

IndraWorks 15VRS

Feldbus-Bibliotheken

**Bibliothek
R911393274**

Ausgabe 04



Titel	IndraWorks 15VRS Feldbus-Bibliotheken
Art der Dokumentation	Bibliothek
Dokumentations-Type	DOK-IWORKS-FB*LIB**V15-LI04-DE-P
Interner Ablagevermerk	RS-1d62254ca090d45fc0a802863e737572-4-de-DE-5
Änderungsverlauf	Ausgabe 04, 2020-06 Siehe Tab. 1-1 "Änderungsverlauf" auf Seite 1
Schutzvermerk	© Bosch Rexroth AG 2020 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 Steuerungsfunktionen SPS und Feldbusse, BeAn (TaDo)

Inhaltsverzeichnis

	Seite
1 Zu dieser Dokumentation.....	1
1.1 Änderungsverlauf.....	1
1.2 Gültigkeit der Dokumentation.....	1
1.3 Bibliotheken und Zielsysteme.....	2
1.4 Gebrauch der Sicherheitshinweise.....	3
1.4.1 Aufbau der Sicherheitshinweise.....	3
1.4.2 Erläuterung der Signalwörter und der Signalgrafik.....	3
1.4.3 Verwendete Symbole.....	5
1.4.4 Erläuterung der Signalgrafik auf dem Gerät.....	5
1.5 Erforderliche und ergänzende Dokumentationen.....	5
1.6 Erforderliche und ergänzende Dokumentationen MLC.....	6
1.7 Erforderliche und ergänzende Dokumentationen MTX.....	12
1.7.1 Auswählen/Zusammenstellen.....	12
1.7.2 Konfigurieren.....	12
1.7.3 In Betrieb nehmen.....	13
1.7.4 Bedienen.....	14
1.7.5 OEM-Entwicklung.....	15
1.7.6 AddOns.....	15
1.8 Bezeichnungen und Abkürzungen.....	15
1.9 Kundenfeedback.....	16
2 Standardschnittstellen an Funktionsbausteinen.....	17
2.1 Motivation.....	17
2.2 Bausteintypen.....	17
2.3 Bausteintypen und ihre Schnittstellen.....	18
2.4 Ein- und Ausgänge zustandsgesteuerter Funktionsbausteine, abschließende Bearbeitung.....	19
2.5 Ein- und Ausgänge flankengesteuerter Funktionsbausteine, abschließende Bearbeitung.....	22
2.6 Ein- und Ausgänge zustandsgesteuerter Funktionsbausteine, permanente Bearbeitung.....	25
2.7 Ein- und Ausgänge flankengesteuerter Funktionsbausteine, permanente Bearbeitung.....	28
3 RIL_ProfibusDP_02.library.....	31
3.1 Einführung und Übersicht.....	31
3.2 Datentypen.....	31
3.2.1 IL_BUSMASTER.....	31
3.2.2 Slave-Diagnosedaten nach Profibus DP-Norm.....	32
3.2.3 Busmaster-Statuswort.....	34
3.2.4 Busmaster-Fehlerwort.....	35
3.2.5 DP_SLAVELIST, Array.....	35
3.2.6 DP_DEVICELIST, Struktur.....	36
3.2.7 DP_MASTERLIST, Array.....	36
3.2.8 DP_MASTERINFO, Struktur.....	37
3.3 Auswahl des DP-Masters.....	37
3.4 IL_DPMasterState.....	37

	Seite
3.5	IL_DPIdent..... 39
3.6	IL_DPV1Read..... 40
3.7	IL_DPV1Write..... 42
3.8	IL_DPReadDiag..... 43
3.9	IL_DPSycFr..... 45
3.10	IL_DPDeviceListGet..... 47
3.11	IL_DPBaudrateGet..... 48
3.12	IL_DPDevInfoGet..... 49
3.13	IL_DPPrjSlaveListGet..... 49
3.14	IL_DPSlaveDiagListGet..... 50
3.15	Fehlerkennungen..... 52
3.15.1	ErrorID / ERROR_CODE..... 52
3.15.2	ErrorIdent..... 52
3.15.3	FM_Error_Code..... 53
3.15.4	Error_Code_OB..... 54
3.15.5	Error_Code_FDL..... 54
3.15.6	Error_Code_DP..... 55
4	RIL_ProfibusDPSlave.library..... 57
4.1	Einführung und Übersicht..... 57
4.2	IL_PBDPSlaveDPV1Polling..... 59
4.3	IL_PBDPSlaveDPV1Response..... 61
4.4	IL_PBDPSlaveDPV1GetWriteData..... 64
4.5	IL_BUSSLAVE..... 67
4.6	IL_FBUS_SLAVE_SERVICE..... 67
4.7	IL_FBUS_SLAVE_RESULT..... 68
5	RIL_ProfinetIO.library..... 71
5.1	Einführung und Übersicht..... 71
5.2	Azyklische Dienste (Acyclic Communication)..... 73
5.2.1	IL_PNIORedRecord..... 73
5.2.2	IL_PNIOServiceRecord..... 76
5.2.3	IL_PNIOWriteRecord..... 81
5.3	Sichern und Wiederherstellen (Backup_Restore)..... 84
5.3.1	IL_PNIODeviceBackup..... 84
5.3.2	IL_PNIODeviceRestore..... 88
5.4	Unterstützung des Geräteauswechs (Device Replacement)..... 91
5.4.1	IL_PNIOAssignName..... 91
5.4.2	IL_PNIOCheckTopology..... 93
5.4.3	IL_PNIODeviceReplacement..... 98
5.4.4	IL_PNIODeviceResetToDefault..... 100
5.4.5	IL_PNIODeviceSetIP..... 102
5.4.6	IL_PNIOGetLiveList..... 104
5.4.7	IL_PNIOGetLogMsg..... 106
5.4.8	IL_PNIOGetPortData..... 107

	Seite
5.4.9 IL_PNIOStoreTopology.....	110
5.4.10 IL_PNIORestoreTopology.....	112
5.5 Diagnose (Diagnostics).....	116
5.5.1 IL_PNIOControllerState.....	116
5.5.2 IL_PNIOControllerStateDetails.....	118
5.5.3 IL_PNIODiagnosis.....	120
5.5.4 IL_PNIOGetConfigDeviceNameList.....	125
5.5.5 IL_PNIOGetControllerList.....	127
5.5.6 IL_PNIOGetDiagDeviceNameList.....	128
5.5.7 IL_PNIORemoteDeviceState.....	129
5.5.8 IL_PNIORemoteDeviceStateDetails.....	131
5.6 I&M (Identification and Maintenance) (IM Data).....	133
5.6.1 IL_PNIO_IM0.....	133
5.6.2 IL_PNIO_IM1.....	134
5.6.3 IL_PNIO_IM2.....	134
5.6.4 IL_PNIO_IM3.....	134
5.6.5 IL_PNIO_IM4.....	134
5.6.6 IL_PNIO_IM_RWMODE.....	135
5.6.7 IL_PNIOReadWriteIMData.....	135
5.7 Kommunikationsschnittstelle (IO-Link).....	140
5.7.1 IL_IOL_DEVICE_STATUS.....	140
5.7.2 IL_IOL_DEVICE_STATUS_DETAILS.....	140
5.7.3 IL_IOL_EVENT_QUALIFIER.....	141
5.7.4 IL_IOL_EVENT_QUALIFIER_INSTANCE.....	141
5.7.5 IL_IOL_EVENT_QUALIFIER_MODE.....	141
5.7.6 IL_IOL_EVENT_QUALIFIER_SOURCE.....	142
5.7.7 IL_IOL_EVENT_QUALIFIER_TYPE.....	142
5.7.8 IL_IOL_SERVICE.....	142
5.7.9 IL_IOL_BACKUP_MODE.....	142
5.7.10 IL_IOL_RESTORE_MODE.....	142
5.7.11 IL_PNIOIoLinkCall.....	143
5.7.12 IL_PNIOIoLinkGetDeviceAccessLock.....	149
5.7.13 IL_PNIOIoLinkIdent.....	154
5.7.14 IL_PNIOIoLinkIdentDetails.....	158
5.7.15 IL_PNIOIoLinkRemoteDeviceState.....	164
5.7.16 IL_PNIOIoLinkSetDeviceAccessLock.....	169
5.7.17 IL_PNIOIoLinkSystemCommand.....	175
5.7.18 IL_PNIOIoLinkBackupDevice.....	180
5.7.19 IL_PNIOIoLinkRestoreDevice.....	185
5.8 Datentypen (Data Types).....	189
5.8.1 IL_BUSMASTER.....	189
5.8.2 IL_FBUS_COMMUNICATION_STATE.....	190
5.8.3 IL_FBUS_SLAVE_STATE.....	190
5.8.4 IL_PNIO_CONTROLLER_STATE.....	190
5.8.5 IL_PNIO_PORT_DATA.....	191
5.8.6 IL_PNIO_REMOTE_DEVICE_INFO.....	191

	Seite
5.8.7 IL_PNIO_REMOTE_DEVICE_STATE.....	192
5.8.8 IL_PNIO_TOPO_STATE.....	192
5.8.9 IL_PNIO_TOPO_STATUS.....	193
5.8.10 PNIO_CHANNELERROR.....	193
5.8.11 PNIO_DIAGINFO.....	194
5.8.12 PNIO_DIAGTYPE.....	194
5.8.13 PNIO_LIVELIST.....	194
5.9 PROFINETIO_TABLE 16#200.....	195
5.9.1 Allgemeines.....	195
5.9.2 Allgemeine Fehlercodes	196
5.9.3 Spezielle Fehlercodes, Profinet IO-Controller.....	198
5.9.4 Spezielle Fehlercodes, Profinet IO-Status	199
5.9.5 Spezielle Fehlercodes, Profinet IO IO-Link Master	209
5.9.6 Spezielle Fehlercodes, Profinet IO IO-Link Device (anwendungsbezogene Fehler)	209
5.10 Beispielprogramme.....	210
6 RIL_ProfinetIODevice.library.....	211
6.1 Einführung und Übersicht.....	211
6.2 Diagnosen (POUs/Diagnosis).....	211
6.2.1 IL_PNIODeviceState.....	211
6.2.2 IL_PNIODeviceStateDetails.....	213
6.2.3 IL_PNIODeviceStateDetailsXMAC.....	214
6.3 Datentypen (DataTypes).....	216
6.3.1 IL_PNIO_DEVICE_STATE.....	216
6.3.2 IL_PNIO_DEVICE_XMAC.....	217
6.3.3 IL_PNIO_DEVICE_XMAC_PORT.....	218
6.4 IL_BUSSLAVE.....	218
6.5 PROFINETIO_TABLE 16#200.....	219
6.5.1 Allgemeines.....	219
6.5.2 Allgemeine Fehlercodes.....	220
6.5.3 Spezielle Fehlercodes, PROFINET IO-Device	222
7 RIL_EtherNetIP.library.....	223
7.1 Einführung und Übersicht.....	223
7.2 Azyklische Dienste (Acyclic_Communication).....	224
7.2.1 IL_ENIPGetAttributeSingle.....	224
7.2.2 IL_ENIPGetIdentityObjAllAttributes.....	226
7.2.3 IL_ENIPSendUnconnectedMessage.....	228
7.2.4 IL_ENIPSetAttributeSingle.....	230
7.3 Sichern und Wiederherstellen (Backup Restore).....	233
7.3.1 IL_ENIPAdapterBackup.....	233
7.3.2 IL_ENIPAdapterRestore.....	236
7.4 Diagnose (Diagnosis).....	239
7.4.1 IL_ENIPRemoteAdapterState.....	239
7.4.2 IL_ENIPRemoteAdapterStateDetails.....	241

	Seite
7.4.3 IL_ENIPScannerState.....	243
7.4.4 IL_ENIPScannerStateDetails.....	245
7.5 Datentypen (DataTypes).....	246
7.5.1 IL_BUSMASTER.....	246
7.5.2 IL_ENIP_IDENTITY_ATTRIBUTES.....	247
7.5.3 IL_ENIP_REMOTE_ADAPTER_STATE_DETAILS.....	248
7.5.4 IL_ENIP_SCANNER_STATE.....	249
7.5.5 IL_FBUS_COMMUNICATION_STATE.....	250
7.5.6 IL_FBUS_SLAVE_STATE.....	250
7.5.7 Auszug aus der Fehlertabelle ETHERNET_IP_TABLE (ERROR_TABLE = 16#0151).....	250
8 RIL_EtherNetIPAdapter.library.....	253
8.1 Einführung und Übersicht.....	253
8.2 Diagnose (Diagnosis).....	253
8.2.1 IL_ENIPAdapterState.....	253
8.2.2 IL_ENIPAdapterStateDetails.....	255
8.3 Diagnose (Diagnosis/EngineeringPortAdapter).....	256
8.3.1 IL_Status.....	256
8.4 Datentypen (DataTypes).....	258
8.4.1 IL_BUSSLAVE.....	258
8.4.2 IL_ENIP_ADAPTER_STATE.....	259
8.4.3 Allgemeine Fehlercodes, EtherNet/IP adapter	260
8.4.4 Spezielle Fehlercodes, EtherNet/IP adapter.....	261
8.4.5 CIP, Allgemeine Fehlercodes.....	262
8.4.6 Erweiterte Fehlercodes für Connection Manager	263
9 RIL_Ethercat.library.....	265
9.1 Einführung und Übersicht.....	265
9.2 Azyklische Kommunikation CoE (Acyclic Communication/CoE).....	266
9.2.1 IL_ECATCoeReadSdo.....	266
9.2.2 IL_ECATCoeWriteSdo.....	269
9.3 Azyklische Kommunikation SoE (Acyclic Communication/SoE).....	271
9.3.1 IL_ECATSoeRead.....	271
9.3.2 IL_ECATSoeWrite.....	275
9.3.3 IL_ECATSoeCommand.....	277
9.4 Diagnose (Diagnostics).....	280
9.4.1 IL_ECATGetNumConfigSlaves.....	280
9.4.2 IL_ECATGetNumOnlineSlaves.....	281
9.4.3 IL_ECATGetConfigEthercatAddrList.....	282
9.4.4 IL_ECATGetOnlineEthercatAddrList.....	284
9.4.5 IL_ECATGetSlaveConfigInfo.....	285
9.4.6 IL_ECATGetSlaveOnlineInfo.....	287
9.4.7 IL_ECATMasterState.....	289
9.4.8 IL_ECATMasterBusStatistic.....	290
9.4.9 IL_ECATRemoteSlaveState.....	292

Inhaltsverzeichnis

	Seite
9.4.10 IL_ECATRemoteSlaveSetErrorCounterPeriod.....	294
9.4.11 IL_ECATRemoteSlaveGetErrorCounterPeriod.....	295
9.4.12 IL_ECATRemoteSlaveGetErrorCounter.....	297
9.4.13 IL_ECATRemoteSlaveClearErrorCounter.....	299
9.4.14 IL_ECATCoeReadDiagHistory.....	300
9.5 Zustandsmaschine (State machine).....	304
9.5.1 IL_ECATMasterGetBusState.....	304
9.5.2 IL_ECATMasterSetBusState.....	305
9.5.3 IL_ECATRemoteSlaveGetDevState.....	307
9.5.4 IL_ECATRemoteSlaveSetDevState.....	308
9.6 Hilfsfunktionen (Utilities).....	310
9.6.1 IL_ECATSoeldnCoding.....	310
9.6.2 IL_ECATSoeAttributeDecoding.....	311
9.6.3 IL_ECATMasterStateDecoding.....	311
9.6.4 IL_ECATSlaveStateDecoding.....	312
9.7 Datentypen (DataTypes).....	312
9.7.1 IL_BUSMASTER.....	312
9.7.2 IL_ECAT_ADDRESS_TYPE.....	313
9.7.3 IL_ECAT_BASE_DATA_TYPE.....	313
9.7.4 IL_ECAT_COE_DIAG_HISTORY_FLAGS.....	314
9.7.5 IL_ECAT_COE_DIAG_HISTORY_STRUCT.....	315
9.7.6 IL_ECAT_COE_DIAG_MSG_EXT.....	316
9.7.7 IL_ECAT_COE_DIAG_MSG_RAW.....	317
9.7.8 IL_ECAT_COE_DIAG_TYPE.....	317
9.7.9 IL_ECAT_DL_STATUS.....	318
9.7.10 IL_ECAT_ESC_FEATURES.....	318
9.7.11 IL_ECAT_MASTER_BUS_STATISTIC.....	319
9.7.12 IL_ECAT_MASTER_STATE.....	319
9.7.13 IL_ECAT_MBX_CONFIG_PROTOCOLS.....	320
9.7.14 IL_ECAT_MBX_ONLINE_PROTOCOLS.....	320
9.7.15 IL_ECAT_MBX_SIZE_INFO.....	321
9.7.16 IL_ECAT_MEMORY_INFO.....	321
9.7.17 IL_ECAT_PORT_ERROR_COUNTER.....	321
9.7.18 IL_ECAT_PORT_STATE.....	322
9.7.19 IL_ECAT_SLAVE_CONFIG_INFO.....	322
9.7.20 IL_ECAT_SLAVE_ERROR_COUNTER.....	323
9.7.21 IL_ECAT_SLAVE_IDENTITY_INFO.....	324
9.7.22 IL_ECAT_SLAVE_ONLINE_INFO.....	324
9.7.23 IL_ECAT_SLAVE_STATE.....	325
9.7.24 IL_ECAT_SOE_ATTRIBUTE.....	326
9.7.25 IL_ECAT_SOE_DATA_LENGTH.....	326
9.7.26 IL_ECAT_SOE_DATA_STATUS.....	327
9.7.27 IL_ECAT_SOE_DATA_TYPE.....	327
9.7.28 IL_ECAT_SOE_ELEMENT.....	327
9.7.29 IL_ECAT_SOE_PARAM_TYPE.....	328
9.7.30 IL_ECAT_STATE.....	328

	Seite
9.8 Fehlercodes.....	328
9.8.1 EtherCAT Fehlertabelle.....	328
10 RIL_MappingList.library.....	331
10.1 Einführung und Überblick.....	331
10.2 POU's.....	333
10.2.1 IL_SlaveMapListInit.....	333
10.2.2 IL_SlaveMapListAddEntry.....	334
10.3 Datentypen (DataTypes).....	337
10.3.1 IL_ADDRESSTYPE.....	337
10.3.2 IL_FIELDBUSOBJECT.....	337
10.3.3 IL_FIELDBUSTYPE.....	338
10.3.4 IL_BUSSLAVE.....	339
11 RIL_SercosIII.library.....	341
11.1 Einführung und Übersicht.....	341
11.2 Azyklische Dienste (AcyclicCommunication).....	343
11.2.1 IL_SIIISvcRead.....	343
11.2.2 IL_SIIISvcWrite.....	348
11.3 Azyklische Dienste/Parameter (AcyclicCommunication/Parameter).....	351
11.3.1 IL_SIIICommand.....	351
11.3.2 IL_SIIISvcReadParameter.....	358
11.3.3 IL_SIIISvcReadRealParameter.....	363
11.3.4 IL_SIIISvcReadStringParameter.....	365
11.3.5 IL_SIIISvcWriteParameter.....	367
11.3.6 IL_SIIISvcWriteRealParameter.....	370
11.3.7 IL_SERCOS_DATA_STATUS.....	372
11.4 Sichern und Wiederherstellen (BackupRestore).....	373
11.4.1 IL_SIIISvcParameterBackup.....	373
11.4.2 IL_SIIISvcParameterRestore.....	379
11.4.3 IL_BACKUP_ERRORS.....	386
11.4.4 IL_BACKUP_OPTIONS.....	386
11.5 Diagnose (Diagnostics).....	387
11.5.1 IL_SIIISvcGetConfigSlaveAdrList.....	387
11.5.2 IL_SIIISvcGetDiagSlaveAdrList.....	388
11.5.3 IL_SIIISvcGetLiveSlaveAdrList.....	390
11.5.4 IL_SIIISvcMasterState.....	391
11.5.5 IL_SIIISvcMasterStateDetails.....	393
11.5.6 IL_SIIISvcRemoteSlaveState.....	394
11.5.7 IL_SIIISvcRemoteSlaveStateDetails.....	396
11.5.8 IL_SIIISvc_MASTER_STATE_DETAILS.....	398
11.5.9 IL_SIIISvc_REMOTE_SLAVE_STATE_DETAILS.....	399
11.5.10 IL_SIIISvc_REMOTE_SLAVE_STATE_EX.....	401
11.5.11 IL_SIIISvc_REMOTE_SLAVE_STATE_EX_BITS.....	401
11.5.12 IL_SIIISvc_DEVICE_STATUS.....	402

Inhaltsverzeichnis

	Seite
11.5.13 IL_SIII_DEVICE_STATUS_UNION.....	402
11.5.14 IL_SIII_DEVICE_STATUS_UNION_BITS.....	403
11.5.15 IL_SIII_TOPOLOGY_PORT_STATUS.....	403
11.5.16 IL_SIII_DIAGNOSTIC_NUMBER.....	404
11.5.17 IL_SIII_DIAGNOSTIC_NUMBER_R.....	405
11.5.18 IL_SIII_DIAGNOSTIC_NUMBER_SOURCE_GROUP.....	405
11.5.19 IL_SIII_DIAGNOSTIC_NUMBER_SOURCE_TYPE.....	405
11.5.20 IL_SIII_DIAGNOSTIC_NUMBER_CLASS.....	406
11.5.21 IL_SIII_DEVICE_TYPE.....	406
11.5.22 IL_SIII_DRIVE_DIAG.....	406
11.5.23 IL_SIII_DRV_ERROR.....	407
11.5.24 IL_SIII_DRV_ERROR_BITS.....	407
11.5.25 IL_SIII_DRV_WARNING.....	408
11.5.26 IL_SIII_DRV_WARNING_BITS.....	408
11.5.27 IL_SIII_DRV_MESSAGE.....	409
11.5.28 IL_SIII_DRV_MESSAGE_BITS.....	409
11.5.29 IL_SIII_IO_DIAG.....	410
11.5.30 IL_SIII_IO_DIAG_STATUS_CODE_TYPE.....	411
11.5.31 IL_SIII_IO_DIAG_STATUS_CODE.....	411
11.5.32 IL_FBUS_COMMUNICATION_STATE.....	418
11.6 Hilfsfunktionen (Utilities).....	418
11.6.1 IL_SIIIAttribute.....	418
11.6.2 IL_SIIIElementsToldn.....	421
11.6.3 IL_SIIIGetSercosPhase.....	422
11.6.4 IL_SIIISetSercosPhase.....	424
11.6.5 IL_SIIIStringToldn.....	426
11.6.6 IL_SIII_DATA_LENGTH.....	427
11.6.7 IL_SIII_DATA_TYPE.....	427
11.6.8 IL_SIII_ELEMENT.....	427
11.6.9 IL_SIII_PARAM_TYPE.....	428
11.6.10 IL_SIII_SERCOS_PHASE.....	428
11.6.11 IL_BUSMASTER.....	429
12 RMB_SercosIII_Util.library.....	431
12.1 Einführung und Übersicht.....	431
12.2 Drives.....	436
12.2.1 MB_SIIIPlcDriveType01.....	436
12.2.2 MB_SIIIPlcDriveVelocityType01.....	441
12.2.3 MB_SIIIPlcDriveTorqueType01.....	444
12.2.4 MB_SIIIPlcDrivePositioningType01.....	448
12.2.5 MB_SIIIPlcDrivePosInterpolationType01.....	454
12.2.6 MB_SIII_CONTROLWORD.....	458
12.2.7 MB_SIII_STATUSWORD.....	459
12.2.8 MB_SIII_DRIVE_SHUTDOWN_MODE.....	460
12.2.9 MB_SIII_HOMING_MODE.....	460
12.2.10 MB_SIII_POSCTRLWORD.....	460

	Seite
12.2.11 MB_SIII_IN_PROC_VELOCITY.....	461
12.2.12 MB_SIII_OUT_PROC_VELOCITY.....	461
12.2.13 MB_SIII_IN_PROC_TORQUE.....	462
12.2.14 MB_SIII_OUT_PROC_TORQUE.....	462
12.2.15 MB_SIII_IN_PROC_POSITIONING.....	462
12.2.16 MB_SIII_OUT_PROC_POSITIONING.....	462
12.2.17 MB_SIII_IN_PROC_POS_INTERPOLATION.....	463
12.2.18 MB_SIII_OUT_PROC_POS_INTERPOLATION.....	463
12.3 PowerSupply.....	464
12.3.1 MB_SIIIPowerSupplyDriveIntegrated.....	465
12.3.2 MB_SIIIPowerSupplyStandalone.....	473
12.3.3 MB_SIIIPowerSupplyDcDcIslandGrid.....	480
12.3.4 MB_POWERSUPPLY_MODE.....	487
12.3.5 MB_POWERSUPPLY_ACTUAL_MODE.....	487
12.3.6 MB_POWERSUPPLY_ERROR_MODE.....	488
12.3.7 MB_POWERSUPPLY_TARGET_MODE.....	489
12.3.8 MB_PS_TARGET_MODE_DC_DC_ISLAND_GRID.....	490
12.3.9 MB_POWERSUPPLY_FUNCTION.....	490
12.4 RexrothEFC.....	491
12.4.1 MB_SIIIEfcCodeToldn.....	491
12.4.2 MB_SIIIPlcDriveEfcFrequencyType01.....	492
12.4.3 MB_EFC_FUNCTION_CODE.....	496
12.4.4 MB_SIII_IN_PROC_EFC_FREQ.....	496
12.4.5 MB_SIII_OUT_PROC_EFC_FREQ.....	496
12.4.6 MB_SIII_EFC_CONTROLWORD.....	497
12.4.7 MB_SIII_EFC_STATUSWORD.....	497
12.5 RexrothHydraulicDrives.....	498
12.5.1 MB_SIIIPlcDriveFlowControl.....	498
12.5.2 MB_SIIIPlcDrivePressureFlowControl.....	503
12.5.3 MB_SIIIPlcDrivePressureSwivelAngleControl.....	508
12.5.4 MB_SIIIPlcDrivePressureSwivelAngleSlave.....	513
12.5.5 MB_SIIIPlcDriveDirectValveControl.....	517
12.5.6 MB_SIIIPlcDriveTorqueDirectValveControl.....	521
12.5.7 MB_SIIIPlcDriveTorqueFlowControl.....	527
12.5.8 MB_SIIIPlcDrivePressureDirectValveControl.....	532
12.5.9 MB_SIIIPlcDriveVelocityAltCtrl.....	537
12.5.10 MB_SIII_HYDR_CONTROL_MODE.....	542
12.5.11 MB_SIII_IN_PROC_FLOW.....	543
12.5.12 MB_SIII_OUT_PROC_FLOW.....	543
12.5.13 MB_SIII_IN_PROC_PRESSURE_FLOW.....	543
12.5.14 MB_SIII_OUT_PROC_PRESSURE_FLOW.....	544
12.5.15 MB_SIII_IN_PROC_SWIVEL.....	545
12.5.16 MB_SIII_OUT_PROC_SWIVEL.....	545
12.5.17 MB_SIII_IN_PROC_SWIVEL_SLAVE.....	546
12.5.18 MB_SIII_OUT_PROC_SWIVEL_SLAVE.....	546
12.5.19 MB_SIII_IN_PROC_SPOOL.....	547

	Seite
12.5.20 MB_SIII_OUT_PROC_SPOOL.....	547
12.5.21 MB_SIII_IN_PROC_TORQUE_SPOOL.....	548
12.5.22 MB_SIII_OUT_PROC_TORQUE_SPOOL.....	548
12.5.23 MB_SIII_IN_PROC_TORQUE_FLOW.....	548
12.5.24 MB_SIII_OUT_PROC_TORQUE_FLOW.....	549
12.5.25 MB_SIII_IN_PROC_PRESSURE_SPOOL.....	549
12.5.26 MB_SIII_OUT_PROC_PRESSURE_SPOOL.....	550
12.5.27 MB_SIII_IN_PROC_VEL_ALT_CTRL.....	550
12.5.28 MB_SIII_OUT_PROC_VEL_ALT_CTRL.....	551
12.6 RexrothSytronix.....	551
12.6.1 MB_SIIIDRn5020.....	551
12.6.2 MB_SIIIFcP5020.....	557
12.6.3 MB_SIII_DRN_ENABLE.....	561
12.6.4 MB_SIII_DRN_VAR_VEL_ENABLE.....	562
12.6.5 MB_SIII_FCP_MASTERSLAVE.....	562
12.6.6 MB_SIII_FCP_PARAGROUP.....	562
12.6.7 MB_SIII_FCP_DUALPUMPSTATUS.....	562
13 RIL_SercosIIIDeviceReplacement.library.....	563
13.1 Einführung und Übersicht.....	563
14 RIL_InlineIOModules.library.....	573
14.1 Einführung und Überblick.....	573
14.2 Analoge Eingangsklemmen.....	576
14.2.1 IL_AI2Sf.....	576
14.2.2 IL_AI4Ef.....	580
14.2.3 IL_AI4IU.....	585
14.2.4 IL_AI8Sfls.....	590
14.2.5 IL_Sgi2f.....	594
14.2.6 IL_Sgi2fOffset.....	600
14.2.7 IL_Temp2Uth.....	600
14.2.8 IL_Temp2Rtd.....	605
14.2.9 IL_Temp4_8Rtd.....	611
14.2.10 IL_Temp4_8RtdEf.....	617
14.3 Analoge Ausgangsklemmen.....	623
14.3.1 IL_AO1Sf.....	623
14.3.2 IL_AO2SfBp.....	626
14.3.3 IL_AO2UI.....	630
14.4 Funktionsklemmen.....	634
14.4.1 IL_CntCount.....	634
14.4.2 IL_CntFrequency.....	641
14.4.3 IL_CntPulse.....	647
14.4.4 IL_CntTime.....	650
14.4.5 IL_Pwm2Frequency.....	656
14.4.6 IL_Pwm2Pwm.....	659

	Seite
14.4.7 IL_Pwm2PulseDirection.....	663
14.4.8 IL_Pwm2SinglePulse.....	666
14.4.9 IL_Ssi.....	669
14.4.10 IL_SsiIn.....	684
14.4.11 IL_Incln.....	689
14.5 Kommunikationsmodule.....	694
14.5.1 IL_RsCom.....	694
14.6 Datentypen.....	702
14.6.1 IL_INLN_AI_CONFIG.....	702
14.6.2 IL_INLN_AI_RANGE.....	702
14.6.3 IL_INLN_AI_DATA.....	703
14.6.4 IL_INLN_DIAG_CODE.....	703
14.6.5 IL_INLN_CNT_INCONFIG.....	704
14.6.6 IL_INLN_TEMP_CONFIG.....	704
14.6.7 IL_INLN_TEMP_CONNECTIONTYPE.....	704
14.6.8 IL_INLN_TEMP_R0.....	705
14.6.9 IL_INLN_TEMP_SENSOR.....	705
14.6.10 IL_INLN_TEMP_EF_CONFIG.....	706
14.6.11 IL_INLN_TEMP_EF_ODS.....	706
14.6.12 IL_INLN_TEMP_EF_R0.....	706
14.6.13 IL_INLN_TEMP_EF_SENSOR.....	707
14.6.14 IL_INLN_RS_CONFIG.....	707
14.6.15 IL_INLN_RS_PROTOCOL.....	708
14.6.16 IL_INLN_RS_BAUDRATE.....	708
14.6.17 IL_INLN_RS_DATAWIDTH.....	709
14.6.18 IL_INLN_RS_INTERFACE.....	709
14.6.19 IL_INLN_RS_MODULETYPE.....	709
14.6.20 IL_INLN_RS_DIAG.....	710
15 RIH_Safety.library.....	711
16 RIH_Safety_ProfibusDP.library.....	713
16.1 IH_PBDPAck_PSDx_Inline.....	713
16.2 IH_PBDPAck_PSDx_S20.....	718
17 RIH_Safety_ProfinetIO.library.....	725
17.1 IH_PNIOAck_PSDx_S20.....	725
17.2 IH_PNIOAck_PSDx_Inline.....	731
18 RIH_Safety_SercosIII.library.....	737
18.1 IH_SIIIAck SSDx_S20.....	737
19 RIH_Dali.library.....	745
19.1 RIH_Dali, Allgemeines.....	745

Inhaltsverzeichnis

	Seite
19.2	IH_DaliDim..... 747
19.3	IH_DaliAddress..... 749
19.4	IH_DaliCallScene..... 751
19.5	IH_DaliRemoveFromGroup..... 753
19.6	IH_DaliDirectPower..... 755
19.7	IH_DaliRemoveFromScene..... 758
19.8	IH_DaliQueryStatus..... 760
19.9	IH_DaliReadWrite..... 763
19.10	IH_DaliQueryActualLevel..... 766
19.11	IH_DaliAddToGroup..... 770
19.12	IH_DaliAddToScene..... 772
19.13	IH_DaliOff..... 774
19.14	IH_DaliOn..... 776
19.15	IH_DaliSearchAddress..... 779
19.16	IH_DaliQuerySceneLevel..... 781
19.17	IH_DaliProgramShortAddress..... 783
19.18	IH_DaliStoreParameter..... 788
19.19	IH_DaliReset..... 791
19.20	IH_DaliQueryDTR..... 794
19.21	IH_DaliQueryParameter..... 796
19.22	IH_DaliQueryGroup..... 800
19.23	IH_DaliQueryRandomAddress..... 803
19.24	Datentypen der Bibliothek RIH_Dali..... 806
19.24.1	IH_INLN_DALI_TOKEN..... 806
19.24.2	IH_INLN_DALI_ADDRTYPE..... 806
19.24.3	IH_INLN_DALI_DIMMODE..... 806
19.24.4	IH_INLN_DALI_LIGHTLEVEL..... 807
19.24.5	IH_INLN_DALI_OPMODE..... 807
19.24.6	IH_INLN_DALI_PARAMETER..... 808
19.25	Fehlerbehandlung mit der Bibliothek RIH_Dali..... 809
20	RIH_Modbus.library..... 811
20.1	RIH_Modbus, Allgemeines..... 811
20.2	IH_MBGetCommEventCounter..... 813
20.3	IH_MBGetCommEventLog..... 816
20.4	IH_MBMaskWriteRegister..... 820
20.5	IH_MBReadCoils..... 823
20.6	IH_MBReadDeviceIdentification..... 827
20.7	IH_MBReadDiagnostics..... 834
20.8	IH_MBReadDiscreteInputs..... 837
20.9	IH_MBReadExceptionStatus..... 841
20.10	IH_MBReadFIFOQueue..... 844
20.11	IH_MBReadFileRecord..... 847
20.12	IH_MBReadHoldingRegisters..... 851
20.13	IH_MBReadInputRegisters..... 855
20.14	IH_MBReadWriteMultipleRegisters..... 859

	Seite
20.15 IH_MBRTUMaster.....	863
20.16 IH_MBRTUSlave.....	868
20.17 IH_MBWriteFileRecord.....	875
20.18 IH_MBWriteMultipleCoils.....	879
20.19 IH_MBWriteMultipleRegisters.....	883
20.20 IH_MBWriteSingleCoil.....	888
20.21 IH_MBWriteSingleRegister.....	891
20.22 Datentypen.....	894
20.22.1 IH_INLN_MB_DATA.....	894
20.22.2 IH_INLN_MB_DEV_IDCODE.....	895
20.22.3 IH_INLN_MB_FIFODATA.....	895
20.22.4 IH_INLN_MB_CONFIG.....	895
20.22.5 IH_INLN_MB_TOKEN.....	896
20.22.6 IH_INLN_MB_SUBREQUESTS.....	897
20.22.7 IH_INLN_MB_RECORD.....	897
20.22.8 IH_INLN_MB_EVENTLOG.....	897
20.23 Fehlerbehandlung mit der Bibliothek RIH_Modbus.....	898
21 RIL_IoLink.library.....	901
21.1 Einführung und Übersicht.....	901
21.2 Allgemeines zu IO-Link.....	901
21.3 IL_SIII_IoLinkProxy.....	903
21.4 IL_IoLinkCall.....	906
21.5 IL_IoLinkErrorText.....	910
21.6 IL_IoLinkIdent.....	911
21.7 RexrothHydraulicValve.....	913
21.7.1 IL_IoLinkDirectValveControlType01.....	913
21.7.2 IL_IOL_OUT_PROC_SPOOL_TYPE01.....	917
21.7.3 IL_IOL_IN_PROC_SPOOL_TYPE01.....	917
22 Modbus RTU - Bibliotheken.....	919
22.1 IoDrvModbusComPort.....	919
22.2 ModbusSlaveComPort.....	919
22.3 IoDrvModbusTCP.....	920
22.4 ModbusTCPSlave.....	920
22.5 IoDrvModbusSerialSlave.....	921
22.6 IoDrvModbusTCPSlave.....	922
22.7 ModbusServer.....	922
22.8 ModbusRequest.....	924
23 RIH_IndraControl_S20.library.....	927
23.1 Einführung und Überblick.....	927
23.2 Fehlercodes.....	930
23.3 Diagnose (Diagnostics).....	931
23.3.1 IH_S20IOModuleStateDetails.....	931

	Seite
23.3.2 IH_S20IOState.....	932
23.3.3 IH_S20IOStateDetails.....	932
23.4 I&M (Identification and Maintenance) (IM Data).....	933
23.4.1 IH_S20ModuleIdentDetails.....	933
23.5 Azyklische Kommunikation (Acyclic Communication).....	934
23.5.1 IH_S20ComConfiguration.....	934
23.5.2 IH_S20ComInterfaceType02.....	935
23.5.3 IH_S20ComInterface.....	937
23.6 Analoge Eingangsmodule (AnalogIO).....	942
23.6.1 IH_S20AI4IU.....	942
23.6.2 IH_S20AI8.....	946
23.6.3 IH_S20AI4RTD.....	948
23.6.4 IH_S20AI4UTH.....	950
23.6.5 IH_S20AI8RTD.....	952
23.6.6 IH_S20AI8UTH.....	954
23.7 Analoge Ausgangsmodule (AnalogIO).....	956
23.7.1 IH_S20AO8.....	956
23.7.2 IH_S20AO4.....	958
23.8 Technologiemodule (TechnologyIO).....	961
23.8.1 IH_S20SsiAo1.....	961
23.8.2 IH_S20CntInc.....	965
23.8.3 IH_S20RSCom.....	970
 24 RIH_IndraControl_S20_KNX.....	 979
24.1 Allgemeines.....	979
24.2 Applikationsbeispiel.....	986
 25 BACnet-Bibliotheken.....	 997
25.1 Übersicht.....	997
25.2 Programmierbeispiele.....	998
25.2.1 Server: NotificationClass → SetRecipientList	998
25.2.2 Server / Client: IBACnetEventConsumer → BACnetEventCallback.....	998
 26 Service und Support.....	 1001
 Index.....	 1003

1 Zu dieser Dokumentation

1.1 Änderungsverlauf

Ausgaben dieser Dokumentation

Ausgabe	Stand	Bemerkung
01	2019-03	Erstausgabe
02	2019-06	Neu: RIH_IndraControl_S20_KNX, RIL_SercosIIIDeviceReplacement Update: MB_SIIIPowerSupplyType03, MB_SIIIPowerSupplyType04
03	2019-10	Neu: Kap. 12.3.3 "MB_SIIIPowerSupplyDcDclslandGrid" auf Seite 480 Update: MB_SIIIPowerSupplyType03 geändert in MB_SIIIPowerSupplyDriveIntegrated , MB_SIIIPowerSupplyType04 geändert in MB_SIIIPowerSupplyStandalone , Kap. 24 "RIH_IndraControl_S20_KNX" auf Seite 979, MB_SIIIPowerSupplyType01 und MB_SIIIPowerSupplyType02 entfernt BACnet: Hinweis ergänzt
04	2020-06	Update: Kap. 24 "RIH_IndraControl_S20_KNX" auf Seite 979

Tab. 1-1: Änderungsverlauf

1.2 Gültigkeit der Dokumentation

Übersicht über Zielgruppen und Produktphasen

In der folgenden Grafik beziehen sich die umrandeten Aktivitäten, Produktphasen und Zielgruppen auf die vorliegende Dokumentation.

