mit all seinen Werten "eingefroren". Nach dem Rücksetzen des Eingangs "Pause" läuft die Regelung weiter. Es kann dabei zu einem Sprung kommen, wenn sich zwischenzeitlich der Istwert stark verändert hat.
- SetPointGradient** Der Funktionsbaustein bietet die Möglichkeit den Anstieg des Sollwertes zu begrenzen. Dies vermindert die Überschwingweite und schont die Anlage. Dabei wird bei jeder neuen Vorgabe des Eingangs "SetPoint" der Sollwert intern rampenförmig (nicht sprungförmig) solange verändert, bis die gewünschte Sollgröße erreicht ist.
- Die Eingabe von "SetPointGradient" erfolgt in [u/s]
- ControlMode** Mit Hilfe dieses Eingangs kann zwischen einer Steuerung im offenen Regelkreis und einer Regelung (optimiert auf Führungs- oder Störverhalten) im geschlossenen Regelkreis gewählt werden.
- Die manuelle Stellgröße "ManContrValue" kann beispielsweise vorab einer Selbsteinstellung zur Steuerung der Regelgröße in den Arbeitspunkt oder bei einem Fehler in der Istwerterfassung verwendet werden.
- In der Default-Einstellung `ControlMode:= CONTROL_TEMP_MODE_ONLY_DISTURBANCE` ist eine sowohl schnelle als auch robuste Streckenidentifikation implementiert. Zu beachten ist hierbei, dass in diesem Modus (aufgrund von Optimierungen) explizit auf Störverhalten optimiert wird. Entsprechend ist kein direkter Wechsel auf einen anderen Optimierungsverfahren ohne erneutes Autotuning möglich.
- Des Weiteren wird eine fehlgeschlagene Identifikation, aufgrund einer unzureichenden Streckenreaktion, in diesem Modus durch Regler- und Streckenparameter NULL signalisiert.
- Regelkreisüberwachung** Der Funktionsbaustein verfügt über eine Gradientenüberwachung des Istwertes, der bei sehr großen Abweichungen anschlägt. Solche können z.B. vorliegen, wenn der Sensor einen Fehlercode ausgibt und dieser Fehlercode vom Anwender nicht abgefangen/ausgewertet wird. Der Gradient ist fest auf 10 [u/ms] eingestellt. Es wird eine einmalige Überschreitung toleriert, bevor ein Fehler generiert wird.

Taskintervallzeit / PWM	Die Taskintervallzeit bildet die Grundlage für die Pulsweitenmodulation (PWM), d.h. die Taskintervallzeit ist die kleinste mögliche Zeitscheibe, die bei der PWM auf "An" bzw. "Aus" gesetzt werden kann. Eine Periode der PWM dauert 20-mal länger als die Taskintervallzeit. Somit können Stellgrößen mit einer Auflösung von 0% bis 100% in 5% Schritten moduliert werden. Die Taskintervallzeit/Zykluszeit sollte in Abhängigkeit von Stellglied und Regelstrecke so eingestellt werden, dass die PWM keine sichtbare Temperaturschwankung innerhalb einer PWM-Periode ($20 \cdot \text{Taskintervallzeit}$) hervorruft. Das heißt, wenn sich der Temperaturgradient infolge der Stellgröße zwischen dem eingeschalteten und ausgeschalteten Zustands der PWM-Periode merklich ändert, ist die Taskintervallzeit zu groß gewählt. Umgekehrt sollte ebenfalls die Taskintervallzeit nicht zu klein gewählt werden, da sonst das Stellglied (z.B. Relaismodul) erhöhten Belastungen ausgesetzt ist. Des Weiteren ist bausteinintern eine Überwachung der Abtastzeit vorhanden, die bei Über-/ Unterschreiten der Abtastzeit um mehr als 20% einen Fehler erzeugt und den Ausgang "Error" auf TRUE setzt.
Selbsteinstellung	Für eine optimale Selbsteinstellung sollten nachfolgende Hinweis beachtet werden: 1. Über einen ausgedehnten Bereich sind Temperaturstrecken meist nicht-linear. Dies bedeutet, dass für eine identische Temperaturänderung bei kleinen Temperaturen eine geringere Stellsignaländerung benötigt wird als bei einer höheren Temperatur. Ein Regler benötigt demzufolge temperaturabhängige Einstellungen, weshalb die Identifikation in dem Temperaturbereich durchgeführt werden sollte, in dem der Regler später eingesetzt wird. 2. Im laufenden Betrieb kann die Selbsteinstellung erneute durch eine steigende Flanke am "StartAutoTune"-Eingang gestartet werden. Dies kann beispielsweise bei einem Füllstandswechsel oder anderweitigen Veränderungen an der Regelstrecke sinnvoll sein. Für eine optimale Identifikation sollte hierbei jedoch eine ausreichende Systemanregung (Istwertänderung) gewährleistet werden. 3. Eine Selbsteinstellung kann nur ausgeführt werden, wenn bei einer Heizung der Sollwert höher als die Ist-Temperatur ist, und bei einer Kühlung die Ist-Temperatur höher als der Sollwert ist.
Spezielle Einflussgrößen der Selbsteinstellung	In Sonderfällen sind nachfolgende Maßnahmen für eine weitere Verbesserung des "AutoTuning" möglich: 1. Mit dem Eingang "ExtendedSettings.IdentAccuracy" können die Abbruchkriterien beeinflusst werden. Abgebrochen wird hierbei entweder bei Erreichen dieser Änderungsrate oder bei der doppelten Anregelzeit mit dieser Schranke, je nachdem welches Kriterium zuerst erfüllt ist. 2. Mit dem Eingang "ExtendedSettings.OblivionFactor" können veränderliche Streckenparameter beim "AutoTuning" berücksichtigt werden. Eine Erniedrigung dieses Eingangs ist bei zeitlich variierenden, und eine Erhöhung bei zeitlich festen Streckenparametern zweckmäßig. 3. Mit dem Eingang "ExtendedSettings.MinDelayTime" wird eine minimale Verzugszeit T_1 als a priori Information vorgegeben. Dieser Wert wird für das "AutoTuning" herangezogen, falls an dem Eingang "ContrParam.Tt" kein Wert vorgegeben ist. Dieser Wert entspricht hierbei der minimal möglichen Streckenverzugszeit, welche nicht kleiner als die Summe sämtlicher sekundärer Transport-, Verzugs- und Filterzeitkonstanten sein kann.

Fehlerbehandlung Der Funktionsbaustein erzeugt in Additional1/Additional2 für die Tabelle F_RELATED_TABLE, 16#0170, folgende Fehlermeldungen:

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#00001600	Aktueller State	Fehler in der Zustandsmaschine, ungültiger Zustand
INPUT_RANGE_ERROR	16#00001601	16#00001601	Eingabewerte für maximalen Ausgabewert ("OutputPowerMax") außerhalb des gültigen Bereichs. Wert muss größer 0[%] bis max. 100[%] sein
INPUT_RANGE_ERROR	16#00001601	16#00001602	Eingabewerte für Alarmschranken ungültig. (Bedingung: Limits.UpperAlarm > Limits.LowerAlarm)
INPUT_RANGE_ERROR	16#00001601	16#00001603	Eingabewert für manuelle Stellgröße ("ManContrValue") ungültig. Wert muss im Bereich von 0[%] und 100[%] liegen
INPUT_RANGE_ERROR	16#00001601	16#00000004	Reglerparameter ungültig. Es liegen entweder keine Reglerparameter vor, oder die Reglerparameter sind ungültig
INPUT_RANGE_ERROR	16#00001601	16#00000005	Streckenzeitkonstante ungültig. Die Zeitkonstante ModelParam.Ts oder ContrParam.Tt ist ungültig
INPUT_RANGE_ERROR	16#00001601	16#00000006	Eingabewert "OblivionFactor" ist ungültig. Wert muss im Bereich von 0.9 und 1 liegen
INPUT_RANGE_ERROR	16#00001601	16#00000007	Eingabewert "MinDelayTime" ist ungültig. Wert muss größer T#0s sein
INPUT_RANGE_ERROR	16#00001601	16#00000008	Eingabewert "IdentAccuracy" ist ungültig. Wert muss größer 0 [%/s] sein
INPUT_RANGE_ERROR	16#00001601	16#00000009	Eingabewert "DampingFactor" ist ungültig. Wert muss im Bereich von 20 und 200 liegen
INPUT_RANGE_ERROR	16#00001601	16#0000000B	Eingabewert "SetPointGradient" ist ungültig. Wert muss größer gleich 0[u/s] sein
INPUT_RANGE_ERROR	16#00001601	16#0000000C	Eingabewerte für Statusschranken ungültig (Bedingung: (Setpoint + Limits.UpperAtSetpoint) > (Setpoint - Limits.LowerAtSetpoint))
INPUT_RANGE_ERROR	16#00001601	16#00000010	Eingabewert "ContrMode" ist ungültig
INPUT_RANGE_ERROR	16#00001601	16#00000011	Eingabewert "AutoTuneMaxPower" ist ungültig. Wert muss im Bereich von 0[%] bis 100[%] liegen
INPUT_RANGE_ERROR	16#00001601	16#00000012	Eingabewert "ExtendedSettings.ControlType" ist ungültig.
INPUT_RANGE_ERROR	16#00001601	16#00000013	Eingabewert "ExtendedSettings.bControl" ist ungültig. Wert muss im Bereich von 0[-] bis 1[-] liegen

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#00001601	16#00000014	Eingabewert "ExtendedSettings.cControl" ist ungültig. Wert muss im Bereich von 0[-] bis 1[-] liegen
SYSTEM_ERROR	16#00001602	Abtastzeit	Jittern der Abtastzeit > 20[%]. Fehler wird erst bei 2 aufeinander folgenden Überschreitungen aktiv
SYSTEM_ERROR	16#00001602	16#00000005	Gradient des Istwertes zu groß, möglicherweise Sensorfehler
CALCULATION_ERROR	16#00001603	16#00000007	Identifizierte Streckenparameter ungültig, Streckenselbstidentifikation fehlgeschlagen
DEVICE_ERROR	16#00001606	Temperatur	Temperaturfehlerschranke überschritten, Temperatur außerhalb der Fehlerschranken

Tab. 7-168: Fehlercodes Funktionsbaustein IL_TempControlType02

Hinweise zu bestimmten Fehlern

- **"Reglerparameter ungültig"**

(Additional1 = 16#1601; Additional2 = 16#0004):

Ursache: Es sind entweder keine Reglerparameter vorhanden, oder die in der Input-Output-Struktur "ContrParam" angelegten Reglerparameter sind fehlerhaft. Damit der Temperaturregler in der Betriebsart `CONTROL_TEMPMODE_OPT_COMMAND` oder `CONTROL_TEMPMODE_OPT_DISTURBANCE` ein sinnvolles Stellsignal erzeugen kann, ist je nach Regelsinn (positiv = Heizen; negativ = Kühlen) mindestens ein proportionaler Regler mit einem Kp ungleich Null nötig.

Fehlerbehebung: Der Temperaturregler kann entweder die Strecken- und Reglerparameter selbst ermitteln, es können die Reglerparameter vorgegeben werden, oder der Regler kann sich die Reglerparameter über vorgegebene Streckenparameter selbst ermitteln. Für die Selbstauslegung ohne vorgegebene Reglerparameter ist es nötig, mit einer steigenden Flanke an "StartAutoTune" die Selbsteinstellung zu starten. Für die Vorgabe von Reglerparametern ist es nötig, in der Input-Output-Struktur "ContrParam" die gewünschten Reglerparameter aufzulegen und mit einer steigenden Flanke am "Preset"-Eingang zu übernehmen. Für die Vorgabe von Streckenparametern ist es nötig, in den Input-Output-Strukturen "ModelParam" und "ContrParam" den proportionalen Reglerparameter ("ContrParam.KP") auf NULL zu setzen, die Steckparameter ("ModelParam.Ks", "ModelParam.Ts" und "ContrParam.Tt") aufzulegen und mit einer steigenden Flanke am "Preset"-Eingang die Reglerberechnung zu starten

- **"Identifizierte Streckenparameter ungültig, Streckenselbstidentifikation fehlgeschlagen"**

(Additional1 = 16#1603; Additional2 = 16#0007):

Die Identifikation des Regelstreckenverhaltens sowie die darauf aufbauende Reglerparametrierung ist fehlgeschlagen.

1. "ControlMode:= `CONTROL_TEMPMODE_ONLY_DISTURBANCE`":

Ursache: Der Eingang "AutoTuneMaxPower" wurde, entsprechend der präsenten Regelstrecke, zu groß gewählt. Infolge dessen konnte das markante Streckenverhalten innerhalb der ersten Hälfte der Istwertänderung nicht identifiziert werden.

Fehlerbehebung:Reduktion des Eingangs "AutoTuneMaxPower".

2. "ControlMode:= CONTROL_TEMPMode_OPT_xxxxx":

Ursache: Das Fehlschlagen der Identifikation kann an einer ungünstig vorgegebenen Streckenverzugszeit "ContrParam.Tt", einer zu großen Taskintervallzeit, einer zum Zeitpunkt der Identifikation nicht ausreichend eingeschwungenen Regelstrecke, einer zu geringen Streckenanregung oder einer fehlerhaften Istwerterfassung (Sensor nicht angeschlossen) liegen.

Fehlerbehebung:Falls die Selbsteinstellung aufgrund einer falschen Abtastzeit fehlgeschlagen ist, kann eine grobe Abschätzung der Streckenverzugszeit als Information für die Selbsteinstellung an dem Eingang "ContrParam.Tt" vorgegeben und/oder die Taskintervallzeit verkleinert werden. Falls die Selbsteinstellung des Temperaturreglers aufgrund einer nicht ausreichend eingeschwungenen Strecke fehlgeschlagen ist, sollte die Selbsteinstellung nochmals mit einer stationär eingeschwungenen Strecke wiederholt werden. Falls die Selbsteinstellung aufgrund einer zu geringen Streckenanregung fehlgeschlagen ist, sollte die Selbsteinstellung mit einer größeren Sollwertänderung wiederholt werden. Für den Fall eines Defektes in der Temperaturerfassung muss dieser behoben werden

7.9.4 IL_TEMPCONTROL_CONTR_PARAM

Die Struktur IL_TEMPCONTROL_CONTR_PARAM enthält die aktuellen Parameter K_P , K_I und K_D des Reglers, sowie die Totzeit T_t .

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme	Beschreibung
KP	REAL	N. Def	N. Def	10000	Bei steigender Flanke an "Preset" oder "StartAutoTune"	Proportionalverstärkung des Reglers
KI	REAL	N. Def	N. Def	0	Bei steigender Flanke an "Preset"	Integrationskonstante des Reglers
KD	REAL	N. Def	N. Def	0	Bei steigender Flanke an "Preset"	Differentiationskonstante des Reglers
Tt	TIME	T#0s	N. Def	T#0s	Bei steigender Flanke an "Preset", "StartAutoTune" oder Änderung der Reglereinstellung ("Damping-Factor", "ControlMode")	Verzugszeit der Strecke

Tab. 7-169: Strukturbeschreibung IL_TEMPCONTROL_CONTR_PARAM

7.9.5 IL_TEMPCONTROL_CONTROL_MODE

Der Aufzählungstyp IL_TEMPCONTROL_CONTROL_MODE dient der Auswahl des Regelkreisverhaltens:

Steuerung im offenen Regelkreis oder Regelung optimiert auf Führungs- oder Störverhalten im geschlossenen Regelkreis

Aufzählungselement	Nummer	Beschreibung
CONTROL_TEMPMODE_MANUAL	1	Baustein steuert den Streckenistwert mit dem am Eingang "ManContrValue" vorgegebenen Wert
CONTROL_TEMPMODE_OPT_COMMAND	2	Baustein regelt optimiert auf Führungsverhalten den "Feedback"-Wert auf den "Setpoint"-Wert
CONTROL_TEMPMODE_OPT_DISTURBANCE	3	Baustein regelt optimiert auf Störverhalten den "Feedback"-Wert auf den "Setpoint"-Wert
CONTROL_TEMPMODE_ONLY_DISTURBANCE	4	Baustein regelt optimiert auf Störverhalten den "Feedback"-Wert auf den "Setpoint"-Wert. Schnelles und zugleich robustes Autotuning explizit für Störverhalten. (Default)

Tab. 7-170: Aufzählungstyp IL_TEMPCONTROL_CONTROL_MODE

7.9.6 IL_TEMPCONTROL_CONTROL_TYPE

Der Aufzählungstyp IL_TEMPCONTROL_CONTROL_TYPE dient zur Auswahl eines spezifischen Reglertyps (PI oder PID), welcher als Basis für die interne Reglerparametrierung ("Autotune" oder "Preset") anhand der Streckenparameter dient.

Unabhängig dieser "Autotuning"-Einstellung, kann über einen "Preset" von entsprechenden Reglerparametern der interne PID-Regler selbst als P-, PI-, PD- oder PID-Regler parametrierbar werden.

Aufzählungselement	Nummer	Beschreibung
CONTROL_TYPE_PI	1	Parametrierung eines PI-Reglers anhand der Streckenparameter
CONTROL_TYPE_PID	2	Parametrierung eines PID-Reglers anhand der Streckenparameter

Tab. 7-171: Aufzählungstyp IL_TEMPCONTROL_CONTROL_TYPE

7.9.7 IL_TEMPCONTROL_EXTENDED_SETTINGS

Die Struktur IL_TEMPCONTROL_EXTENDED_SETTINGS beinhaltet die erweiterten Einstellungsmöglichkeiten der Selbsteinstellung.

Die Standardeinstellungen sind für übliche Strecken ausreichend.

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme	Beschreibung
IdentAccuracy	REAL	>0	N. Def	3.0	Zyklisch	Abbruchkriterium der Identifikation
OblivionFactor	REAL	0.9	1.0	0.98	Zyklisch	Vergessenheitsfaktor für veränderliche Strecken
MinDelayTime	TIME	>T#0s	N. Def	T#1s	Bei steigender Flanke von "Start-AutoTune"	Minimale Verzugszeit der Strecke
ControlType	IL_TEMP- CONTROL_ CONTROL_ TYPE, Seite 507	CONTROL_ TYPE_PI	CONTROL_ TYPE_PID	CONTROL_ TY- PE_PI	Zyklisch	Reglertyp (PI- oder PID-Regler)
bControl	REAL	0	1.0	1.0	Zyklisch	Gewichtungsfaktor für P-Anteil
cControl	REAL	0	1.0	1.0	Zyklisch	Gewichtungsfaktor für D-Anteil

Tab. 7-172: Strukturbeschreibung IL_TEMPCONTROL_EXTENDED_SETTINGS

7.9.8 IL_TEMPCONTROL_LIMITS

Die Struktur IL_TEMPCONTROL_LIMITS enthält die Alarmschranken in [u] (Einheiten). Initialwert ist jeweils 0u (nicht aktiv).