Beispiel: In der Produktphase "Inbetriebnahme" kann die Zielgruppe "Technologie" mit Hilfe dieser Dokumentation die Aktivitäten "optimieren" und "erproben" ausführen.

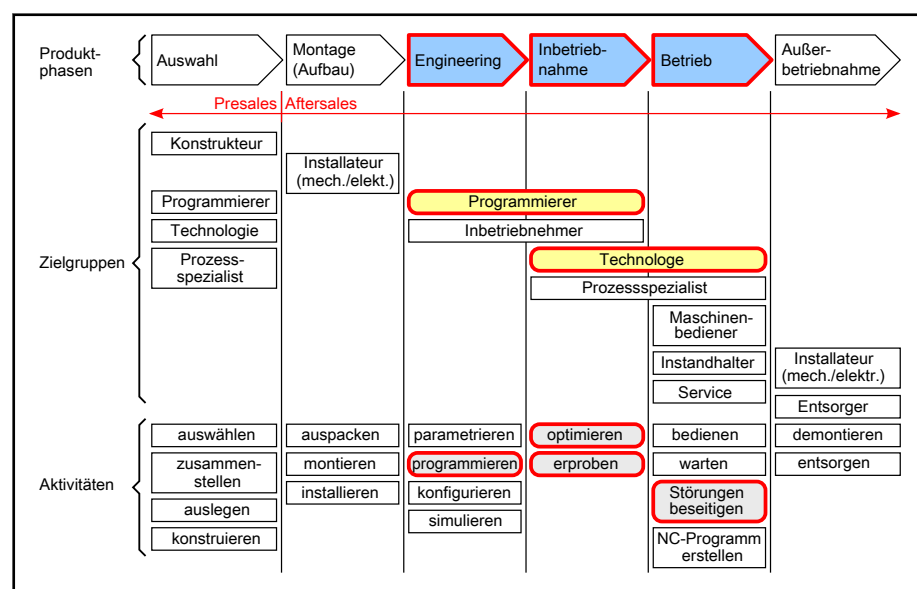


Abb. 1-1: Zuordnung der vorliegenden Dokumentation zu den Zielgruppen, Produktphasen und den Aktivitäten der Zielgruppe

Zielgruppe Diese Dokumentation richtet sich an Anwender, die sich über die verfügbaren Feldbus-Bibliotheken der Steuerungen MLC oder MTX informieren möchten.

Anwendungsphasen Diese Dokumentation unterstützt den Anwender in den Phasen

- Feldbus-Applikation und -Diagnose im SPS-Anwenderprogramm

1.3 Bibliotheken und Zielsysteme

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

Name	Beschreibung	MLC	MTX
RIL_ProfibusDP_02, Seite 31	Profibus-DP Master V1-Dienste, Diagnose, Sync und Freeze	(x)	(x)
RIL_ProfibusDPSlave, Seite 57	Profibus-DP Slave, V1-Dienste, Diagnose	x	x
RIL_ProfinetIO, Seite 71	Diagnose- und Kommunikationsdienste für Profinet IO-Controller	x	x
RIL_ProfinetIODevice, Seite 211	Diagnose- und Kommunikationsdienste für Profinet IO-Device	x	x
RIL_EtherNetIP, Seite 223	Diagnose- und Kommunikationsdienste für Ethernet IP Scanner	x	x
RIL_EtherNetIPAdapter, Seite 253	Diagnose- und Kommunikationsdienste für Ethernet IP Adapter und Ethernet IP adapter (Engineering-Schnittstelle)	x	x
RIL_EtherCAT, Seite 265	Schnittstelle von der SPS-Programmierungsumgebung zum EtherCAT Master	x	
RIL_MappingList, Seite 331	Mapping-Tabelle für azyklische Zugriffe auf Feldbuslaves (DP-Slave, Profinet IO-Device und ENIP-Adapter)	x	x
RIL_SercosIII, Seite 341	Diagnose- und Kommunikationsdienste für Sercos III	x	x
RMB_SercosIII_Util, Seite 431	Vereinfachte Achsansteuerung über den Sercos Bus, der Antrieb wird als SPS-Gerät angelegt	x	x
RIL_SercosIIIDeviceReplacement, Seite 563	Ermöglicht einen PC-losen Gerätetausch	x	x
RIL_InlineIOModules, Seite 573	Zugriffsmöglichkeiten eines IEC-Programms auf Rexroth-InlineIO-Module	x	x
RIH_Safety, Seite 711	Unterstützt den Zugriff der Standardsteuerung auf Daten der Sicherheitssteuerung SafeLogic durch die Bereitstellung von Datentypen	x	x
RIH_Safety_ProfibusDP, Seite 713	Enthält einen Quittierungs-Funktionsbausteinen für die Safety-EA-Module von Bosch Rexroth, die an der SafeLogic betrieben werden (inline)	x	x
RIH_Safety_ProfinetIO, Seite 725	Enthält Quittierungs-Funktionsbausteine für die Safety-EA-Module von Bosch Rexroth, die an der SafeLogic betrieben werden (inline und S20)	x	x
RIH_Safety_SercosIII, Seite 737	Enthält einen Quittierungs-Funktionsbausteinen für die Safety-EA-Module von Bosch Rexroth, die an der SafeLogic betrieben werden (S20)	x	x
RIH_Dali, Seite 745	Ist mit den Inline-Modulen R-IB IL DALI/PWR-PAC (R911172212) und R-IB IL DALI-PAC (R911172213) zu verwenden. Das Inline-Modul kann an die Steuerung oder einen Inline-Buskoppler angeschlossen werden (Profibus, Profinet, Sercos)	x	x
RIH_Modbus, Seite 811	Stellt Funktionsbausteine zur Kommunikation mit dem Modbus-RTU-Protokoll als Master oder Client bereit. ist mit den Inline-Modulen R-IB IL RS232 PRO , R-IB IL RS485/422 PRO oder R-IB IL RS-Uni (Inline-Module für serielle Datenübertragung) zu verwenden. Die Inline-Module können an die Steuerung oder einen Inline-Buskoppler angeschlossen werden (Profibus, Profinet, Sercos)	x	x
RIL_IoLink.library, Seite 901	Enthält feldbusunabhängige Funktionsbausteine zum Ansteuern von IO-Link-Geräten	x	x

Name	Beschreibung	MLC	MTX
RIH_IndraControl_S20, Seite 927	Zugriffsmöglichkeiten des IEC-Programms auf das IndraControl S20-System mithilfe von Funktionsbausteinen	x	x
RIH_IndraControl_S20_KNX, Seite 979	Zugriffsmöglichkeiten des SPS-Programms auf das IndraControl S20 KNX-Modul mithilfe von Funktionsbausteinen	x	x
BACnet.library, Seite 997	Schnittstelle von der SPS-Programmierungsumgebung zum BACnet-Stack	x	

x Bibliothek existiert / ist verfügbar für System

(x) Bibliothek ist in Bearbeitung

Tab. 1-2: Feldbus-Bibliotheken zu IndraLogic 2G

1.4 Gebrauch der Sicherheitshinweise

1.4.1 Aufbau der Sicherheitshinweise

Die Sicherheitshinweise sind wie folgt aufgebaut:

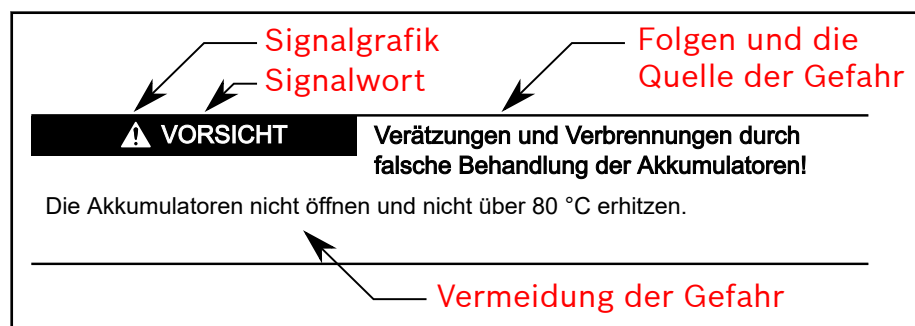


Abb. 1-2: Aufbau der Sicherheitshinweise

1.4.2 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.

1.4.3 Verwendete Symbole

Fingerzeige werden wie folgt dargestellt:



Dies ist ein Hinweis.

Tipps werden wie folgt dargestellt:



Dies ist ein Tipp.

1.4.4 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.

1.5 Erforderliche und ergänzende Dokumentationen

Die im Folgenden aufgeführten Dokumente beinhalten zusätzliche Informationen zu diesem Thema.

IndraWorks 14VRS IndraLogic 2G SPS-Programmiersystem

DOK-IWORKS-IL2GPRO*V14-APRS-DE-P, R911343570

Diese Dokumentation beschreibt das SPS-Programmierwerkzeug IndraLogic 2G und seine Verwendung. Beschrieben werden die Basisverwendung, Erste Schritte, die Visualisierung, Menüpunkte und die Editoren.

IndraWorks 15VRS Feldbusse

DOK-IWORKS-FB*****V15-APRS-DE-P, R911393282

Die Dokumentation dient der Beschreibung der von den Systemen IndraMotion MLC und IndraMotion MTX unterstützten Feldbus- und lokalen Peripherieanschlüssen. Der Schwerpunkt dieser Dokumentation liegt in der Konfiguration, Parametrierung, Inbetriebnahme und Diagnose der verschiedenen Peripherieanschlüssen. Sie ist die Grundlage für die Online-Hilfe.

IndraWorks 14VRS Basis-Bibliotheken IndraLogic 2G

DOK-IL*2G*-BASLIB**V14-LIRS-DE-P, R911343918

Das vorliegende Handbuch beschreibt die systemübergreifenden SPS-Bibliotheken.

IndraMotion 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 die IndraMotion MLC. Die Dokumentation enthält ferner die Fehlerreaktion der Bausteine.

Die Automatisierungsklemmen der Produktfamilie Rexroth Inline

DOK-CONTRL-ILSYSINS***-AWRS-DE-P, R911317017

Diese Dokumentation beschreibt die Automatisierungsklemmen der Produktfamilie Rexroth Inline.

Rexroth Inline-Buskoppler für Sercos mit digitalen Ein- und Ausgängen R-IB IL S3 BK DI8 DO4-PAC

DOK-CONTRL-ILS3BKDI8DO-KBRS-DE-P, R911170559

Diese Dokumentation beschreibt einen Buskoppler, der zum Einsatz innerhalb eines Sercos III-Netzwerkes vorgesehen ist. Er stellt das Bindeglied zum Inline-Installationssystem dar.

Rexroth Inline-Buskoppler für Profibus DP mit digitalen Ein- und Ausgängen R-IL PB BK DI8 DO4-PAC

DOK-CONTRL-ILPBBKDI8DO-AWRS-DE-P, R911170475

Diese Dokumentation beschreibt einen Buskoppler, der zum Einsatz innerhalb eines Profibus DP-Netzwerkes vorgesehen ist. Er stellt das Bindeglied zum Inline-Installationssystem dar.

Rexroth Inline-Buskoppler für Profinet IO mit digitalen Ein- und Ausgängen R-IL PN BK DI8 DO4-PAC

DOK-CONTRL-ILPNBKDIO**-DARS-DE-P, R911328681

Diese Dokumentation beschreibt einen Buskoppler, der zum Einsatz innerhalb eines Profinet IO-Netzwerkes vorgesehen ist. Er stellt das Bindeglied zum Inline-Installationssystem dar.

Rexroth Inline-Buskoppler für Ethernet IP mit digitalen Ein- und Ausgängen R-IL ETH BK DI8 DO4 2TX-PAC

DOK-CONTRL-ILETHBKDIO*-DARS-DE-P, R911328678

Diese Dokumentation beschreibt einen Buskoppler, der zum Einsatz innerhalb eines Ethernet IP-Netzwerkes vorgesehen ist. Er stellt das Bindeglied zum Inline-Installationssystem dar.

1.6 Erforderliche und ergänzende Dokumentationen MLC

Dokumentationstitel mit Typenschlüssel und Materialnummer

Systemübersicht
MLC 15VRS Systemübersicht DOK-MLC***-SYSTEM**V15-PRRS-DE-P, R911394935 Diese Dokumentation stellt eine Übersicht der möglichen Hardware-/Softwarekomponenten des Automationssystems MLC in der genannten Version dar. Die Dokumentation dient als Auswahlhilfe zur Zusammenstellung eines Systems.

Tab. 1-3: MLC-Dokumentationsübersicht - Systemübersicht

Erste Schritte
IndraMotion MLC 14VRS Erste Schritte DOK-MLC***-F*STEP**V14-CORS-DE-P, R911341516 Diese Dokumentation beschreibt die "ersten Schritte" mit der MLC und RobotControl. Beschrieben werden die Hard- und Softwarevoraussetzungen und das Erstellen eines Projekts.
MLC 15VRS Projektkonvertierung DOK-MLC***-PROCONV*V15-APRS-DE-P, R911396594 Diese Dokumentation beschreibt die Projektkonvertierung des Systems MLC.

Tab. 1-4: MLC-Dokumentationsübersicht - Erste Schritte

Engineering
IndraWorks 15VRS Software-Installation DOK-IWORKS-SOFTINS*V15-CORS-DE-P, R911393449 Die Dokumentation beschreibt den Installationsprozess von IndraWorks.
IndraWorks 15VRS Engineering DOK-IWORKS-ENGINEE*V15-APRS-DE-P, R911393302 Diese Dokumentation beschreibt die Verwendung von IndraWorks, in das sich die Rexroth Engineering-Tools integrieren. Beschrieben werden das Arbeiten mit IndraWorks und die Bedienung der Oszilloskop-Funktion.
MLC 15VRS Funktionsbeschreibung DOK-MLC***-FUNC****V15-APRS-DE-P, R911393304 Die Dokumentation dient der Beschreibung der Wizards, Kontextmenüs, Dialoge, Steuerungsinbetriebnahme, Gerätekonfiguration, Funktionalitäten des Systems MLC.

Tab. 1-5: MLC-Dokumentationsübersicht - Engineering

Diagnose und Service
MLC 15VRS Diagnosen DOK-XLCMLC-DIAG****V15-RERS-DE-P, R911393323 Diese Dokumentation enthält alle im Steuerungssystem MLC implementierten Steuerungsdiagnosen.
MLC 15VRS Parameter DOK-XLCMLC-PARAM***V15-RERS-DE-P, R911393321 Diese Dokumentation beschreibt die Parameter des Systems MLC, sowie das Wechselspiel zwischen Parametrieren und Programmieren. Beschrieben werden die Achsparameter, die Steuerungsparameter, die Kinematik-Parameter, die Mess-taster- und die Nockenschaltwerksparameter.
WebAssistant DOK-GENERL-WEB*ASSIST*-APRS-DE-P, R911381468 Diese Dokumentation beschreibt den WebAssistant. Der WebAssistant ist ein webbasiertes Diagnosewerkzeug, das zum Zugriff auf ein Steuerungssystem über eine Ethernet-Hochgeschwindigkeitsverbindung dient. Der WebAssistant ermöglicht OEMs, Endanwendern und Kundenbetreuern von überall auf ein System zuzugreifen und Ferndiagnosen zu stellen. Voraussetzung ist ein PC mit Internet Explorer Version 11 oder neuer, Firefox Version 45 oder neuer oder Google Chrome Version 54 oder neuer. Die folgenden Steuerungsvarianten werden unterstützt: • MLC XM21/XM22/XM42/VPx

Tab. 1-6: MLC-Dokumentationsübersicht - Diagnose und Service

Firmware
Rexroth IndraDrive MPx-21, Funktionen ¹⁾ DOK-INDRV*-MP*-21VRS**-APRS-DE-P, R911385759 Diese Dokumentation beschreibt alle funktionellen Eigenschaften der IndraDrive-Firmware in den Varianten MPB-20, MPM-20, MPC-20 und MPE-20.
IndraDrive MPx-16 bis MPx-21 und PSB, Parameter DOK-INDRV*-GEN1-PARA**-RERS-DE-P, R911328650 Diese Dokumentation enthält die Beschreibung aller, in der Firmware für Antriebsregelgeräte der IndraDrive-Familie implementierten Parameter. Sie unterstützt die Parametrierung der Antriebsregelgeräte.

Rexroth IndraDrive MPx-16 bis MPx-21 und PSB, Diagnosen

DOK-INDRV*-GEN1-DIAG**-RERS-DE-P, R911326539

Diese Dokumentation enthält die Beschreibung aller in den folgenden Firmwares implementierten Diagnosen:

- Antriebsregelgerät-Firmwares MPx-16 bis MPx-21
- Firmwares PSB der Versorgungsgeräte des Typs "HMU05" und "KMOV03"
- Firmwares der Versorgungsgeräte des Typs "HMOV01".

Sie unterstützt den Maschinenbediener und den Anlagenprogrammierer bei der Fehlerbehebung.

Hardware**Rexroth IndraDrive Steuerteile CSB02, CSE02, CSH02, CDB02**

DOK-INDRV*-Cxx02*****-PRRS-DE-P, R911338961

Rexroth IndraDrive Cs Antriebssysteme mit HCS01

DOK-INDRV*-HCS01*****-PRRS-DE-P, R911322209

Rexroth IndraDrive Mi Antriebssysteme mit KCU02, KSM02, KMS02/03, KMOV03

DOK-INDRV*-KCU02+KSM02-PRxx-DE-P, R911335702

Rexroth IndraDrive ML Antriebssysteme mit HMU05

DOK-INDRV*-Hxx05*****-PRxx-DE-P, R911344278

1) in Vorbereitung
 Tab. 1-7: Übersicht der Dokumentationen für Antriebsregelgeräte

SPS**IndraWorks 15VRS SPS-Programmiersystem IndraLogic 2G**

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.

IndraWorks 15VRS Basis-Bibliotheken IndraLogic 2G

DOK-IL*2G*-BASLIB**V15-LIRS-DE-P, R911398632

Das vorliegende Handbuch beschreibt die systemübergreifenden SPS-Bibliotheken.

MLC 15VRS PLCopen-Bibliotheken

DOK-MLC***-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.

Tab. 1-8: MLC-Dokumentationsübersicht - SPS

Feldbusse
IndraWorks 15VRS Feldbusse DOK-IWORKS-FB*****V15-APRS-DE-P, R911393282 Die Dokumentation dient der Beschreibung der von den Systemen MLC und MTX unterstützten Feldbus- und lokalen Peripherieanschlüssen. Der Schwerpunkt dieser Dokumentation liegt in der Konfiguration, Parametrierung, Inbetriebnahme und Diagnose der verschiedenen Peripherieanschlüssen. Sie ist die Grundlage für die Online-Hilfe.
IndraWorks 15VRS Feldbus-Bibliotheken DOK-IWORKS-FB*LIB**V15-LIRS-DE-P, R911393274 Das vorliegende Handbuch beschreibt die Bibliotheken der Feldbusse: RIL_ProfibusDP_02, RIL_ProfibusDP Slave, RIL_ProfinetIO, RIL_ProfinetIODevice, RIL_EtherNetIPAdapter, RIL_MappingList, RIL_SERCOSIII, RIH_IndraControl_S20 einschließlich deren Diagnosen, sowie die Fehlerreaktion der Bausteine.
Sercos Systemhandbuch für I/O-Geräte DOK-CONTRL-ILS3*****-APxx-DE-P, R911333511 Diese Dokumentation beschreibt die Projektierung, Parametrierung, Inbetriebnahme und Diagnose von I/O-Geräten mit Sercos Schnittstelle.

xx entsprechender Ausgabestand
Tab. 1-9: MLC-Dokumentationsübersicht - Feldbusse

HMI
IndraWorks 14VRS HMI DOK-IWORKS-HMI*****V14-APRS-DE-P, R911343567 Diese Dokumentation beschreibt die Funktionen, die Konfiguration und die Bedienung der Bedienoberflächen IndraWorks HMI Engineering und IndraWorks HMI Operation.
IndraWorks 14VRS WinStudio 7.4 DOK-IWORKS-WINSTUD*V14-APRS-DE-P, R911341584 Dieses "Benutzerhandbuch und Technische Referenz" soll Sie dabei unterstützen, mit Ihrer Software "Rexroth WinStudio" optimale Ergebnisse zu erzielen. Das Dokument bietet technische Informationen und Schritt-für-Schritt-Anleitungen zum Anlegen von webfähigen HMI/SCADA-Programmen.
Rexroth IndraLogic XLC IndraMotion MLC 14VRS HMI-Anbindung DOK-XLCMLC-HMI*****V14-APRS-DE-P, R911341496 Dieses Dokument beschreibt die von der XLC und MLC unterstützten Visualisierungssysteme sowie deren Anbindung.

Tab. 1-10: MLC-Dokumentationsübersicht - HMI

Technologie
MLC 15VRS Generic Application Template DOK-MLC**-TF*GAT**V15-APRS-DE-P, R911400165 Dieses Handbuch stellt dem IndraLogic SPS-Programmierer ein strukturiertes Template, welches zum Hinzufügen und Bearbeiten des SPS-Programmiercodes verwendet werden kann, zur Verfügung. Beschrieben werden das Template selbst, der Template Wizard und Beispielapplikationen.
MLC 15VRS Technologie-Bibliotheken DOK-MLC***-TF*LIB**V15-LIRS-DE-P, R911396133 Diese Dokumentation beschreibt die Funktionsbausteine, Funktionen und Datentypen der Bibliotheken "ML_TechInterface.library", "ML_TechMotion.library", "RMB_TechCam.library" und "ML_TechBase.library" sowie Bibliotheken zur Wicklerfunktionalität, Registerreglerfunktionalität und Querschneiderfunktionalität.

Technologie
IndraWorks 14VRS Energy Efficiency Management DOK-IWORKS-4EE****V14-APRS-DE-P, R911339228 Diese Dokumentation beschreibt die HMI-Anbindung des Energiemanagement-Systems und die Funktionsbausteine, Funktionen und Datentypen der Bibliothek "RIL_4EE ". Die Dokumentation enthält ferner die Fehlerreaktion der Bausteine.
Rexroth IndraMotion MLC 14VRS RegisterControl (Bibliothek) DOK-MLC***-REGI*CO*V14-LIRS-DE-P, R911341508 Diese Dokumentation beschreibt die Ein- und Ausgänge der einzelnen Funktionsbausteine und gibt Hinweise für deren Verwendung.
IndraMotion MLC 14VRS RegisterControl (Anwendungsbeschreibung) DOK-MLC***-REGI*CO*V14-APRS-DE-P, R911341506 Dieses Dokument beschreibt die Anwendung der integrierten Registerregelung für eine Tiefdruckmaschine. Beschrieben werden die Komponenten des Markenstromsensors, die HMI-Anwendung und die Fehlerbehebung. Diese Anleitung dient dazu, die Registerregelung zu bedienen und auf Fehler zu reagieren sowie Diagnosen abzufragen. Die Zielgruppe für diese Dokumentation sind Maschineneinrichter und Maschinenbediener.
Rexroth IndraMotion MLC 14VRS Anwendung Wicklerfunktionen DOK-MLC***-TF*WIND*V14-APRS-DE-P, R911341512 Diese anwendungsbezogene Systemdokumentation beschreibt die Anwendung der Wickler-Technologiefunktionen.
Rexroth IndraWorks 14VRS CamBuilder DOK-IWORKS-CAMBUIL*V14-APRS-DE-P, R911387567 Diese Dokumentation dient zur Beschreibung der Grundlagen und der Bedienung des CamBuilder, des Werkzeugs zur Kurvenscheibenbearbeitung.
IndraMotion MLC 15VRS Robot-Control V2 DOK-MLC***-ROCO****V15-RERS-DE-P, R911396138 Die Dokumentation informiert über die Robot-Control V2. Die Programmierung über PLCopen steht hierbei im Vordergrund. Beschrieben werden die Programmstruktur, Variablen, Funktionen, Bewegungsanweisungen sowie die notwendigen Parameter des Systems. Die Dokumentation enthält ferner die Beschreibung der Bibliothek "ML_Robot". Robot-Control V2 wird unterstützt durch ein Kinematik-Interface. Die Komponenten dieses Interfaces werden in der Bibliothek "ML_KinTech" beschrieben.

Tab. 1-11: MLC-Dokumentationsübersicht - Technologie

Open Core Engineering - OCE
IndraLogic XLC IndraMotion MLC 14VRS Erste Schritte MLPI (Onlinehilfe) DOK-XLCMLC-MLPI****V14-CORS-EN-E, R911341033 Dieses Dokument beschreibt die Installation und Inbetriebnahme der MLPI-Schnittstelle.
IndraLogic XLC IndraMotion MLC 14VRS Automation Interface (Onlinehilfe) DOK-XLCMLC-AUT*INT*V14-APRS-EN-E, R911341499 Diese Dokumentation beschreibt den script-basierten Zugriff auf IndraWorks-Projektdaten über die Schnittstelle des Automation Interfaces.

Tab. 1-12: MLC-Dokumentationsübersicht - Open Core Engineering - OCE

SafeLogic
IndraWorks SafeLogic 14VRS Die ersten Schritte DOK-IWORKS-SL*STEP*V14-CORS-DE-P, R911341519 Dieses Dokument beschreibt die Erstinbetriebnahme des Safety-Funktionsmoduls und dessen externer Safety-Peripherie. Am Beispiel eines Projektes werden alle notwendigen Schritte zur Inbetriebnahme einer einfachen Safety-Anwendung durchgeführt.
IndraLogic XLC IndraMotion MLC 14VRS SafeLogic Systemübersicht DOK-XLCMLC-SL**SYS*V14-PRRS-DE-P, R911341695 Dieses Dokument beschreibt die Projektierung, die Erstellung und die Inbetriebnahme von sicherheitstechnischen Vorrichtungen.
IndraWorks 14VRS SafeLogic - Projektkonfiguration, Stand 1.2.0.0. DOK-IWORKS-SL**PRJ*V14-APRS-DE-P, R911341693 Dieses Dokument beschreibt die Projektierung von Safety-Anwendungen.
IndraControl CFL01.1-F1 Safety-Funktionsmodul DOK-CONTRL-SL**FM*****-ITRS-DE-P, R911336575 Dieses Dokument liegt der Hardware bei und beschreibt die Inbetriebnahme, die Installation, die Außerbetriebnahme und Entsorgung der Hardware.

Tab. 1-13: MLC-Dokumentationsübersicht - SafeLogic

Hardware
IndraControl XM21, XM22 Steuerungen DOK-CONTRL-IC*XM2*****-ITRS-DE-P, R911340666 Diese Dokumentation dient der Beschreibung der Steuerungen IndraControl XM21/XM22.
IndraControl XM42 Steuerungen DOK-CONTRL-XM4X*CTRL**-ITRS-DE-P, R911345565 Diese Dokumentation dient der Beschreibung der Steuerungen IndraControl XM42.
IndraControl XFE 01.1 Erweiterungsmodule Profibus, RT-Ethernet, Sercos, CAN DOK-CONTRL-XFE**EXTMOD-ITRS-DE-P, R911345569 Diese Dokumentation dient der Beschreibung aller Erweiterungsmodule der XM-Steuerungen.

xx entsprechender Ausgabestand

Tab. 1-14: MLC-Dokumentationsübersicht - Hardware

Hydraulik
IndraMotion MLC 14VRS Hydraulische Funktionen DOK-MLC***-TF*HMOT*V14-LIRS-DE-P, R911341905 Diese Dokumentation beschreibt die Funktionsbausteine, Funktionen und Datentypen der Bibliotheken MH_TechHydrBase und MH_TechHydrMotion. Die Dokumentation enthält ferner die Fehlerreaktion der Bausteine.
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Sequentielle Programmierung DOK-XLCMLC-SEQPROG*V14-LIRS-DE-P, R911341908 Diese Dokumentation beschreibt die Möglichkeiten der Sequentiellen Programmierung, auch in Verbindung mit GAT, und die Bibliothek MH_HydrControlProg.

Tab. 1-15: MLC-Dokumentationsübersicht - Hydraulik

1.7 Erforderliche und ergänzende Dokumentationen MTX

1.7.1 Auswählen/Zusammenstellen

Dokumentationstitel mit Typenschlüssel und Materialnummer

MTX 15VRS Systembeschreibung DOK-MTX***-SYS*DES*V15-PRRS-DE-P, R911397741 Diese Dokumentation beschreibt das Steuerungssystem MTX. Beschrieben werden die Ausführungen, Technischen Daten, Schnittstellen und die Konfiguration der Steuerungsbaugruppen.
MTX 14VRS SafeLogic Systemübersicht DOK-MTX***-SL**SYS*V14-PRRS-DE-P, R911341697 Diese Dokumentation beschreibt den Einsatz der Sicherheitssteuerung SafeLogic im System MTX.
IndraControl XM42 Steuerungen DOK-CONTRL-XM4X*CTRL**-ITRS-DE-P, R911345565 Diese Dokumentation dient der Beschreibung der Steuerungen IndraControl XM42.

RS entsprechender Ausgabestand
Tab. 1-16: *MTX-Dokumentationsübersicht - Auswählen/Zusammenstellen*

1.7.2 Konfigurieren

Dokumentationstitel mit Typenschlüssel und Materialnummer

MTX 15VRS Maschinenparameter DOK-MTX***-MA*PAR**V15-RERS-DE-P, R911400175 Diese Dokumentation beschreibt den Aufbau und die Anpassung der verfügbaren Parameter der MTX. Ebenfalls beschrieben werden Funktion und Bedienung des NC-Konfigurators.
MTX 15VRS SPS-Interface DOK-MTX***-PLC*INT*V15-PRRS-DE-P, R911400171 Diese Dokumentation beschreibt die Interface-Signale und die Programmbausteine für die integrierte SPS.
MTX 15VRS Funktionsbeschreibung - Grundlagen DOK-MTX***-NC*F*BA*V15-RERS-DE-P, R911401400 Diese Dokumentation beschreibt die Basisfunktionen der MTX. Erläutert werden grundlegende Inbetriebnahmeschritte und die Funktionen der Steuerung als Beschreibung und Handlungsanweisung.
MTX 15VRS Funktionsbeschreibung - Erweiterung DOK-MTX***-NC*F*EX*V15-RERS-DE-P, R911393315 Diese Dokumentation beschreibt die erweiterten Funktionen der MTX. Erläutert werden grundlegende Inbetriebnahmeschritte und die Funktionen der Steuerung als Beschreibung und Handlungsanweisung.
MTX 15VRS Funktionsbeschreibung - Sonderfunktionen DOK-MTX***-NC*F*SP*V15-RERS-DE-P, R911393308 Diese Dokumentation beschreibt die Sonderfunktionen der MTX. Erläutert werden grundlegende Inbetriebnahmeschritte und die Funktionen der Steuerung als Beschreibung und Handlungsanweisung.

MTX Freiformflächenbearbeitung

DOK-MTX***-FREEFORM***-APRS-DE-P, R911341434

Diese Dokumentation beschreibt die Vorgehensweise bei der Freiformflächenbearbeitung mit der MTX-Steuerung. CNC-Programme, die von einem CAD/CAM-System generiert werden, dienen als Grundlage für den gesamten Bearbeitungsprozess. Es wird ein Überblick über folgende Themen gegeben: Beschreibung der MTX- bzw. CNC-Funktionen zur Freiformflächenbearbeitung, Vorgehensweise bei der NC-Parametrierung, Rahmenbedingungen bei der Erzeugung von CNC-Programmen mit CAM-Software.

MTX Konvertierung von MTX-Projekten

DOK-MTX***-PROCONV****-PRRS-DE-P, R911342483

Diese Dokumentation unterstützt bei der Konvertierung von MTX 1.x Projekten nach MTX 2G. Es wird die Vorgehensweise beschrieben und auf mögliche Schwierigkeiten bei der Konvertierung hingewiesen.

RS

entsprechender Ausgabestand

Tab. 1-17:

MTX-Dokumentationsübersicht - Konfigurieren

1.7.3 In Betrieb nehmen

Dokumentationstitel mit Typenschlüssel und Materialnummer

MTX 15VRS Inbetriebnahme

DOK-MTX***-STARTUP*V15-CORS-DE-P, R911393280

Diese Dokumentation beschreibt die Inbetriebnahme der Steuerung MTX. Neben einer Gesamtübersicht werden die Inbetriebnahme und Konfiguration der Achsen und Bedienoberfläche und der SPS-Daten erläutert.

IndraWorks 15VRS Basis-Bibliotheken IndraLogic 2G

DOK-IL*2G*-BASLIB**V15-LIRS-DE-P, R911398632

Das vorliegende Handbuch beschreibt die systemübergreifenden SPS-Bibliotheken.

IndraWorks 15VRS Feldbusse

DOK-IWORKS-FB*****V15-APRS-DE-P, R911393282

Die Dokumentation dient der Beschreibung der von den Systemen MLC und MTX unterstützten Feldbus- und lokalen Peripherieanschlüssen. Der Schwerpunkt dieser Dokumentation liegt in der Konfiguration, Parametrierung, Inbetriebnahme und Diagnose der verschiedenen Peripherieanschlüssen. Sie ist die Grundlage für die Online-Hilfe.

IndraWorks 14VRS WinStudio 7.4

DOK-IWORKS-WINSTUD*V14-APRS-DE-P, R911341584

Dieses "Benutzerhandbuch und Technische Referenz" soll Sie dabei unterstützen, mit Ihrer Software "Rexroth WinStudio"™ optimale Ergebnisse zu erzielen. Das Dokument bietet technische Informationen und Schritt-für-Schritt-Anleitungen zum Anlegen von webfähigen HMI/SCADA-Programmen.

IndraWorks 15VRS Software-Installation

DOK-IWORKS-SOFTINS*V15-CORS-DE-P, R911393449

Die Dokumentation beschreibt den Installationsprozess von IndraWorks.

IndraWorks 15VRS Engineering

DOK-IWORKS-ENGINEE*V15-APRS-DE-P, R911393302

Diese Dokumentation beschreibt die Verwendung von IndraWorks, in das sich die Rexroth Engineering-Tools integrieren. Beschrieben werden das Arbeiten mit IndraWorks und die Bedienung der Oszilloskop-Funktion.

IndraWorks 15VRS SPS-Programmiersystem IndraLogic 2G

DOK-IWORKS-IL2GPRO*V15-APRS-DE-P, R911396134

Diese Dokumentation beschreibt das SPS-Programmierwerkzeug IndraLogic 2G und seine Verwendung. Beschrieben werden die Basisverwendung, Erste Schritte, die Visualisierung, Menüpunkte und die Editoren.

Zu dieser Dokumentation

IndraWorks 15VRS HMI

DOK-MLC***-HMI*****V15-APRS-DE-P, R911399269

Diese Dokumentation beschreibt die Funktionen, die Konfiguration und die Bedienung der Bedienoberflächen IndraWorks HMI Engineering und IndraWorks HMI Operation.

MTX 15VRS Technologie Strahlschneiden

DOK-MTX***-TECHCUT*V15-CORS-DE-P, R911399148

Diese Dokumentation beschreibt die Technologie Strahlschneiden, über die Maschinen mittels eines NC-geführten Bearbeitungskopfs Materialien schneiden können.

RS

entsprechender Ausgabestand

Tab. 1-18:

MTX-Dokumentationsübersicht - In Betrieb nehmen

1.7.4 Bedienen

Dokumentationstitel mit Typenschlüssel und Materialnummer

MTX 12VRS Satzvorlauf

DOK-MTX***-BLK*RUN*V12-APRS-DE-P, R911334378

Die Dokumentation soll den Maschinenhersteller in die Lage versetzen, die Funktion "Satzvorlauf" an seiner Maschine für den Endanwender einzurichten.

MTX 15VRS Programmierhandbuch

DOK-MTX***-NC**PRO*V15-RERS-DE-P, R911393317

Das vorliegende Handbuch beschreibt die Standard-Programmierung der Steuerung MTX. Beschrieben werden neben den Grundlagen der NC-Programmierung sowohl die Anwendung der NC-Funktionen nach DIN 66025, als auch der NC-Funktionen mit Hochsprachensyntax und CPL-Funktionen.

MTX 15VRS Standard-NC-Bedienung

DOK-MTX***-NC*OP***V15-APRS-DE-P, R911393312

Diese Dokumentation beschreibt die Bedienung der Standard-Bedienoberfläche der NC-Steuerung MTX. Beschrieben werden die Bedienung der Oberfläche, die NC-Programmerstellung und die Werkzeugverwaltung.

MTX 15VRS Multitouch

DOK-MTX***-MULTI***V15-APRS-DE-P, R911393310

In dieser Dokumentation wird die Bedienung der Multitouch-Bedienoberfläche der NC-Steuerung MTX beschrieben.

MTX 15VRS Standard-NC-Zyklen

DOK-MTX***-NC*CYC**V15-PRRS-DE-P, R911394939

Diese Dokumentation beschreibt für die Steuerung MTX die Anwendung der Standard-Zyklen der verschiedenen Technologien.