Wird die Struktur mit Werten belegt, müssen

`Limits.UpperAlarm > Limits.LowerAlarm`

und

`(Setpoint + Limits.UpperAtSetpoint) >= (Setpoint - Limits.LowerAtSetpoint)`

sein, ansonsten wird ein Fehler erzeugt.

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme	Beschreibung
UpperAlarm	REAL	N. Def	N. Def	0	Zyklisch	Obere Istwertgrenze, ab der ein Alarm ausgelöst wird (absolut)
LowerAlarm	REAL	N. Def	N. Def	0	Zyklisch	Untere Istwertgrenze, ab der ein Alarm ausgelöst wird (absolut)
UpperAtSetpoint	REAL	N. Def	N. Def	0	Zyklisch	Obere Istwertgrenze für SetpointReached (relativ)
LowerAtSetpoint	REAL	N. Def	N. Def	0	Zyklisch	Untere Istwertgrenze für SetpointReached (relativ)

Tab. 7-173: Strukturbeschreibung IL_TEMPCONTROL_LIMITS

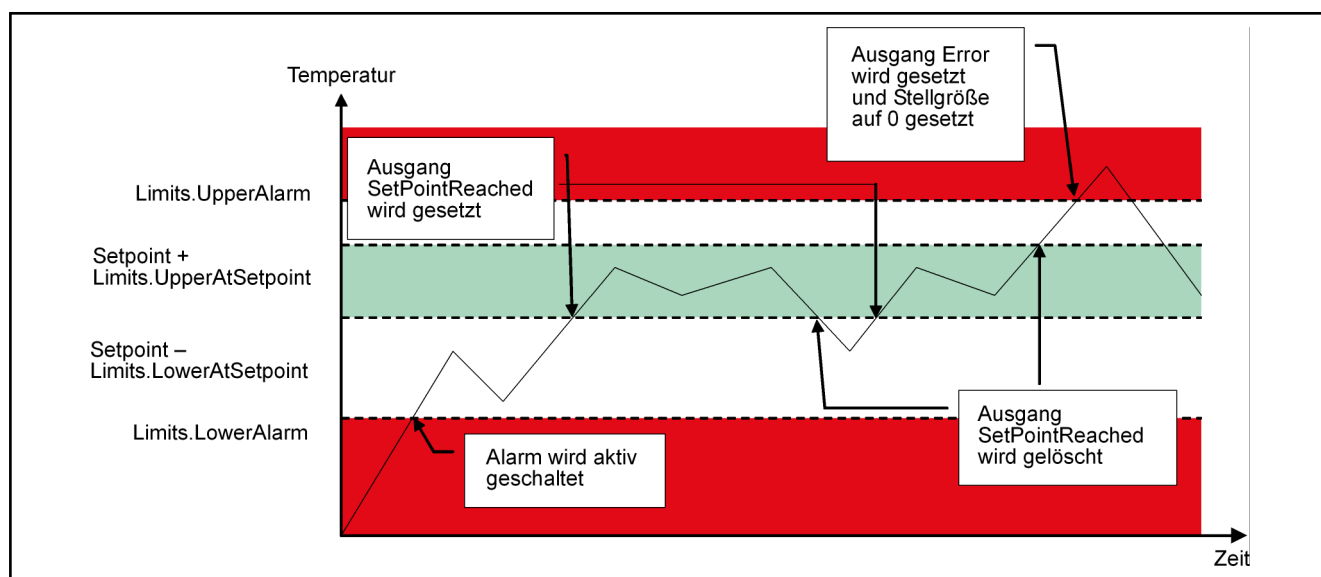


Abb. 7-111: Wirkungsweise der Alarm- und Statusschranken

7.9.9 IL_TEMPCONTROL_MODEL_PARAM

Die Struktur IL_TEMPCONTROL_MODEL_PARAM enthält die Parameter K_s und T_s der Regelstrecke.

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme	Beschreibung
K_s	REAL	N. Def	N. Def	0	Bei steigender Flanke an "Preset", "StartAutoTune" oder Änderung der Reglereinstellung ("DampingFactor", "ControlMode")	Verstärkung der Regelstrecke
T_s	TIME	T#0s	N. Def	T#0s	Steigende Flanke an "StartAutoTune", "Preset" oder Änderung der Reglereinstellung ("DampingFactor", "ControlMode")	Ausgleichszeitkonstante der Regelstrecke

Tab. 7-174: Strukturbeschreibung IL_TEMPCONTROL_MODEL_PARAM

7.10 Zweipunktregler

7.10.1 IL_TwoPosCtrlType01

Kurzbeschreibung Der Funktionsbaustein IL_TwoPosCtrlType01 realisiert einen Zweipunktregler mit Hysterese und ist für einfache Regelungen geeignet.



Der Funktionsbaustein "ML_TwoPosCtrlType01 (ML_TechBase-Bibliothek)" wurde mit Version 14VRS umbenannt und in die "RIL_LoopControl-Bibliothek" verlegt.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek
MLC 15VRS	RIL_LoopControl

Tab. 7-175: Referenztablette Funktionsbaustein IL_TwoPosCtrlType01

Schnittstellenbeschreibung

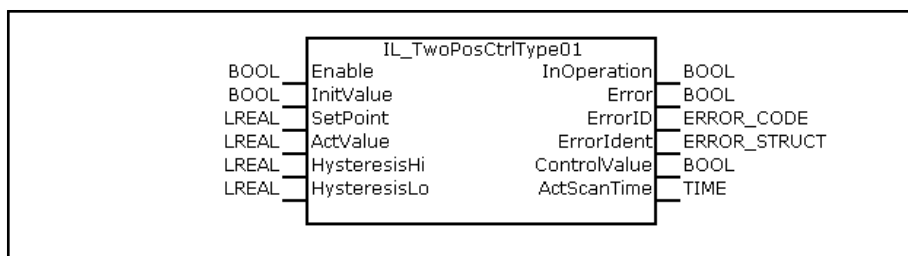


Abb. 7-112: Funktionsbaustein IL_TwoPosCtrlType01

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (zyklisch, zustandsgesteuert)
	InitValue	BOOL	Vorgabewert für Stellgröße y
	SetPoint	LREAL	Führungsgröße w (Sollwert)
	ActValue	LREAL	Regelgröße x (Istwert)
	HysteresisHi	LREAL	Obere Ansprechschwelle der Hysterese
	HysteresisLo	LREAL	Untere Ansprechschwelle der Hysterese
VAR_OUTPUT	InOperation	BOOL	Berechnung der Stellgröße y ohne Fehler abgeschlossen, Ausgangsvariable "ControlValue" gültig
	Error	BOOL	Berechnung der Stellgröße "ControlValue" mit Fehler abgeschlossen, Ausgangsvariable "ErrorIdent" gültig
	ErrorID	ERROR_CODE	Bei gesetztem "Error"-Ausgang enthält dieser Ausgang eine grobe Klassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Beschreibung der Diagnose im Fehlerfall Fehlerbehandlung, Seite 511
	ControlValue	BOOL	Reglerausgabewert - Stellgröße y
	ActScanTime	TIME	Reglerzykluszeit [ms]

Tab. 7-176: Schnittstellenvariablen Funktionsbaustein IL_TwoPosCtrlType01

Spezifikation

Der Funktionsbaustein IL_TwoPosCtrlType01 kann über folgende Eingangsvariablen in seinen Eigenschaften angepasst werden:

"HysteresisHi":

- Der Eingang muss mit einem positiven Wert bzw. Null vorbelegt werden, der Wert Null schaltet die positive Hysterese aus.

"HysteresisLo":

- Der Eingang muss mit einem negativen Wert bzw. Null vorbelegt werden, der Wert Null schaltet die negative Hysterese aus.



Bei der Berechnung der Stellgröße "ControlValue" ist "FALSE" dominant, d.h. bei "HysteresisLo" == "HysteresisHi" == Null ist bei einer Regeldifferenz von Null ("SetPoint" == "ActualValue") der Ausgang "ControlValue" stets "FALSE".

Funktionsbeschreibung

Der Funktionsbaustein IL_TwoPosCtrlType01 belegt nach der Bearbeitungsfreigabe des Funktionsbausteins mittels "Enable" im ersten Reglerzyklus den internen Reglerzustand sowie den Reglerausgang auf den am Eingang "InitValue" anliegenden Presetwert vor und berechnet in den Folgezyklen zyklisch die Stellgröße "ControlValue".

Im Fall der fehlerfreien Bearbeitung des Funktionsbausteins wird die Gültigkeit der Stellgröße durch den Ausgang "InOperation" signalisiert, die Elemente der Ausgangsstruktur "ErrorID" und "ErrorIdent" werden in diesem Fall nicht aktualisiert.