MTX 15VRS NC-Simulation

DOK-MTX***-NC*SIM**V15-APRS-DE-P, R911393272

Diese Dokumentation beschreibt für die Steuerung MTX die NC-Simulation.

MTX 15VRS Messfunktionen

DOK-MTX***-MES*FUN*V15-APRS-DE-P, R91194937

Diese Dokumentation beschreibt die Messzyklen der Steuerung MTX.

RS

entsprechender Ausgabestand

Tab. 1-19:

MTX-Dokumentationsübersicht - Bedienen

1.7.5 OEM-Entwicklung

Dokumentationstitel mit Typenschlüssel und Materialnummer

MTX 13VRS Automation Interface DOK-MTX***-AUT*INT*V13-APRS-DE-P, R911337273 Diese Dokumentation beschreibt den scriptbasierten Zugriff auf IndraWorks-Projektdaten über die Schnittstelle des Automation Interfaces. Beschrieben werden die verschiedenen Objekte, einschließlich Code-Beispielen. Eine Beschreibung des Automation Builders schließt sich an.
MTX 15VRS OPC-Kommunikation DOK-MTX***-OPC*COM*V15-PRRS-DE-P, R911399271 Diese Dokumentation beschreibt die Syntax und den Aufbau der Items für die Kommunikation mit den Bosch Rexroth-Geräten.
IndraWorks OPC UA Kommunikation DOK-IWORKS-OPC*UA*****-APRS-DE-P, R911379308 Diese Dokumentation beschreibt die OPC UA Kommunikation der Steuerungssysteme MLC und MTX.

RS entsprechender Ausgabestand
Tab. 1-20: MTX-Dokumentationsübersicht - OEM-Entwicklung

1.7.6 AddOns

Dokumentationstitel mit Typenschlüssel und Materialnummer

MTX 11VRS Aktionsrekorder DOK-MTX***-ACR*****V11-APxx-DE-P, R911329942 Diese Dokumentation beschreibt den MTX Aktionsrekorder. Beschrieben werden sowohl die Installation und die Inbetriebnahme, einschließlich Schnittstellensignale, als auch die Anwendung und die Bedienung.
MTX 15VRS Efficiency Workbench MTX cta, MTX ega DOK-MTX***-EWB*****V15-APRS-DE-P, R911400177 Diese Dokumentation beschreibt die Funktionsweise und den Einsatzbereich der Analysetools MTX cta und MTX ega.
MTX Remote Condition Monitoring DOK-MTX***-RCM*****V01-APRS-DE-P, R911334382 Diese Dokumentation beschreibt die Bedienung des Remote Condition Monitoring Systems.
MTX visIREC Anwenderdokumentation DOK-MTX***-VISIREC*V01-APRS-DE-P, R911344241 Diese Dokumentation beschreibt das Analysewerkzeug visIREC, das zur Optimierung des Freiformflächenbearbeitungsprozesses eingesetzt werden kann. 2D- bzw. 4D-Anzeige der bahnbezogenen Daten. 2D- bzw. 4D-Anzeige der koordinatenbezogenen Daten. Analyse von kritischen Bereichen (Bahn- und Orientierungsabweichung). Vergleich zwischen den programmierten und den interpolierten NC-Sätzen.

xx / RS entsprechender Ausgabestand
Tab. 1-21: MTX-Dokumentationsübersicht - AddOns

1.8 Bezeichnungen und Abkürzungen

Begriff	Erklärung
EFC	Rexroth Frequenzumrichter
ENIP	Ethernet IP

PNIO	Profinet IO
DALI	Digital Addressable Lighting Interface ist ein Protokoll für die Licht- und Gebäudeautomation gemäß der technischen Norm IEC 62386

Tab. 1-22: Verwendete Bezeichnungen und Abkürzungen

1.9 Kundenfeedback

Anregungen, Wünsche oder Verbesserungen von unseren Kunden haben bei uns einen hohen Stellenwert. Senden Sie uns Ihre Anmerkungen zu den Dokumentationen per E-Mail an Feedback.Documentation@boschrexroth.de. Sie können direkt im elektronischen PDF-Dokument Kommentare einfügen und uns die PDF-Datei zusenden.

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

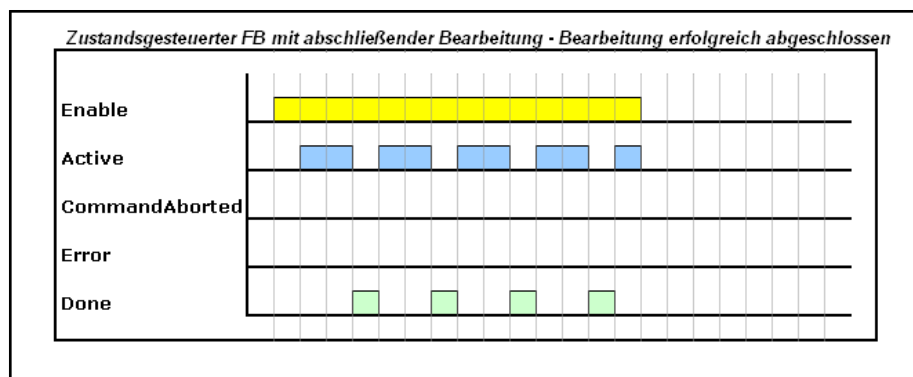


Abb. 2-1: I/O zustandsgesteuerter Funktionsbausteine, abschließende Bearbeitung ohne "Shutdown". Bearbeitung erfolgreich abgeschlossen

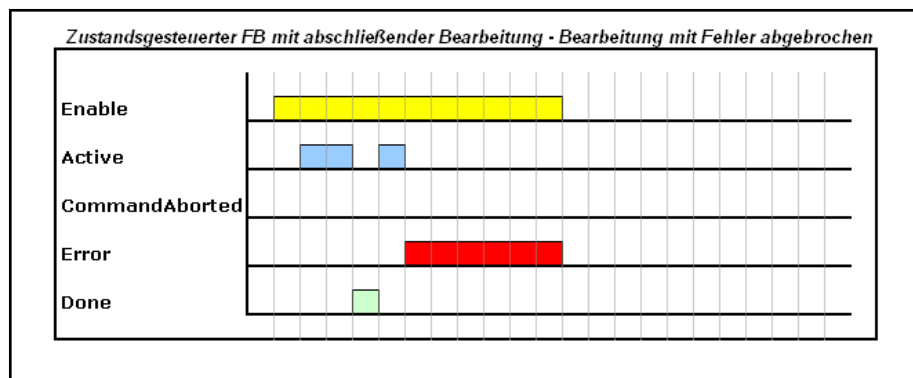


Abb. 2-2: I/O zustandsgesteuerter Funktionsbausteine, abschließende Bearbeitung ohne "Shutdown". Bearbeitung mit Fehler abgeschlossen

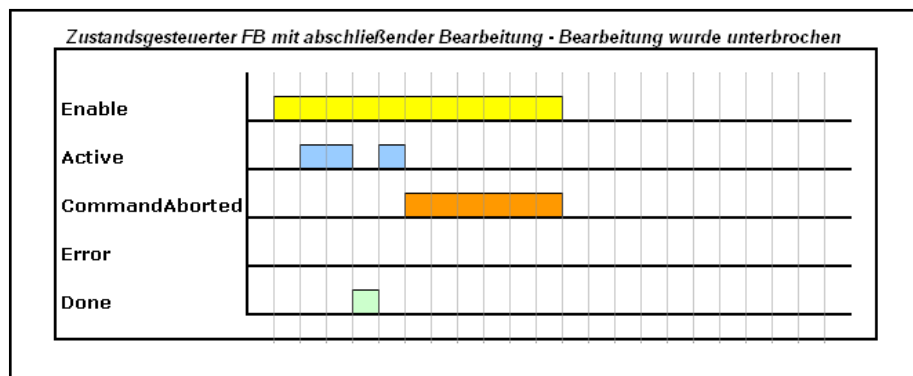


Abb. 2-3: I/O zustandsgesteuerter Funktionsbausteine, abschließende Bearbeitung ohne "Shutdown". Bearbeitung unterbrochen

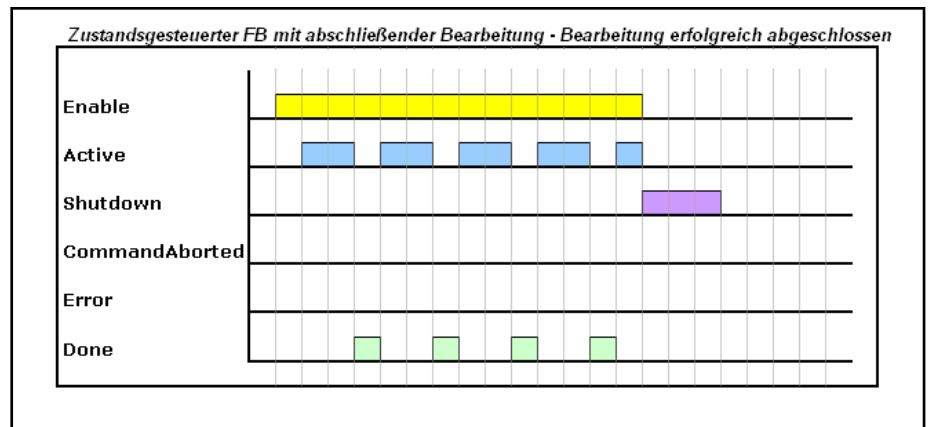


Abb. 2-4: I/O zustandsgesteuerter Funktionsbausteine, abschließende Bearbeitung mit "Shutdown". Bearbeitung erfolgreich abgeschlossen

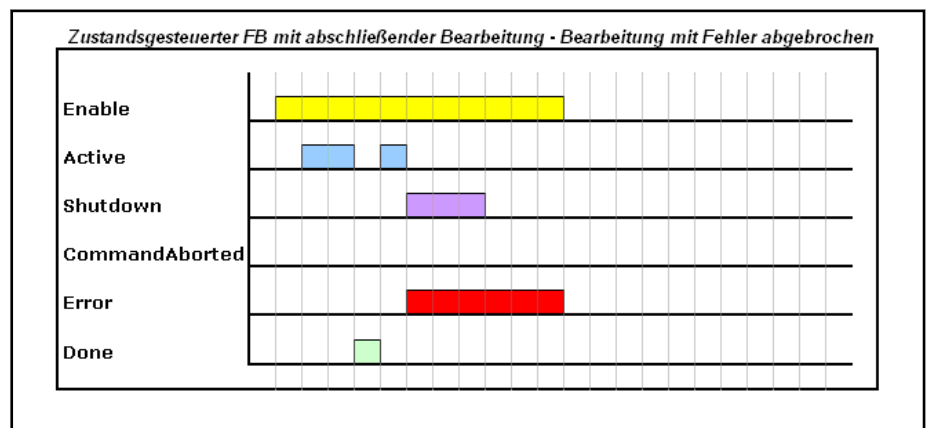


Abb. 2-5: I/O zustandsgesteuerter Funktionsbausteine, abschließende Bearbeitung mit "Shutdown". Bearbeitung mit Fehler abgeschlossen

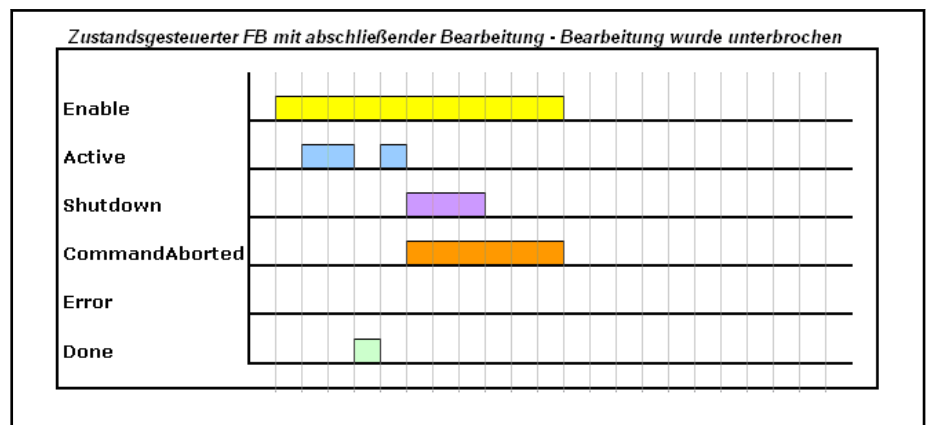


Abb. 2-6: I/O zustandsgesteuerter Funktionsbausteine, abschließende Bearbeitung mit "Shutdown". Bearbeitung wurde unterbrochen

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).

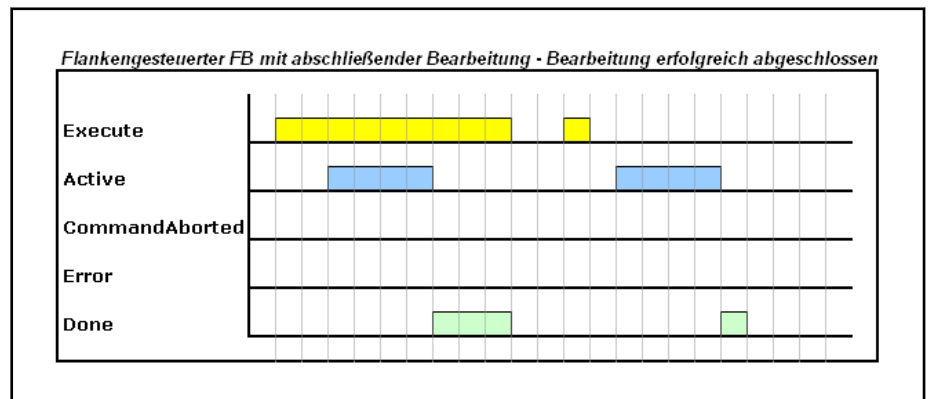


Abb. 2-7: I/O flankengesteuerter Funktionsbausteine, abschließende Bearbeitung ohne "Shutdown". Bearbeitung erfolgreich abgeschlossen

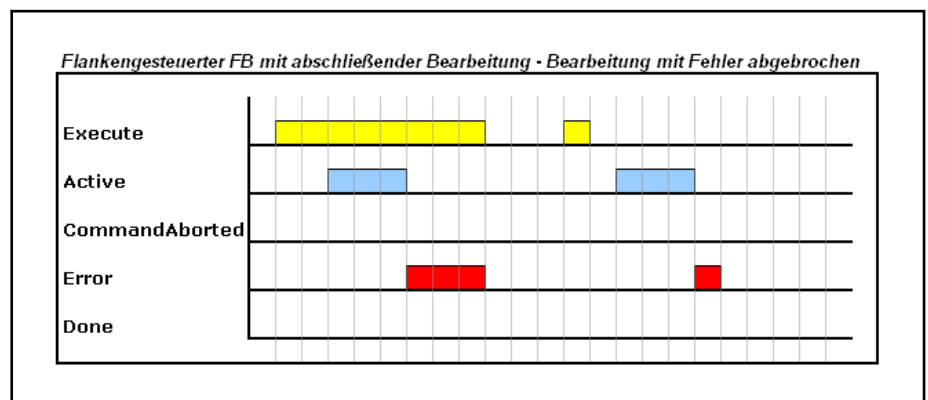


Abb. 2-8: I/O flankengesteuerter Funktionsbausteine, abschließende Bearbeitung ohne "Shutdown". Bearbeitung mit Fehler abgeschlossen

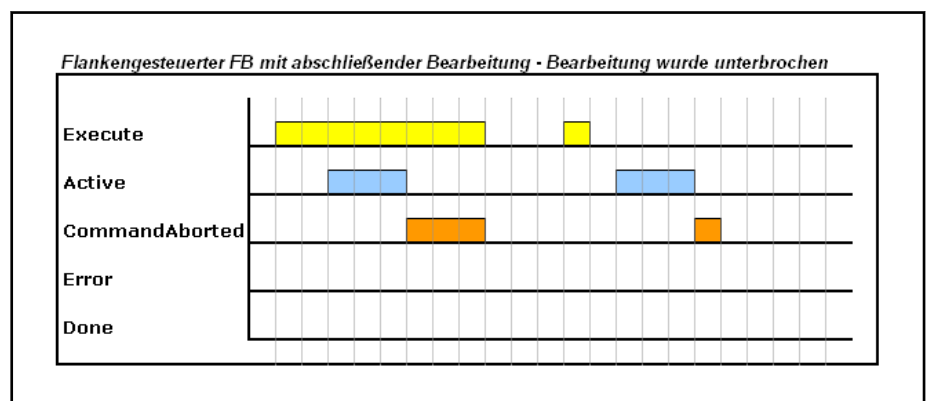


Abb. 2-9: I/O flankengesteuerter Funktionsbausteine, abschließende Bearbeitung ohne "Shutdown". Bearbeitung unterbrochen

Standardschnittstellen an Funktionsbausteinen

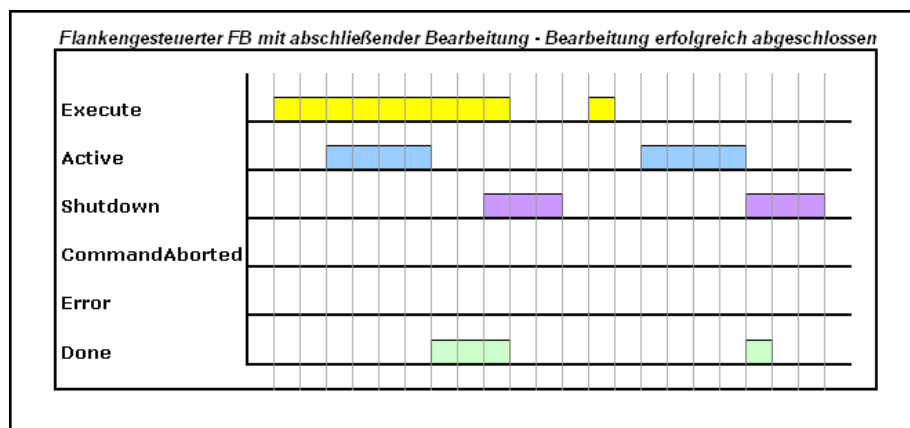


Abb. 2-10: I/O flankengesteuerter Funktionsbausteine, abschließende Bearbeitung mit "Shutdown". Bearbeitung erfolgreich abgeschlossen

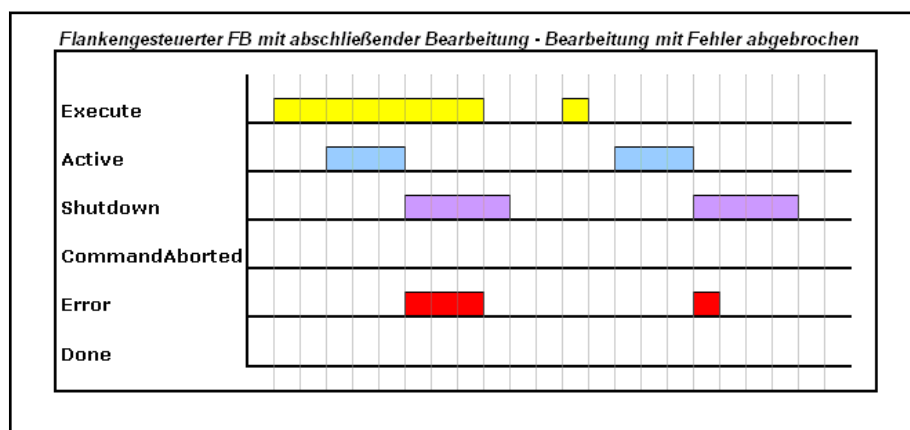


Abb. 2-11: I/O flankengesteuerter Funktionsbausteine, abschließende Bearbeitung mit "Shutdown". Bearbeitung mit Fehler abgeschlossen

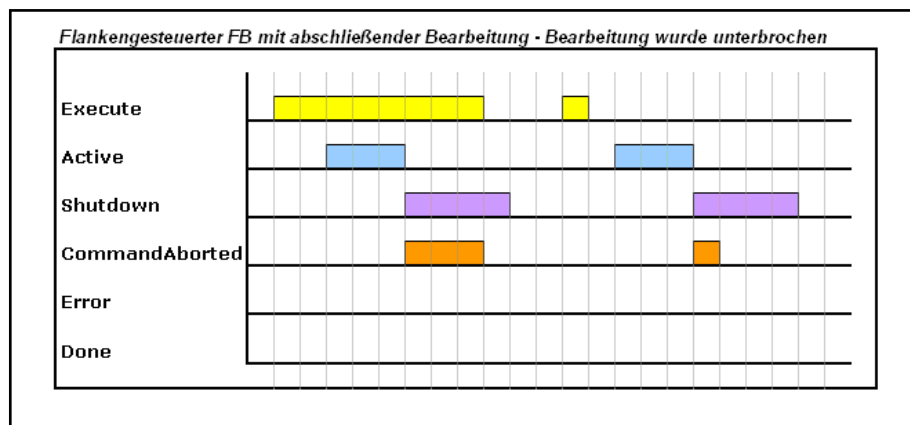


Abb. 2-12: I/O flankengesteuerter Funktionsbausteine, abschließende Bearbeitung mit "Shutdown". Bearbeitung unterbrochen

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 Variablenamen 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).

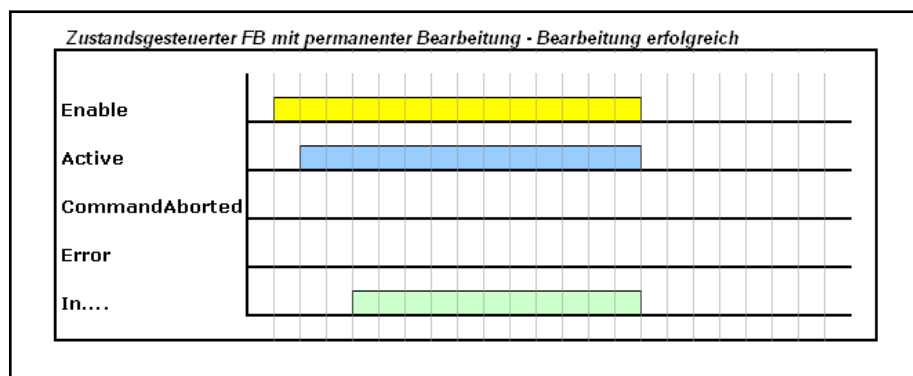


Abb. 2-13: I/O zustandsgesteuerter Funktionsbausteine, permanente Bearbeitung ohne "Shutdown". Bearbeitung erfolgreich

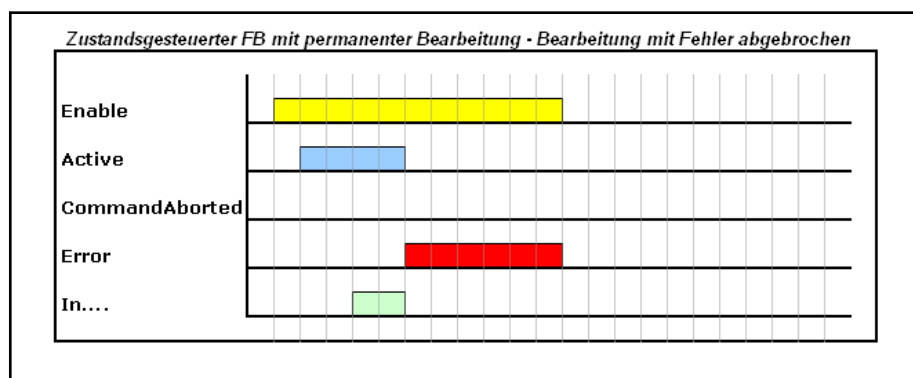


Abb. 2-14: I/O zustandsgesteuerter Funktionsbausteine, permanente Bearbeitung ohne "Shutdown". Bearbeitung mit Fehler abgebrochen

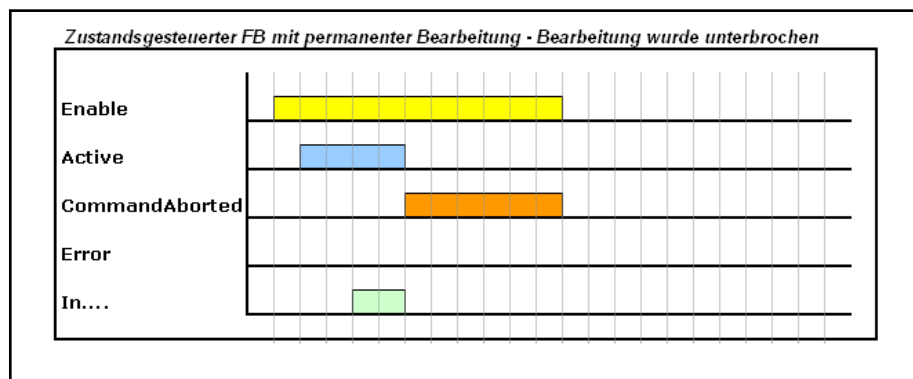


Abb. 2-15: I/O zustandsgesteuerter Funktionsbausteine, permanente Bearbeitung ohne "Shutdown". Bearbeitung unterbrochen

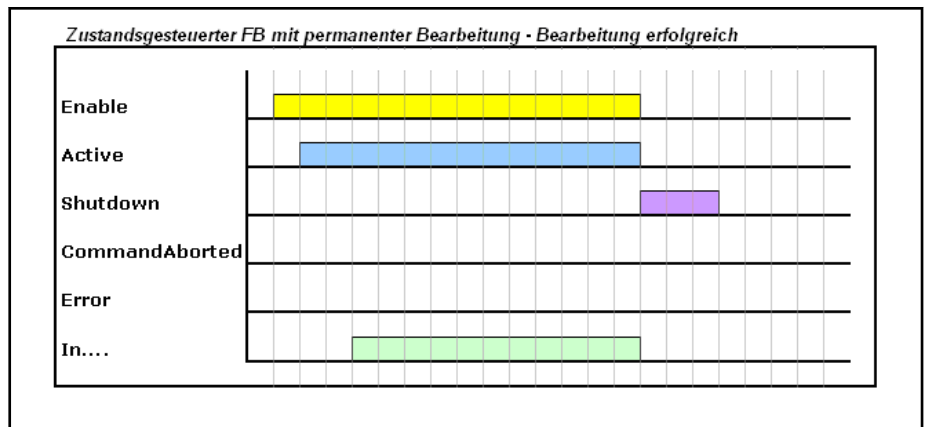


Abb. 2-16: I/O zustandsgesteuerter Funktionsbausteine, permanente Bearbeitung mit "Shutdown". Bearbeitung erfolgreich

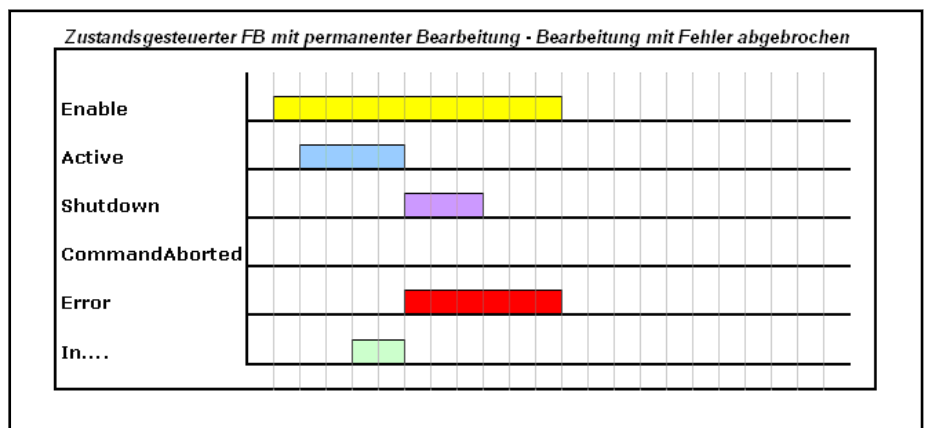


Abb. 2-17: I/O zustandsgesteuerter Funktionsbausteine, permanente Bearbeitung mit "Shutdown". Bearbeitung mit Fehler abgebrochen

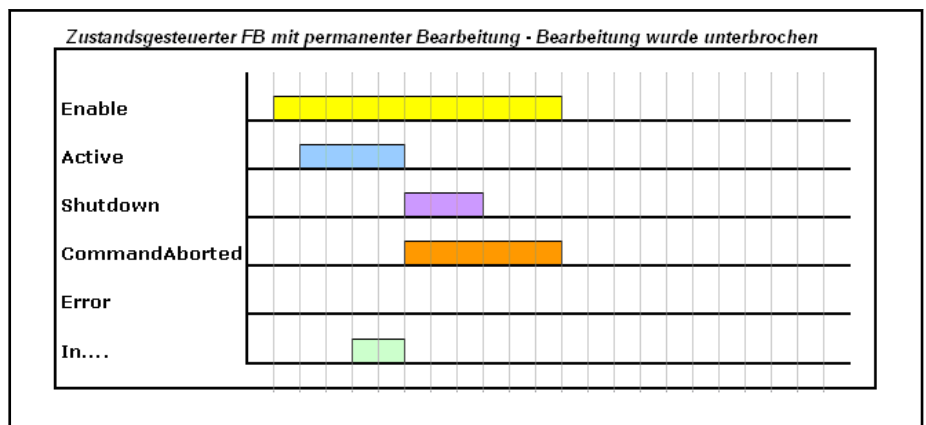


Abb. 2-18: I/O zustandsgesteuerter Funktionsbausteine, permanente Bearbeitung mit "Shutdown". Bearbeitung unterbrochen

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).

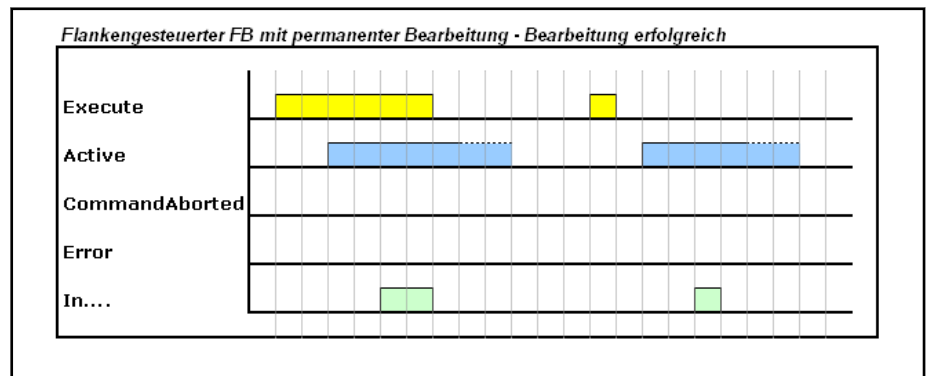


Abb. 2-19: I/O flankengesteuerter Funktionsbausteine, permanente Bearbeitung ohne "Shutdown". Bearbeitung erfolgreich

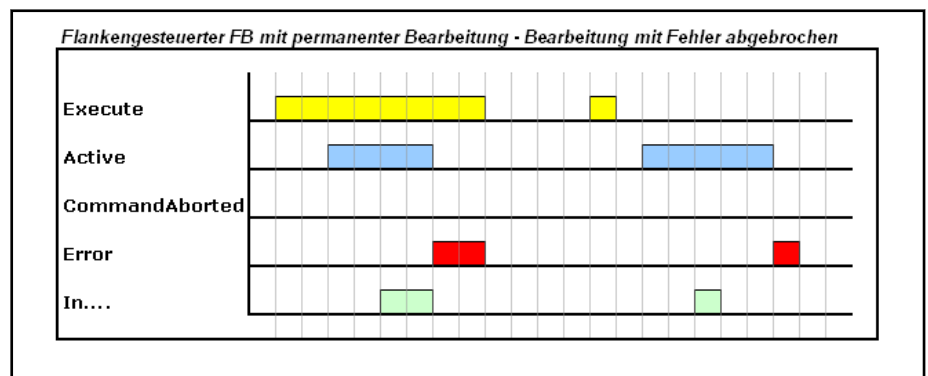


Abb. 2-20: I/O flankengesteuerter Funktionsbausteine, permanente Bearbeitung ohne "Shutdown". Bearbeitung mit Fehler abgebrochen

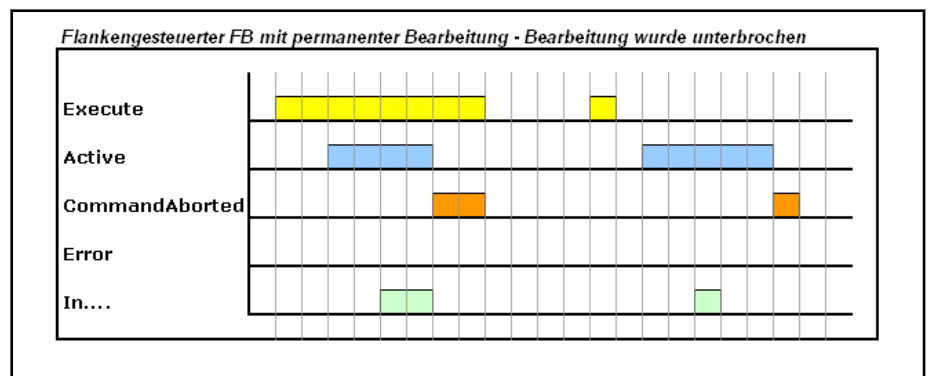


Abb. 2-21: I/O flankengesteuerter Funktionsbausteine, permanente Bearbeitung ohne "Shutdown". Bearbeitung unterbrochen

Standardschnittstellen an Funktionsbausteinen

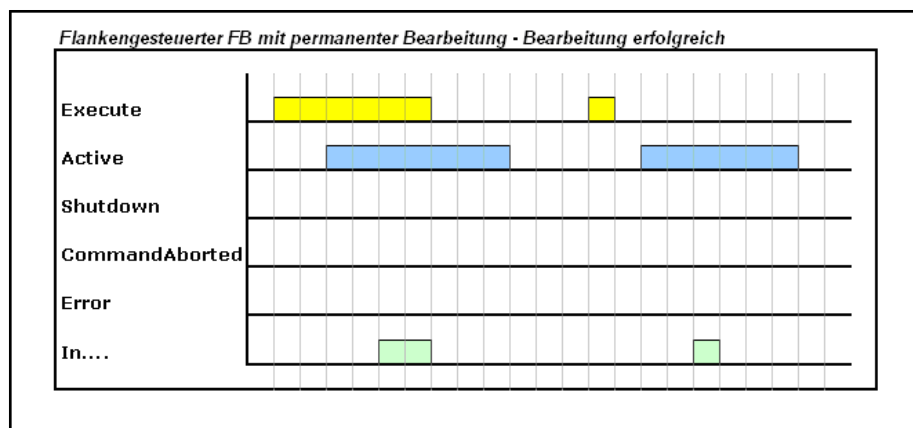


Abb. 2-22: I/O flankengesteuerter Funktionsbausteine, permanente Bearbeitung mit "Shutdown". Bearbeitung erfolgreich

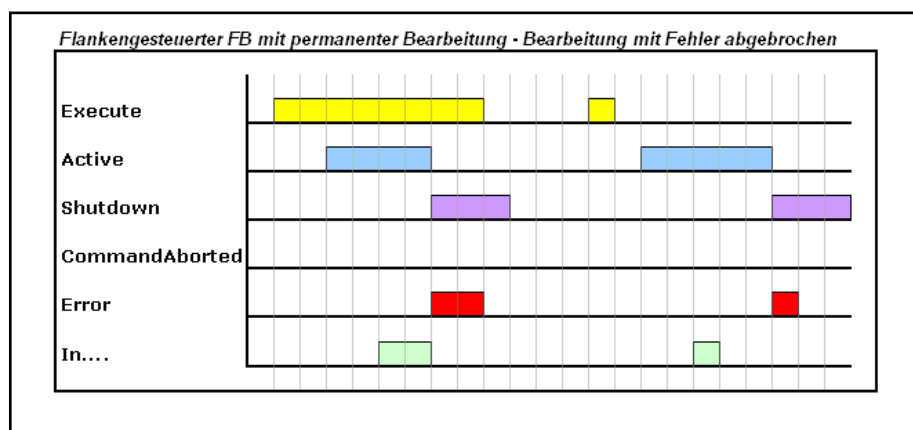


Abb. 2-23: I/O flankengesteuerter Funktionsbausteine, permanente Bearbeitung mit "Shutdown". Bearbeitung mit Fehler abgebrochen

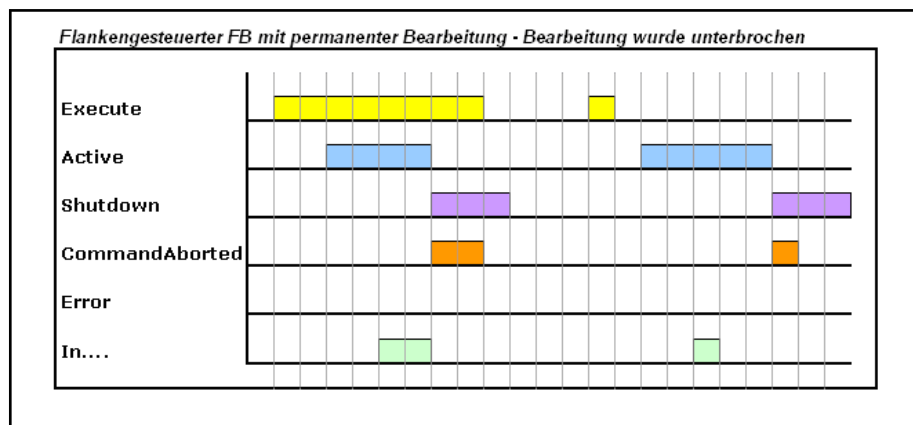


Abb. 2-24: I/O flankengesteuerter Funktionsbausteine, permanente Bearbeitung mit "Shutdown". Bearbeitung unterbrochen

3 RIL_ProfibusDP_02.library

3.1 Einführung und Übersicht

Historie Mit dem Einsatz des Funktionsmoduls Profibus-DP Master und des vorhandenen Onboard-Masters stehen Steuerungen zur Realisierung von Systemen mit mehreren DP-Mastern zur Verfügung.

Dazu wird die RIL_ProfibusDP_02.lib bereitgestellt, die mehrere DP-Master unterscheiden kann.

Diese Bibliothek wird auf die Version 01V02 erweitert, um die Kompatibilität sicherzustellen.



Die RIL_ProfibusDP.lib Version 01V01 ist zur Verwendung auf Systemen mit mehreren DP-Mastern ungeeignet.