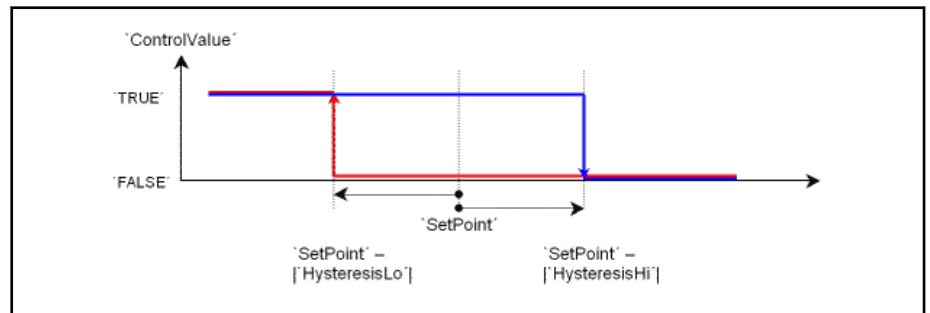


Abb. 7-113: Hysterese Funktionsbaustein IL_TwoPosCtrlType01

Tritt während der Bearbeitung des Funktionsbausteins ein Fehler auf, so wird dieser durch den Ausgang "Error" signalisiert, sowie durch die in diesem Fall aktualisierten Elemente der Ausgangsstruktur "ErrorID" und "ErrorIdent" spezifiziert. Die Ausgänge "ControlValue" und "ActScanTime" werden im Fehlerfall nicht aktualisiert.

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle **MLC_TABLE**, 16#0030. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR (16#0006)	16#F0260030	0	Eingänge "HysteresisHi" oder "HysteresisLo" sind ungültig
STATE_MACHINE_ERROR (16#0005)	16#F0260031	xx	Ungültiger Zustand der Zustandsmaschine xx gibt den ungültigen Zustand an

Tab. 7-177: Fehlercodes Funktionsbaustein IL_TwoPosCtrlType01

7.11 Utilities

7.11.1 IL_COORD_TYPE01

Diese Datenstruktur definiert eine zweidimensionale Koordinate mit X und Y REAL-Werten.

Programm:

```

TYPE IL_COORD_TYPE01 :
STRUCT
  X: REAL;    // X coord
  Y: REAL;    // Y coord
END_STRUCT
END_TYPE

```

Anwendung in

- [IL_DeviationResponseCurve](#), Seite 375,
- [IL_PType01](#), Seite 360,
- [IL_PType02](#), Seite 364,

7.11.2 IL_SUPPRESSION_LIMITS

IL_SUPPRESSION_LIMITS ist ein Aufzählungstyp und aktiviert die Unterdrückungsfenster mit folgender Belegung:

Element	Beschreibung
IL_SUPPRESSION_OFF	Unterdrückung ist deaktiviert, alle Werte kommen unverändert an den Ausgang.
IL_SUPPRESSION_LIMIT_1	Unterdrückungsfenster #1 aktiv ("SuppressionLimit1Low" und "SuppressionLimit1High")
IL_SUPPRESSION_LIMIT_2	Unterdrückungsfenster #2 aktiv ("SuppressionLimit2Low" und "SuppressionLimit2High")

Tab. 7-178: Aufbau des Aufzählungstyps IL_SUPPRESSION_LIMITS

Anwendung in

- [IL_SuppressionType03, Seite 465](#)
- [IL_SuppressionType04, Seite 470](#)

7.11.3 IL_SUPPRESSION_MODE

IL_SUPPRESSION_MODE ist ein Aufzählungstyp und beschreibt die Funktion der Eingänge "SuppressionLimitLow*" und "SuppressionLimitHigh*" mit folgender Belegung:

Element	Beschreibung
IL_SUPPRESSION_PERCENTAGE	Werte sind prozentual zum gemittelten Eingangswert
IL_SUPPRESSION_ABSOLUTE_VALUE	Werte sind absolute Werte zum gemittelten Eingangswert
IL_SUPPRESSION_STATISTICAL	Werte sind statistisch zur Standardabweichung σ des Eingangswertes (z.B. $0.5 \cdot \sigma$, $1.0 \cdot \sigma$, $4.0 \cdot \sigma$,...)

Tab. 7-179: Aufbau des Aufzählungstyps IL_SUPPRESSION_MODE

Anwendung in

- [IL_SuppressionType01, Seite 455](#)
- [IL_SuppressionType02, Seite 460](#)
- [IL_SuppressionType03, Seite 465](#)
- [IL_SuppressionType04, Seite 470](#)

8 RIL_HMI_Uilities.library

8.1 Einführung und Übersicht



Für die Bibliothek ist das Basis-Funktionspaket PLC des Systems MLC 15VRS ausreichend.

Anwendungsgebiete und Funktionen / Funktionsbausteine von RIL_HMI_Uilities

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine	Beschreibung
POUs (POUs)	
IL_MKeys, Seite 513	unterstützt die Bedienung von Maschinenfunktionstasten am Visualisierungsgerät
IL_UdpKeys, Seite 514	ermöglicht dem Anwender Tastendrucke von einem HMI-Gerät sicher an die Steuerung zu übertragen

Tab. 8-1: RIL_HMI_Uilities-Bibliothek

8.2 IL_MKeys

Kurzbeschreibung

Der Funktionsbaustein IL_MKeys unterstützt die Bedienung von Maschinenfunktionstasten am Visualisierungsgerät zur Ausführung von Maschinenfunktionen durch die SPS.

Wenn die Maschinenfunktionstasten als SPS-Eingänge verfügbar sind, müssen sie auf die Eingänge R1-R8 und L1-L8 verschaltet werden.

Zielsystem	Bibliothek / Ordner
MLC 15VRS	RIL_HMI_Uilities / POU's
Basis-Funktionspaket PLC	

Tab. 8-2: Bibliothekszuordnung

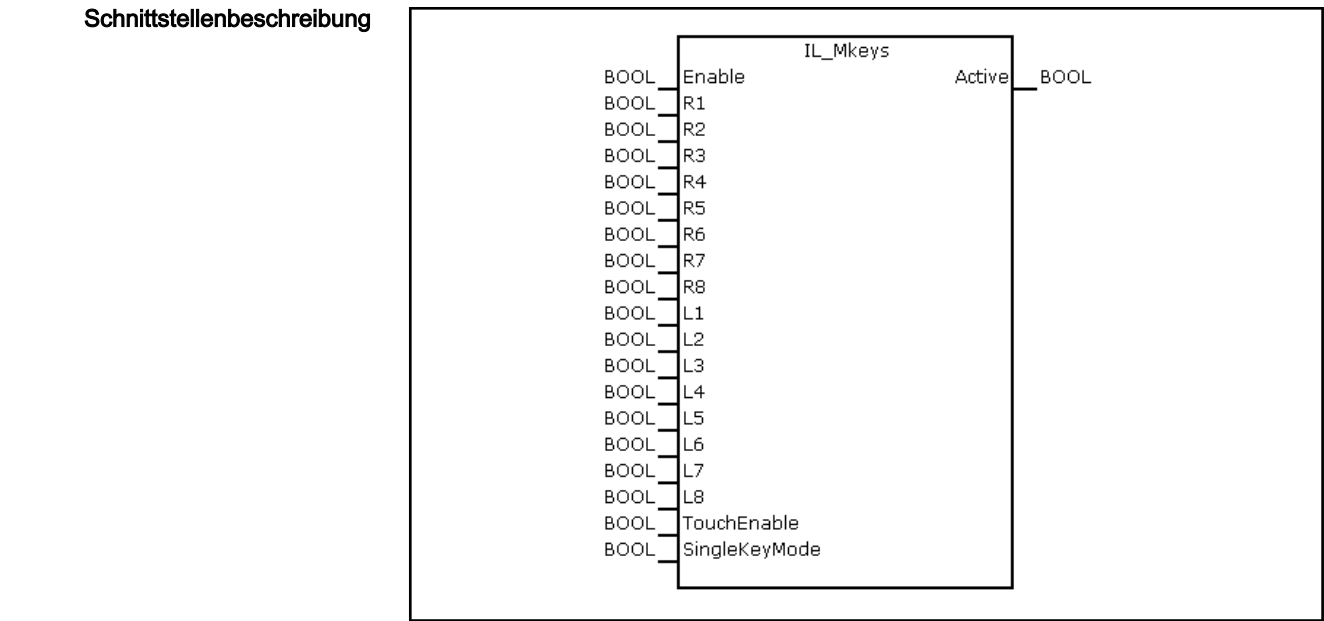


Abb. 8-1: Funktionsbaustein IL_MKeys

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Aktiviert die Funktion des Bausteins.
	R1-R8	BOOL	Hardwaretasten <R1> bis <R8>.
	L1-L8	BOOL	Hardwaretasten <L1> bis <L8>.
	TouchEnable	BOOL	Ermöglicht die Bedienung mittels Touchscreen bzw. Mausklick auf die Tasten am Visualisierungsgerät.
	SingleKeyMode	BOOL	Betriebsart, die garantiert, dass nur eine Taste zu jedem Zeitpunkt gedrückt werden kann
VAR_OUTPUT	Active	BOOL	Ist TRUE, wenn eine M-Tastenbelegung aktiv ist.

Tab. 8-3: Funktionsbaustein IL_MKeys

 **VORSICHT**

Gefahr bei sicherheitskritischen Maschinenfunktionen!

Da bei Touchbedienung das Loslassen der Tasten nicht mit absoluter Sicherheit in der SPS erkannt werden kann, muss bei sicherheitskritischen Maschinenfunktionen der Eingang "TouchEnable" mit einem Hardware-Zustimmungstaster belegt werden.



Aus Kompatibilitätsgründen enthält die Bibliothek RIL_HMI_Uilities.library auch die in IndraLogic 1x verfügbaren Funktionsbausteine MKEYS und HMICE_MKEYS.

Bei Neuimplementierungen in IndraLogic 2G wird jedoch empfohlen, diese Bausteine nicht mehr zu verwenden.