Anwendungsgebiete und Funktionsbausteine von RIL_ProfibusDP_02

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine	Beschreibung
IL_DPMasterState, Seite 37	Ermittelt den Zustand eines lokalen Busmasters, sowie der konfigurierten Slaves
IL_DPBAudrateGet, Seite 48	Lesen der aktuellen Baudrate
IL_DPDeviceListGet, Seite 47	Liste der aktiven DP-Devices im System
IL_DPDevInfoGet, Seite 49	Lesen der Busmaster Informationsstruktur
IL_DPGetCfg, in Vorbereitung	
IL_DPIdent, Seite 39	Funktion zum Zusammenbauen von Profibus IDs
IL_DPPrjSlaveListGet, Seite 49	Liste der projektierten Slaves
IL_DPReadDiag, Seite 43	Lesen der Diagnosedaten eines Slaves nach DP-Norm
IL_DPSlaveDiagListGet, Seite 50	Liste der aktuellen Slavediagnose
IL_DPSycFr, Seite 45	Synchronisieren/Freezen Steuerkommando
IL_DPV1Read, Seite 40	lesender V1 Dienst
IL_DPV1Write, Seite 42	schreibender V1 Dienst

Tab. 3-1: RIL_ProfibusDP_02-Bibliothek

3.2 Datentypen

3.2.1 IL_BUSMASTER

Kurzbeschreibung Der Aufzählungstyp IL_BUSMASTER aus der RIL_FieldbusTypes.library dient dazu, den jeweiligen Profibus-DP Master auszuwählen. In einem MLC/MTX-System können neben der onboard Realtime-Ethernet-Schnittstelle über Funktionsmodule weitere Controller vorhanden sein.

Die Profibus-DP Master werden anhand ihrer aufsteigenden Reihenfolge in der Konfiguration unterschieden.

Die Indizierung der Master erfolgt in der Reihenfolge:

1. Onboard-Master
2. Funktionsmodul 1

3. Funktionsmodul 2
4. Funktionsmodul 3
5. Funktionsmodul 4



Ist beispielsweise der Onboard-Master kein Profibus Master und ist auf dem Funktionsmodul 1 ein Profibus Master konfiguriert, so erhält der Master auf dem Funktionsmodul 1 den Wert IL_BUSMASTER_0.

Element	Wert	Beschreibung
IL_BUSMASTER_0	0	Erstes Gerät
IL_BUSMASTER_1	1	Zweites Gerät
IL_BUSMASTER_2	2	Drittes Gerät
IL_BUSMASTER_3	3	Viertes Gerät
IL_BUSMASTER_4	4	Fünftes Gerät

Tab. 3-2: Aufzählungstyp IL_BUSMASTER

Beispiel zur Adressierung der verschiedenen Instanzen der Profibus Master auf einer Steuerung:

Konfiguriert ist:	Adressierung mit:
Onboard: Profibus-DP Master	IL_BUSMASTER_0
FM1: Profibus-DP Slave	IL_BUSSLAVE_0
FM2: Profibus-DP Master	IL_BUSMASTER_1
FM3: Profibus-DP Master	IL_BUSMASTER_2

3.2.2 Slave-Diagnosedaten nach Profibus DP-Norm

Überblick Aufbau der Profibus-spezifischen Diagnoseinformationen nach Profibus-DP Norm.

Die Slave-Diagnosedaten gliedern sich in zwei Teile:

- Allgemeiner Teil mit einer festen Länge von 6 Byte
- Erweiterte Diagnose (Slave spezifisch mit variabler Länge)

Offset	Type	Bezeichnung	Beschreibung
0	BYTE	Stationsstatus_1	Siehe unten
1	BYTE	Stationsstatus_2	Siehe unten
2	BYTE	Stationsstatus_3	Siehe unten
3	BYTE	Master_Add	Busadresse des Masters, der den Slave parametrier hat
4	WORD	Ident_Number	Ident-Nummer des Slaves
6-243		Ext_Diag_Data	Erweiterte Diagnose nach Profibus DP-Norm

Tab. 3-3: Slave-Diagnosedaten

Stationsstatus Die folgende Beschreibung des Stationsstatus 1 bis 3 ist ein Auszug aus Profibus DP-Norm.

Bit	Bezeichnung	Beschreibung
7	Master_Lock	Der DP-Slave wurde von einem anderen Master parametrieren. Dieses Bit wird vom DP-Master (Klasse 1) gesetzt, wenn die Adresse in Octet 4 ungleich 255 und ungleich der eigenen Adresse ist. Der DP-Slave setzt dieses Bit fest auf Null
6	Prm_Fault	Dieses Bit wird vom DP-Slave gesetzt, falls das letzte Parametertelegramm fehlerhaft war, z. B. falsche Länge, falsche Ident_Number, ungültige Parameter
5	Invalid_Slave_Response	Dieses Bit wird vom DP-Master gesetzt, sobald von einem angesprochenen DP-Slave eine unplausible Antwort empfangen wird. Der DP-Slave setzt dieses Bit fest auf Null
4	Not_Supported	Dieses Bit wird vom DP-Slave gesetzt, sobald eine Funktion angefordert wurde, die von diesem DP-Slave nicht unterstützt wird
3	Ext_Diag	Dieses Bit wird vom DP-Slave gesetzt. Ist das Bit gesetzt, muss im Slave-spezifischen Diagnosebereich (Ext_Diag_Data) ein Diagnoseeintrag vorliegen. Ist das Bit nicht gesetzt, kann im Slave-spezifischen Diagnosebereich (Ext_Diag_Data) eine Statusmeldung vorliegen. Die Bedeutung dieser Statusmeldung ist anwendungsspezifisch zu vereinbaren
2	Cfg_Fault	Dieses Bit wird vom DP-Slave gesetzt, sobald die vom DP-Master zuletzt erhaltenen Konfigurationsdaten mit denen, die der DP-Slave ermittelt hat, nicht übereinstimmen
1	Station_Not_Ready	Dieses Bit wird vom DP-Slave gesetzt, wenn der DP-Slave noch nicht für den Datenaustausch bereit ist
0	Station_Non_Existent	Dieses Bit setzt der DP-Master, falls dieser DP-Slave nicht über den Bus erreichbar ist. Ist dieses Bit gesetzt, enthalten die Diagnosebits den Zustand der letzten Diagnosemeldung oder den Initialwert. Der DP-Slave setzt dieses Bit fest auf Null

Tab. 3-4: Stationsstatus_1

Bit	Bezeichnung	Beschreibung
7	Deactivated	Dieses Bit wird vom DP-Master gesetzt, sobald der DP-Slave im DP-Slave-Parametersatz als nicht aktiv gekennzeichnet und aus der zyklischen Bearbeitung herausgenommen wurde. Der DP-Slave setzt dieses Bit immer auf Null
6	reserviert	-
5	Sync_Mode	Dieses Bit wird vom DP-Slave gesetzt, sobald er das Sync-Steuerkommando erhalten hat. Eine Änderung dieser Bits führt nicht zu einer Diagnosemeldung. Deshalb geben diese Bits in der Regel nicht den aktuellen Stand wieder
4	Freeze_Mode	Dieses Bit wird vom DP-Slave gesetzt, sobald er das Freeze-Steuerkommando erhalten hat. Eine Änderung dieser Bits führt nicht zu einer Diagnosemeldung. Deshalb geben diese Bits in der Regel nicht den aktuellen Stand wieder
3	WD_On	Watchdog Dieses Bit wird vom DP-Slave gesetzt, sobald seine Ansprechüberwachung aktiviert ist
2	1	Dieses Bit wird vom DP-Slave fest auf "1" gesetzt

Bit	Bezeichnung	Beschreibung
1	Stat_Diag	Statische Diagnose Setzt der DP-Slave dieses Bit, muss der DP-Master solange Diagnoseinformationen abholen, bis dieses Bit wieder gelöscht wird. Der DP-Slave setzt dieses Bit beispielsweise dann, wenn er keine gültigen Nutzdaten zur Verfügung stellen kann
0	Prm_Req	Setzt der DP-Slave dieses Bit, muss er neu parametrisiert und konfiguriert werden. Das Bit bleibt solange gesetzt, bis eine Parametrierung erfolgt ist. Dieses Bit wird vom DP-Slave gesetzt

Tab. 3-5: Stationsstatus_2

Bit	Bezeichnung	Beschreibung
7	Ext_Diag_Overflow	Ist dieses Bit gesetzt, liegen mehr Diagnoseinformationen vor, als in Ext_Diag_Data angegeben wurde. Der DP-Slave setzt dieses Bit beispielsweise, wenn mehr Kanaldiagnosen vorliegen, als der DP-Slave in seinen Sendepuffer eintragen kann; oder der DP-Master setzt dieses Bit, wenn der DP-Slave mehr Diagnoseinformationen sendet, als der DP-Master in seinem Diagnosepuffer aufnehmen kann
6	reserviert	-
5	reserviert	-
4	reserviert	-
3	reserviert	-
2	reserviert	-
1	reserviert	-
0	reserviert	-

Tab. 3-6: Stationsstatus_3

3.2.3 Busmaster-Statuswort

Das Busmaster-Statuswort "BmState" gibt einen Überblick über den Zustand des Busmasters und der Slaves am Feldbus. Hier lässt sich z. B. erkennen, ob bei mindestens einem Busmaster eine Slave-Diagnose ansteht. Jedes gesetzte Bit (TRUE) in "BmState" repräsentiert einen Status:

Bit	Status	Beschreibung
0	BMS_BMF	Busmaster-Fehler: Dieses Bit zeigt an, dass ein Busmasterfehler vorliegt. In diesem Fall enthält das Busmaster-Fehlerwort genauere Informationen
1	BMS_KSD	Klassifizierte Slave-Diagnose: Ist dieses Bit gesetzt, meldet mindestens ein Slave eine klassifizierte Diagnose. Welche klassifizierte(n) Diagnose(n) gesetzt ist/sind, lässt sich anhand der Bits 8 bis 13 ermitteln
2	BMS_SD	Slave-Diagnose: Ist dieses Bit gesetzt, so meldet mindestens ein Slave Diagnose
3	-	reserviert
4	-	reserviert
5	-	reserviert
6	-	reserviert

Bit	Status	Beschreibung
7	BMS_AKTIV	Aktivkennung: Dieses Bit muss immer den Wert 1 haben. Ist dies nicht der Fall, liegt ein fataler Fehler in der Software des Busmasters vor
8	BMS_SNE	Ein oder mehrere Slaves sind über den Bus nicht erreichbar
9	BMS_SKF	Ein oder mehrere Slaves melden Konfigurationsfehler
10	BMS_DPS	Ein oder mehrere Slaves melden statische Diagnose
11	BMS_EXD	Ein oder mehrere Slaves melden erweiterte Diagnose
12	BMS_SNB	Ein oder mehrere Slaves sind nicht bereit für den zyklischen Datenaustausch
13	BMS_SF	Ein oder mehrere Slaves melden einen sonstigen Fehler
14	-	reserviert
15	-	reserviert

Tab. 3-7: Status-Kodierung in "BmState"

3.2.4 Busmaster-Fehlerwort

Im Busmaster-Fehlerwort "BmError" werden gravierende Fehler angezeigt, die einen Betrieb am Feldbus unmöglich machen. Jedes gesetzte Bit (TRUE) in "BmError" repräsentiert einen Fehler:

Bit	Fehler ¹⁾	Beschreibung
0	IL_BMF_HW_ERR	Hardware-Fehler
1	IL_BMF_MPS_ERR	Masterparametersatz (Feldbus-Konfigurationsdatei) fehlt oder fehlerhaft
2	IL_BMF_BUS_ERR	Fehler am Feldbus (z. B. Kurzschluss)
3	IL_BMF_SW_ERR	Systemfehler im Peripherie-Treiber (d. h., die Treiber-Software hat einen schweren Fehler festgestellt)

Tab. 3-8: Fehler-Kodierung in "BmError"

3.2.5 DP_SLAVELIST, Array

Kurzbeschreibung Die Bitliste "DP_SLAVELIST" (DP_BITLIST) hat eine feste Länge von 16 Bytes (128 Bit).

Typdeklaration

```
TYPE DP_SLAVELIST :  
  ARRAY [0..15] OF BYTE;  
END_TYPE
```

Kodierung der Bitliste Jedes Bit der Bitliste ist einer Busadresse des Slaves (Profibus: FDL-Adresse) zugeordnet. Das niederwertigste Bit im ersten Feldelement (ARRAY[0]) ist dem Profibus-Teilnehmer mit der Adresse "0" zugeordnet:

¹⁾ "IL_BMF_OK" zeigt an, dass kein Fehler vorliegt

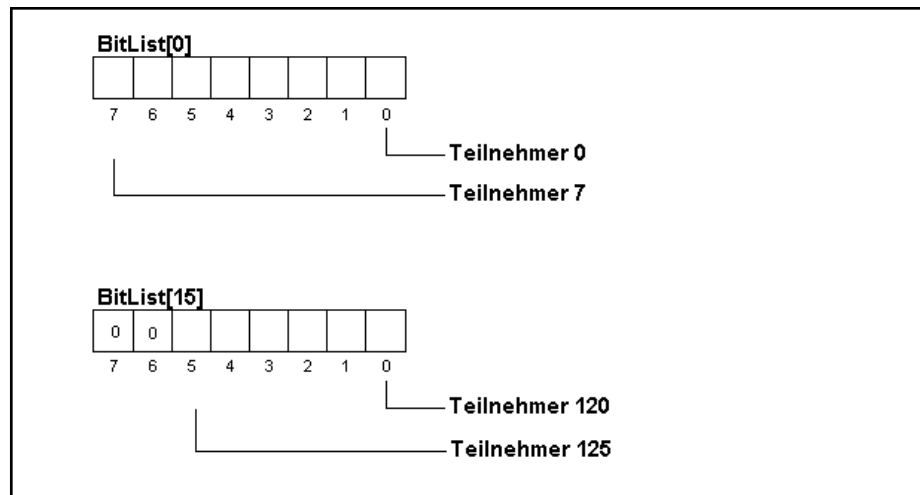


Abb. 3-1: Kodierung der Bitliste

3.2.6 DP_DEVICELIST, Struktur

Kurzbeschreibung Dieser Datentyp fasst die Informationen bezüglich eines DP-Masters zusammen:

Beispiel *Programm:*

```

TYPE DP_DEVICELIST:
STRUCT
  bMasterAdr      : BYTE;  (* master addressing *)
  bMasterBusAdr   : BYTE;  (* master bus address *)
  wMasterError    : WORD;  (* cf. IL_DPDevInfoGet *)
  wMasterState    : WORD;  (* cf. IL_DPDevInfoGet *)
  udBaudrate      : UDINT; (* cf. IL_DPBaudrateGet *)
  dFirmwareVersion : DINT;  (* driver firmware version *)
  dHardwareVersion : DINT;  (* hardware version *)
  dAddInfo1       : DINT;  (* 3S module identification *)
  dAddInfo2       : DINT;  (* res *)
END_STRUCT
END_TYPE

TYPE DP_MASTERLIST:
  ARRAY [0..5] of DP_DEVICELIST; (* list for 6 masters *)
END_TYPE

```



Ein eventuell vorhandener Onboard-Slave wird auch in dieser Liste geführt.

3.2.7 DP_MASTERLIST, Array

Kurzbeschreibung Dieser Datentyp fasst die Informationen von max. 6 DP-Master zusammen:

Beispiel

```

TYPE DP_DEVICELIST:
STRUCT
  bMasterAdr      : BYTE;  (* master addressing *)
  bMasterBusAdr   : BYTE;  (* master bus address *)
  wMasterError    : WORD;  (* cf. IL_DPDevInfoGet *)
  wMasterState    : WORD;  (* cf. IL_DPDevInfoGet *)
  udBaudrate      : UDINT; (* cf. IL_DPBaudrateGet *)
  dFirmwareVersion : DINT;  (* driver firmware version *)
  dHardwareVersion : DINT;  (* hardware version *)
  dAddInfo1       : DINT;  (* 3S module identification *)
  dAddInfo2       : DINT;  (* res *)
END_STRUCT
END_TYPE

TYPE DP_MASTERLIST:
  ARRAY [0..5] of DP_DEVICELIST; (* list for 6 masters *)
END_TYPE

```


3.2.8 DP_MASTERINFO, Struktur

Kurzbeschreibung Dieser Datentyp fasst Informationen über den Zustand des Busmasters zusammen.

Beispiel

```
TYPE DP_MASTERINFO: (*DP_DEVICEINFO:*)
STRUCT
    BmState      : WORD;      (* bus master status *)
    BmError      : WORD;      (* bus master error word *)
END_STRUCT
END_TYPE
```

3.3 Auswahl des DP-Masters

Adressierung Die DP-Master werden anhand ihrer Reihenfolge in der DP-Konfiguration unterschieden:

0 .. n (5) Nummer des DP-Masters in aufsteigender Reihenfolge der Konfiguration.

Es werden nur DP-Master gezählt.

Verwendung Die Funktionen und Funktionsbausteine, die auf eine bestimmte Instanz eines Masters zugreifen müssen, erhalten den Input-Parameter "Master" vom Typ BYTE.

Die Funktionen und Funktionsbausteine, die der Profibus Guideline 2182 entsprechen, besitzen einen DWORD Parameter "ID", der als Slot-Handle interpretiert wird. Darin ist ein Byte für die Auswahl des Masters reserviert. Zur Erzeugung der Parameter-ID kann die Funktion **DP_SLOT** benutzt werden.

Die Funktionen und Funktionsbausteine, die den DWORD Parameter "Ident" nutzen, können diesen Parameter mit der Funktion **IL_DPIdent** generieren.

Byte	Inhalte	Beschreibung
0	MASTER	ID des DP-Systems: Kennzeichnung des DP-Masters (oder auch des onboard Slaves)
1	SEGMENT	Nummer des DP-Segmentes (0)
2	STATION	Nummer des DP-Slaves (Busadresse)
3	SLOT	Nummer des Slots innerhalb des Slaves

Tab. 3-9: Slot-Handle: Parameter-ID

3.4 IL_DPMasterState

Kurzbeschreibung Mit dem Funktionsbaustein **IL_DPMasterState** wird der Zustand eines lokalen Busmasters sowie der konfigurierten Slaves ermittelt. Die Anzeige des Ergebnisses erfolgt bitcodiert im Ausgangsparameter "State".

Zielsystem	Bibliothek/Ordner
MLC	RIL_ProfibusDP_02
MTX	RIL_ProfibusDP_02

Tab. 3-10: Referenztablelle

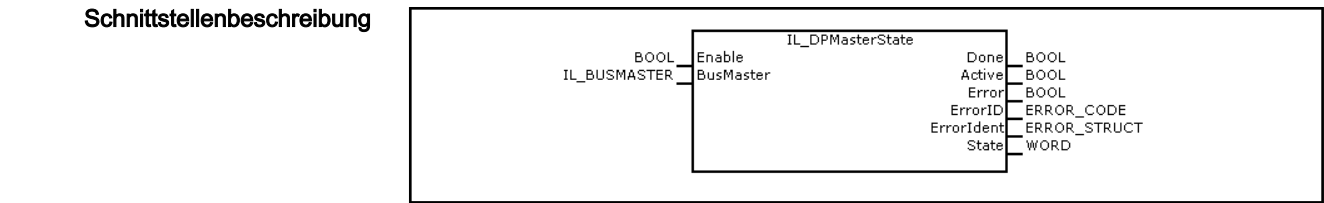


Abb. 3-2: Schnittstellenbeschreibung FB IL_DPMasterState

I/O-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (kontinuierlich, zustandsgesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Masters in der Reihenfolge des Konfigurationsbaumes. Ist nur ein Profibus-DP Master vorhanden, besitzt dieser Master immer den Wert IL_BUSMASTER_0 Siehe auch Kap. 3.2.1 "IL_BUSMASTER" auf Seite 31
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	State	WORD	Master-Diagnosestatus (siehe Elemente des Ausgangs State)

Tab. 3-11: I/O-Schnittstelle des FB IL_DPMasterState

Im Ausgang "State" ist eine Grobklassifizierung von Diagnosen binär kodiert. Die Bits sind in der RIL_FieldbusTypes.library als Globale Konstanten "FieldbusStates" enthalten.

Ausgang "State" (WORD)

BIT-Nr.	Bezeichnung	Beschreibung
0	IL_FBM_BUS_OFF	Feldbus nicht bereit
1	IL_FBM_MASTER_FAILURE	Fehler des Busmasters (bus master failure)
2	IL_FBM_SLAVE_NOT_REACHABLE	Ein oder mehrere Feldbuslaves sind nicht erreichbar (slave not reachable)
3	IL_FBM_SLAVE_DIAGNOSIS	Ein oder mehrere Feldbuslaves melden Diagnose (slave diag)
4	IL_FBM_SLAVE_ERROR	Ein oder mehrere Feldbuslaves melden Fehler (slave error)

Tab. 3-12: Elemente des Ausgangs "State"

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Zustand TRUE von "Enable"

Tab. 3-13: Min-, Max- und Default-Werte der IL_DPMasterState-Eingänge

- Funktionsbeschreibung**
- Ein wirksamer Aufruf von **IL_DPMasterState** und die Übernahme des Parameters "BusMaster" erfolgt bei Enable = TRUE
 - Bei Enable = FALSE ist der Aufruf ohne Wirkung. In diesem Fall werden alle Ausgangsparameter auf "0" gesetzt
 - Das Ergebnis des Aufrufs wird zunächst in den Ausgangsparametern Done, Active und Error angezeigt. Bei Active = FALSE ist der Aufruf abgeschlossen und die anderen Parameter sind auszuwerten
 - Done = TRUE und Error = FALSE kennzeichnen einen fehlerfreien Verlauf des Aufrufs
 - Done = FALSE und Error = TRUE zeigen einen fehlerhaften Verlauf des Aufrufs an. In diesem Fall beschreiben die Parameter ErrorID und ErrorIdent die Fehlerursache
 - Nach fehlerfreiem Verlauf des Aufrufs ist der effektive Ausgangsparameter State gültig und kann durch die aufrufende Instanz ausgewertet werden
 - Bei fehlerhaftem Verlauf ist der effektive Ausgangsparameter State ungültig
 - Bei gültigem Ausgangsparameter wird in State der Zustand der Busmaster-Instanz sowie der konfigurierten Slaves in bitcodiertem Format angezeigt
 - State = 0 zeigt einen fehlerfreien Zustand des Bussystems an
 - Error = TRUE kennzeichnet einen fehlerhaften Verlauf des Aufrufs. In diesem Fall definieren ErrorID und ErrorIdent die Fehlerursache. Der Ausgangsparameter State ist ungültig

Fehlerbehandlung

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#1001	BusMaster-Instanz	Ungültige BusMaster-Instanz (Wertebereich)
DEVICE_ERROR	16#2001	BusMaster-Instanz	Busmaster-Instanz nicht verfügbar
ACCESS_ERROR	16#2002	ERROR_CODE	Fehler beim Zugriff

Tab. 3-14: Fehlercodes Funktionsbaustein IL_DPMasterState

3.5 IL_DPIdent

Kurzbeschreibung Die Funktion **IL_DPIdent** setzt einen Ident-Handle aus den einzelnen Komponenten zusammen.

Zielsystem	Bibliothek/Ordner
MLC	RIL_ProfibusDP_02
MTX	RIL_ProfibusDP_02

Tab. 3-15: Referenztablelle

Schnittstellenbeschreibung

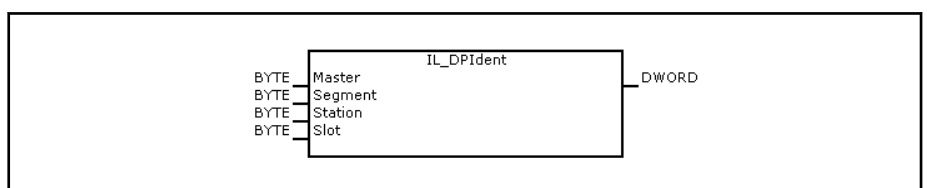


Abb. 3-3: Schnittstellenbeschreibung Funktion IL_DPIdent

I/O-Typ	Name	Typ	Beschreibung
VAR_INPUT	Master	BYTE	Kennzeichnung des DP-Masters (vgl. oben)
	Segment	BYTE	Nummer des DP-Segmentes (0)
	Station	BYTE	Nummer des DP-Slaves (Busadresse): Soll das Kommando nur für einen bestimmten Slave gelten, dann muss hier die Busadresse des Slaves eingetragen werden (0...125). Nur für IL_DPSycFr: Soll das Kommando aber für alle Slaves einer Gruppe eingetragen werden, dann muss hier die Globaladresse (= 127) eingetragen werden
	Slot	BYTE	Nummer des Slots innerhalb des Slaves (lt. Spezifikation des Slaves), Wertebereich: 0...254
Funktionswert		DWORD	Ident-Handle

Tab. 3-16: I/O-Schnittstelle der Funktion IL_DPIdent

Funktionsbeschreibung

Aus den 4 Bytewerten wird das DWORD "Ident" gebildet. Dieses DWORD wird als Eingangsparameter für die nachfolgend beschriebenen Funktionsbausteine benötigt.

3.6 IL_DPV1Read

Kurzbeschreibung

Mit dem Funktionsbaustein **IL_DPV1Read** wird ein lesender DPV1-Zugriff durchgeführt. Der Datenaustausch auf dem Profibus-DP findet azyklisch statt. Für die zu lesenden Prozessdaten muss ein Zielbereich über eine Zeiger-Adressierung (POINTER) definiert werden.

Zielsystem	Bibliothek/Ordner
MLC	RIL_ProfibusDP_02
MTX	RIL_ProfibusDP_02

Tab. 3-17: Referenztablelle

Schnittstellenbeschreibung

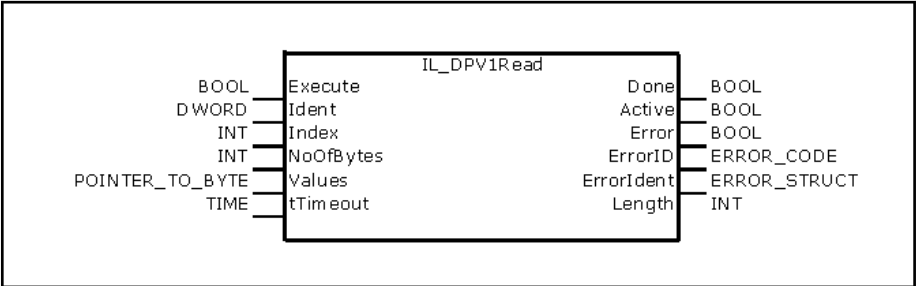


Abb. 3-4: Schnittstellenbeschreibung FB IL_DPV1Read

I/O-Typ	Name	Typ	Beschreibung
VAR_INPUT	Execute	BOOL	Funktionsaktivierung; ein aktivierter Funktionsbaustein kann nicht unterbrochen werden
	Ident	DWORD	Ident-Handle (siehe Funktion Kap. 3.5 "IL_DPIdent" auf Seite 39)
	Index	INT	Index der Prozessdaten (Feldnummer)
	NoOfBytes	INT	Maximale Länge der zu lesenden Daten; Anzahl der verfügbaren Bytes auf dem Pointer "Values"
	Values	POINTER TO BYTE	Zeiger auf Datenpuffer für die Zieldaten
	tTimeout	TIME	Zeitbegrenzung für Warten auf Daten
VAR_OUTPUT	Done	BOOL	Fertigmeldung (erfolgreich)
	Active	BOOL	Aktivitätsanzeige
	Error	BOOL	Fertigmeldung (nicht erfolgreich)
	ErrorID	ERROR_CODE	Siehe Kap. 3.15.1 "ErrorID / ERROR_CODE" auf Seite 52
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur aus ErrorTable PB_DP_TABLE
	Length	INT	Länge der gelesenen Daten in Byte

Tab. 3-18: I/O-Schnittstelle des FB IL_DPV1Read

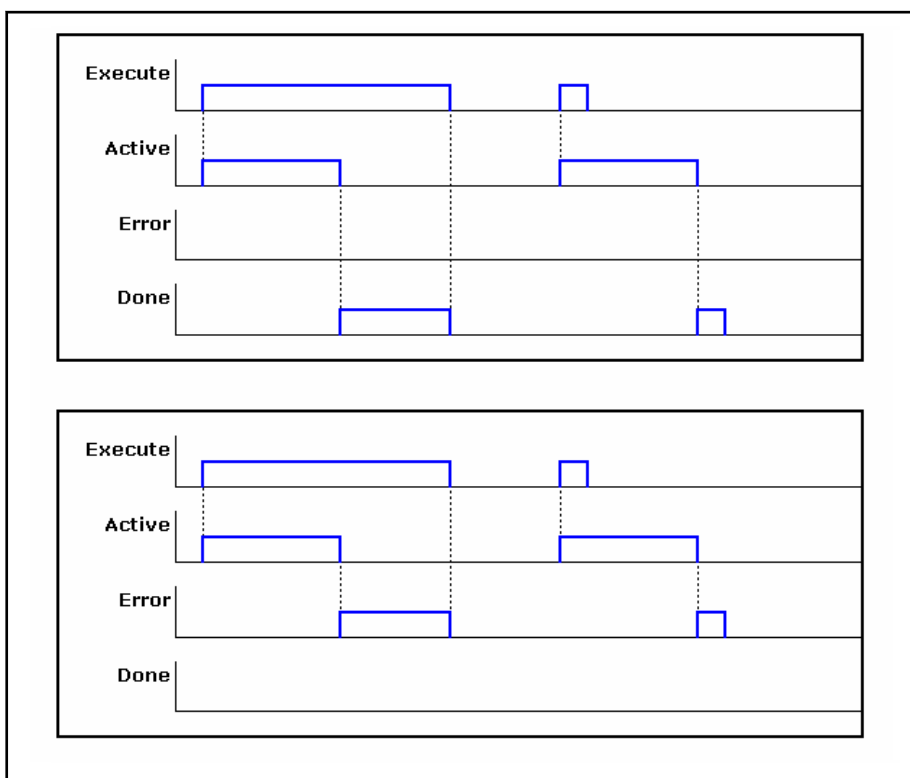
Signal-Zeit-Diagramm

Abb. 3-5: Signal-Zeit-Diagramm IL_DPV1Read

Funktionsbeschreibung

Der Master (Klasse 1) greift mit diesem Zugriff auf einen DP-V1 Slave zu. Der Master liest den durch "Slaveadresse", "Slot" und "Index" adressierten Datensatz des Slaves. Die Adressierung mit "Slot" und "Index" sowie die Interpretation der Daten ist Slave-spezifisch und der jeweiligen Dokumentation des Slaves zu entnehmen. Die Ausführungszeit der Funktion ist unter anderem von der Busbelastung und der eingestellten Baudrate abhängig.

Diese Funktion ist nur für Slaves möglich die am DP-Buszyklus teilnehmen.

3.7 IL_DP1Write

Kurzbeschreibung Mit dem Funktionsbaustein **IL_DP1Write** wird ein schreibender DPV1 Zugriff durchgeführt. Der Datenaustausch auf dem Profibus-DP findet azyklisch statt. Die zu schreibenden Prozessdaten werden über eine Zeiger-Adressierung (POINTER) übergeben.

Zielsystem	Bibliothek /Ordner
MLC	RIL_ProfibusDP_02
MTX	RIL_ProfibusDP_02

Tab. 3-19: Referenztablelle

Schnittstellenbeschreibung

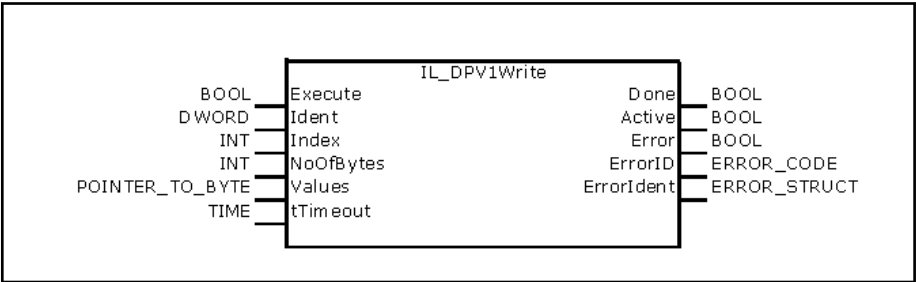


Abb. 3-6: Schnittstellenbeschreibung FB IL_DP1Write

I/O-Typ	Name	Typ	Beschreibung
VAR_INPUT	Execute	BOOL	Funktionsaktivierung; ein aktivierter Funktionsbaustein kann nicht unterbrochen werden.
	Ident	DWORD	Ident-Handle (siehe Funktion IL_DPIdent)
	Index	INT	Index der Prozessdaten (Feldnummer)
	NoOfBytes	INT	Maximale Länge der zu schreibenden Daten; Anzahl der verfügbaren Bytes auf dem Pointer "Values"
	Values	POINTER TO BYTE	Zeiger auf Datenpuffer für die Daten
	tTimeout	TIME	Zeitbegrenzung für Warten auf Abschluss des Schreibvorgangs
VAR_OUTPUT	Done	BOOL	Fertigmeldung (erfolgreich)
	Active	BOOL	Aktivitätsanzeige
	Error	BOOL	Fertigmeldung (nicht erfolgreich)
	ErrorID	ERROR_CODE	Siehe Kap. 3.15.1 "ErrorID / ERROR_CODE" auf Seite 52
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur aus ErrorTable PB_DP_TABLE

Tab. 3-20: I/O-Schnittstelle des FB IL_DP1Write

Signal-Zeit-Diagramm

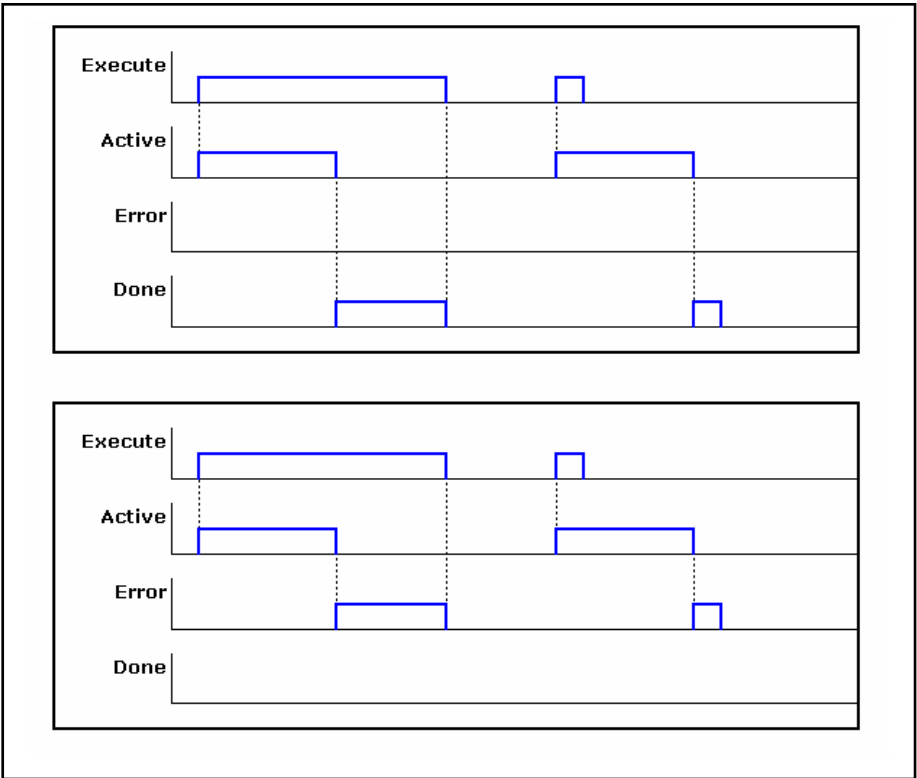


Abb. 3-7: Signal-Zeit-Diagramm IL_DPWrite

Funktionsbeschreibung Der Master (Klasse 1) greift mit diesem Zugriff auf einen DP-V1-Slave zu. Er liest den durch Slave-Adresse, Slot und Index adressierten Datensatz des Slaves. Die Adressierung mit Slot und Index sowie die Interpretation der Daten ist Slave-spezifisch und der jeweiligen Dokumentation des Slaves zu entnehmen. Die Ausführungszeit der Funktion ist unter anderem von der Busbelastung und der eingestellten Baudrate abhängig.

Diese Funktion ist nur für Slaves möglich die am DP-Buszyklus teilnehmen.

3.8 IL_DPReadDiag

Kurzbeschreibung Mit dem Funktionsbaustein **IL_DPReadDiag** werden die Diagnosedaten eines Slaves vom DP-Master (DPM1) gelesen. Der Datenpuffer der Diagnosedaten muss bereitgestellt werden, um diesen über einen Zeiger (POINTER) zu adressieren.