8.3

IL_UdpKeys

Kurzbeschreibung

Der Funktionsbaustein IL_UdpKeys ermöglicht dem Anwender Tastendrucke von einem HMI-Gerät sicher an die Steuerung zu übertragen.

Die Verbindung erfolgt über UDP, d.h. dieser Baustein funktioniert nur, wenn der PC bzw. Embedded PC das UDP-Protokoll unterstützen (z.B. VEH30.2).

Jedes Mal, wenn der Funktionsbaustein aufgerufen wird, werden die letzten Daten kopiert.

Im Fall einer Verbindungsunterbrechung werden alle Ausgänge auf 0 zurückgesetzt. Das ermöglicht dem Anwender das Programmieren von sicheren Eigenschaften, wobei der Sicherheitsmode der 0 in den Ausgangsdaten des Funktionsbausteins zugeordnet ist.

Zielsystem	Bibliothek / Ordner
MLC 15VRS Basis-Funktionspaket PLC	RIL_HMI_Utilitys / POU's

Tab. 8-4: Bibliothekszuordnung

Schnittstellenbeschreibung

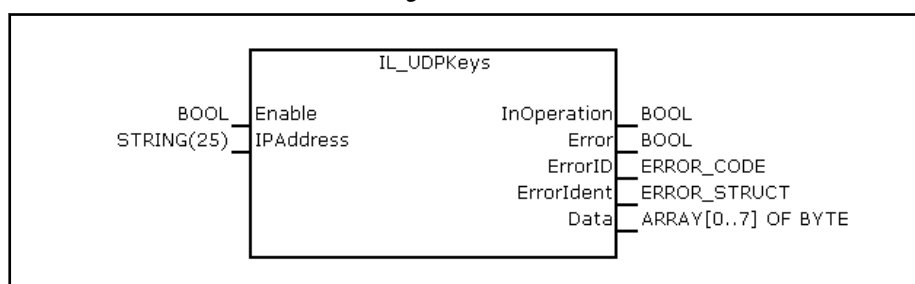


Abb. 8-2: Funktionsbaustein IL_UDPKeys

	Name	Typ	Kommentar
VAR_INPUT	IPAddress	STRING(25)	Eingegebene IP-Adresse ".-Notation, z. B. "10.104.73.193"
	Enable	BOOL	Solange der Eingang TRUE ist, werden die Daten ständig aktualisiert. Wenn inaktiv, dann sind die Ausgangsdaten alle 0.
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, dann sind die Ausgangsdaten gültig und werden aktualisiert.
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb des Funktionsbausteins aufgetreten ist. Die Ausgangsdaten werden alle auf 0 gesetzt.
	ErrorID	ERROR_CODE	Fehlerkennung (siehe Fehlermeldungen , Seite 516).
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur mit weiterer Unterteilung der Fehler.
	Data	ARRAY [0..7] OF BYTE	Enthält Daten als Byte-Array. Die Bedeutung der jeweiligen Bits ist in der Dokumentation des HMI beschrieben.

Tab. 8-5: Funktionsbaustein IL_UDPKeys

Installationshinweis

Um den Funktionsbaustein nutzen zu können, muss das UDP-Protokoll auf dem HMI-Gerät aktiviert werden.

Details dazu entnehmen Sie bitte der Projektierungsanleitung des jeweiligen Handbediengerätes.

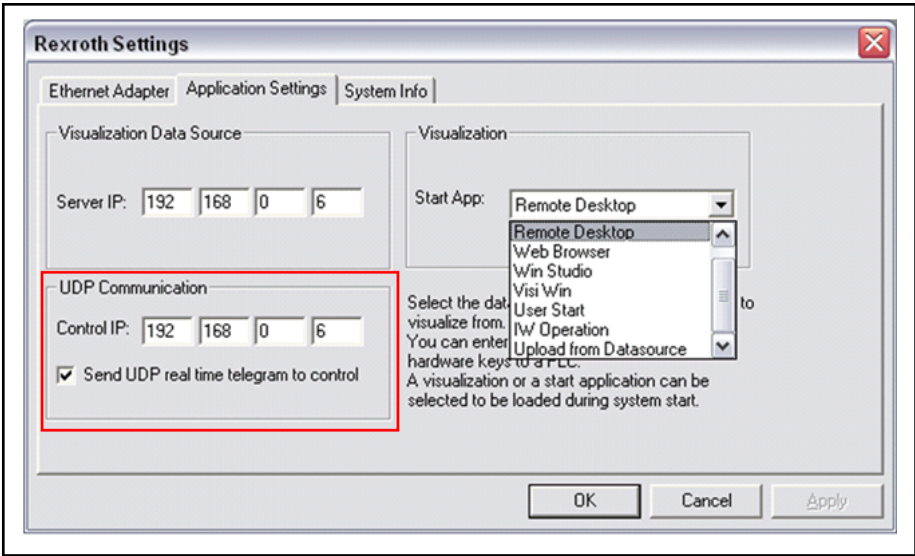


Abb. 8-3: Rexroth-Settings, Einstellung UDP-Communication zur Übertragung des Tastaturcodes

Fehlermeldungen Der Funktionsbaustein erzeugt in Additional1 und Additional2 für die Tabelle **ETHERNET_TABLE, 16#0150** folgende Fehlermeldungen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	10	0	Falsche IP-Syntax
COMMUNICATION_ERROR (16#0002)	11	0	Unbekannte IP-Adresse oder keine Verbindung
DEVICE_ERROR (16#0008)	12	0	Ungültige interne Puffergröße
STATE_MACHINE_ERROR (16#0005)	1	0	Ungültiger Zustand der Zustandsmaschine

Tab. 8-6: Generierte Fehlernummern des Funktionsbausteins IL_UdpKeys

 Aus Kompatibilitätsgründen enthält die Bibliothek RIL_HMI_Uilities.library auch den in IndraLogic 1x verfügbaren Funktionsbaustein IL_VExKeys.

Bei Neuimplementierungen in IndraLogic 2G wird jedoch empfohlen, diesen Baustein nicht mehr zu verwenden.

9 IL_VCP_DP.library

9.1 Einführung und Übersicht



Für die Bibliothek ist das Basis-Funktionspaket PLC des Systems MLC 15VRS ausreichend.

Mit dieser Bibliothek werden Rexroth-VCP-Kleinbedienterminals mit Profibus-Anschluss an die SPS gekoppelt. Dazu existieren drei Funktionsbausteine, die sich alternativ verwenden lassen:

- [VCP_PBS16_A4096](#) , Seite 517
- [VCP_PBS32_A4096](#) , Seite 518
- [VCP_PBS32_A65536](#) , Seite 519

9.2 VCP_PBS16_A4096

Kurzbeschreibung

Dieser Funktionsbaustein (FB) aktiviert das Profibus DP-Protokoll für die Kleinbedienterminals VCPxx. Zudem wird das E/A-Abbild der physikalischen Adressen zwischen SPS und Bedienterminal übertragen.

Die Datenbreite der Datenübertragung beträgt 16 Byte. Die Größe des über ein ARRAY zur Verfügung stehenden Adressbereichs beträgt 4096 Byte (Ein- und Ausgänge in Summe).

Schnittstellenbeschreibung

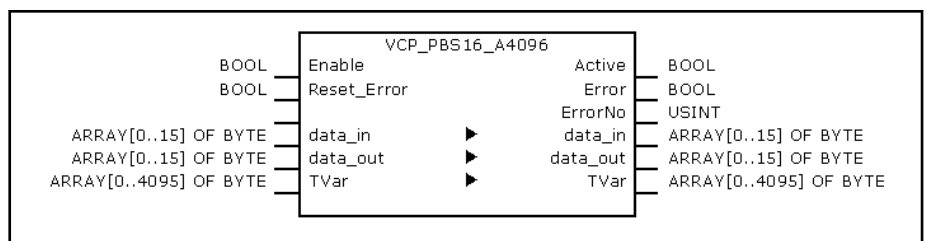


Abb. 9-1: VCP_PBS16_A4096

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	TRUE: FB wird bearbeitet FALSE: FB wird nicht bearbeitet
	Reset_Error	BOOL	TRUE: Rücksetzen von "Error" (auf FALSE) und "ErrorNo" wird auf 0 gesetzt
VAR_OUTPUT	Active	BOOL	TRUE, solange "Enable" ebenfalls TRUE ist
	Error	BOOL	TRUE, sobald ein Fehler auftritt. Lässt sich mit "Reset-Error" zurück setzen.
	ErrorNo	USINT	Fehlerart: 2: InputRangeError 4: CalculationError

	Name	Typ	Kommentar
VAR_IN_OUT	data_in	ARRAY [0..15] OF BYTE	Daten zur Ankopplung der physikalischen Eingänge des Kleinbedienterminals
	data_out	ARRAY [0..15] OF BYTE	Daten zur Ankopplung der physikalischen Ausgänge des Kleinbedienterminals
	TVar	ARRAY [0..4095] OF BYTE	Array, das zum Lesen vom und zum Schreiben auf das Kleinbedienterminal verwendet wird.

Tab. 9-1: Schnittstelle von VCP_PBS16_A4096

VI-Composer Bei der Konfiguration mit dem Rexroth VI-Composer beziehen sich die Adressen der Variablenliste auf das jeweilige Byte im Array "TVar", das im SPS-Programm zum Datenaustausch benutzt wird.

Fehlerbehandlung Sobald ein Fehler auftritt, wird die Kommunikation unterbrochen und die Anzeige des Kleinbedienterminals zeigt COMMUNICATION ERROR, ERROR CODE 110.

Die Fehlerart (ErrorNo) zeigt an, ob es sich um einen Adressberechnungsfehler (CalculationError) handelt, oder ob eine zu große Datenbreite für Data_in oder Data_out gewählt wurde (InputRangeError).

9.3 VCP_PBS32_A4096

Kurzbeschreibung Dieser Funktionsbaustein (FB) aktiviert das Profibus DP-Protokoll für die Kleinbedienterminals VCPxx. Zudem wird das E/A-Abbild der physikalischen Adressen zwischen SPS und Bedienterminal übertragen.

Die Datenbreite der Datenübertragung beträgt 32 Byte. Die Größe des über ein ARRAY zur Verfügung stehenden Adressbereichs beträgt 4096 Byte (Ein- und Ausgänge in Summe).

Schnittstellenbeschreibung

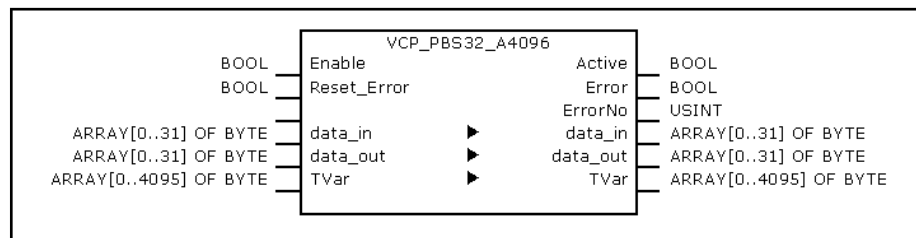


Abb. 9-2: VCP_PBS32_A4096

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	TRUE: FB wird bearbeitet FALSE: FB wird nicht bearbeitet
	Reset_Error	BOOL	TRUE: Rücksetzen von "Error" (auf FALSE) und "ErrorNo" wird auf 0 gesetzt
VAR_OUTPUT	Active	BOOL	TRUE, solange "Enable" ebenfalls TRUE ist
	Error	BOOL	TRUE, sobald ein Fehler auftritt. Lässt sich mit "Reset-Error" zurück setzen.
	ErrorNo	USINT	Fehlerart: 2: InputRangeError 4: CalculationError

	Name	Typ	Kommentar
VAR_IN_OUT	Data_in	ARRAY [0..31] OF BYTE	Daten zur Ankopplung der physikalischen Eingänge des Kleinbedienterminals
	Data_out	ARRAY [0..31] OF BYTE	Daten zur Ankopplung der physikalischen Ausgänge des Kleinbedienterminals
	TVar	ARRAY [0..4095] OF BYTE	Array, das zum Lesen vom und zum Schreiben auf das Kleinbedienterminal verwendet wird.

Tab. 9-2: Schnittstelle von VCP_PBS32_A4096

VI-Composer Bei der Konfiguration mit dem Rexroth VI-Composer beziehen sich die Adressen der Variablenliste auf das jeweilige Byte im Array "TVar", das im SPS-Programm zum Datenaustausch benutzt wird.

Fehlerbehandlung Sobald ein Fehler auftritt, wird die Kommunikation unterbrochen und die Anzeige des Kleinbedienterminals zeigt COMMUNICATION ERROR, ERROR CODE 110.

Mit der Fehlerart (ErrorNo) wird angezeigt, dass es sich um einen Adressberechnungsfehler (CalculationError) handelt.

9.4 VCP_PBS32_A65536

Kurzbeschreibung Dieser Funktionsbaustein (FB) aktiviert das Profibus DP-Protokoll für die Kleinbedienterminals VCPxx. Zudem wird das E/A-Abbild der physikalischen Adressen zwischen SPS und Bedienterminal übertragen.

Die Datenbreite der Datenübertragung beträgt 32 Byte. Die Größe des über ein ARRAY zur Verfügung stehenden Adressbereichs beträgt 65536 Byte (Ein- und Ausgänge in Summe).

Schnittstellenbeschreibung

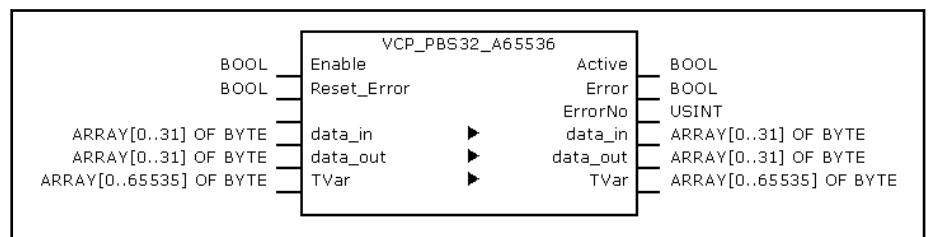


Abb. 9-3: VCP_PBS32_A65536

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	TRUE: FB wird bearbeitet FALSE: FB wird nicht bearbeitet
	Reset_Error	BOOL	TRUE: Rücksetzen von "Error" (auf FALSE) und "ErrorNo" wird auf 0 gesetzt
VAR_OUTPUT	Active	BOOL	TRUE, solange "Enable" ebenfalls TRUE ist
	Error	BOOL	TRUE, sobald ein Fehler auftritt. Lässt sich mit "Reset-Error" zurück setzen.
	ErrorNo	USINT	Fehlerart: 2: InputRangeError 4: CalculationError

	Name	Typ	Kommentar
VAR_IN_OUT	Data_in	ARRAY [0..31] OF BYTE	Daten zur Ankopplung der physikalischen Eingänge des Kleinbedienterminals
	Data_out	ARRAY [0..31] OF BYTE	Daten zur Ankopplung der physikalischen Ausgänge des Kleinbedienterminals
	TVar	ARRAY [0..65535] OF BYTE	Array, das zum Lesen vom und zum Schreiben auf das Kleinbedienterminal verwendet wird.

Tab. 9-3: Schnittstelle von VCP_PBS32_A65536

VI-Composer Bei der Konfiguration mit dem Rexroth VI-Composer beziehen sich die Adressen der Variablenliste auf das jeweilige Byte im Array "TVar", das im SPS-Programm zum Datenaustausch benutzt wird.

Fehlerbehandlung Sobald ein Fehler auftritt, wird die Kommunikation unterbrochen und die Anzeige des Kleinbedienterminals zeigt COMMUNICATION ERROR, ERROR CODE 110.

Mit der Fehlerart (ErrorNo) wird angezeigt, dass es sich um einen Adressberechnungsfehler (CalculationError) handelt.

10 Service und Support

Für Ihre schnelle und optimale Unterstützung verfügen wir über ein dichtes weltweites Servicenetz. Unsere Experten stehen Ihnen mit Rat und Tat zur Seite. Sie erreichen uns täglich **rund um die Uhr - auch an Wochenenden und Feiertagen**.

Service Deutschland

Unser technologieorientiertes Competence Center in Lohr deckt alle Belange rund um den Service für elektrische Antriebe und Steuerungen ab.

Sie erreichen unsere **Service-Hotline** und unseren **Service-Helpdesk** unter:

Telefon: **+49 9352 40 5060**
Fax: **+49 9352 18 4941**
E-Mail: service.svc@boschrexroth.de
Internet: <http://www.boschrexroth.com>

Auf unseren Internetseiten finden Sie ergänzende Hinweise zu Service, Reparatur (z. B. Anlieferadressen) und Training.

Service weltweit

Außerhalb Deutschlands nehmen Sie bitte zuerst Kontakt mit Ihrem Ansprechpartner auf. Die Hotline-Rufnummern entnehmen Sie bitte den Vertriebsadressen im Internet.

Vorbereitung der Informationen

Wir können Ihnen schnell und effizient helfen, wenn Sie folgende Informationen bereithalten:

- Eine detaillierte Beschreibung der Störung und der Umstände
- Angaben auf dem Typenschild der betreffenden Produkte, insbesondere Typenschlüssel und Seriennummern
- Ihre Kontaktdaten (Telefon-, Faxnummer und E-Mail-Adresse)

Index

A

Active.....	6
ANSI Z535.6-2006.....	2
Asynchrone FTP-Funktionsbausteine.....	307, 318
Asynchrone ICMP-Funktionsbausteine.....	304
Asynchrone Kommunikationsbausteine.....	323
AXIS_GROUP_REF.....	23
AXIS_REF.....	21

B

Basis-Bibliotheken	
Überblick.....	1
Basisklasse	
IL_EnableDoneBase.....	87
IL_EnableInOpBase.....	90
IL_EnableInOpShutdownBase.....	92
IL_ExecuteDoneAbortedBase.....	102
IL_ExecuteDoneAbortedShutdownBase.....	105
IL_ExecuteDoneBase.....	97
IL_ExecutelnOpBase.....	99
Basisklassen für FB-Entwicklung.....	82
Bausteintypen.....	5
Bausteintypen und ihre Schnittstellen.....	6
Bibliothek	
IL_VCP_DP.....	517
RIL_CommonTypes.....	19
RIL_HMI_Uilities.....	513
RIL_LoopControl.....	339
RIL_ModbusTCP.....	201
RIL_SocketComm.....	255
RIL_Uilities.....	65
Big-Endian.....	268
Byte-Order.....	268