Zielsystem	Bibliothek/Ordner
MLC	RIL_ProfibusDP_02
MTX	RIL_ProfibusDP_02

Tab. 3-21: Referenztablelle

Schnittstellenbeschreibung

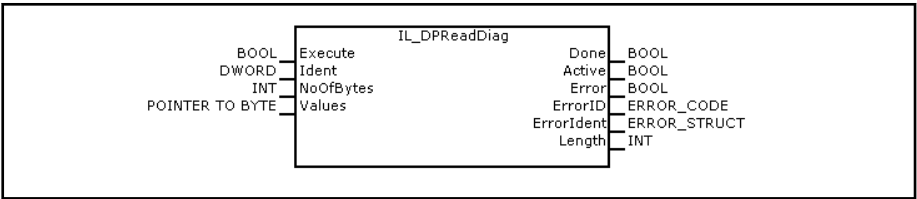


Abb. 3-8: Schnittstellenbeschreibung FB IL_DPReadDiag

I/O-Typ	Name	Typ	Beschreibung
VAR_INPUT	Execute	BOOL	Funktionsaktivierung; ein aktivierter Funktionsbaustein kann nicht unterbrochen werden
	Ident	DWORD	Ident-Handle (siehe Funktion IL_DPIdent)
	NoOfBytes	INT	Maximale Länge der zu lesenden Daten; Anzahl der verfügbaren Bytes auf dem Pointer "Values"
	Values	POINTER TO BYTE	Zeiger auf Datenpuffer der Slave-Diagnosedaten. Siehe Kap. 3.2.2 "Slave-Diagnosedaten nach Profibus DP-Norm" auf Seite 32
VAR_OUTPUT	Done	BOOL	Fertigmeldung (erfolgreich)
	Active	BOOL	Aktivitätsanzeige
	Error	BOOL	Fertigmeldung (nicht erfolgreich)
	ErrorID	ERROR_CODE	Siehe Kap. 3.15.1 "ErrorID / ERROR_CODE" auf Seite 52
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur aus ErrorTable PB_DP_TABLE
	Length	INT	Länge der Diagnosedaten in Byte

Tab. 3-22: I/O-Schnittstelle des FB IL_DPReadDiag

Signal-Zeit-Diagramm

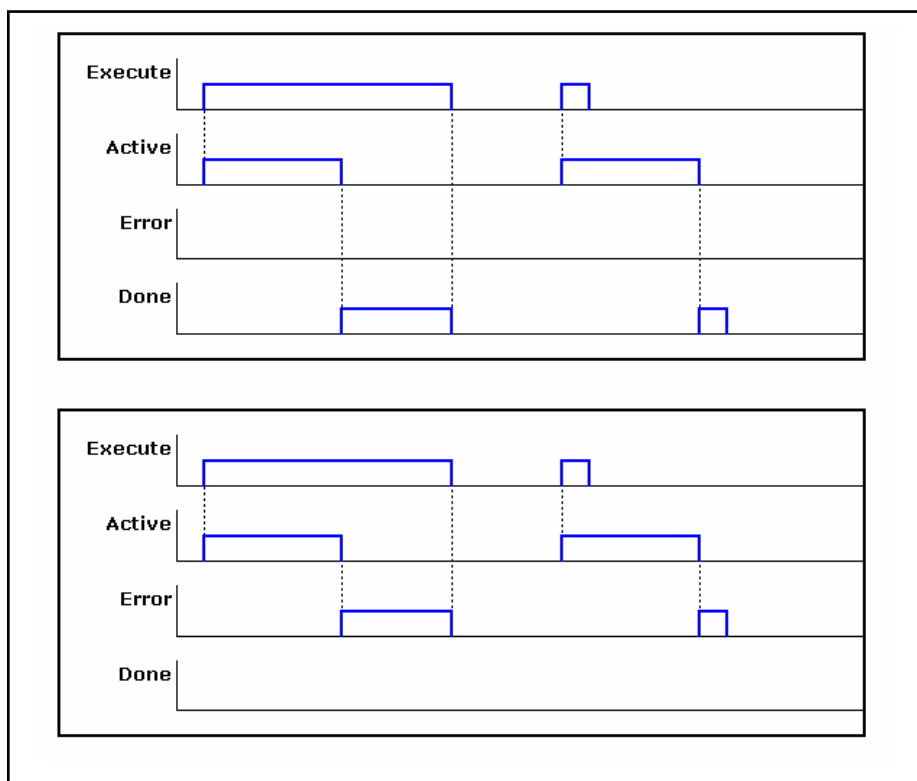


Abb. 3-9: Signal-Zeit-Diagramm IL_DPReadDiag

Funktionsbeschreibung

Der Master liest die Diagnoseinformationen der Slaves aus und speichert sie Slave-bezogen ab. Der Slave löst durch Anforderungsbits im zyklischen Telegrammverkehr den Diagnose-Request des Masters aus. Für den Inhalt der Diagnosedaten ist der Slave verantwortlich. Die vorliegende Funktion löst keinen Telegrammverkehr am Profibus aus, sie greift lediglich auf die im Master vorliegenden Diagnoseinfos zu.

3.9 IL_DPSycFr

Kurzbeschreibung Mit dem Funktionsbaustein **IL_DPSycFr** lassen sich Steuerkommandos zur Synchronisation von Ein- und Ausgängen realisieren.

Profibus-DP bietet die Möglichkeit, dass ein Master ein sogenanntes "Global Control Telegramm" an eine Gruppe von Slaves sendet. Das Global Control Telegramm enthält die Steuerkommandos "Freeze" und "Sync":

- Mit dem **Freeze**-Steuerkommando speichern alle Slaves der adressierten Gruppe gleichzeitig die aktuellen **Eingangsdaten** ab (Eingänge synchronisieren)
- Mit dem **Sync**-Steuerkommando übernehmen alle Slaves der adressierten Gruppe gleichzeitig die aktuellen **Ausgangsdaten** (Ausgänge synchronisieren)

Zielsystem	Bibliothek/Ordner
MLC	RIL_ProfibusDP_02
MTX	RIL_ProfibusDP_02

Tab. 3-23: Referenztabelle

Schnittstellenbeschreibung

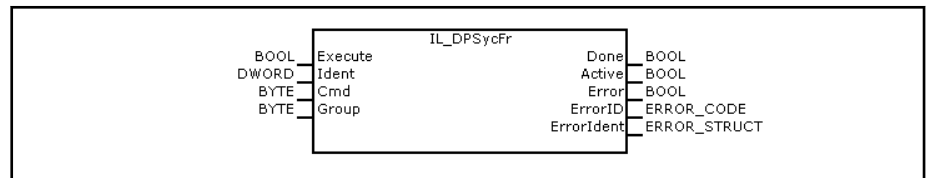


Abb. 3-10: Schnittstellenbeschreibung FB IL_DPSycFr

I/O-Typ	Name	Typ	Beschreibung
VAR_INPUT	Execute	BOOL	Funktionsaktivierung; ein aktivierter Funktionsbaustein kann nicht unterbrochen werden
	Ident	DWORD	Ident-Handle, siehe IL_DPIdent (Slot ist nicht relevant sollte "0" sein)
	Cmd	BYTE	Steuerkommando (siehe unten)
	Group	BYTE	Selektiert eine oder mehrere Gruppen, auf die sich das Kommando bezieht. Jedes Bit ist einer Gruppe zugeordnet
VAR_OUTPUT	Done	BOOL	Fertigmeldung (erfolgreich)
	Active	BOOL	Aktivitätsanzeige
	Error	BOOL	Fertigmeldung (nicht erfolgreich)
	ErrorID	ERROR_CODE	Siehe Kap. 3.15.1 "ErrorID / ERROR_CODE" auf Seite 52
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur aus ErrorTable PB_DP_TABLE

Tab. 3-24: I/O-Schnittstelle des FB IL_DPSycFr

Signal-Zeit-Diagramm

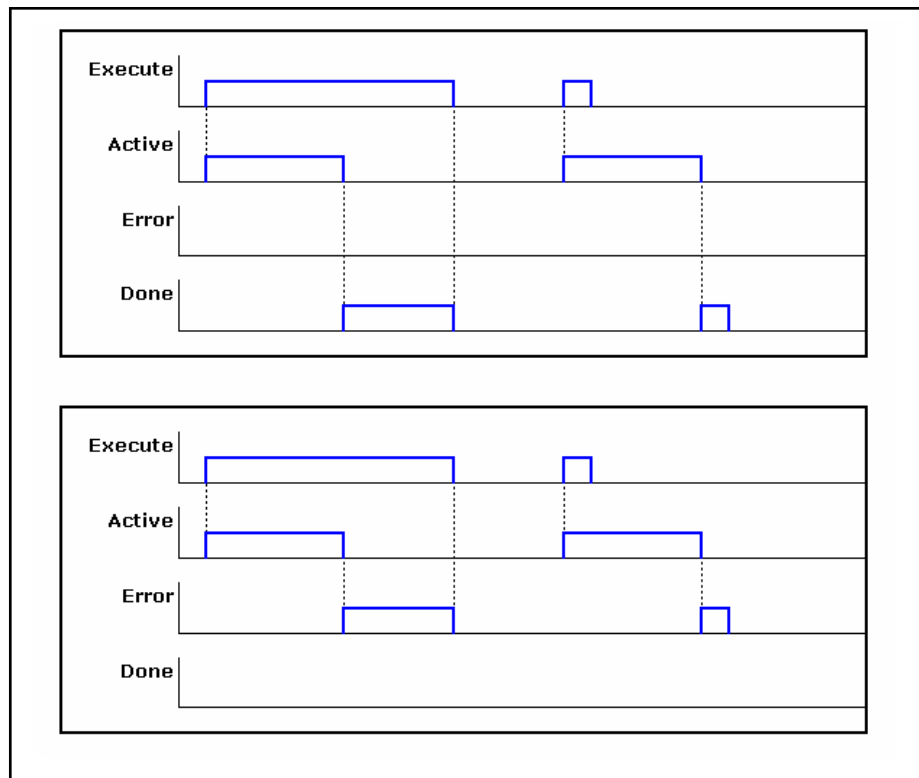


Abb. 3-11: Signal-Zeit-Diagramm IL_DPSycFr

Funktionsbeschreibung

Beispiel: Um alle Slaves einer Gruppe zu adressieren, hat die ID den Wert 16#007f0000.

Um die ID aus den einzelnen Komponenten zu bilden, dient die Funktion IL_DPIdent.



Für das Synchronisieren der Ausgänge muss gewährleistet werden, dass zuvor alle Slaves die aktuellen Ausgangsdaten erhalten haben, bevor diese das Sync-Kommando empfangen.

Rufen Sie dazu den Funktionsbaustein **IL_DPSycFr** von der gleichen SPS-Task auf, von der aus auch die Ausgangsdaten der Slaves geschrieben werden:

In einer SPS-Task werden zuerst die Ausgangsdaten für die zu synchronisierenden Slaves geschrieben. Anschließend wird mit **IL_DPSycFr** das Sync-Kommando gestartet. Solange das Sync-Kommando noch nicht fertig ist (BUSY), dürfen die Ausgangsdaten nicht mehr verändert werden.

Bei Verwendung von Sync oder Freeze muss im IndraWorks-Projektexplorer eine Zuordnung der Gruppen beim Master und den entsprechenden Slaves eingestellt werden. Weitere Informationen hierzu erhalten Sie in der Dokumentation oder der Online-Hilfe zu IndraWorks.

Mögliche Steuerkommandos

IL_DP_CMD_UNFREEZE	16#04	Freeze mode release
IL_DP_CMD_FREEZE	16#08	command Freeze

IL_DP_CMD_UNSYNC	16#10	Sync mode release
IL_DP_CMD_SYNC	16#20	command Sync

Tab. 3-25: Mögliche Steuerkommandos von IL_DPSycF

3.10 IL_DPDeviceListGet

Kurzbeschreibung Der Funktionsbaustein **IL_DPDeviceListGet** ermittelt eine Liste der aktuell vorhandenen DP-Master.

Zielsystem	Bibliothek/Ordner
MLC	RIL_ProfibusDP_02
MTX	RIL_ProfibusDP_02

Tab. 3-26: Referenztable

Schnittstellenbeschreibung

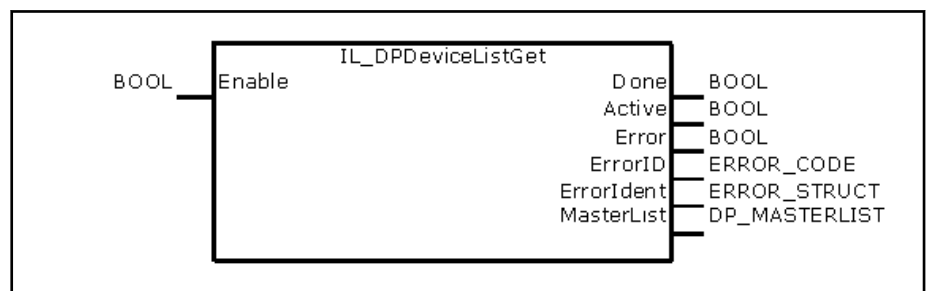


Abb. 3-12: Schnittstellenbeschreibung FB IL_DPDeviceListGet

I/O-Typ	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Funktionsfreigabe
VAR_OUTPUT	Done	BOOL	Fertigmeldung (erfolgreich)
	Active	BOOL	Aktivitätsanzeige
	Error	BOOL	Fertigmeldung (nicht erfolgreich)
	ErrorID	ERROR_CODE	Siehe Kap. 3.15.1 "ErrorID / ERROR_CODE" auf Seite 52
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur aus ErrorTable PB_DP_TABLE
	MasterList	DP_MASTERLIST	Liste der vorhandenen (maximal sechs) DP-Master

Tab. 3-27: I/O-Schnittstelle des FB IL_DPDeviceListGet

Signal-Zeit-Diagramm

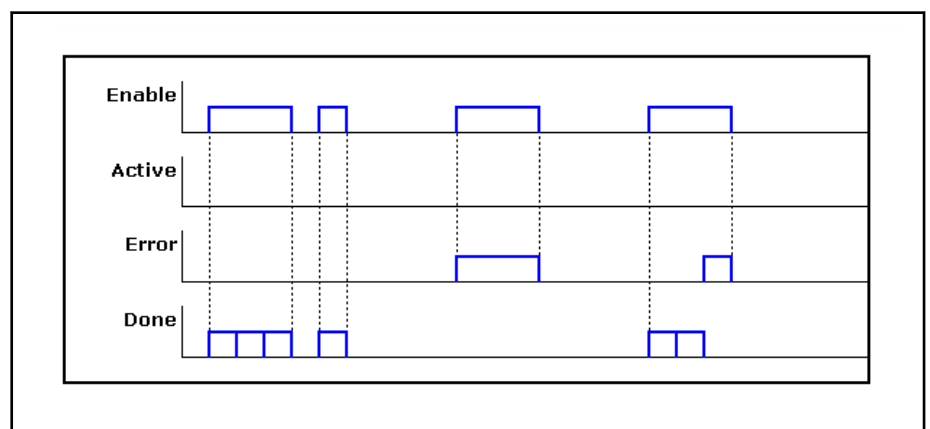


Abb. 3-13: Signal-Zeit-Diagramm IL_DPDeviceListGet

Funktionsbeschreibung

Ziel dieses Funktionsbausteins ist es, die auf der SPS aktiven und adressierbaren Master-Instanzen anzuzeigen. Damit kann der Anwender auch überwachen, ob eine passende Konfiguration geladen und richtig erkannt wurde.

3.11 IL_DPBAudrateGet

Kurzbeschreibung

Der Funktionsbaustein **IL_DPBAudrateGet** ermittelt die Baudrate des angeschlossenen Feldbuses. Die Baudrate wird in Bits pro Sekunde angegeben.

Zielsystem	Bibliothek/Ordner
MLC	RIL_ProfibusDP_02
MTX	RIL_ProfibusDP_02

Tab. 3-28: Referenztablelle

Schnittstellenbeschreibung

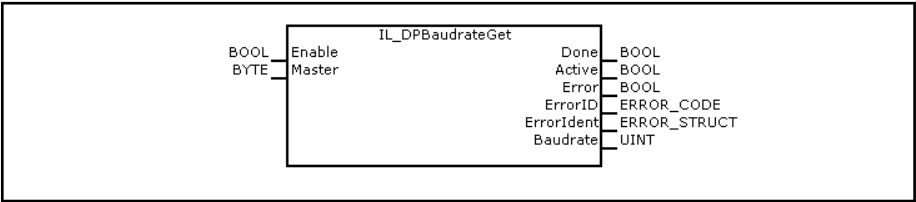


Abb. 3-14: Schnittstellenbeschreibung FB IL_DPBAudrateGet

I/O-Typ	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Funktionsfreigabe
	Master	BYTE	MasterID (siehe oben)
VAR_OUTPUT	Done	BOOL	Fertigmeldung (erfolgreich)
	Active	BOOL	Aktivitätsanzeige
	Error	BOOL	Fertigmeldung (nicht erfolgreich)
	ErrorID	ERROR_CODE	Siehe Kap. 3.15.1 "ErrorID / ERROR_CODE" auf Seite 52
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur aus ErrorTable PB_DP_TABLE
	Baudrate	UDINT	Baudrate in 1/s

Tab. 3-29: I/O-Schnittstelle des FB IL_DPBAudrateGet

Signal-Zeit-Diagramm

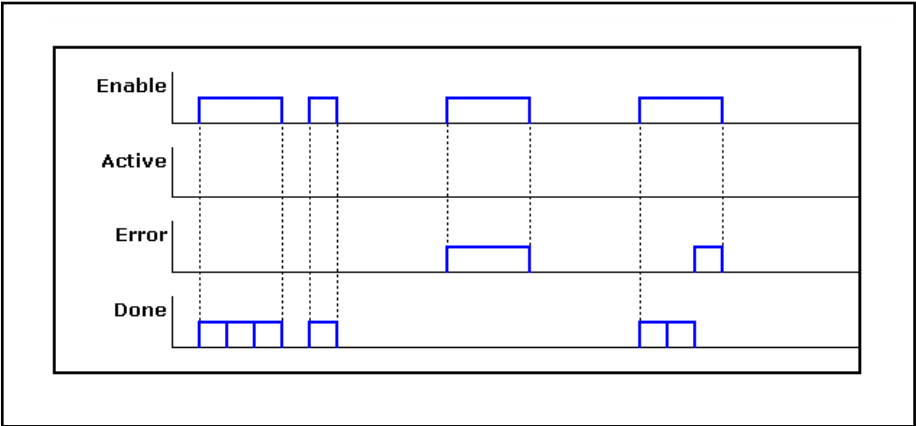


Abb. 3-15: Signal-Zeit-Diagramm IL_DPBAudrateGet

Funktionsbeschreibung

Ermittlung des Betriebszustandes des adressierten PB-Masters. Die "BmState" und "BmError" sind mit 1.3.2 und 1.3.3 zu kodieren.

3.12 IL_DPDevInfoGet

Kurzbeschreibung Der Funktionsbaustein **IL_DPDevInfoGet** ermittelt eine Information über den Betriebszustand des betreffenden Profibus-DP Gerätes.

Zielsystem	Bibliothek/Ordner
MLC	RIL_ProfibusDP_02
MTX	RIL_ProfibusDP_02

Tab. 3-30: Referenztable

Schnittstellenbeschreibung

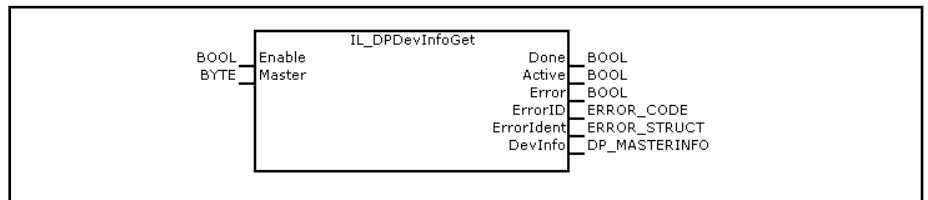


Abb. 3-16: Schnittstellenbeschreibung FB IL_DPDevInfoGet

I/O-Typ	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Funktionsfreigabe
	Master	BYTE	MasterID (siehe oben)
VAR_OUTPUT	Done	BOOL	Fertigmeldung (erfolgreich)
	Active	BOOL	Aktivitätsanzeige
	Error	BOOL	Fertigmeldung (nicht erfolgreich)
	ErrorID	ERROR_CODE	Siehe Kap. 3.15.1 "ErrorID / ERROR_CODE" auf Seite 52
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur aus ErrorTable PB_DP_TABLE
	DevInfo	DP_MASTERINFO	enthält Informationen über den Zustand des Busmasters

Tab. 3-31: I/O-Schnittstelle des FB IL_DPDevInfoGet

Signal-Zeit-Diagramm

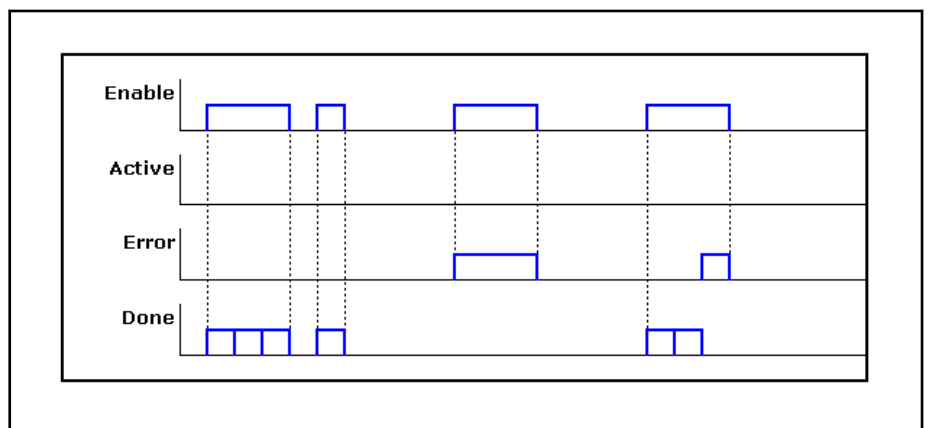


Abb. 3-17: Signal-Zeit-Diagramm IL_DPDevInfoGet

3.13 IL_DPPrjSlaveListGet

Kurzbeschreibung Der Funktionsbaustein **IL_DPPrjSlaveListGet** liefert die Liste der projizierten Slaves. Die Liste enthält alle in der Masterkonfigurationsdatei vorhandenen Slaves.

Zielsystem	Bibliothek/Ordner
MLC	RIL_ProfibusDP_02
MTX	RIL_ProfibusDP_02

Tab. 3-32: Referenztabelle

Schnittstellenbeschreibung

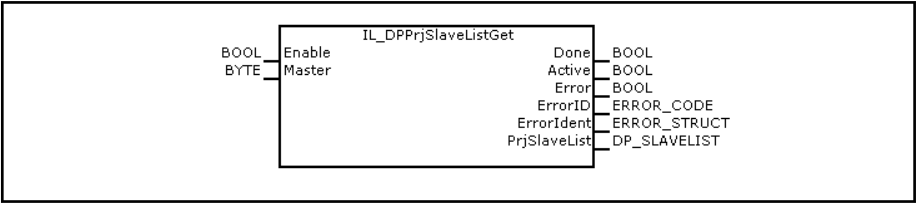


Abb. 3-18: Schnittstellenbeschreibung FB IL_DPPrjSlaveListGet

I/O-Typ	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Funktionsfreigabe
	Master	BYTE	MasterID (siehe oben)
VAR_OUTPUT	Done	BOOL	Fertigmeldung (erfolgreich)
	Active	BOOL	Aktivitätsanzeige
	Error	BOOL	Fertigmeldung (nicht erfolgreich)
	ErrorID	ERROR_CODE	Siehe Kap. 3.15.1 "ErrorID / ERROR_CODE" auf Seite 52
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur aus ErrorTable PB_DP_TABLE
	PrjSlaveList	DP_SLAVELIST	Bitliste mit gesetztem Bit für jeden Diagnose meldenden Slave Siehe Kap. 3.2.5 "DP_SLAVELIST, Array" auf Seite 35

Tab. 3-33: I/O-Schnittstelle des FB IL_DPPrjSlaveListGet

Signal-Zeit-Diagramm

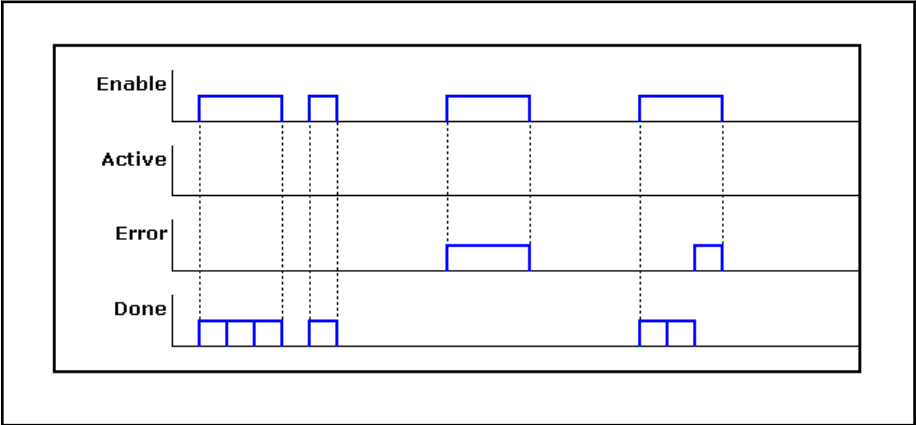


Abb. 3-19: Signal-Zeit-Diagramm IL_DPPrjSlaveListGet

Funktionsbeschreibung Für jeden projizierten Slave eines Masters wird ein Bit in der Bitliste gesetzt. Der Funktionsbaustein stützt sich auf die vorhandenen Konfigurationsdaten.

3.14 IL_DPSlaveDiagListGet

Kurzbeschreibung Der Funktionsbaustein **IL_DPSlaveDiagListGet** liefert die Liste der Slaves, die eine Diagnose melden.

Zielsystem	Bibliothek/Ordner
MLC	RIL_ProfibusDP_02
MTX	RIL_ProfibusDP_02

Tab. 3-34: Referenztafel

Schnittstellenbeschreibung

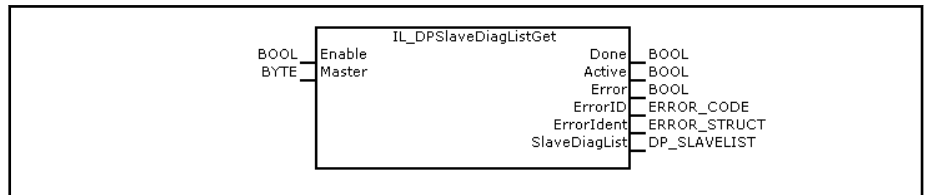


Abb. 3-20: Schnittstellenbeschreibung FB IL_DPSlaveDiagListGet

I/O-Typ	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Funktionsfreigabe
	Master	BYTE	MasterID (siehe oben)
VAR_OUTPUT	Done	BOOL	Fertigmeldung (erfolgreich)
	Active	BOOL	Aktivitätsanzeige
	Error	BOOL	Fertigmeldung (nicht erfolgreich)
	ErrorID	ERROR_CODE	Siehe Kap. 3.15.1 "ErrorID / ERROR_CODE" auf Seite 52
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur aus ErrorTable PB_DP_TABLE
	SlaveDiagList	DP_SLAVELIST	Bitliste mit gesetztem Bit für jeden Diagnose meldenden Slave Siehe Kap. 3.2.5 "DP_SLAVELIST, Array" auf Seite 35

Tab. 3-35: I/O-Schnittstelle des FB IL_DPSlaveDiagListGet

Signal-Zeit-Diagramm

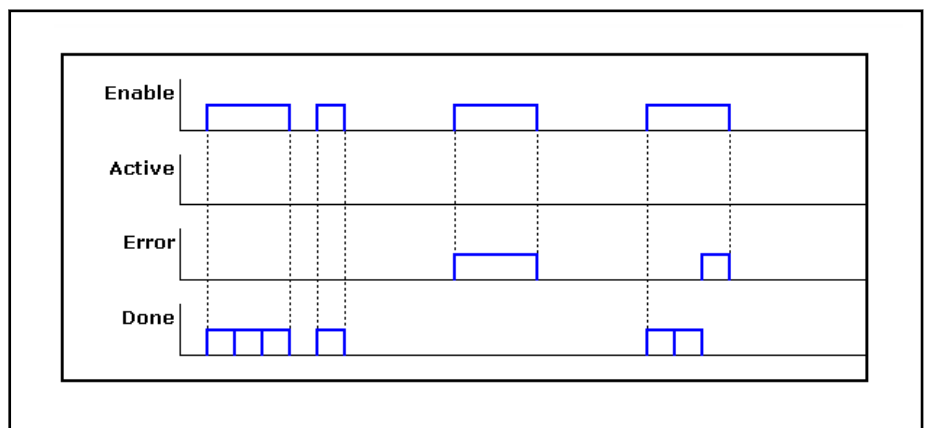


Abb. 3-21: Signal-Zeit-Diagramm IL_DPSlaveDiagListGet

Funktionsbeschreibung

In der "SlaveDiagList" wird für jeden Diagnose meldenden Slave ein Bit gesetzt. Damit können die Diagnosen gezielt gelesen werden. Es werden keine DP-Dienste durch diesen Baustein ausgelöst. Es werden im Master vorhandene Daten ausgewertet.

3.15 Fehlerkennungen

3.15.1 ErrorID / ERROR_CODE

Bibliothek	Bereich
RIL_CommonTypes.lib	Datatypes of POU diagnosis - includes

Tab. 3-36: *ERROR_CODE*, definiert in *RIL_CommonTypes*

Enumerator	Code	Beschreibung
NONE_ERROR	16#0000	Kein Fehlercode verfügbar
INPUT_INVALID_ERROR	16#0001	Ungültige Eingangsbelegung
COMMUNICATION_ERROR	16#0002	Kommunikationsfehler
RESOURCE_ERROR	16#0003	Quelle nicht vorhanden
ACCESS_ERROR	16#0004	Fehlerhafter oder ungültiger bei Zugriff auf Daten
STATE_MACHINE_ERROR	16#0005	Zustandsmaschine auf ungültigen Wert
INPUT_RANGE_ERROR	16#0006	Der Wert einer oder mehrerer Eingänge liegt außerhalb der definierten Grenzen
CALCULATION_ERROR	16#0007	Berechnungsfehler
DEVICE_ERROR	16#0008	Antriebsfehler
OTHER_ERROR	16#7FFE	Undefinierter Fehler (keiner der anderen IDs zuordenbar)
SYSTEM_ERROR	16#7FFF	Systemfehler

Tab. 3-37: *Mögliche ErrorIDs*

3.15.2 ErrorIdent

Aufbau ErrorIdent ist eine Struktur aus drei Elementen. Die Struktur ist mit "0" vorbelegt.

- Error_Table: PB_DP_TABLE (16#0130) Profibus-DP Fehler
- ErrorAdditional1
- ErrorAdditional2

ErrorAdditional1 In ErrorAdditional1 wird eine übergeordnete Unterscheidung nach der Fehlerquelle getroffen. Es werden folgende Fehlerquellen unterschieden:

16#0001: Onboard Profibus Device

16#0002: FunctionModule (FM) Profibus Device

16#0003: netXDevice

16#0005: Spezifischer Fehler, weitere Informationen finden Sie in der Profibus-DP Norm, [Kap. 3.15.6 "Error_Code_DP" auf Seite 55](#) oder in der gerätespezifischen Dokumentation

16#0100: Function not supported

16#0101: Device(Master) not found

16#0102: Slave not configured

16#0103: IO-driver not ready

16#0104: Timeout-Error

16#0107: Parameter-Error

ErrorAdditional2

Für "Onboard Profibus Device" ist ErrorAdditional2 ebenfalls in 4 Bytes unterteilt.

Die Bytes haben folgende Bedeutung:

Byte-Nr.	Bedeutung	Beschreibung
Byte 3	Error_Source	Hier wird nach Fehlerursprung unterschieden: 16#00 Profibus (Slave) 16#10 Masterstack 16#20 Profibus FDL-Layer
Byte 2	Error_Code_DP Error_Code_OB Error_Code_FDL	Für Error_Source = 16#00 Für Error_Source = 16#10 Für Error_Source = 16#20
Byte 1	AddInfo_1	Reserved
Byte 0	AddInfo_2	Reserved

Tab. 3-38: Codierung des Onboard Profibus

Für "FunctionModule (FM) Profibus Device" ist ErrorAdditional2 ebenfalls in 4 Bytes unterteilt.

Die Bytes haben folgende Bedeutung:

Byte-Nr.	Bedeutung	Beschreibung
Byte 3	FM_Error_Code	Fehlercode des FM-Masters
Byte 2	Error_Code_DP	Siehe folgende Tabellen
Byte 1	Error_Code_1	DP-User specific
Byte 0	AddInfo_1	Reserved

Tab. 3-39: Codierung des Funktionsmoduls

3.15.3 FM_Error_Code

Die folgende Fehlerbeschreibung entspricht der "Error Response Definitions" des Hilscher Profibus-DP Masters.

FM_Error_Code	Fehler
16#02	Kein Speicher oder Buffer auf dem Slave für diesen Dienst vorhanden
16#03	Der Slave unterstützt keine DPV1-Dienste
16#09	Der Slave hat keine Daten gesendet
16#11	Der Slave hat nicht geantwortet oder ist nicht am Bus
16#12	DP-Master ist nicht am Profibus (Verkabelung prüfen)
16#19	Der Slave arbeitet nicht DPV1-konform
16#36	Der Zugriff wurde vom Slave abgelehnt. Error_Code_DP auswerten!
16#81	DPV1 ist auf dem Master nicht konfiguriert
16#82	Der Slave hat mit nicht plausiblen Parametern geantwortet

16#83	Es ist schon ein Dienst in Bearbeitung; keine parallelen Dienste erlaubt
16#84	Datengröße übersteigt die konfigurierte Größe
16#85	Falsche Parameter im Request
16#9a	Unbekanntes Kommando
16#F0	Ungültiger Zustand

Tab. 3-40: Funktionsmodul Error-Code

3.15.4 Error_Code_OB

Error_Code_OB	Fehler
16#11	Ungültige Auftrags-Parameter
16#23	RequestList voll
16#25	Fehler bei SemTake
16#31	Nicht erlaubter Aufruf
16#32	Ungültige Aufrufparameter
16#33	Ungültige Datenlänge
16#34	Fehlerhafter Aufrufzustand
16#35	Slave nicht konfiguriert
16#36	Slave konfiguriert, aber nicht im zyklischen Betrieb
16#61	DPV1-Request an nicht-DPV1-Slave
16#62	Slave antwortet nicht innerhalb Timeout
16#63	DPV1-Telegramm-Formatfehler
16#64	Auftrag wurde zurückgezogen
16#65	Zugehörigen RQB nicht gefunden
16#66	Ungültiger Parameter
16#67	Unbekannter AMPRO2-Opcode

Tab. 3-41: Onboard Error-Code

3.15.5 Error_Code_FDL

Error_Code_FDL	Fehler
16#61	FE: Format-Error in a Request-APDU
16#62	NI: Service not implemented
16#63	AD: Access denied
16#64	EA: Area to large (Up-/Download)
16#65	LE: Data-block-length to large (Up-/Download)
16#66	RE: Format-Error in a Response-APDU
16#67	IP: Invalid Parameter
16#68	SC: Sequence Conflict
16#69	SE: Sequence Error

Error_Code_FDL	Fehler
16#6A	NE: Area non-existent
16#60	Kein Slave gefunden
16#6B	DI: Data Incomplete
16#6C	NC: Master Parameter Set not compatible

Tab. 3-42: Fieldbus Data Link Layers (FDL) Error-Code

3.15.6 Error_Code_DP

Aufbau Die Bedeutung von "Error_Code_DP" entspricht dem "Error_Code_1" aus der DPV1-Norm. Die Bits 4..7 des Fehlerbytes bilden die "Error_Class", die Bits 0..3 den "Error_Code".

7	6	5	4	3	2	1	0	Meaning
								Error Code
								Error Class

Tab. 3-43: Aufbau Error_Code_DP

Error_Class	Meaning	Error_Code
0 to 9	reserved ¹⁾	
10	Application	0 = read error 1 = write error 2 = module failure 3 to 7 = reserved ¹⁾ 8 = version conflict 9 = feature not supported 10 to 15 = user specific
11	Access	0 = invalid index 1 = write length error 2 = invalid slot 3 = type conflict 4 = invalid area 5 = state conflict 6 = access denied 7 = invalid range 8 = invalid parameter 9 = invalid type 10 to 15 = user specific

Error_Class	Meaning	Error_Code
12	resource	0 = read constrain conflict 1 = write constrain conflict 2 = resource busy 3 = resource unavailable 4 to 7 = reserved ¹⁾ 8 to 15 = user specific
13 to 15	User specific	

¹⁾ reserved values are intended to be passed unchanged to the user

Tab. 3-44: DP Error-Code

Additional Info: User(Slave) specific

4 RIL_ProfibusDPSlave.library

4.1 Einführung und Übersicht

Die hier beschriebene Funktionalität realisiert die azyklischen DPV1-Dienste READ und WRITE beim Profibus-DP-Slave. Damit werden Zugriffe auf Datenobjekte der Slave-Anwendungsebene ermöglicht.

Die Funktion erfordert beim **READ**-Dienst die kombinierte Nutzung von zwei Funktionsbausteinen (FBs).

Mit dem FB IL_PBDPSlaveDPV1Polling wird die Aktivität einer DPV1-Service-Anforderung zyklisch abgefragt. Eine aktive "READ"-Anforderung wird durch das Anwenderprogramm auf ihre Zulässigkeit geprüft und durch Aufruf des FBs IL_PBDPSlaveDPV1Response beantwortet.

Dabei erfolgt der Zugriff auf das betreffende Datenobjekt in der Weise, dass dem FB IL_PBDPSlaveDPV1Response als Parameter ein Pointer auf das Datenobjekt und die gültige Datenlänge übergeben wird.

Das Kopieren der Daten erfolgt durch den FB.

Ein "WRITE"-Dienst erfordert die kombinierte Nutzung von drei Funktionsbausteinen.

Mit dem FB IL_PBDPSlaveDPV1Polling wird die Aktivität einer DPV1-Service-Anforderung zyklisch abgefragt. Eine aktive "WRITE"-Anforderung wird durch das Anwenderprogramm auf ihre Zulässigkeit geprüft. Gegebenenfalls werden die Nutzdaten durch Aufruf des FBs IL_PBDPSlaveDPV1GetWriteData in das Zielobjekt kopiert. Dazu wird dem FB IL_PBDPSlaveDPV1GetWriteData als Parameter ein Pointer auf das Datenobjekt und die gültige Datenlänge übergeben.

Das Kopieren der Daten erfolgt durch den FB.

Die Beantwortung erfolgt wieder durch den Aufruf des FBs IL_PBDPSlaveDPV1Response.

Die Datenlänge pro Zugriff kann maximal 240 Byte betragen (Grenze durch V1-Kanal des Profibus DP). Größere Datenmengen können z.B. über eine Fragmentierung über mehrere Feldbusobjekte à 240 Bytes erreicht werden.

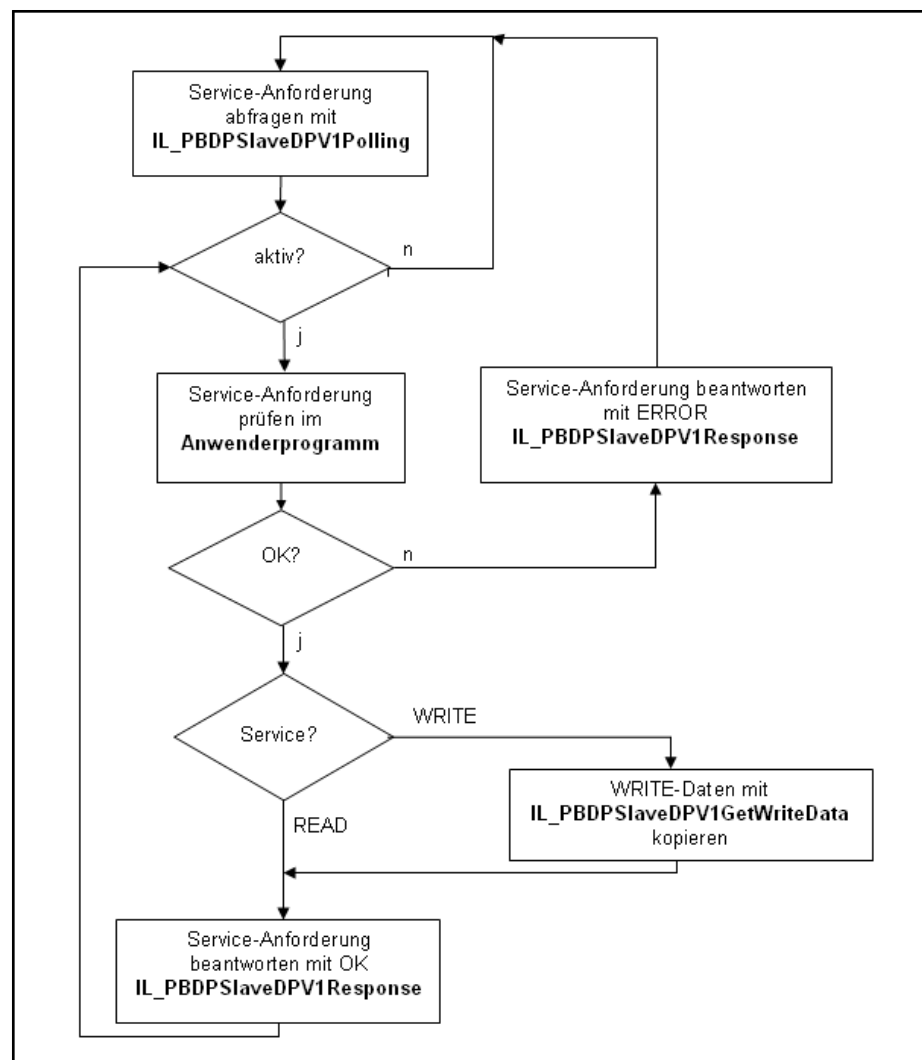


Abb. 4-1: Zusammenwirken der Funktionsbausteine der RIL_ProfibusDPSlave-Bibliothek

Die Bibliothek enthält folgende Bestandteile:

- [FB IL_PBDPSlaveDPV1Polling auf Seite 59](#), mit dem Polling-FB wird die Aktivität einer DPV1-Service-Anforderung abgefragt. Der Aufruf erfolgt zyklisch.
- [FB IL_PBDPSlaveDPV1Response auf Seite 61](#), mit dem Response-FB wird eine aktive DPV1-Service-Anforderung beantwortet. Der Aufruf erfolgt abhängig vom Ergebnis des Polling-FB.
- [FB IL_PBDPSlaveDPV1GetWriteData auf Seite 64](#), mit dem GetWrite-Data-FB werden bei einer DPV1-WRITE-Service-Anforderung die Daten in das Anwendungsobjekt kopiert.
- [FB IL_FBUS_SLAVE_RESULT auf Seite 68](#), dient dazu, eine Kennung des Ergebnisses der Datenübertragung im V1-Kanal zu kodieren.
- [FB IL_FBUS_SLAVE_SERVICE auf Seite 67](#), dient dazu, eine Service-Kennung des DP V1-Kanals darzustellen.

Auswahl des Slaves Der Aufzählungstyp IL_BUSSLAVE aus der RIL_FieldbusTypes.library dient dazu, den jeweiligen ProfibusDP-Slave auszuwählen.

Siehe auch: [FB IL_BUSSLAVE auf Seite 67](#)

4.2 IL_PBDPSlaveDPV1Polling

Kurzbeschreibung Mit dem DPV1-Polling-FB wird die Aktivität einer DPV1-Service-Anforderung abgefragt. Eine aktive Service-Anforderung zeigt der FB durch die Ausgangsparameter Service, Slot, Index, ServiceID und DataLength an.

Der Parameter Service definiert die Art des Zugriffs (Read bzw. Write) oder ist Idle, wenn keine Service-Anforderung aktiv ist.

Die Parameter Slot und Index definieren das Datenobjekt, auf das zugegriffen werden soll.

ServiceID ist ein interner Ident-Parameter, der beim Aufruf des Response- sowie des GetWriteData-FBs wieder identisch zu übergeben ist.

DataLength gibt die Zugriffslänge an, d.h. die Anzahl der zu schreibenden Daten bzw. die Anzahl der maximal zu lesenden Daten.

Aufgabe des Anwendungsprogramms ist es, eine aktive Service-Anforderung auf ihre Zulässigkeit zu prüfen und durch Aufruf des FBs IL_PBDPSlaveDPV1Response (s.u.) zu beantworten.

Zielsystem	Bibliothek /Ordner
MLC	RIL_ProfibusDPSlave
MTX	RIL_ProfibusDPSlave

Tab. 4-1: Referenztable

Schnittstellenbeschreibung

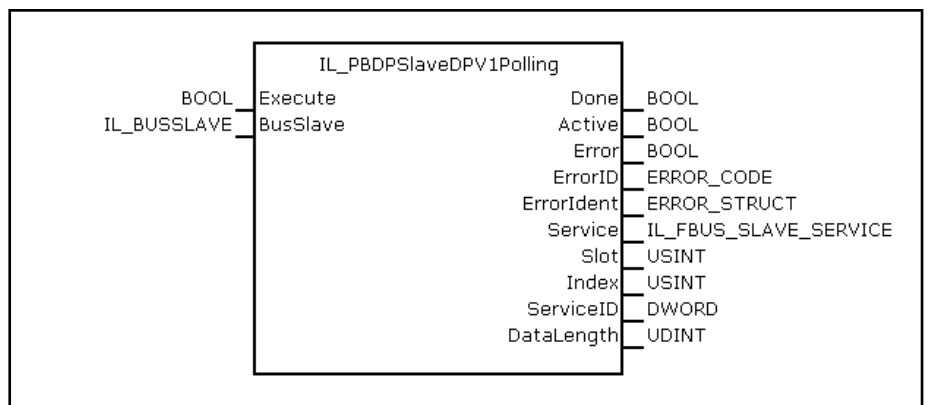


Abb. 4-2: Schnittstelle FB IL_PBDPSlaveDPV1Polling

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins
	BusSlave	IL_BUSSLAVE	Instanz des Slaves in der Reihenfolge des Konfigurationbaumes. Ist nur ein Profibus DP Slave vorhanden, besitzt dieser immer den Wert IL_BUSSLAVE_0. FB IL_BUSSLAVE auf Seite 67
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall

E/A-Typ	Name	Datentyp	Beschreibung
	ErrorIdent	ERROR_STRUCT	Detaildiagnose
	Service	IL_FBUS_SLAVE_SERVICE	Service-Kennung (Read, Write oder Idle) FB_IL_FBUS_SLAVE_SERVICE auf Seite 67
	Slot	USINT	Adressparameter Slot
	Index	USINT	Adressparameter Index
	ServiceID	DWORD	Identparameter ServiceID
	DataLength	UDINT	Datenlänge (es sind Datenlängen ≤ 240 Bytes möglich)

Tab. 4-2: E/A-Schnittstelle FB_IL_PBDPSlaveDPV1Polling

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
BusSlave	IL_BUSSLAVE	IL_BUSSLAVE_0	IL_BUSSLAVE_4	IL_BUSSLAVE_0	Zustand TRUE von Enable

Tab. 4-3: Min-/Max und Default-Werte der Eingänge IL_PBDPSlaveDPV1Polling

Funktionsbeschreibung

- Ein wirksamer Aufruf von IL_PBDPSlaveDPV1Polling und die Übernahme des Parameters BusSlave erfolgt mit steigender Flanke von Execute.
- Bei Execute = FALSE ist der Aufruf ohne Wirkung. In diesem Fall werden alle Ausgangsparameter auf '0' gesetzt.
- Das Ergebnis des Aufrufs wird zunächst in den Ausgangsparametern Done, Active und Error angezeigt. Bei Active = FALSE ist der Aufruf abgeschlossen und die anderen Parameter sind auszuwerten.
- Done = TRUE und Error = FALSE kennzeichnen einen fehlerfreien Verlauf des Aufrufs.
- Done = FALSE und Error = TRUE zeigen einen fehlerhaften Verlauf des Aufrufs an. In diesem Fall beschreiben die Parameter ErrorID und ErrorIdent die Fehlerursache.
- Nach fehlerfreiem Verlauf des Aufrufs sind die effektiven Ausgangsparameter Service, Slot, Index, ServiceID und DataLength gültig und können durch die aufrufende Instanz ausgewertet werden.
- Bei fehlerhaftem Verlauf sind die effektiven Ausgangsparameter ungültig.
- Bei gültigen Ausgangsparametern wird in Service die Art der Anforderung (READ, WRITE oder IDLE) angezeigt. Bei READ oder WRITE definieren Slot und Index das Objekt, auf das zugegriffen werden soll und DataLength die Zugriffslänge.
- Bei Service = IDLE liegt keine aktive Service-Anforderung vor. Eine Behandlung durch das Anwendungsprogramm ist in diesem Fall nicht erforderlich.

Fehlerbehandlung

Error = TRUE kennzeichnet einen fehlerhaften Verlauf des Aufrufs.

Die Fehler des Bausteines sind in der Fehlertabelle PB_DP_TABLE (ERROR_TABLE = 16#0130) zu finden.

In diesem Fall definieren ErrorID und ErrorIdent (Additional1 und Additional2) die Fehlerursache.

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#1001	BusSlave-Instanz	Ungültige BusSlave-Instanz
STATE_MACHINE_ERROR	16#1011	0	mehrfaches Pollen eines aktiven Service
SYSTEM_ERROR	16#1021	0	Slave-Interface nicht verfügbar
SYSTEM_ERROR	16#1022	Interface-Fehler	Slave-Interface-Fehler

Tab. 4-4: Fehlercodes FB IL_PBDPSlaveDPV1Polling

4.3 IL_PBDPSlaveDPV1Response

Kurzbeschreibung Mit dem DPV1-Response-FB wird eine zuvor mit IL_PBDPSlaveDPV1Polling (s.o.) erkannte aktive DPV1-Service-Anforderung beantwortet. Abhängig davon, ob die Anforderung zulässig ist oder nicht, wird der FB mit unterschiedlichen Parametern ValueAdr, SizeOfValue und Result aufgerufen.

Bei zulässiger "READ"-Service-Anforderung wird dem FB im Parameter ValueAdr ein Pointer auf das Datenobjekt und in SizeOfValue die entsprechende Datenlänge übergeben. Das kopieren der Daten erfolgt durch den Response-FB.

Beim "WRITE"-Service erfolgt das kopieren der Daten vor dem Aufruf des Response-FBs durch den FB IL_PBDPSlaveDPV1GetWriteData (s.u.). Der Parameter ValueAdr ist in diesem Fall ohne Bedeutung.

Eine unzulässige Anforderung wird mit ValueAdr = 0, SizeOfValue = 0 und einer entsprechenden Fehlerkennung in Result beantwortet.

Zielsystem	Bibliothek /Ordner
MLC	RIL_ProfibusDPSlave
MTX	RIL_ProfibusDPSlave

Tab. 4-5: Referenztable

Schnittstellenbeschreibung

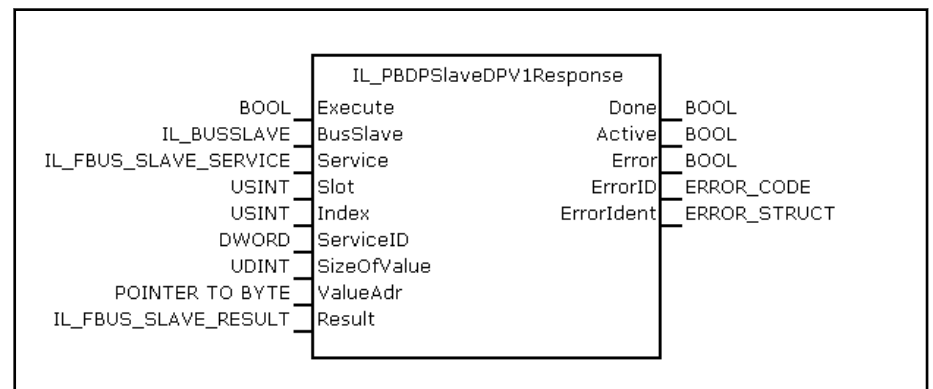


Abb. 4-3: Schnittstelle IFB L_PBDPSlaveDPV1Response

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins
	BusSlave	IL_BUSSLAVE	Instanz des Slaves in der Reihenfolge des Konfigurationbaumes. Ist nur ein Profibus DP Slave vorhanden, besitzt dieser immer den Wert IL_BUSSLAVE_0. FB IL_BUSSLAVE auf Seite 67
	Service	IL_FBUS_SLAVE_SERVICE	Kennung des Service, der abgeschlossen werden soll. Dieser Wert wird mit dem aktiven Service verglichen. FB IL_FBUS_SLAVE_SERVICE auf Seite 67
	Slot	USINT	Adressparameter Slot des Service, der abgeschlossen werden soll. Dieser Wert wird mit dem aktiven Slot verglichen.
	Index	USINT	Adressparameter Index des Service, der abgeschlossen werden soll. Dieser Wert wird mit dem aktiven Index verglichen.
	ServiceID	DWORD	Identparameter ServiceID des Service, für den Daten kopiert werden sollen (Parameter von IL_PBDPSlaveDPV1Polling). Dieser Wert wird mit der aktiven ServiceID verglichen.
	SizeOfValue	UINT	Maximale Größe des Datenbereichs von ValueAdr
	ValueAdr	POINTER TO BYTE	Pointer auf Daten
	Result	IL_FBUS_SLAVE_RESULT	Ergebnis des Zugriffs (wird über den V1-Response an den anfragenden Profibus-Master zurückgesendet) FB IL_FBUS_SLAVE_RESULT auf Seite 68
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCT	Detailldiagnose

Tab. 4-6: E/A-Schnittstelle FB IL_PBDPSlaveDPV1Response

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusSlave	IL_BUSSLAVE	IL_BUSSLAVE_0	IL_BUSSLAVE_4	IL_BUSSLAVE_0	Steigende Flanke von Execute
Service	IL_FBUS_SLAVE_SERVICE	IL_SLAVE_SERVICE_IDLE	IL_SLAVE_SERVICE_WRITE	IL_SLAVE_SERVICE_IDLE	Steigende Flanke von Execute
Slot	USINT	0	255	0	Steigende Flanke von Execute
Index	USINT	0	255	0	Steigende Flanke von Execute
ServiceID	DWORD	0	16#FFFFFFFF	0	Steigende Flanke von Execute
SizeOfValue	UDINT	0 (1)	240	0	Steigende Flanke von Execute

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
ValueAdr	POINTER TO BYTE		n.def.	0	Steigende Flanke von Execute
Result	IL_FBUS_SLAVE_RESULT	IL_SLAVE_RESULT_OK	n.def.	IL_SLAVE_RESULT_OK	Steigende Flanke von Execute

Tab. 4-7: Min-/Max und Default-Werte der Eingänge IL_PBDPSlaveDPV1Response

Funktionsbeschreibung

- Ein wirksamer Aufruf von IL_PBDPSlaveDPV1Response und die Übernahme der Eingangsparameter erfolgt mit steigender Flanke von Execute.
- Bei Execute = FALSE ist der Aufruf ohne Wirkung. In diesem Fall werden alle Ausgangsparameter auf '0' gesetzt.
- Die Eingangsparameter Service, Slot und Index definieren das Datenobjekt, auf das zugegriffen wurde und sind zwingend identisch zu den entsprechenden Ausgangsparametern von IL_PBDPSlaveDPV1Polling.
- ServiceID ist der Identparameter des aktiven Service und ist zwingend identisch zu dem entsprechenden Ausgangsparameter von IL_PBDPSlaveDPV1Polling
- SizeOfValue ist bei einem WRITE-Service identisch zu DataLength des IL_PBDPSlaveDPV1Polling, kann aber bei einem READ-Service auch kleiner sein.
- ValueAdr ist ein Pointer auf das adressierte Datenobjekt und ist '0' im Fall einer ungültigen Service-Anforderung.
- Result definiert einen Fehlerfall beim Zugriff auf das angeforderte Datenobjekt.
- Das Ergebnis des Aufrufs wird zunächst in den Ausgangsparametern Done, Active und Error angezeigt.
Bei Active = FALSE ist der Aufruf abgeschlossen und die anderen Parameter sind auszuwerten.
- Done = TRUE und Error = FALSE kennzeichnen einen fehlerfreien Verlauf des Aufrufs.
- Done = FALSE und Error = TRUE zeigen einen fehlerhaften Verlauf des Aufrufs an.
In diesem Fall beschreiben die Parameter ErrorID und ErrorIdent die Fehlerursache.

Fehlerbehandlung

Error = TRUE kennzeichnet einen fehlerhaften Verlauf des Aufrufs.

Die Fehler des Bausteines sind in der Fehlertabelle PB_DP_TABLE (ERROR_TABLE = 16#0130) zu finden.

In diesem Fall definieren ErrorID und ErrorIdent (Additional1 und Additional2) die Fehlerursache.

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#2001	BusSlave-Instanz	Ungültige BusSlave-Instanz
INPUT_RANGE_ERROR	16#2002	Service	Ungültiger Service
INPUT_RANGE_ERROR	16#2003	Slot	Ungültiger Slot
INPUT_RANGE_ERROR	16#2004	Index	Ungültiger Index

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#2005	ServiceID	Ungültige ServiceID
INPUT_RANGE_ERROR	16#2006	SizeOfValue	Ungültige SizeOfValue
INPUT_RANGE_ERROR	16#2007	ValueAdr	Ungültige ValueAdr
INPUT_RANGE_ERROR	16#2008	Result	Result ungültig
STATE_MACHINE_ERROR	16#2011	0	Kein Service aktiv
COMMUNICATION_ERROR	16#2012	Timeout [ms]	Service nicht rechtzeitig bearbeitet
SYSTEM_ERROR	16#2021	0	Slave-Interface nicht verfügbar
SYSTEM_ERROR	16#2022	Interface-Fehler	Slave-Interface-Fehler

Tab. 4-8: Fehlercodes FB IL_PBDPSlaveDPV1Response

4.4 IL_PBDPSlaveDPV1GetWriteData

Kurzbeschreibung Mit dem DPV1-GetWriteData-FB werden bei einer zuvor mit IL_PBDPSlaveDPV1Polling erkannten aktiven und zulässigen DPV1-WRITE-Service-Anforderung die Nutzdaten in das betreffende Zielobjekt kopiert.

Dazu werden dem FB im Parameter ValueAdr ein Pointer auf das Datenobjekt und in SizeOfValue die entsprechende Datenlänge übergeben.

Zielsystem	Bibliothek /Ordner
MLC	RIL_ProfibusDPSlave
MTX	RIL_ProfibusDPSlave

Tab. 4-9: Referenztabelle

Schnittstellenbeschreibung

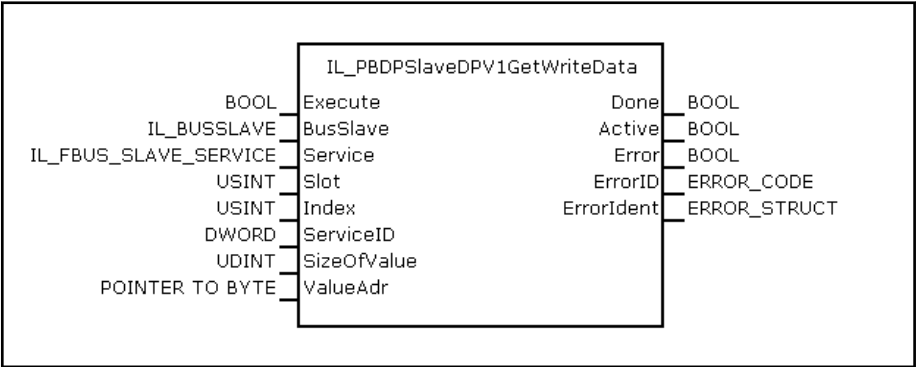


Abb. 4-4: Schnittstelle FB IL_PBDPSlaveDPV1GetWriteData

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusSlave	IL_BUSSLAVE	Instanz des Slaves in der Reihenfolge des Konfigurationsbaumes. Ist nur ein Profibus DP Slave vorhanden, besitzt dieser immer den Wert IL_BUSSLAVE_0. FB IL_BUSSLAVE auf Seite 67

E/A-Typ	Name	Datentyp	Beschreibung
	Service	IL_FBUS_SLAVE_SERVICE	Kennung des Service, für den Daten kopiert werden sollen. Dieser Wert wird mit dem aktiven Service verglichen. FB IL_FBUS_SLAVE_SERVICE auf Seite 67
	Slot	USINT	Adressparameter Slot des Service, für den Daten kopiert werden sollen. Dieser Wert wird mit dem aktiven Slot verglichen.
	Index	USINT	Adressparameter Index des Service, für den Daten kopiert werden sollen. Dieser Wert wird mit dem aktiven Index verglichen.
	ServiceID	DWORD	Identparameter ServiceID des Service, für den Daten kopiert werden sollen (Parameter von IL_PBDPSlaveDPV1Polling). Dieser Wert wird mit der aktiven ServiceID verglichen.
	SizeOfValue	UINT	Maximale Größe des Datenbereichs von ValueAdr
	ValueAdr	POINTER TO BYTE	Pointer auf Daten
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCT	Detaildiagnose

Tab. 4-10: E/A-Schnittstelle FB IL_PBDPSlaveDPV1GetWriteData

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusSlave	IL_BUSSLAVE	IL_BUSSLAVE_0	IL_BUSSLAVE_4	IL_BUSSLAVE_0	Steigende Flanke von Execute
Service	IL_FBUS_SLAVE_SERVICE	IL_SLAVE_SERVICE_IDLE	IL_SLAVE_SERVICE_WRITE	IL_SLAVE_SERVICE_IDLE	Steigende Flanke von Execute
Slot	USINT	0	255	0	Steigende Flanke von Execute
Index	USINT	0	255	0	Steigende Flanke von Execute
ServiceID	DWORD	0	16#FFFFFFFF		Steigende Flanke von Execute
SizeOfValue	UDINT	0 (1)	240	0	Steigende Flanke von Execute
ValueAdr	POINTER TO BYTE	n.def.	n.def.	0	Steigende Flanke von Execute

Tab. 4-11: Min-/Max und Default-Werte der Eingänge IL_PBDPSlaveDPV1GetWriteData

Funktionsbeschreibung

- Ein wirksamer Aufruf von IL_PBDPSlaveDPV1GetWriteData und die Übernahme der Eingangsparameter erfolgt mit steigender Flanke von Execute.

- Bei Execute = FALSE ist der Aufruf ohne Wirkung. In diesem Fall werden alle Ausgangsparameter auf '0' gesetzt.
- Bei Execute = TRUE (konstant) bleiben die Ausgangsparameter erhalten.
- Die Eingangsparameter Service, Slot und Index definieren das Datenobjekt, auf das zugegriffen wurde und sind zwingend identisch zu den entsprechenden Ausgangsparametern von IL_PBDPSlaveDPV1Polling.
- ServiceID ist der Identparameter des aktiven Service und ist zwingend identisch zu dem entsprechenden Ausgangsparameter von IL_PBDPSlaveDPV1Polling.
- SizeOfValue ist bei einem WRITE-Service identisch zu DataLength des IL_PBDPSlaveDPV1Polling, kann aber bei einem READ-Service auch kleiner sein.
- ValueAdr ist ein Pointer auf das Zielobjekt.
- Das Ergebnis des Aufrufs wird zunächst in den Ausgangsparametern Done, Active und Error angezeigt. Bei Active = FALSE ist der Aufruf abgeschlossen und die anderen Parameter sind auszuwerten.
- Done = TRUE und Error = FALSE kennzeichnen einen fehlerfreien Verlauf des Aufrufs.
- Done = FALSE und Error = TRUE zeigen einen fehlerhaften Verlauf des Aufrufs an. In diesem Fall beschreiben die Parameter ErrorID und ErrorIdent die Fehlerursache.

Fehlerbehandlung Error = TRUE kennzeichnet einen fehlerhaften Verlauf des Aufrufs.

Die Fehler des Bausteines sind in der Fehlertabelle PB_DP_TABLE (ERROR_TABLE = 16#0130) zu finden.

In diesem Fall definieren ErrorID und ErrorIdent (Additional1 und Additional2) die Fehlerursache.

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#3001	BusSlave-Instanz	Ungültige BusSlave-Instanz
INPUT_RANGE_ERROR	16#3002	Service	Ungültiger Service
INPUT_RANGE_ERROR	16#3003	Slot	Ungültiger Slot
INPUT_RANGE_ERROR	16#3004	Index	Ungültiger Index
INPUT_RANGE_ERROR	16#3005	ServiceID	Ungültige ServiceID
INPUT_RANGE_ERROR	16#3006	SizeOfValue	Ungültige SizeOfValue
INPUT_RANGE_ERROR	16#3007	ValueAdr	Ungültige ValueAdr
STATE_MACHINE_ERROR	16#3011	0	Kein Service aktiv
COMMUNICATION_ERROR	16#3012	Timeout [ms]	Service nicht rechtzeitig bearbeitet
SYSTEM_ERROR	16#3021	0	Slave-Interface nicht verfügbar
SYSTEM_ERROR	16#3022	Interface-Fehler	Slave-Interface-Fehler

Tab. 4-12: Fehlercodes FB IL_PBDPSlaveDPV1GetWriteData

4.5 IL_BUSSLAVE

Kurzbeschreibung Der Aufzählungstyp IL_BUSSLAVE dient dazu, den jeweiligen Profibus DP Slave auszuwählen. In einem Steuerungssystem können neben der onboard Profibus-Schnittstelle über Funktionsmodule weitere Slaves vorhanden sein.

Die Profibus DP Slaves werden anhand ihrer aufsteigenden Reihenfolge in der Konfiguration unterschieden.

Die Indizierung der Slaves geschieht dabei in der Reihenfolge

- Onboard Slave
- Funktionsmodul 1
- Funktionsmodul 2
- Funktionsmodul 3
- Funktionsmodul 4

D.h. ist beispielsweise der onboard Slave kein Profibus Slave und ist auf dem FM1 ein Profibus Slave konfiguriert, so erhält der Slave auf dem FM1 den Wert IL_BUSSLAVE_0.

Element	Wert	Beschreibung
IL_BUSSLAVE_0	0	Erstes Gerät
IL_BUSSLAVE_1	1	Zweites Gerät
IL_BUSSLAVE_2	2	Drittes Gerät
IL_BUSSLAVE_3	3	Viertes Gerät
IL_BUSSLAVE_4	4	Fünftes Gerät

Tab. 4-13: Elemente des Aufzählungstyps IL_BUSSLAVE

Beispiel zur Adressierung der verschiedenen Instanzen des Profibus Slaves auf einer Steuerung:

Konfiguriert ist:	Adressierung mit:
Onboard: Profibus DP Slave	IL_BUSSLAVE_0
FM1: Profibus DP Master	IL_BUSMASTER_0
FM2: Profibus DP Slave	IL_BUSSLAVE_1
FM3: Profibus DP Slave	IL_BUSSLAVE_2

Tab. 4-14: Beispiel 1 zum Aufzählungstyp IL_BUSSLAVE

4.6 IL_FBUS_SLAVE_SERVICE

Kurzbeschreibung Der Aufzählungstyp IL_FBUS_SLAVE_SERVICE dient dazu, eine Service-Kennung des DP V1-Kanals darzustellen. Hiermit wird dem Anwender angezeigt, welcher Anforderungszustand im V1-Kanal aktuell vorhanden ist.

Die Bits sind als Defines in der Bibliothek "RIL_FIELDBUS_TYPES" enthalten.

Element	Wert	Beschreibung
IL_SLAVE_SERVICE_IDLE	0	Keine aktive Anforderung
IL_SLAVE_SERVICE_READ	1	Lese-Anforderung liegt vor
IL_SLAVE_SERVICE_WRITE	2	Schreib-Anforderung liegt vor

Tab. 4-15: Elemente des Datentyps IL_FBUS_SLAVE_SERVICE

4.7 IL_FBUS_SLAVE_RESULT

Kurzbeschreibung Der Aufzählungstyp IL_FBUS_SLAVE_RESULT dient dazu, eine Kennung des Ergebnisses der Datenübertragung im V1-Kanal zu kodieren. Hiermit gibt der Anwender an, ob die Datenübertragung fehlerfrei war bzw. welcher Fehlercode über den V1-Kanal an den anfragenden Profibus-Master zurückgesendet wird.

Die Bits sind als Defines in der Bibliothek "RIL_FIELDBUS_TYPES" enthalten.

Element	Wert	Beschreibung
IL_SLAVE_RESULT_OK	16#00	Kein Fehler beim Zugriff
IL_SLAVE_RESULT_INVALID_SLOT	16#B2	Fehlerhafter Slot
IL_SLAVE_RESULT_INVALID_INDEX	16#B0	Fehlerhafter Index
IL_SLAVE_RESULT_ACCESS_DENIED	16#B6	Zugriff nicht erlaubt
IL_SLAVE_RESULT_READ_ONLY	16#BA	Schreibzugriff nicht erlaubt
IL_SLAVE_RESULT_READ_ERROR	16#A0	Fehler beim Lesevorgang (auf die SPS-Anwendung bezogen)
IL_SLAVE_RESULT_WRITE_ERROR	16#A1	Fehler beim Schreibvorgang (auf die SPS-Anwendung bezogen)
IL_SLAVE_RESULT_WRITE_LENGTH_ERROR	16#B1	Schreibzugriff mit zu langer Datenlänge (es sollen mehr Daten geschrieben werden, als für das Datenobjekt möglich sind)
IL_SLAVE_RESULT_READ_LENGTH_ERROR	16#BB	Lesezugriff mit zu kurzer Datenlänge (es sollen weniger Daten gelesen werden, als für das Datenobjekt notwendig sind)
IL_SLAVE_RESULT_RESOURCE_BUSY	16#C2	Datenzugriff zur Zeit nicht möglich
IL_SLAVE_RESULT_USER_ERROR_0	16#F0	Bedeutung ist anwendungsspezifisch
IL_SLAVE_RESULT_USER_ERROR_1	16#F1	Bedeutung ist anwendungsspezifisch
IL_SLAVE_RESULT_USER_ERROR_2	16#F2	Bedeutung ist anwendungsspezifisch
IL_SLAVE_RESULT_USER_ERROR_3	16#F3	Bedeutung ist anwendungsspezifisch
IL_SLAVE_RESULT_USER_ERROR_4	16#F4	Bedeutung ist anwendungsspezifisch
IL_SLAVE_RESULT_USER_ERROR_5	16#F5	Bedeutung ist anwendungsspezifisch
IL_SLAVE_RESULT_USER_ERROR_6	16#F6	Bedeutung ist anwendungsspezifisch
IL_SLAVE_RESULT_USER_ERROR_7	16#F7	Bedeutung ist anwendungsspezifisch
IL_SLAVE_RESULT_USER_ERROR_8	16#F8	Bedeutung ist anwendungsspezifisch
IL_SLAVE_RESULT_USER_ERROR_9	16#F9	Bedeutung ist anwendungsspezifisch

Element	Wert	Beschreibung
IL_SLAVE_RESULT_USER_ERROR_10	16#FA	Bedeutung ist anwendungsspezifisch
IL_SLAVE_RESULT_USER_ERROR_11	16#FB	Bedeutung ist anwendungsspezifisch
IL_SLAVE_RESULT_USER_ERROR_12	16#FC	Bedeutung ist anwendungsspezifisch
IL_SLAVE_RESULT_USER_ERROR_13	16#FD	Bedeutung ist anwendungsspezifisch
IL_SLAVE_RESULT_USER_ERROR_14	16#FE	Bedeutung ist anwendungsspezifisch
IL_SLAVE_RESULT_USER_ERROR_15	16#FF	Bedeutung ist anwendungsspezifisch

Tab. 4-16: Elemente des Datentyps IL_FBUS_SLAVE_RESULT

5 RIL_ProfinetIO.library

5.1 Einführung und Übersicht

Diese Bibliothek beschreibt die Zugriffsmöglichkeiten des IEC-Programms auf den netX-Profinet IO-Controller.

Bei der Realisierung der Bibliothek wurde auf der RIL_UTILITIES aufgesetzt.

Es wurden die Methoden der Basisfunktionsbausteine (Basisklassen) IL_EnableDoneBase und IL_ExecuteDoneBase verwendet und erweitert oder überschrieben.



Dieser Hinweis gilt für folgende Profinet-Funktionsbausteine:

- Azyklische Kommunikation
- Backup & Restore
- Device Replacement
- IO-Link,
- IM-Data,
- IL_PNIORemoteDeviceStateDetails
- IH_PNIOAck_PSDx_Inline
- IH_PNIOAck_PSDx_S20

Bei der Programmierung ist darauf zu achten, dass nur ein azyklischer Dienst (DCP-Dienste oder Read-Implicit-Dienst) ausgeführt werden kann. Eine neue azyklische Anfrage kann erst dann gestartet werden, wenn die zuvor gestartete Anfrage beendet ist. Starten Sie daher erst einen neuen Funktionsbaustein mit positiver Execute-Flanke, sobald die Bearbeitung aller anderen verwendeten Funktionsbausteine abgeschlossen ist. Die Bearbeitung ist abgeschlossen, sobald der Done- oder Error-Ausgang gesetzt wird.

Anwendungsgebiete und Funktionsbausteine von RIL_ProfinetIO

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine	Beschreibung
Azyklische Dienste (Acyclic Communication)	
IL_PNIOWriteRecord, Seite 81	azyklisches Schreiben
IL_PNIOReadRecord, Seite 73	azyklisches Lesen
IL_PNIOServiceRecord, Seite 76	dient der Abwicklung des Profinet-Dienst-Schnittstellenzugriffs, der z. B. für die IO-Link-Integration oder für PROFIenergy benötigt wird
Sichern und Wiederherstellen (Backup_Restore)	
IL_PNIODeviceBackup, Seite 84	dient zum Sichern der Konfiguration und der Parameter eines Profinet IO-Device auf eine Datei mittels azyklischen Lesens
IL_PNIODeviceRestore, Seite 88	dient zum Wiederherstellen der Parameter eines Profinet IO-Device mittels azyklischen Schreibens aus der Sicherungsdatei
Unterstützung des Geräteausbaus (Device Replacement)	
IL_PNIOAssignName, Seite 91	weist einem Profinet IO-Device den Device-Namen (Identstring) zu

IL_PNIOCheckTopology, Seite 93	überprüft die gespeicherte Topologie der Profinet-Gerätetauschfunktion anhand der aktuell vorhandenen Profinet-Topologie. Dies erfolgt durch Auslesen der Port-Informationen aus den Profinet IO-Device
IL_PNIODeviceReplacement, Seite 98	liefert eine Diagnose über den Status der automatischen Gerätetauschfunktion. Die Gerätetauschfunktion kann auch deaktiviert werden
IL_PNIODeviceResetToDefault, Seite 100	setzt ein Profinet IO-Device auf den Auslieferungszustand (Factory-Default) zurück
IL_PNIODeviceSetIP, Seite 102	setzt die IP-Einstellungen eines Profinet IO-Device
IL_PNIOGetLiveList, Seite 104	Ermittlung der Liste der aktuell erreichbaren Profinet IO-Teilnehmer
IL_PNIOGetLogMsg, Seite 106	holt einen Logger-Text
IL_PNIOGetPortData, Seite 107	liefert die Port-Daten ("Nachbar-Informationen") von einem Profinet IO-Device
IL_PNIOStoreTopology, Seite 110	generiert die Netzwerktopologie-Informationen und sichert diese Informationen in einer XML-Datei
IL_PNIORestoreTopology, Seite 112	versucht die gespeicherte Topologie des PNIO Controllers wiederherzustellen (z.B. Anwendungsfall "Kreuztausch" von PN Devices).
Diagnose (Diagnostics)	
IL_PNIOControllerState, Seite 116	Ermittlung des Einfachstatus
IL_PNIOControllerStateDetails, Seite 118	Ermittlung des Status des Profinetstacks
IL_PNIODiagnosis, Seite 120	erstellt eine umfassende Diagnose eines PROFINET IO-Controllers und von dessen angeschlossenen Geräten
IL_PNIOGetConfigDeviceNameList, Seite 125	Ermittlung der Liste der konfigurierten Profinet IO-Devices
IL_PNIOGetControllerList, Seite 127	liefert die Anzahl der konfigurierten PROFINET IO-Controller
IL_PNIOGetDiagDeviceNameList, Seite 128	Ermittlung der Liste der diagnosemeldenden Profinet IO-Devices
IL_PNIORemoteDeviceState, Seite 129	Ermittlung des Einfachstatus eines beliebigen Profinet IO-Devices
IL_PNIORemoteDeviceStateDetails, Seite 131	Ermittlung des Status eine beliebigen Profinet IO-Devices
I&M (Identification and Maintenance) (IM Data)	
IL_PNIOReadWriteIMData, Seite 135	kann I&M-Datensätze (Identifikation & Wartung) des Profinet IO-Device mit verschiedenen Lese-/ Schreibmodi lesen bzw. schreiben
Kommunikationsschnittstelle (IO-Link)	
IL_PNIOIoLinkBackupDevice, Seite 180	dient zum Sichern der Konfiguration und Parameter des IOL-Device auf eine Datei über Profinet IO IO-Link-Gateways
IL_PNIOIoLinkCall, Seite 143	ist für die Lese- und Schreibdienste der IOL-Datenobjekte über Profinet IO IO-Link-Gateways zuständig
IL_PNIOIoLinkGetDeviceAccessLock, Seite 149	liest den Status des IO-Link-Deviceparameters Device Access Locks des IO-Link-Device, das an den Profinet-Koppler angeschlossen ist
IL_PNIOIoLinkIdent, Seite 154	wird verwendet, um grundlegende Informationen zur Identifizierung eines IO-Link-Device zu lesen, das an einen bestimmten IO-Link-Port von Profinet angeschlossen ist
IL_PNIOIoLinkIdentDetails, Seite 158	wird verwendet, um Identifizierungsparameter und gerätespezifische Informationen eines IO-Link-Device zu lesen, das an einen bestimmten IO-Link-Port von Profinet angeschlossen ist

IL_PNIOIoLinkRemoteDeviceState, Seite 164	liest den aktuellen Diagnosestatus des IO-Link-Device, das an einen bestimmten IO-Link-Port auf Profinet angeschlossen ist
IL_PNIOIoLinkRestoreDevice, Seite 185	dient zum Wiederherstellen der Parameter eines IOL-Device aus der Sicherungsdatei über das Profinet IO IO-Link-Gateway
IL_PNIOIoLinkSetDeviceAccessLock, Seite 169	ermöglicht die Steuerung des Profinet IO-Device-Verhaltens durch Überschreiben des Parameters "Device Access Locks" des IO-Link-Device, das an den Profinet-Koppler angeschlossen ist
IL_PNIOIoLinkSystemCommand, Seite 175	wird verwendet, um einen Systembefehl auf ein IO-Link-Device zu schreiben, das an einen bestimmten IO-Link-Port von Profinet angeschlossen ist

Tab. 5-1: RIL_ProfinetIO-Bibliothek

Auswahl des Profinet IO-Controllers

Adressierung: Die Profinet IO-Controller werden anhand ihrer aufsteigenden Reihenfolge in der Konfiguration unterschieden:

Die Zählung beginnt beim ersten Profinet IO-Controller mit "0".