C

CommandAborted.....	6
CONTROLS.....	21

D

Datentypen zum Aufsynchronisieren in den lageeregelten Synchron-Betriebsarten.....	51
Datentypen zum Referenzieren von Objek- ten bei Motion-Funktionsbausteinen.....	20
Datentypen zur Fehlerdiagnose bei Motion- Funktionsbausteinen.....	54
Datentypen zur Unterstützung von Kurven- scheiben, Motion- und FlexProfilen.....	26
Datentypen zur Unterstützung von Mess- taster-Funktionen.....	58
Datentypen zur Wiedergabe des Datum und absoluter Zeitangaben bei Motion-Funkti- onsbausteinen.....	57
Datum- und Zeitberechnung.....	179
Debugging.....	133
Dokumentation	

Änderungsverlauf.....	1
Done.....	6
Doppelt-verkettete Liste.....	190
Doppelt-verkettete Listen.....	189

E

Ein- und Ausgänge flankengesteuerter Funktionsbausteine, abschließende Bearbei- tung.....	10
Ein- und Ausgänge flankengesteuerter Funktionsbausteine, permanente Bearbeitung...	16
Ein- und Ausgänge zustandsgesteuerter Funktionsbausteine, abschließende Bearbei- tung.....	7
Ein- und Ausgänge zustandsgesteuerter Funktionsbausteine, permanente Bearbeitung...	13
Enable.....	6
ENUM_INT_OPTION.....	165
ENUM_OPT_UTF.....	167
ENUM_REAL_OPTION.....	166
ENUM_TRIM_OPT.....	178
ERROR_CODE.....	54
ERROR_STRUCT.....	55
ERROR_TABLE.....	56
Error_Tables	
ETHERNET_TABLE.....	334
ETHERNET_TABLE.....	334
16#00000001-16#00000212.....	335
16#00001001-16#00001002.....	336
16#00002001-16#00002004.....	336
16#00002101-16#00002108.....	337
Eventgesteuerte Funktionsbausteine.....	342
Execute.....	6
ExecuteLock.....	6
EXT_SYS_TIME_DATE.....	57

F

Fehlercodes	
IL_AmplitudeControllerType01.....	442
IL_AmplitudeControllerType02.....	445
IL_AngleGeneratorType01.....	427
IL_AngleGeneratorType02.....	429
IL_ContinuousAdjustType01.....	76
IL_ContinuousAdjustType02.....	79
IL_CRC16.....	130
IL_DeadtimeType01.....	371
IL_DeadtimeType02.....	374
IL_DeviationResponseCurve	377
IL_DT1Type01.....	379
IL_DT1Type02.....	381
IL_FIRType01.....	405
IL_FIRType02.....	407
IL_FTPGetAsync.....	310
IL_FTPPutAsync.....	313
IL_IIRType01.....	409

Index

IL_IIRType02.....	412	IL_TransportDelayType01.....	396
IL_IncrementalAdjustType01.....	81	IL_TransportDelayType02.....	399
IL_InterpolationLinear.....	139	IL_TwoPosCtrlType01.....	511
IL_LinkedList.....	194	IL_UDPClose.....	304
IL_LocalTime.....	183	IL_UDPCloseAsync.....	299
IL_ModbusTCPClient.....	229	IL_UDPCyclic.....	331
IL_ModbusTCPReadCoils.....	231	IL_UDPInitial.....	301
IL_ModbusTCPReadDeviceIdentification....	253	IL_UDPInitialAsync.....	295
IL_ModbusTCPReadDiscreteInputs.....	234	IL_UdpKeys.....	516
IL_ModbusTCPReadHoldingRegisters.....	236	IL_UDPRecvfrom.....	303
IL_ModbusTCPReadInputRegisters.....	239	IL_UDPRecvfromAsync.....	298
IL_ModBusTCPReadWriteMultipleRegis- ters.....	251	IL_UDPSendto.....	302
IL_ModbusTCPServer.....	219	IL_UDPSendtoAsync.....	296
IL_ModbusTCPTType02.....	225	IL_UDPSocketAsync.....	293
IL_ModBusTCPWriteMultipleCoils.....	245	IL_ValueLimit.....	400
IL_ModBusTCPWriteMultipleRegisters.....	248	IL_ValueWindow.....	402
IL_ModBusTCPWriteSingleCoil.....	241	Fehlertabellen	
IL_ModBusTCPWriteSingleRegister.....	243	ETHERNET_TABLE.....	334
IL_NoiseGeneratorType01.....	420	Filter.....	402
IL_NoiseGeneratorType02.....	422	FTP.....	267
IL_PDT1Type01.....	384		
IL_PDT1Type02.....	388	G	
IL_PhaseController.....	431	Gefahrenhinweise.....	1
IL_PIDType01.....	347	Grundelemente - Regelungstechnik.....	368
IL_PIDType02.....	351		
IL_PIDType03.....	355	H	
IL_PIDType04.....	360	Helpdesk.....	521
IL_Ping.....	307	Hotline.....	521
IL_PingAsync.....	306		
IL_PIType01.....	364	I	
IL_PIType02.....	368	I/O Zeitverhalten der Funktionsbausteine.....	255
IL_PRBSGeneratorType01.....	424	ICMP.....	267
IL_PRBSGeneratorType02.....	426	IIL_IIRType01.....	407
IL_PT1Type01.....	390	IIL_IIRType02.....	409
IL_PT1Type02.....	393	IL_AMP_CONTROL.....	445
IL_SignalGeneratorType01.....	435	IL_AmplitudeControllerType01.....	439
IL_SignalGeneratorType02.....	439	IL_AmplitudeControllerType02.....	442
IL_StateVariableType01.....	414	IL_AngleGeneratorType01.....	426
IL_StateVariableType02.....	417	IL_AngleGeneratorType02.....	428
IL_SuppressionType01.....	460	IL_Ascii_To_Char.....	164
IL_SuppressionType02.....	465	IL_AsciiStreamToBase64.....	148
IL_SuppressionType03.....	470	IL_AsciiToByte.....	130
IL_SuppressionType04.....	476	IL_Base64ToAsciiStream.....	149
IL_TCPClose.....	290	IL_BoolToByte.....	111
IL_TCPCloseAsync.....	284	IL_BoolToDWord.....	112
IL_TCPConnectionAsync.....	273	IL_BoolToWord.....	115
IL_TCPCyclic.....	327	IL_BufferGetElementCount.....	168
IL_TCPInitial.....	286	IL_BufferToString.....	151
IL_TCPInitialAsync.....	277	IL_ByteToAscii.....	131
IL_TCPRecv.....	288	IL_ByteToBool.....	119
IL_TCPRecvAsync.....	281	IL_ByteToDWord.....	116
IL_TCPRecvPacket.....	289	IL_CENTERLINE_CONTROL.....	445
IL_TCPRecvPacketAsync.....	283	IL_Char_To_Ascii.....	164
IL_TCPSend.....	287	IL_Code.....	169
IL_TCPSendAsync.....	279	IL_CONCAT.....	156
IL_TempControlType01.....	494	IL_ConcatByte.....	117
IL_TempControlType02.....	505	IL_ConcatWord.....	118

IL_CONN_MODE.....	332	IL_LowWord.....	125
IL_CONN_STATUS.....	333	IL_LRealToString.....	132
IL_ContinuousAdjustType01.....	72	IL_MaxValue.....	139
IL_ContinuousAdjustType02.....	76	IL_MID.....	161
IL_COORD_PAIR_TYPE01.....	140	IL_MKeys.....	513
IL_COORD_TYPE01.....	511	IL_MODBUS_STATS.....	225
IL_CRC16.....	129	IL_ModbusTCPClient.....	226
IL_CURVETYPE.....	446	IL_ModbusTCPReadCoils.....	229
IL_Date.....	179	IL_ModbusTCPReadDeviceIdentification.....	251
IL_DateAndTime.....	180	IL_ModbusTCPReadDiscreteInputs.....	231
IL_DeadtimeType01.....	369	IL_ModbusTCPReadHoldingRegisters.....	234
IL_DeadtimeType02.....	372	IL_ModbusTCPReadInputRegisters.....	236
IL_DELETE.....	157	IL_ModbusTCPReadWriteMultipleRegisters....	248
IL_DeviationResponseCurve.....	375	IL_ModbusTCPServer.....	216
IL_DT1Type01.....	377	IL_ModbusTCPServerType02.....	219
IL_DT1Type02.....	379	IL_ModbusTCPWriteMultipleCoils.....	243
IL_DWordToBool.....	120	IL_ModbusTCPWriteMultipleRegisters.....	245
IL_DWordToByte.....	123	IL_ModbusTCPWriteSingleCoil.....	239
IL_EnableDoneBase.....	87	IL_ModbusTCPWriteSingleRegister.....	241
IL_EnableInOpBase.....	90	IL_NoiseGeneratorType01.....	419
IL_EnableInOpShutdownBase.....	92	IL_NoiseGeneratorType02.....	420
IL_ExecuteDoneAbortedBase.....	102	IL_NULL.....	20
IL_ExecuteDoneAbortedShutdownBase.....	105	IL_PDT1Type01.....	381
IL_ExecuteDoneBase.....	97	IL_PDT1Type02.....	385
IL_ExecuteInOpBase.....	99	IL_PhaseController.....	429
IL_ExtSysTimeDate.....	180	IL_PIDType01.....	343
IL_FB_STATES.....	110	IL_PIDType02.....	347
IL_FILTER_TYPE.....	417	IL_PIDType03.....	351
IL_FIND.....	158	IL_PIDType04.....	355
IL_FIR_COEFF.....	417	IL_Ping.....	306
IL_FIRType01.....	403	IL_PingAsync.....	304
IL_FIRType02.....	405	IL_PIType01.....	360
IL_FTPGetAsync.....	307	IL_PIType02.....	364
IL_FTPGetFileAsync.....	313	IL_PRBSGeneratorType01.....	422
IL_FTPPutAsync.....	310	IL_PRBSGeneratorType02.....	424
IL_FTPPutFileAsync.....	315	IL_PT1Type01.....	388
IL_GetModuloType01.....	196	IL_PT1Type02.....	390
IL_GetModuloType02.....	197	IL_Real_To_Str.....	165
IL_GetModuloType03.....	198	IL_RealToString.....	131
IL_GetModuloType04.....	199	IL_REPLACE.....	162
IL_HighByte.....	123	IL_RIGHT.....	163
IL_HighResTimeDiff.....	134	IL_SignalGeneratorType01.....	431
IL_HighResTimeTick.....	134	IL_SignalGeneratorType02.....	435
IL_HighWord.....	124	IL_StateVariableType01.....	412
IL_HistogramType01.....	447	IL_StateVariableType02.....	414
IL_HistogramType02.....	449	IL_StatisticType01.....	451
IL_IIR_COEFF.....	417	IL_StatisticType02.....	453
IL_IncrementalAdjustType01.....	79	IL_StrCompare.....	145
IL_INSERT.....	158	IL_StrCompareNoCase.....	146
IL_Int_To_Str.....	164	IL_StrCopy.....	154
IL_InterpolationLinear.....	137	IL_StrGetElement.....	171
IL_LEFT.....	159	IL_String_To_Unicode.....	166
IL_LEN.....	160	IL_StringToBuffer.....	153
IL_LINKED_LIST_MAINTENANCE.....	194	IL_StrInverse.....	170
IL_LINKED_LIST_MODE.....	194	IL_StrLChr.....	172
IL_LinkedList.....	190	IL_StrLower.....	144
IL_LocalTime.....	181	IL_StrNCompare.....	146
IL_LowByte.....	124	IL_StrNCompareNoCase.....	147