Wenn nur ein Profinet IO-Controller konfiguriert ist, ist seine Controller-Instanz immer "0" (siehe auch: [IL_BUSMASTER, Seite 189](#)).

Adressierung der Profinet IO-Devices

Device-Name: Bei Profinet werden die Geräte über den Device-Namen "IdentStr" adressiert.

Der Device-Name muss in der Konfiguration eingetragen sein, siehe "Station Name".

PROFINETIO_TABLE 16#200

Die [PROFINETIO_TABLE 16#200, Seite 195](#), enthält die Fehlernummern, die von den Funktionsbausteinen der Bibliothek ausgegeben werden.

5.2 Azyklische Dienste (Acyclic Communication)

5.2.1 IL_PNIORReadRecord

Kurzbeschreibung

Der Funktionsbaustein **IL_PNIORReadRecord** liest azyklische Daten von Profinet IO-Devices.

Die Profinet IO-Devices werden über ihren Device-Namen in "IdentStr" adressiert.

Zielsystem	Bibliothek/Ordner
MLC	RIL_ProfinetIO/POUs/AcyclicCommunication
MTX	RIL_ProfinetIO/POUs/AcyclicCommunication

Tab. 5-2: Referenztable

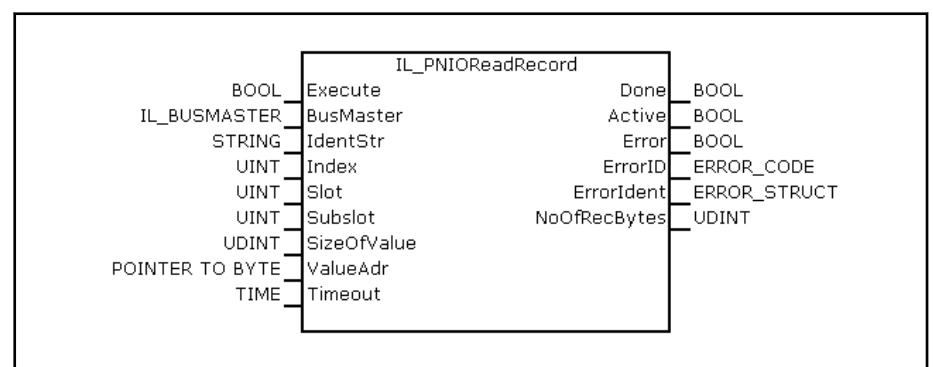


Abb. 5-1: Schnittstellenbeschreibung FB IL_PNIORReadRecord

I/O-Typ	Name	Typ	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Feldbus-Masters in der Reihenfolge des Konfigurationbaumes siehe auch Kap. 5.8.1 "IL_BUSMASTER" auf Seite 189
	IdentStr	STRING	Identifikationsstring des Feldbusteilnehmers
	Index	UINT	Index des adressierten Datensatzes. Die Indizes 16#8000 bis 16#FFFF sind für Profinet reserviert. Indizes von 0 bis 16#7FFF sind user- bzw. applikationsspezifisch
	Slot	UINT	Adressierung des Modulslots. Bei einem Profinet IO-Device adressiert ein Slot ein logisches oder physikalisches Modul (beispielsweise ein I/O-Modul)
	Subslot	UINT	Adressierung des Submoduls. Ein Modul kann sich aus mehreren logischen oder physikalischen Untermodulen zusammensetzen. Es ist bei jedem Modul (Slot) mindestens das Untermodul mit der Nummer 1 vorhanden
	SizeOfValue	UDINT	Größe des mit "ValueAdr" adressierten Datenbereichs
	ValueAdr	POINTER TO BYTE	Anfangsadresse des Datenbereichs
	Timeout	TIME	Timeout - Überwachung des FBs in ms. T#0ms = Timeout - Überwachung ist nicht aktiv
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	NoOf RecBytes	UDINT	Anzahl der empfangenen Bytes im Datenbereich

Tab. 5-3: I/O-Schnittstelle des FB IL_PNIReadRecord

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an "Execute"
IdentStr	STRING			''	Steigende Flanke an "Execute"
Index	UINT	0	16#ffff	0	Steigende Flanke an "Execute"
Slot	UINT	1	16#7fff	0	Steigende Flanke an "Execute"

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Subslot	UINT	1	16#8fff	0	Steigende Flanke an "Execute"
SizeOfValue	UDINT	0	n.def.	0	Steigende Flanke an "Execute"
ValueAdr	POINTER TO BYTE	n.def	n.def	0	Steigende Flanke an "Execute"
Timeout	TIME	T#0ms	T#3600s	T#2s	Steigende Flanke an "Execute"

Tab. 5-4: Min-, Max- und Default-Werte der IL_PNIReadRecord-Eingänge

Funktionsbeschreibung

Das Profinet IO-Device wird über den Device-Namen in "IdentStr" angesprochen. Es muss Teil der Konfiguration des Profinet IO-Controllers sein. Der Datensatz auf dem Profinet IO-Device wird über die Kombination von Slot und Subslot sowie den Index adressiert.

"Slot" und "Subslot" adressieren die verschiedenen logischen oder physikalischen Module eines Profinet IO-Device. Über den Index wird die Art der Information ausgewählt. Der Bereich zwischen 16#8000 und 16#ffff ist durch die IEC 61158-6-10 V23 vergeben. Der Wertebereich 0 bis 16#7fff kann durch den Hersteller des Profinet IO-Devices vergeben sein. Näheres ist der entsprechenden Dokumentation des Profinet IO-Devices zu entnehmen.

Beispielsweise können die I&M0-Daten eines Profinet IO-Devices mit dem Index 16#AFF0 sowie "Slot 0" und "Subslot 1" angesprochen werden.

Die Daten werden in der maximal zulässigen Länge ("SizeOfValue") vom Profinet IO-Device angefordert. Die Anzahl der tatsächlich vom Profinet IO-Device gelieferten und gültigen Datenbytes werden in "NoOfRecBytes" zurückgegeben.

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle **PROFINETIO_TABLE** (ERROR_TABLE = 16#0200).

Der Funktionsbaustein kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Bemerkung
INPUT_RANGE_ERROR	16#0001	0	"ValueAdr" ist nicht gesetzt
INPUT_RANGE_ERROR	16#0001	1	"SizeOfValue" ist nicht gesetzt
INPUT_RANGE_ERROR	16#0001	2	"IdentStr" ist ungültig
SYSTEM_ERROR	16#0101	internal use	Profinet IO-Controller nicht in Konfigurationstabelle
COMMUNICATION_ERROR	16#0104	0	Zeitüberschreitung
STATE_MACHINE_ERROR	16#0105	16#0001 16#0002	Fehler in der State Machine

ErrorID	Additional1	Additional2	Bemerkung
COMMUNICATION_ERROR	16#0003	Profinet IO-Controller-Diagnose - Siehe Kap. 5.9.2 "Allgemeine Fehlercodes " auf Seite 196 - Siehe Kap. 5.9.3 "Spezielle Fehlercodes, Profinet IO-Controller " auf Seite 198	
COMMUNICATION_ERROR	16#0004	Profinet IO-Status Siehe Kap. 5.9.4 "Spezielle Fehlercodes, Profinet IO-Status " auf Seite 199	

Tab. 5-5: Fehlercodes FB IL_PNIReadRecord

5.2.2 IL_PNIOServiceRecord

Kurzbeschreibung Der Funktionsbaustein **IL_PNIOServiceRecord** dient der Abwicklung des Profinet-Dienst-Schnittstellenzugriffs, der z. B. für die IO-Link-Integration oder für PROFlenergy benötigt wird.

Zielsystem	Bibliothek/Ordner
MLC	RIL_ProfinetIO/POUs/IO-Link
MTX	RIL_ProfinetIO/POUs/IO-Link

Tab. 5-6: Referenztabelle

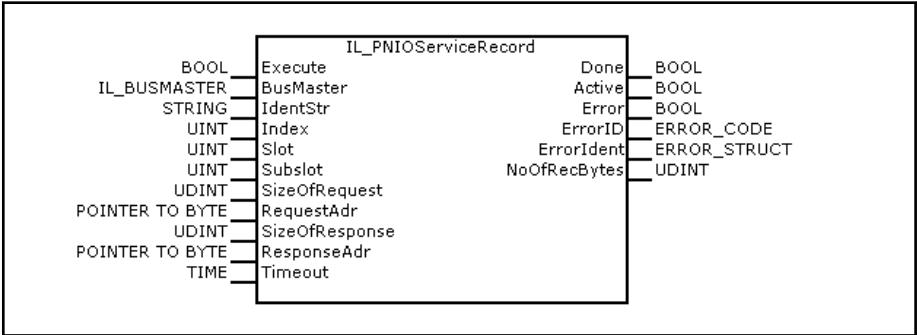


Abb. 5-2: Schnittstellenbeschreibung FB IL_PNIOServiceRecord

I/O-TYP	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Profinet IO-Controllers in der Reihenfolge des Konfigurationbaumes. Ist nur ein Profinet IO-Controller vorhanden, besitzt dieser Controller immer den Wert IL_BUSMASTER_0 Siehe auch Kap. 5.8.1 "IL_BUSMASTER" auf Seite 189
	IdentStr	STRING	Identifikationsstring des Profinet IO-Device
	Index	UINT	Index des adressierten Datensatzes. Die Indizes 16#8000 bis 16#FFFF sind für Profinet reserviert. Die Indizes von 0 bis 16#7FFF sind user- bzw. applikationsspezifisch
	Slot	UINT	Adressierung des Modulslots. Der Slot adressiert ein logisches oder physikalisches Modul (beispielsweise ein I/O-Modul)
	SubSlot	UINT	Adressierung des Submoduls. Ein Modul kann sich aus mehreren logischen oder physikalischen Submodulen zusammensetzen. Es ist bei jedem Modul (Slot) mindestens das Submodul mit der Nummer 1 vorhanden
	SizeOfRequest	UDINT	Länge der Anforderungsdaten. (<= SizeOf(RequestAdr))
	RequestAdr	POINTER TO BYTE	Anfangsadresse des angeforderten Datenbereichs. Adr() Operator verwenden
	SizeOfResponse	UDINT	Größe des Antwortadr.-Arrays. Adr SizeOf() Operator verwenden
	ResponseAdr	POINTER TO BYTE	Anfangsadresse des Antwortdatenbereichs. Adr() Operator verwenden
	Timeout	TIME	Timeout-Überwachung des Funktionsbausteins
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten ungültig
	Fehler	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall (aus RIL_CommonTypes)
	ErrorIdent	ERROR_STRUCT	Detailldiagnose (ErrorTable PROFINETIO_TABLE)
	NoOfRecBytes	UDINT	Anzahl der als Antwort empfangenen Bytes

Tab. 5-7: I/O-Schnittstelle des FB IL_PNIOServiceRecord

Min-, Max- und Default-Werte: In der folgenden Tabelle sind die Min-, Max- und Default-Werte der Funktionsbausteineingänge aufgelistet.

Name	Typ	Min. Wert	Max. Wert	Default-Wert	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
BusMaster 12 16	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an „Execute“
IdentStr	STRING	---		---	Steigende Flanke an „Execute“
Index	UINT	0	16#ffff	0	Steigende Flanke an „Execute“
Slot	UINT	0	16#7fff	0	Steigende Flanke an „Execute“
SubSlot	UINT	1	16#8fff	1	Steigende Flanke an „Execute“
SizeOfRequest	UDINT	0	----	0	Steigende Flanke an „Execute“
RequestAdr	POINTER TO BYTE	0	----	0	Steigende Flanke an „Execute“
SizeOfResponse	UDINT	0	----	0	Steigende Flanke an „Execute“
ResponseAdr	POINTER TO BYTE	0	----	0	Steigende Flanke an „Execute“
Timeout	TIME	T#0ms	----	T#2s	Steigende Flanke an „Execute“

Tab. 5-8: Min-, Max- und Default-Werte der IL_PNIOServiceRecord-Eingänge

Funktionsbeschreibung

Der Funktionsbaustein **IL_PNIOServiceRecord** dient der Abwicklung des Profinet-Dienst-Schnittstellenzugriffs, der z. B. für die IO-Link-Integration oder für PROFlenergy benötigt wird. Der Zugriff auf die Serviceschnittstelle kann über die Funktionsbausteine IL_PNIOWriteRecord (Schreibdienstanforderung) und IL_PNIOReadRecord (Lesedienstanforderung) erfolgen.

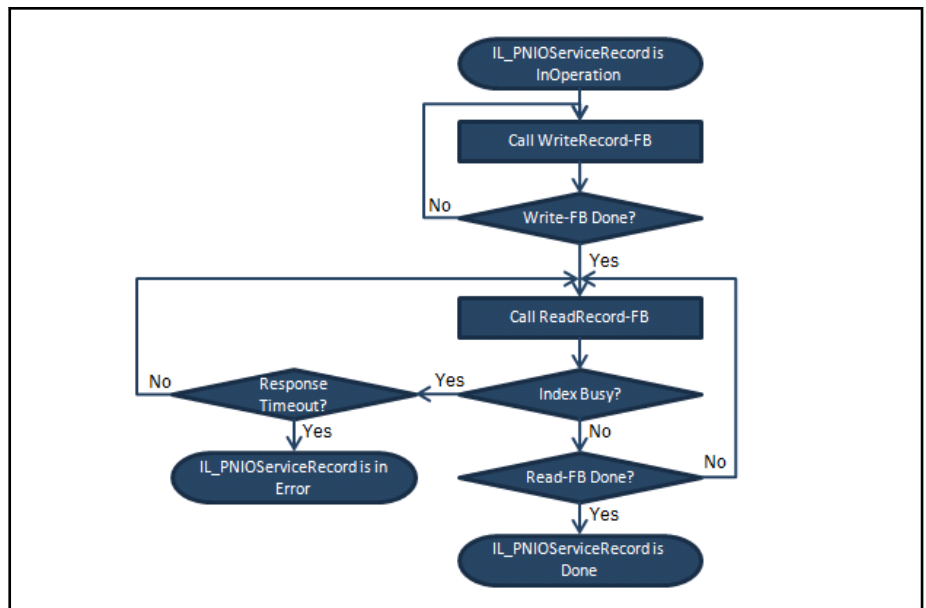


Abb. 5-3: Ablaufdiagramm des FBs **IL_PNIOServiceRecord** der Bibliothek *RIL_ProfinetIO.library*

Das Profinet IO-Device wird über den Profinet Device-Namen in "IdentStr" adressiert. Es muss Teil der Konfiguration des Profinet IO-Controllers sein. Der Datensatz auf dem Profinet IO-Device wird über die Kombination von "Slot" und "SubSlot" sowie den "Index" adressiert.

"Slot" und "SubSlot" adressieren die verschiedenen logischen oder physikalischen Module eines Profinet IO-Devices. Über den Index wird die Art der Information ausgewählt. Der Bereich zwischen 16#8000 und 16#ffff ist durch IEC 61158-6-10 V23 vergeben. Der Wertebereich 0 bis 16#7fff kann durch den Hersteller des Profinet IO-Devices vergeben sein. Näheres ist der entsprechenden Dokumentation des Profinet IO-Devices zu entnehmen. Der angegebene Datenbereich "RequestAdr" wird in seiner gesamten Größe "SizeOfValue" zum Profinet IO-Device transferiert. Die Reaktion des Profinet IO-Devices auf Bereichsüberschreitungen ist herstellerspezifisch.

Die Daten werden in der maximal zulässigen Länge ("SizeOfResponse") vom Profinet IO-Device angefordert. Die Anzahl der tatsächlich vom Profinet IO-Device gelieferten und gültigen Datenbytes wird in "NoOfRecBytes" zurückgegeben.

Beispielprogramm

Der Funktionsbaustein **IL_PNIOServiceRecord** dient der Abwicklung des Profinet-Serviceschnittstellenzugriffs.

Beispiel: **IL_PNIOServiceRecord** liest den Herstellernamen des Profinet IO-Device, das an Port 2 des IOL-Masters angeschlossen ist.

Deklaration

```

VAR
  fbIL_PNIOServiceRecord: IL_PNIOServiceRecord;
  bExecute: BOOL;
  bDone: BOOL;
  bActive: BOOL;
  bError: BOOL;
  enErrorID: ERROR_CODE;
  stErrorIdent: ERROR_STRUCT;
  arRequestAdr: ARRAY[0..7] OF BYTE;
  udiNoOfRecBytes: UDINT;
  arResponseAdr: array[0..232] of byte;
END_VAR
  
```

Implementierung

```
fbIL_PNIOServiceRecord(  
  Execute:= bExecute,  
  Done=> bDone,  
  Active=> bActive,  
  Error=> bError,  
  ErrorID=> enErrorID,  
  ErrorIdent=> stErrorIdent,  
  BusMaster:= IL_BusMaster.IL_BUSMASTER_0,  
  IdentStr:='iolBalluff',  
  Index:= UINT#255,  
  Slot:= UINT#1,  
  Subslot:= UINT#1,  
  SizeOfRequest:= UDINT#8,  
  RequestAdr:= ADR(arRequestAdr),  
  SizeOfResponse:= SIZEOF(arResponseAdr),  
  ResponseAdr:= ADR(arResponseAdr),  
  Timeout:= T#2S,  
  NoOfRecBytes=> udiNoOfRecBytes);
```

Das Ergebnis des IL_PNIOServiceRecord wird in nachfolgender Abbildung zeigt:

Expression	Type	Value
* fbIL_PNIOServiceRecord	IL_PNIOServiceRecord	
bExecute	BOOL	TRUE
bDone	BOOL	TRUE
bActive	BOOL	FALSE
bError	BOOL	FALSE
enErrorID	ERROR_CODE	NONE_ERROR
* stErrorIdent	ERROR_STRUCT	
= arRequestAdr	ARRAY [0..7] OF BYTE	
arRequestAdr[0]	BYTE	16#08 → For "CALL", fixed
arRequestAdr[1]	BYTE	16#02 → Port number
arRequestAdr[2]	BYTE	16#FE → FI_Index (MSB)
arRequestAdr[3]	BYTE	16#4A → FI_Index (LSB)
arRequestAdr[4]	BYTE	16#03 → Task (Read)
arRequestAdr[5]	BYTE	16#00 → IOL Index (MSB)
arRequestAdr[6]	BYTE	16#10 → IOL Index (LSB)
arRequestAdr[7]	BYTE	16#00 → IOL Subindex
udiNoOfRecBytes	UDINT	16#0000000F
= arResponseAdr	ARRAY [0..232] OF BYTE	
arResponseAdr[0]	BYTE	16#08
arResponseAdr[1]	BYTE	16#02
arResponseAdr[2]	BYTE	16#FE
arResponseAdr[3]	BYTE	16#4A
arResponseAdr[4]	BYTE	16#00
arResponseAdr[5]	BYTE	16#00
arResponseAdr[6]	BYTE	16#10
arResponseAdr[7]	BYTE	16#00
arResponseAdr[8]	BYTE	16#42
arResponseAdr[9]	BYTE	16#41
arResponseAdr[10]	BYTE	16#4C
arResponseAdr[11]	BYTE	16#4C
arResponseAdr[12]	BYTE	16#55
arResponseAdr[13]	BYTE	16#46
arResponseAdr[14]	BYTE	16#46

Abb. 5-4: IL_PNIOServiceRecord, Beispiel, online

Fehlerbehandlung

Der Funktionsbaustein IL_PNIOServiceRecord nutzt die Fehlertabelle PRO-FINETIO_TABLE (ERROR_TABLE = 16#0200).
Der Funktionsbaustein kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Bemerkung
INPUT_RANGE_ERROR	16#0001	16#0000	"ValueAdr" ("RequestAdr" oder "ResponseAdr") ist nicht gesetzt
INPUT_RANGE_ERROR	16#0001	16#0001	"SizeOfValue" ("SizeOfRequest" oder "SizeOfResponse") ist nicht gesetzt
INPUT_RANGE_ERROR	16#0001	16#0002	"IdentStr" ist ungültig
SYSTEM_ERROR	16#0101	internal use	Profinet IO-Controller nicht in Konfigurationstabelle
COMMUNICATION_ERROR	16#0104	16#0000	Zeitüberschreitung
STATE_MACHINE_ERROR	16#0105	internal use	Fehler in der State Machine
COMMUNICATION_ERROR	16#0003	Profinet IO-Controller-Diagnose - General Error-Codes, Bus-Master Kap. 5.9.2 "Allgemeine Fehlercodes" auf Seite 196 - Special Error-Codes, Profinet IO-Controller Kap. 5.9.3 "Spezielle Fehlercodes, Profinet IO-Controller" auf Seite 198	
COMMUNICATION_ERROR	16#0004	PROFINET IO-Status Error-Codes, Profinet IO-Status Kap. 5.9.4 "Spezielle Fehlercodes, Profinet IO-Status" auf Seite 199	

Tab. 5-9: Fehlercodes IL_PNIOWriteRecord

5.2.3 IL_PNIOWriteRecord

Kurzbeschreibung Der Funktionsbaustein **IL_PNIOWriteRecord** schreibt azyklische Daten auf Profinet IO-Devices.

Die Profinet IO-Devices werden über ihren Device-Namen adressiert. Die Profinet IO-Devices müssen in der Konfiguration vorhanden sein.

Zielsystem	Bibliothek/Ordner
MLC	RIL_ProfinetIO/POUs/AcyclicCommunication
MTX	RIL_ProfinetIO/POUs/AcyclicCommunication

Tab. 5-10: Referenztable

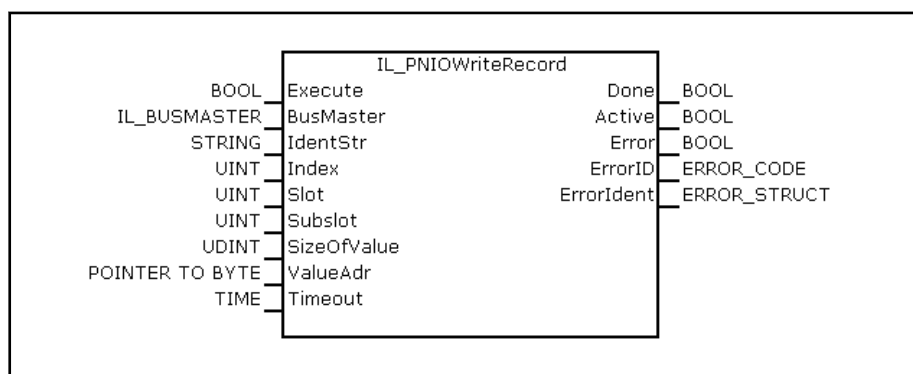


Abb. 5-5: Schnittstellenbeschreibung FB IL_PNIOWriteRecord

I/O-Typ	Name	Typ	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Profinet IO-Controllers in der Reihenfolge des Konfigurationbaumes. Ist nur ein Profinet IO-Controller vorhanden, besitzt dieser Controller immer den Wert IL_BUSMASTER_0 siehe auch Kap. 5.8.1 "IL_BUSMASTER" auf Seite 189
	IdentStr	STRING	Stationsname des Profinet IO-Devices, auf das zugegriffen werden soll
	Index	UINT	Index des adressierten Datensatzes. Die Indizes 16#8000 bis 16#FFFF sind für Profinet reserviert. Indizes von 0 bis 16#7FFF sind user- bzw. applikationsspezifisch
	Slot	UINT	Adressierung des Modulslots. In einem modularen Profinet IO-Device adressiert man über "Slot" die physikalischen Module. In einem kompakten Profinet IO-Device adressiert "Slot" eine logische Funktion oder ein virtuelles Modul
	Subslot	UINT	Adressierung des Submoduls. Über "Subslot" kann man die physikalische Schnittstellen von Sub-Modulen eines Moduls adressieren. Allgemein ist "Subslot" die zweite Struktur-Ebene eines Profinet IO-Devices
	SizeOfValue	UDINT	Maximale Größe des Datenbereichs von "ValueAdr"
	ValueAdr	POINTER TO BYTE	Zeiger auf die Datenablage
	Timeout	TIME	Timeoutwert für den Abbruch der Funktion
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall (aus RIL_CommonTypes)
	ErrorIdent	ERROR_STRUCT	Detailldiagnose (ErrorTable PROFINETIO_TABLE)

Tab. 5-11: I/O-Schnittstelle des FB IL_PNIOWriteRecord

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an "Execute"
IdentStr	STRING			""	Steigende Flanke an "Execute"

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Index	UINT	0	16#ffff	0	Steigende Flanke an "Execute"
Slot	UINT	1	16#7fff	0	Steigende Flanke an "Execute"
Subslot	UINT	1	16#8fff	0	Steigende Flanke an "Execute"
NoOfBytes	UDINT	0	n.def.	0	Steigende Flanke an "Execute"
ValueAdr	POINTER TO BYTE	n.def.	n.def.	0	Steigende Flanke an "Execute"
Timeout	TIME	T#0ms	T#3600s	T#2s	Steigende Flanke an "Execute"

Tab. 5-12: Min-, Max- und Default-Werte der IL_PNIOWriteRecord-Eingänge

Funktionsbeschreibung

Das Profinet IO-Device wird über den Device-Namen angesprochen. Es muss Teil der Konfiguration des Profinet IO-Controllers sein. Der Datensatz auf dem Profinet IO-Device wird über die Kombination von "Slot" und "Subslot" sowie den "Index" adressiert.

"Slot" und "Subslot" adressieren die verschiedenen logischen oder physikalischen Module eines Profinet IO-Devices. Über den Index wird die Art der Information ausgewählt. Der Bereich zwischen 16#8000 und 16#ffff ist durch die IEC 61158-6-10 V23 vergeben. Der Wertebereich 0 bis 16#7fff kann durch den Hersteller des Profinet IO-Devices vergeben sein. Näheres ist der entsprechenden Dokumentation des Profinet IO-Devices zu entnehmen.

Der angegebene Datenbereich "ValueAdr" wird in seiner gesamten Größe "SizeOfValue" zum Profinet IO-Device transferiert. Die Reaktion des Profinet IO-Devices auf Bereichsüberschreitungen ist herstellerspezifisch.

Fehlerbehandlung

Der Funktionsbaustein nutzt die **Fehlertabelle PROFINETIO_TABLE** (ERROR_TABLE = 16#0200).

Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Bemerkung
INPUT_RANGE_ERROR	16#0001	0	"ValueAdr" ist nicht gesetzt
INPUT_RANGE_ERROR	16#0001	1	"SizeOfValue" ist nicht gesetzt
INPUT_RANGE_ERROR	16#0001	2	"IdentStr" ist nicht gesetzt
SYSTEM_ERROR	16#0101	internal use	Profinet IO-Controller nicht in Konfigurationstabelle
COMMUNICATION_ERROR	16#0104	0	Zeitüberschreitung
STATE_MACHINE_ERROR	16#0105	0	Fehler in der State Machine

ErrorID	Additional1	Additional2	Bemerkung
COMMUNICATION_ERROR	16#0003	Profinet IO-Controller-Diagnose (siehe Additional2 Code) - Siehe Kap. 5.9.2 "Allgemeine Fehlercodes " auf Seite 196 - Siehe Kap. 5.9.3 "Spezielle Fehlercodes, Profinet IO-Controller" auf Seite 198	
COMMUNICATION_ERROR	16#0004	Profinet IO-Status Siehe Kap. 5.9.4 "Spezielle Fehlercodes, Profinet IO-Status " auf Seite 199	

Tab. 5-13: Fehlercodes FB IL_PNIOWriteRecord

5.3 Sichern und Wiederherstellen (Backup_Restore)

5.3.1 IL_PNIODeviceBackup

Kurzbeschreibung Der Funktionsbaustein **IL_PNIODeviceBackup** dient zum Sichern der Konfiguration und der Parameter eines Profinet IO-Device auf eine Datei mittels azyklischen Lesens ([Kap. 5.2.1 "IL_PNIOWriteRecord" auf Seite 73](#)).

Zielsystem	Bibliothek/Ordner
MLC	RIL_ProfinetIO/POUs/Backup Restore

Tab. 5-14: Referenztablelle

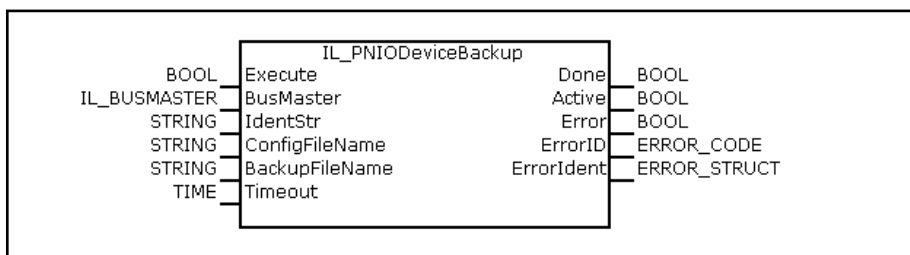


Abb. 5-6: Schnittstellenbeschreibung FB IL_PNIODeviceBackup

I/O-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Profinet IO-Controllers in der Reihenfolge des Konfigurationsbaums. Ist nur ein Profinet I/O-Controller vorhanden, besitzt dieser Controller immer den Wert IL_BUSMASTER_0 Siehe auch Kap. 5.8.1 "IL_BUSMASTER" auf Seite 189
	IdentStr	STRING	Identifikationsstring des Profinet IO-Device
	ConfigFileName	STRING	ConfigFile mit "Slot", "SubSlot" und "Index" der Parameter, die aus dem Profinet IO-Device zu lesen sind. Name der Konfigurationsdatei ohne Extension (*.pnc)
	BackupFileName	STRING	Name der Sicherungsdatei ohne Extension (*.pnb)
	Timeout	TIME	Timeout-Überwachung des Funktionsbausteins

I/O-Typ	Name	Datentyp	Beschreibung
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibt die Diagnose im Fehlerfall (aus RIL_Common-Types)
	ErrorIdent	ERROR_STRUCT	Detaildiagnose (Fehlertabelle PROFINETIO_TABLE) "Fehlerbehandlung" auf Seite 88

Tab. 5-15: I/O-Schnittstelle des FB IL_PNIODeviceBackup

Min-, Max- und Default-Werte: In der folgenden Tabelle sind die Min-, Max- und Default-Werte der Funktionsbausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an "Execute"
Timeout	TIME	T#0ms		T#2s	Steigende Flanke an "Execute"

Tab. 5-16: Min-, Max- und Default-Werte der IL_PNIODeviceBackup-Eingänge

Funktionsbeschreibung

Der Funktionsbaustein **IL_PNIODeviceBackup** dient zum Sichern der Konfiguration und der Parameter eines Profinet IO Device auf eine Datei nach azyklischem Lesen (siehe [Kap. 5.2.1 "IL_PNIORReadRecord" auf Seite 73](#)) aus dem Profinet IO-Device.

Der Funktionsbaustein liest die in der Konfigurationsdatei "ConfigFileName" aufgeführten Parameter aus dem Profinet IO-Device aus und speichert die Daten in der Datei "BackupFileName".

Die Konfigurations- und Sicherungsdateien werden auf der Compact-Flash-Karte der Steuerung (USER-Partition, /ata0b/) abgelegt.

Der Funktionsbaustein erstellt bei jeder Ausführung eine neue Datei. Ist eine Datei bereits vorhanden, wird die Datei überschrieben.

Die Konfigurationsdatei ist im Textformat gespeichert und hat die Extension "*.pnc".

Basisstruktur der ConfigFile-Datei	Beispiel der ConfigFile-Datei
Slot, SubSlot, Index	16#0000, 16#0001, 16#0040
Slot, SubSlot, Index	16#0000, 16#0001, 16#AFF0
Slot, SubSlot, Index	16#0001, 16#0001, 16#8000
...	...
...	...
...	...
Slot, SubSlot, Index	16#0002, 16#0001, 16#C000

Bei steigender Flanke am Eingang "Execute", liest der Funktionsbaustein die Parameter aus dem Profinet IO-Device aus und speichert sie in die Sicherungsdatei.

Basissicherungsdatei	Beispiel der Sicherungsdatei
Slot, SubSlot, Index, DataSize, Data	16#0000, 16#0001, 16#0040, 16#0001, 16#58
Slot, SubSlot, Index, DataSize, Data1, Data2, Data3	16#0000, 16#0001, 16#AFF0, 16#0003, 16#20, 16#82, 16#32
...	...
...	...
Slot, SubSlot, Index, DataSize, Data1, Data2	16#0000, 16#0001, 16#AFF0, 16#0002, 16#45, 16#56

Beispielprogramm Dieses Beispiel beschreibt das Erstellen des Backups der Profinet IO-Device Parameter zur Ablage auf der Compact-Flash-Karte in der IndraControl-Steuerung.

- Festzulegen ist der Konfigurationsdateiname in "strConfigFileName"
- Die Konfigurationsdatei ist auf der Compact-Flash-Karte (USER-Partition, /ata0b/) der Steuerung zu speichern

Schritte zur Erzeugung und Ablage der Konfigurationsdatei

1. Notepad öffnen und Backup-Parameterinformationen eingeben.

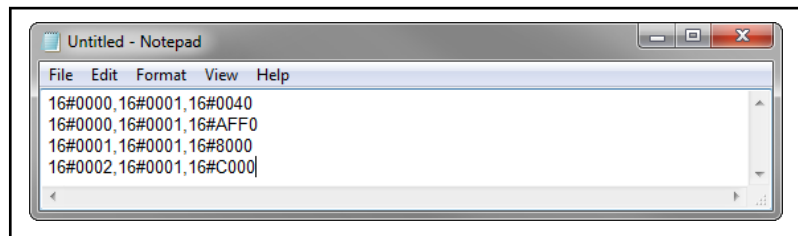


Abb. 5-7: Konfigurationsdatei

2. Konfigurationsdatei als "ConfigFile.enc" speichern.
3. Rechte-Maus-Klick auf die Steuerung im Projektextplorer und navigieren zu **Gerätedaten ► Explorer**.

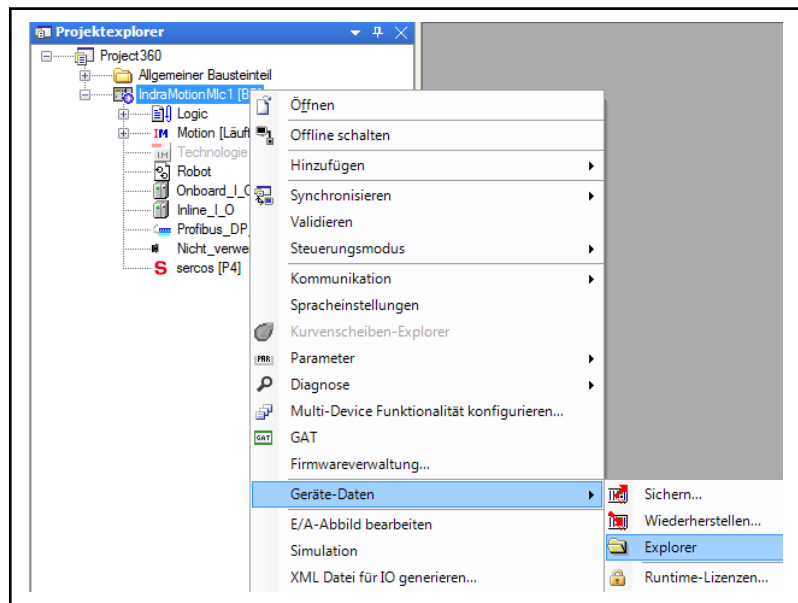


Abb. 5-8: Kontextmenü der Steuerung

4. Navigieren zur User-Partition auf der Compact-Flash-Karte der Steuerung.

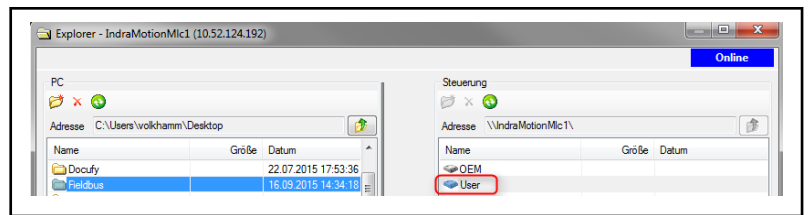


Abb. 5-9: User-Partition

5. Übertragung der Konfigurationsdatei zur User-Partition.