Index

IL_StrNCopy.....	155
IL_StrRChr.....	173
IL_StrSetElement.....	174
IL_StrSetEndOfString.....	171
IL_StrSplit.....	175
IL_StrTrimAll.....	176
IL_StrTrimSpace.....	177
IL_StrUpper.....	144
IL_SUPPRESSION_LIMITS.....	511
IL_SUPPRESSION_MODE.....	512
IL_SuppressionType01.....	455
IL_SuppressionType02.....	460
IL_SuppressionType03.....	465
IL_SuppressionType04.....	470
IL_SwapDWord.....	127
IL_SwapWord.....	128
IL_SysTime64.....	183
IL_SysTime64ToSysTimeDate.....	184
IL_SysTimeDate.....	184
IL_SysTimeDateToSysTime64.....	185
IL_TCPClose.....	290
IL_TCPCloseAsync.....	283
IL_TCPConnectionAsync.....	269
IL_TCPCyclic.....	323
Fehlercodes.....	327
Verbindungsmode IL_CONN_MODE.....	332
Verbindungsmode IL_CONN_STATUS.....	333
IL_TCPInitial.....	285
IL_TCPInitialAsync.....	273
IL_TCPRecv.....	287
IL_TCPRecvAsync.....	279
IL_TCPRecvPacket.....	288
IL_TCPRecvPacketAsync.....	281
IL_TCPSend.....	286
IL_TCPSendAsync.....	277
IL_TEMPCONTROL_CONTR_PARAM.....	506
IL_TEMPCONTROL_CONTROL_MODE.....	506
IL_TEMPCONTROL_CONTROL_TYPE.....	507
IL_TEMPCONTROL_EXTENDED_SETTINGS.....	507
IL_TEMPCONTROL_LIMITS.....	508
IL_TEMPCONTROL_MODEL_PARAM.....	509
IL_TempControlType01.....	477
IL_TempControlType02.....	495
IL_TFTPGetFileAsync.....	318
IL_TFTPPutFileAsync.....	321
IL_TimeOfDay.....	185
IL_TIMEZONE.....	186
IL_Toggle.....	136
IL_TransportDelayType01.....	393
IL_TransportDelayType02.....	396
IL_TwoPosCtrlType01.....	509
IL_UDPClose.....	304
IL_UDPCloseAsync.....	298
IL_UDPCyclic.....	328
Fehlercodes.....	331
Verbindungsmode IL_CONN_MODE.....	332
Verbindungsmode IL_CONN_STATUS.....	333

IL_UDPInitial.....	300
IL_UDPInitialAsync.....	293
IL_UdpKeys.....	514
IL_UDPRecvfrom.....	302
IL_UDPRecvfromAsync.....	297
IL_UDPSendto.....	301
IL_UDPSendtoAsync.....	295
IL_UDPSocketAsync.....	291
IL_Unicode_To_String.....	167
IL_ValueLimit.....	399
IL_ValueWindow.....	401
IL_VCP_DP.library.....	517
IL_WordToBool.....	125
InGear.....	6
InOperation.....	6
InSync.....	6
InTorque.....	6
InVelocity.....	6
IP-Kommunkation	
Byte-Reihenfolge.....	268

K

KIN_OBJECTS.....	24
Kleinbedienterminal.....	517
Konvertierung	
Bit-Stream Connect.....	111
Bit-Stream Split.....	119
ByteSwap.....	127
Datatype Change.....	130

L

Little-Endian.....	268

M

Mathematische Funktionen	
IL_InterpolationLinear.....	137
IL_MaxValue.....	139
MB_ContinuousAdjustType01 (alt).....	72
MB_ContinuousAdjustType02 (alt).....	76
MB_DIFF_EVAL.....	59
MB_EXPECT_WINDOW_MODE.....	60
MB_GetModuloType01 (alt).....	196
MB_GetModuloType02 (alt).....	197
MB_GetModuloType03 (alt).....	198
MB_GetModuloType04 (alt).....	199
MB_IncrementalAdjustType01 (alt).....	79
MB_INTERPOLATION_MODE.....	60
MB_POSITION_EVAL.....	61
MB_PROBE_CONFIG.....	63
MB_PROBE_CONFIG, Besonderheiten bei	
Sercos-basierten Achsen.....	63
MB_PROBE_DATA.....	62
MB_PROBE_INPUT.....	61
MB_PROBE_MODULE.....	61
MC_CAM_ID.....	26
MC_START_MODE.....	51
MC_STEP_MODE.....	27

MC_SYNC_DIRECTION	53
ML_InterpolationLinear (alt).....	137
ML_MaxValue (alt).....	139
ML_TwoPosCtrlType01 (alt).....	509
ModBus	
Pro-face GP-Pro EX 2.2.....	208
Siemens WinCC-flexible 2007.....	204
Modbus TCP.....	201
ModbusTCP Client.....	226
ModbusTCP Server.....	202
Modulowert-Berechnung.....	195

O

OBJECTS.....	22

P

Prüfsummen	
CRC16.....	129

R

Regler	
PI.....	343
PID.....	343
RIL_CommonTypes, Datentypen an der Schnittstelle von Funktionsbausteinen.....	19
RIL_CommonTypes.library.....	19
RIL_HMI_Utilitys.library.....	513
RIL_LoopControl.library.....	339
RIL_ModbusTCP.library.....	201
RIL_SocketComm, vereinfachte TCP oder UDP Kommunikation.....	255
RIL_SocketComm.library.....	255
RIL_Utilitys.library.....	65

S

Service-Hotline.....	521
Shutdown.....	6
Sicherheitshinweise.....	1
Signalgeneratoren.....	418
Signalgrafik.....	2
Signalwörter.....	2
Standardschnittstellen an Funktionsbausteinen...	5
Statistikbausteine.....	446
Support.....	521
Symbole.....	3
Synchrone ICMP-Funktionen.....	306
SYS_TIME_DATE.....	58
SYS_TIME64.....	57

T

TCP.....	256
TCP Asynchrone Funktionsbausteine.....	269
TCP Synchrone Funktionen.....	284
Temperaturregler.....	477
TFTP.....	267
TOUCHPROBE_REF.....	25
TP_OBJECTS.....	26

U

UDP.....	260
UDP Asynchrone Funktionsbausteine.....	290
UDP Synchrone Funktionen.....	299

V

VCP_PBS16_A4096.....	517
VCP_PBS32_A4096.....	518
VCP_PBS32_A65536.....	519
Verkettete Liste.....	190
Verstell-Funktionen	
IL_ContinuousAdjustType02.....	72, 76
IL_IncrementalAdjustType01.....	79
Vertippen von Variablen.....	69
Berechnung der Änderungsgeschwindig- keit.....	69
Einführung und Übersicht.....	69
Grundregeln.....	70
Komponenten.....	81
VI-Composer.....	518, 519, 520

W

Warnhinweise.....	1

Z

Zeichenfolgen ohne Längenbegrenzung.....	141
Zeit- und Datumberechnung.....	179
Zeitdiskrete Funktionsbausteine.....	341
Zu dieser Dokumentation.....	1
Gültigkeit der Dokumentation.....	1
Zweipunktreger.....	509
Zweipunktreger mit Hysterese.....	509
Zyklische Kommunikation.....	268

Notizen

Notizen

Bosch Rexroth AG

Electric Drives and Controls

Postfach 13 57

97803 Lohr, Deutschland

Bgm.-Dr.-Nebel-Str. 2

97816 Lohr, Deutschland

Tel. +49 9352 18 0

Fax +49 9352 18 8400

www.boschrexroth.com/electrics



R911398632