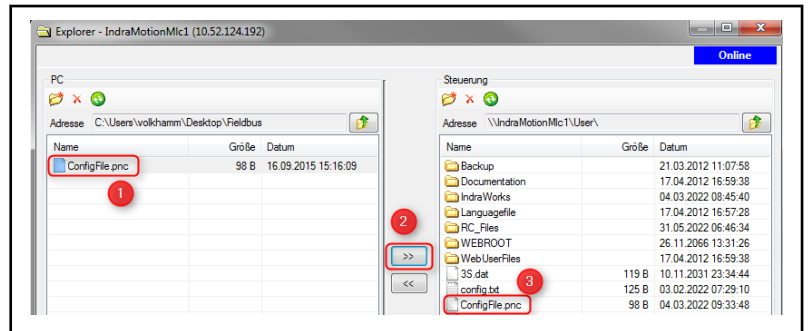


Abb. 5-10: Übertragung der Konfigurationsdatei

- Festlegen des Sicherungsdateinames "strBackupFileName"
Die Eingabe der Dateinamen hat ohne Extension zu erfolgen, z. B. "Backupfile"
- Mit steigender Flanke an "bExecute" beginnt die Bearbeitung:
 - Lesen der durch die Konfigurationsdatei vorgegebenen Parameter vom durch "IdentStr" adressierten Profinet IO-Device
 - Ablegen der Daten auf der Sicherungsdatei

Deklaration

```

PROGRAM ProgIL_PNIODiviceBackup
VAR
    fbIL_PNIODiviceBackup: IL_PNIODiviceBackup;
    bExecute: BOOL; // FB running when TRUE
    strIdentString: STRING := 'Lx5-PN'; // Station name of PN
    strConfigFileName: STRING := 'ConfigFile'; // Config file name
    strBackupFileName: STRING := 'BackupFile'; // Backup file name
    tTimeout: TIME := T#3S; // Timeout monitoring of FB
    bDone: BOOL; // FB completed its task
    bActive: BOOL; // FB working on actual task
    bError: BOOL; // Indicates an error
    enErrorID: RIL_COMMONTYPES.ERROR_CODE; // Class of error
    enErrorIdent: RIL_COMMONTYPES.ERROR_STRUCT; // Detailed info on error
END_VAR

```

Implementierung

```

fbIL_PNIODiviceBackup(
    Execute:= bExecute,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> enErrorIdent,
    BusMaster:= IL_BUSMASTER_IL_BUSMASTER_0,
    IdentStr:= strIdentString,
    ConfigFileName:= strConfigFileName,
    BackUpFileName:= strBackupFileName,
    Timeout:= tTimeout);

```

Fehlerbehandlung Der Funktionsbaustein **IL_PNIODeviceBackup** nutzt die Fehlertabelle PROFINETIO_TABLE (ERROR_TABLE = 16#0200).

Der Funktionsbaustein kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Bemerkungen
INPUT_RANGE_ERROR	16#0001	16#0002	"IdentStr" ist ungültig
COMMUNICATION_ERROR	16#0003	Profinet IO-Controller Diagnose • Kap. 5.9.2 "Allgemeine Fehlercodes " auf Seite 196 • Kap. 5.9.3 "Spezielle Fehlercodes, Profinet IO-Controller" auf Seite 198	
COMMUNICATION_ERROR	16#0004	Profinet IO-Status • Kap. 5.9.4 "Spezielle Fehlercodes, Profinet IO-Status " auf Seite 199	
SYSTEM_ERROR	16#0101	internal use	Profinet IO-Controller nicht in der Konfigurationstabelle gefunden
COMMUNICATION_ERROR	16#0104	16#0000	Zeitüberschreitung
INPUT_INVALID_ERROR	16#3002	16#0001	"ConfigFileName" ist ungültig
INPUT_INVALID_ERROR	16#3002	16#0002	"BackupFileName" ist ungültig
ACCESS_ERROR	16#3003	16#0000	"Configuration file" konnte nicht gelesen werden
ACCESS_ERROR	16#3003	16#0001	"Configuration file" konnte nicht geöffnet werden
ACCESS_ERROR	16#3003	16#0002	"Configuration file" Format des ist ungültig
ACCESS_ERROR	16#3003	16#0003	"Backup file" konnte nicht geöffnet werden
ACCESS_ERROR	16#3003	16#0004	"Backup file" schreiben oder lesen nicht möglich

Tab. 5-17: Fehlercodes IL_PNIODeviceBackup

5.3.2 IL_PNIODeviceRestore

Kurzbeschreibung Der Funktionsbaustein **IL_PNIODeviceRestore** dient zum Wiederherstellen der Parameter eines Profinet IO-Device mittels azyklischen Schreibens aus der Sicherungsdatei ([Kap. 5.2.3 "IL_PNIOWriteRecord" auf Seite 81](#)).

Zielsystem	Bibliothek/Ordner
MLC	RIL_ProfinetIO/ POU's/Backup Restore

Tab. 5-18: Referenztablelle

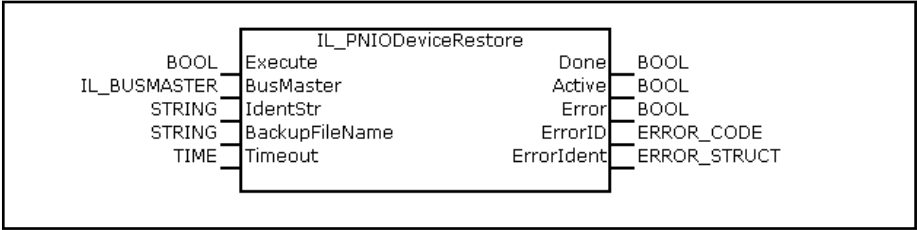


Abb. 5-11: Schnittstellenbeschreibung FB IL_PNIODeviceRestore

I/O-Typ	Name	Typ	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Profinet IO-Controllers in der Reihenfolge des Konfigurationsbaums. Ist nur ein Profinet I/O-Controller vorhanden, besitzt dieser Controller immer den Wert IL_BUSMASTER_0. Siehe auch Kap. 5.8.1 "IL_BUSMASTER" auf Seite 189
	IdentStr	STRING	Identifikationsstring des Profinet IO-Device
	BackupFileName	STRING	Name der Sicherungsdatei ohne Extension (*.pnb)
	Timeout	TIME	Timeout-Überwachung des Funktionsbausteins
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibt die Diagnose im Fehlerfall (aus RIL_Common-Types)
	ErrorIdent	ERROR_STRUCT	Detailldiagnose (Fehlertabelle PROFINETIO_TABLE) "Fehlerbehandlung" auf Seite 90

Tab. 5-19: I/O-Schnittstelle des FB IL_PNIODeviceRestore

Min-, Max- und Default-Werte: In der folgenden Tabelle sind die Min-, Max- und Default-Werte der Funktionsbausteineingänge aufgelistet.

Name	Typ	Min. Wert	Max. Wert	Defaultwert	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an „Execute“
Timeout	TIME	T#0ms		T#2s	Steigende Flanke an „Execute“

Tab. 5-20: Min-, Max- und Default-Werte der IL_PNIODeviceRestore-Eingänge

Funktionsbeschreibung

Der Funktionsbaustein **IL_PNIODeviceRestore** dient zum Wiederherstellen der Parameter eines Profinet IO Slave-Device mittels azyklischen Schreibens aus der Sicherungsdatei ([Kap. 5.2.3 "IL_PNIOWriteRecord" auf Seite 81](#)).

Der Funktionsbaustein liest die Parameter aus der Sicherungsdatei des Profinet IO Slave-Device, die auf der Compact-Flash-Karte der Steuerung abliegen.

Das Profinet IO-Device wird über "IdentStr" adressiert.

"Slot", "SubSlot" und "Index" werden genutzt, um einen einzelnen "Wert" des Profinet IO-Devices zu adressieren.

Bei steigender Flanke an "Execute" liest der Funktionsbaustein alle Parameter aus der Sicherungsdatei aus und schreibt die Parameterwerte auf das Profinet IO-Device auf die durch "Slot", "SubSlot" und "Index" spezifizierten Stellen.

Basisstruktur der Sicherungsdatei	Beispiel der Sicherungsdatei
Slot, Sub, Slot, Index, DataSize, Data	16#0000, 16#0001, 16#0040, 16#0001, 16#58
Slot, SubSlot, Index, DataSize, Data1, Data2, Data3	16#0000, 16#0001, 16#AFF0, 16#0003, 16#20, 16#82, 16#32
...	...
...	...
...	...
Slot, SubSlot, Index, DataSize, Data1, Data2	16#0000, 16#0001, 16#AFF0, 16#0002, 16#45, 16#56



Die Variable am Eingang "BackupFileName" sollte nur aus dem Dateinamen ohne Erweiterung bestehen.

Die Extension (*.pnb) wird intern verwaltet.

Beispielprogramm

In diesem Beispielprojekt wird das Wiederherstellen der Profinet IO-Device Geräteparameter von der – auf der Compact-Flash-Karte in der IndraControl-Steuerung gespeicherten – Sicherungsdatei beschrieben.

Vorgegeben sind der Profinet-Stationsname "IdentStr" des Profinet IO-Device und der Dateiname der Sicherungsdatei ohne Extension.

Bei steigender Flanke am "bExecute"-Eingang wird der Vorgang ausgeführt: Lesen der Parameter von der Sicherungsdatei und Schreiben auf das durch "strIdent" zu identifizierende Profinet IO-Device.

Deklaration

```
PROGRAM ProgIL_PNIODDeviceRestore
VAR
    fbIL_PNIODDeviceRestore: IL_PNIODDeviceRestore; // FB instance
    bExecute: BOOL; // FB running when TRUE
    strIdentString: STRING := 'Lx5-PN'; // Station name of PN
    strFileName: STRING := 'BackupFile'; // Backup file name
    tTimeout: TIME := T#2S; // Timeout monitoring the FB
    bDone: BOOL; // FB completed its task
    bActive: BOOL; // FB working on actual task
    bError: BOOL; // Indicates an error
    enErrorID: RIL_COMMONTYPES.ERROR_CODE; // Class of error
    enErrorIdent: RIL_COMMONTYPES.ERROR_STRUCT; // Detailed info on error
END_VAR
```

Implementierung

```
fbIL_PNIODDeviceRestore(
    Execute:= bExecute,
    Done:= bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> enErrorIdent,
    BusMaster:= IL_BUSMASTER.IL_BUSMASTER_0,
    IdentStr:= strIdentString,
    BackUpFileName:= strFileName,
    Timeout:= tTimeout);
```

Fehlerbehandlung

Der Funktionsbaustein **IL_PNIODDeviceRestore** nutzt die Fehlertabelle PROFINETIO_TABLE (ERROR_TABLE = 16#0200).

Der Funktionsbaustein kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Bemerkungen
INPUT_RANGE_ERROR	16#0001	16#0002	"IdentStr" ist ungültig
COMMUNICATION_ERROR	16#0003	Profinet IO-Controller Diagnose - Kap. 5.9.2 "Allgemeine Fehlercodes " auf Seite 196 - Kap. 5.9.3 "Spezielle Fehlercodes, Profinet IO-Controller" auf Seite 198	
COMMUNICATION_ERROR	16#0004	Profinet IO-Status Kap. 5.9.4 "Spezielle Fehlercodes, Profinet IO-Status " auf Seite 199	
SYSTEM_ERROR	16#0101	internal use	Profinet IO-Controller nicht in der Konfigurationstabelle gefunden
COMMUNICATION_ERROR	16#0104	16#0000	Zeitüberschreitung
INPUT_INVALID_ERROR	16#3002	16#0002	"BackupFileName" ist ungültig
ACCESS_ERROR	16#3003	16#0002	"Backup file" Format ist ungültig
ACCESS_ERROR	16#3003	16#0003	"Backup file" konnte nicht geöffnet werden
ACCESS_ERROR	16#3003	16#0004	"Backup file" schreiben/lesen nicht möglich

Tab. 5-21: Fehlercodes FB IL_PNIODeviceRestore

5.4 Unterstützung des Geräteausbaus (Device Replacement)

5.4.1 IL_PNIOAssignName

Kurzbeschreibung Der Funktionsbaustein **IL_PNIOAssignName** weist einem Profinet IO-Device den Device-Namen ("IdentStr") zu (das Gerät wird **benannt**). Das Profinet IO-Device wird durch seine MAC-Adresse adressiert.

Zielsystem	Bibliothek/Ordner
MLC	RIL_ProfinetIO/POUs/DeviceReplacement
MTX	RIL_ProfinetIO/POUs/DeviceReplacement

Tab. 5-22: Referenztable

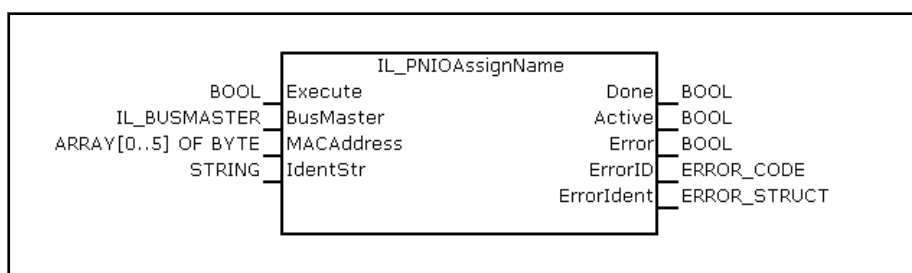


Abb. 5-12: Schnittstellenbeschreibung FB IL_PNIOAssignName

I/O-Typ	Name	Typ	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Profinet IO-Controllers in der Reihenfolge des Konfigurationsbaumes. Ist nur ein Profinet IO-Controller vorhanden, besitzt dieser Controller immer den Wert IL_BUSMASTER_0 siehe auch Kap. 5.8.1 "IL_BUSMASTER" auf Seite 189
	MACAddress	ARRAY[0..5] OF BYTE	MAC-Adresse des Feldbusteilnehmers
	IdentStr	STRING	Neuer Identifikationsstring des Feldbusteilnehmers
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall (aus RIL_CommonTypes)
	ErrorIdent	ERROR_STRUCT	Detailldiagnose (ErrorTable PROFINETIO_TABLE)

Tab. 5-23: I/O-Schnittstelle des FB IL_PNIOAssignName

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an "Execute"
MACAddress	ARRAY[0..5] OF BYTE	[0,0,0,0,0,0]	-	[0,0,0,0,0,0]	Steigende Flanke an "Execute"
IdentStr	STRING	-	-	"	Steigende Flanke an "Execute"

Tab. 5-24: Min-, Max- und Default-Werte der IL_PNIOAssignName-Eingänge

Funktionsbeschreibung

Der Funktionsbaustein **IL_PNIOAssignName** weist den Stationsnamen ("IdentStr") einem Profinet IO-Device zu. Die Adressierung erfolgt über die MAC-Adresse des Gerätes. Die MAC-Adresse kann dem Funktionsbaustein **IL_PNIOGetLiveList** entnommen werden. Der Stationsname "IdentStr" ist ein nicht case-sensitiver String.



Der Stationsname darf aus folgenden Zeichen bestehen:

- Kleinbuchstaben ohne Umlaute "a" bis "z".
Großbuchstaben ohne Umlaute "A" bis "Z" sind zulässig, werden aber intern in Kleinbuchstaben gewandelt.
Die Stationsnamen sind nicht case-sensitiv.
- Ziffern "0" bis "9" (dürfen nicht am Anfang des Namens stehen).
- Sonderzeichen Bindestrich "-" und Punkt "." (dürfen weder am Anfang noch am Ende des Namens stehen).
- Der Name darf max. 240 Zeichen lang sein. Bei manchen Geräten / Herstellern gilt die Beschränkung 127 Zeichen.
Der Name kann mittels Punkten "." in Teilstrings gegliedert werden. Ein solcher Teilstring darf max. 63 Zeichen lang sein.
Der Name darf nicht mit der Zeichenfolge "port-xyz" (mit x,y,z = 0...9) beginnen.
Der Name darf nicht die Form "a.b.c.d" (mit a,b,c,d = 0...999) haben.

Der Dienst kann nur einmal pro Profinet IO-Controller genutzt werden. Es wird der Profinet IO-Dienst DCP-Set genutzt.

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle **PROFINETIO_TABLE** (ERROR_TABLE = 16#0200).

Der Funktionsbaustein kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Bemerkung
INPUT_RANGE_ERROR	16#0001	2	"IdentStr" ist nicht gesetzt
INPUT_RANGE_ERROR	16#0001	3	"IdentStr" ist ungültig
INPUT_RANGE_ERROR	16#0001	4	MAC-Adresse ist ungültig (0)
SYSTEM_ERROR	16#0101	internal use	Profinet IO-Controller nicht in Konfigurationstabelle
COMMUNICATION_ERROR	16#0104	0	Zeitüberschreitung
STATE_MACHINE_ERROR	16#0105	0	Fehler in der State Machine
COMMUNICATION_ERROR	16#0003	Profinet IO-Controller-Diagnose • Siehe Kap. 5.9.2 "Allgemeine Fehlercodes " auf Seite 196 • Siehe Kap. 5.9.3 "Spezielle Fehlercodes, Profinet IO-Controller" auf Seite 198	
COMMUNICATION_ERROR	16#0004	Profinet IO-Status • Siehe Kap. 5.9.4 "Spezielle Fehlercodes, Profinet IO-Status " auf Seite 199	

Tab. 5-25: Fehlercodes FB IL_PNIOAssignName

5.4.2 IL_PNIOCheckTopology

Kurzbeschreibung Der Funktionsbaustein **IL_PNIOCheckTopology** überprüft die gespeicherte Topologie der Profinet-Gerätetauschfunktion anhand der aktuell vorhande-

nen Profinet-Topologie. Dies erfolgt durch Auslesen der Port-Informationen aus den Profinet IO-Devices.

Der Funktionsbaustein kann zum Beispiel eingesetzt werden, um Änderungen in der Verkabelung oder in der Anordnung der Geräte zu erkennen.

Zielsystem	Bibliothek/Ordner
MLC	RIL_ProfinetIO/POUs

Tab. 5-26: Referenztablelle

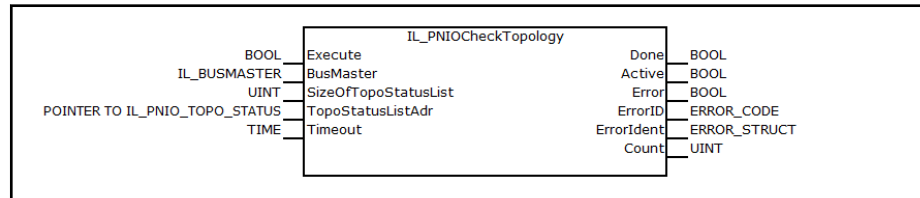


Abb. 5-13: Schnittstellenbeschreibung FB IL_PNIOCheckTopology

I/O-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Profinet IO-Controllers in der Reihenfolge des Konfigurationsbaums. Ist nur ein Profinet IO-Controller vorhanden, besitzt dieser Controller immer den Wert IL_BUSMASTER_0 Siehe auch Kap. 5.8.1 "IL_BUSMASTER" auf Seite 189
	SizeOfTopoStatusList	UINT	Größe des Arrays "TopoStatusListAdr" in Byte (z. B. SIZE-OF() verwenden)
	TopoStatusListAdr	POINTER TO IL_PNIO_TOPO_STATUS	Zeiger auf ein Array vom Typ IL_PNIO_TOPO_STATUS (z.B. ADR() verwenden). Die Größe dieses Arrays muss beim Eingang "SizeOfTopoStatusList" übergeben werden. Es werden nur Geräte mit einem Fehler angezeigt Kap. 5.8.9 "IL_PNIO_TOPO_STATUS" auf Seite 193
	Timeout	TIME	Timeout-Überwachung des Funktionsbausteins
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibt die Diagnose im Fehlerfall (aus RIL_Common-Types)
	ErrorIdent	ERROR_STRUCT	Detailldiagnose (Fehlertabelle PROFINETIO_TABLE) "Fehlerbehandlung" auf Seite 98
	Count	UINT	Anzahl der gültigen Elemente in "TopoStatusListAdr"

Tab. 5-27: I/O-Schnittstelle des FB IL_PNIOCheckTopology

Min-, Max- und Default-Werte: In der folgenden Tabelle sind die Min-, Max- und Default-Werte der Funktionsbausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an "Execute"
SizeOfTopoStatusList	UINT	0	n.def.	0	Steigende Flanke an "Execute"
TopoStatusListAdr	POINTER TO IL_PNIO_TOPO_STATUS	-	-	0	Kontinuierlich
Timeout	TIME	T#0S	n.def.	T#20S	Steigende Flanke an "Execute"

Tab. 5-28: Min-, Max- und Default-Werte der IL_PNIOCheckTopology-Eingänge

Funktionsbeschreibung

Profinet-IO-Devices tauschen über das "Link Layer Discovery Protocol" (LLDP) Nachbarinformationen aus, d. h. Profinet IO-Devices kennen den "Stationsnamen" ihrer Nachbargeräte und auch über welche Ports (eigener Port und Port des Nachbarn) die Verkabelung eingerichtet ist (siehe exemplarische Profinet-Topologie in nachfolgender Abbildung).

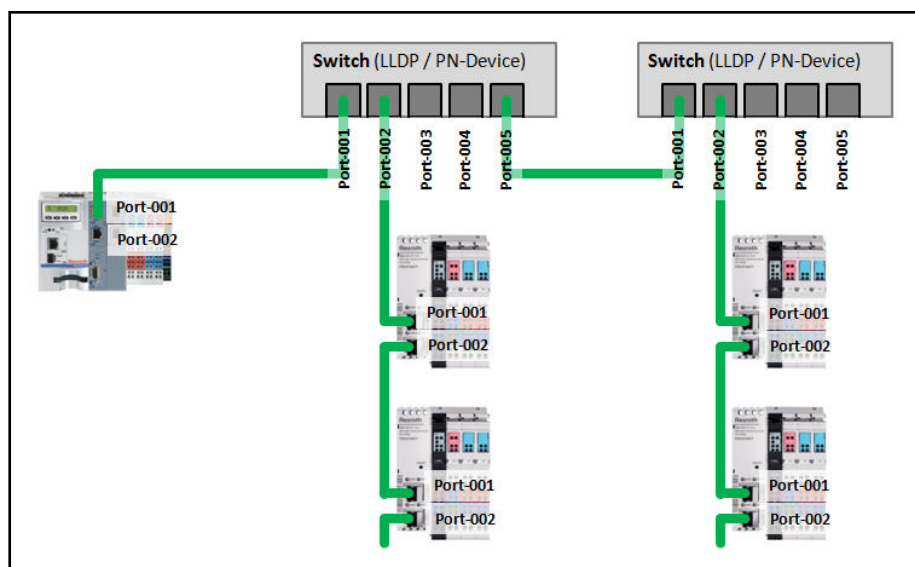


Abb. 5-14: Beispiel Profinet-Topologie

Der Profinet IO-Controller speichert diese Nachbarinformationen ab, um bei einem Gerätetausch das neue Profinet IO-Device (identischer Typ, bei richtiger Topologie/Verkabelung) automatisch mit dem gespeicherten Stationsnamen benennen zu können.

Voraussetzung hierfür ist, dass die komplette Profinet-Konfiguration in fehlerfreier Kommunikation ist und auch eingesetzte Switches müssen LLDP unterstützen bzw. als Profinet IO-Device konfiguriert sein. Nur auf diese Weise kann der Profinet IO-Controller die Nachbarinformationen gültig auslesen und abspeichern.

Der Funktionsbaustein **IL_PNIOCheckTopology** liest von allen konfigurierten Profinet IO-Devices die aktuellen Nachbar-Informationen aus und vergleicht diese mit den gespeicherten Daten der Gerätetauschfunktion.

Hierbei werden die Stationsnamen der Nachbargeräte sowie die PortIDs des jeweiligen Gerätes verglichen (Gerätetyp/DeviceID und Hersteller/VendorID wird nicht überprüft).

Ergebnis ist ein "TopoState" zu jedem Profinet IO-Device.

Mit diesem Topologie-Vergleich können beispielsweise folgende Anwendungsfälle umgesetzt werden:

- aktuelle Topologie bzw. Verkabelung prüfen, ob diese Topologie noch mit der gespeicherten Topologie der Gerätetauschfunktion übereinstimmt. Nur dann kann der Profinet IO-Controller bei einem Gerätetausch ein automatisches Benennen durchführen.

Verkabelung der Ports wurde vertauscht:

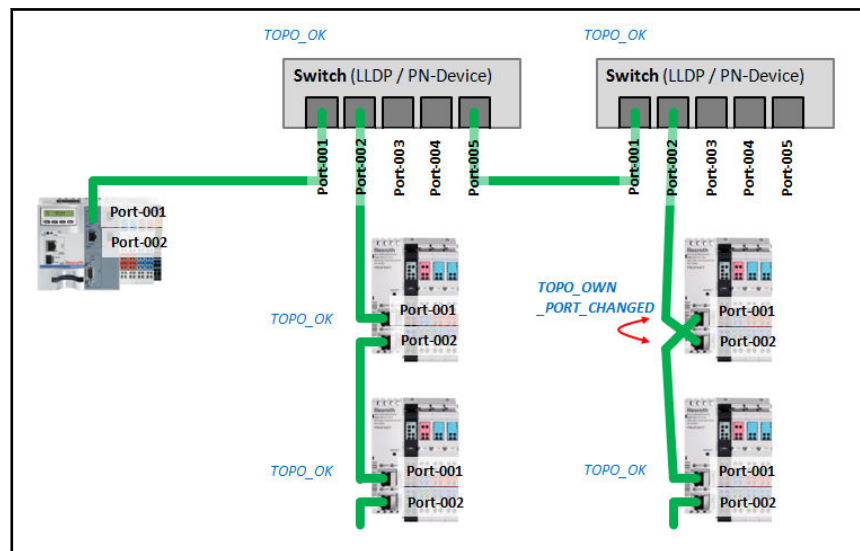


Abb. 5-15: Beispiel Profinet-Topologie, Verkabelung der Ports wurde vertauscht

- **Kreuztausch:** um zu Überprüfen, ob ein Fehler bei einer Anlage durch ein Profinet-IO-Device verursacht wird, wird zum Beispiel dieses Profinet IO-Device mit einem identischen Profinet IO-Device "kreuzgetauscht", um ein "Mit-Wandern" des Fehlers zu prüfen. Da der Profinet IO-Controller beim Verbindungsaufbau der zyklischen Kommunikation die Topologie **nicht** überprüft, würde bei einem Kreuztausch die Kommunikation weiterlaufen, auch wenn die Devices inkl. Stationsname nun getauscht sind.

Der **IL_PNIOCheckTopology** kann erkennen, dass die aktuellen Nachbarinformationen nicht mehr übereinstimmen und im Anwenderprogramm könnten in der Folge die getauschten Geräte zurückgesetzt werden ("ResetToDefault"). Auf diese Weise könnte dann die Gerätetauschfunktion die zurückgesetzten Profinet IO-Devices dann wieder entsprechend der gespeicherte Konfiguration benennen.

Nachfolgende Abbildung zeigt einen Beispiel-Kreuztausch. "M1Dev1" von Modul 1 wurde getauscht mit "M2Dev1" von Modul2.

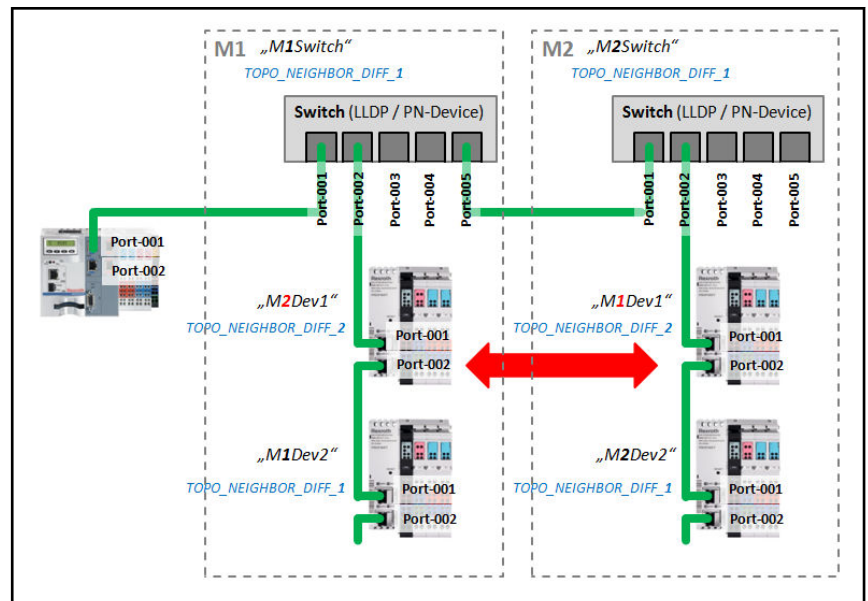


Abb. 5-16: Beispiel Profinet-Topologie, Kreuztausch

- Lokalisierung von Fehlern u. Problemen:** die normale Profinet IO-Diagnose zeigt bereits an, wenn ein Profinet IO-Device nicht erreichbar ist. Beim Topologievergleich würde dieser Fehler "Read-Error/Not-Reachable" in gleicher Weise auch identifiziert werden und zudem auch die betroffenen Nachbargeräte, was bei der Lokalisierung eines Problems weiterhelfen kann.

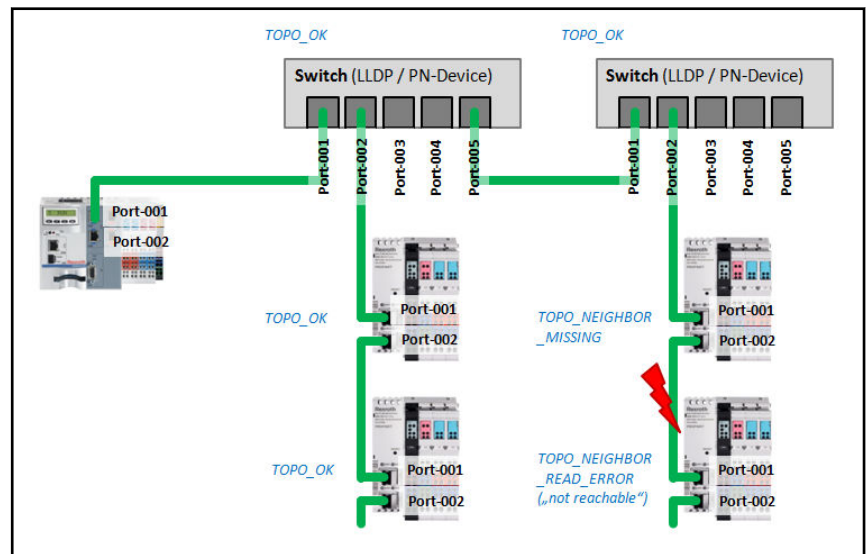


Abb. 5-17: Beispiel Profinet-Topologie, Lokalisierung von Fehlern und Problemen

**Eingang "Timeout":**

Ist ein Profinet IO-Device nicht erreichbar, entsteht beim internen "ReadImplicit"-Dienst ein Timeout. Dieser Timeout beträgt 3 sec und ist von außen nicht änderbar.

Sind mehrere Profinet IO-Devices nicht erreichbar, ist beim Eingang "Timeout" des **IL_PNIOCheckTopology** ein entsprechend hoher Wert einzustellen; ansonsten wird dieser FB einen "User-Timeout"-Error anzeigen.

**MAC-Adresse in "TopoStatusList":**

Die MAC-Adressen der Profinet IO-Devices werden nur beim Hochlauf des Profinet IO-Controllers ausgelesen. D. h. findet ein Gerätetausch mit einem Ersatzgerät während dem laufenden Betrieb (also ohne Neustart des Profinet IO-Controllers) statt, dann findet keine Aktualisierung der MAC-Adressen statt!

**Abhängigkeit zur Gerätetauschfunktion:**

Voraussetzung zur Ausführung der "CheckTopology"-Funktion ist, dass die Gerätetauschfunktion die Topologie gültig gespeichert hat (siehe z. B. "ReadyForReplace" bei **IL_PNIODDeviceReplacement**).

Fehlerbehandlung

Der Funktionsbaustein **IL_PNIOCheckTopology** nutzt die Fehlertabelle PROFINETIO_TABLE (ERROR_TABLE = 16#0200).

Der Funktionsbaustein kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Bemerkungen
INPUT_RANGE_ERROR	16#0001	16#0000	"TopoStatusListAdr" ist "0"
INPUT_RANGE_ERROR	16#0001	16#0001	"SizeOfTopoStatusList" ist "0"
INPUT_RANGE_ERROR	16#0001	16#0101	Profinet IO-Device nicht gefunden (Name/IP)
INPUT_RANGE_ERROR	16#0001	16#0102	Puffer zu klein
ACCESS_ERROR	16#0005	16#0000	Bei mindestens einem PN-Device konnten keine Port-Daten (Anzahl Ports = 0) ausgelesen werden; vermutlich werden hier die Nachbarschafts-Daten (LLDP) nicht unterstützt. Eine Überprüfung der Topologie ist nicht möglich.
ACCESS_ERROR	16#0005	16#0100	Gespeicherte Daten stehen nicht zur Verfügung ("ReadyForReplace" ist nicht gesetzt)
SYSTEM_ERROR	16#0101	internal use	Profinet IO-Controller nicht in Konfigurationstabelle
COMMUNICATION_ERROR	16#0104	internal use	Zeitüberschreitung
STATE_MACHINE_ERROR	16#0105	internal use	Fehlerhafter Zustand (intern)

Tab. 5-29: Fehlercodes **IL_PNIOCheckTopology**

5.4.3 IL_PNIODDeviceReplacement

Kurzbeschreibung

Der Funktionsbaustein **IL_PNIODDeviceReplacement** liefert eine Diagnose über den Status der automatischen Gerätetauschfunktion. Die Gerätetauschfunktion kann auch deaktiviert werden.

Zielsystem	Bibliothek/Ordner
MLC	RIL_ProfinetIO/POUs
MTX	RIL_ProfinetIO/POUs

Tab. 5-30: Referenztable

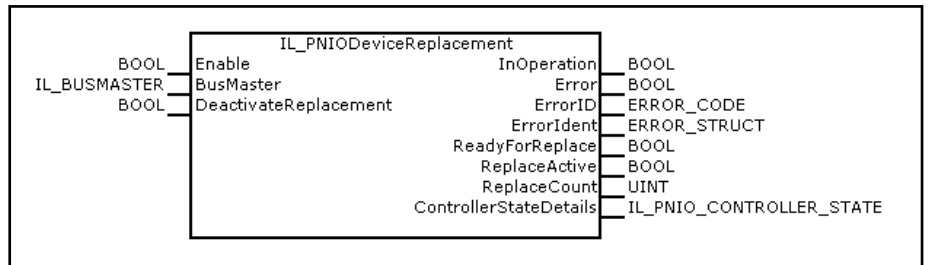


Abb. 5-18: Schnittstellenbeschreibung FB IL_PNIODeviceReplacement

I/O-Typ	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Profinet IO-Controllers in der Reihenfolge des Konfigurationbaumes. Ist nur ein Profinet IO-Controller vorhanden, besitzt dieser Controller immer den Wert IL_BUSMASTER_0 siehe auch Kap. 5.8.1 "IL_BUSMASTER" auf Seite 189
	DeactivateReplacement	BOOL	Mit TRUE wird die automatische Gerätetauschfunktion des Profinet IO-Controllers deaktiviert. Ist Enable=FALSE wird der Defaultzustand wieder hergestellt, d. h. die Gerätetauschfunktion ist wieder aktiviert
VAR_OUTPUT	InOperation	BOOL	Zyklischer Bearbeitungszustand
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall (aus RIL_CommonTypes)
	ErrorIdent	ERROR_STRUCT	Detailldiagnose (ErrorTable PROFINETIO_TABLE)
	ReadyForReplace	BOOL	Zeigt an, ob die Profinet-Topologie vollständig erkannt u. gespeichert wurde und die Gerätetauschfunktion damit "funktionsfähig" ist
	ReplaceActive	BOOL	Zeigt an, dass für ein konfiguriertes Profinet IO-Device, dessen Kommunikation nicht läuft, ein Ersatzgerät gesucht wird
	ReplaceCount	UINT	Anzahl automatischer Benennungen durch den Profinet IO-Controller aufgrund der gespeicherten Topologie
	StateDetails	IL_PNIO_CONTROLLER_STATE	Detaillierte Statusinformationen des Profinet IO-Controller-Stacks Siehe auch Kap. 5.8.4 "IL_PNIO_CONTROLLER_STATE" auf Seite 190

Tab. 5-31: I/O-Schnittstelle des FB IL_PNIODeviceReplacement

Name	Typ	Min	Max	Default	Übernahme
Enable	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an "Enable"
DeactivateReplacement	BOOL	-	-	FALSE	Kontinuierlich

Funktionsbeschreibung Der Funktionsbaustein **IL_PNIODeviceReplacement** liefert eine Diagnose über den Status der automatischen Gerätetauschfunktion. Zudem kann die Gerätetauschfunktion auch deaktiviert werden.

Mit dem Eingang "DeactivateReplacement" kann die Gerätetauschfunktion deaktiviert werden. Ist `Enable=FALSE`, wird der Defaultzustand wieder hergestellt, d. h. die automatische Gerätetauschfunktion des Profinet IO-Controllers ist aktiviert.

Der Ausgang "ReadyForReplace" zeigt an, ob der Profinet-Controller die Profinet-Topologie vollständig erkannt und gespeichert hat (Voraussetzung: alle konfigurierten Profinet IO-Devices sind in fehlerfreier Kommunikation), d. h. die automatische Gerätetauschfunktion ist dann "funktionsfähig".

"ReplaceActive" zeigt an, dass für ein konfiguriertes Profinet IO-Device, dessen Kommunikation nicht läuft, ein Ersatzgerät gesucht wird (durch Prüfung der gespeicherten Nachbarschaftsinformationen).

Folgendes ist zu beachten:

Die Aktualisierung dieses Flags findet in einem langsameren Zyklus (mehrere Sekunden) statt als die Profinet-Diagnose, die die aktive oder inaktive Kommunikation eines Gerätes anzeigt.

Die Anzahl der automatisch durchgeführten Benennungen (seit Einschalten der Steuerung) wird in "ReplaceCount" ausgegeben.

Der Ausgang "ControllerStateDetails" ist identisch zum Funktionsbaustein [Kap. 5.5.2 "IL_PNIOControllerStateDetails" auf Seite 118](#), d. h. es werden Detail-Diagnosen zum Profinet IO-Controller ausgegeben, z. B. Anzahl der konfigurierten Profinet IO-Devices und Anzahl der Profinet IO-Devices in aktiver Kommunikation.

Fehlerbehandlung Der Funktionsbaustein **IL_PNIODeviceReplacement** nutzt die **Fehlertabelle PROFINETIO_TABLE** (ERROR_TABLE = 16#0200).

Der Funktionsbaustein kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Bemerkungen
SYSTEM_ERROR	16#0005	internal use	Interner Fehler (RIL_ProfinetIO.library)
SYSTEM_ERROR	16#0101	internal use	Controller nicht in der Konfigurationstabelle gefunden

Tab. 5-32: Fehlercodes FB IL_PNIODeviceReplacement

5.4.4 IL_PNIODeviceResetToDefault

Kurzbeschreibung Der Funktionsbaustein **IL_PNIODeviceResetToDefault** setzt ein Profinet IO-Device auf den Auslieferungszustand (Factory-Default) zurück. Die Adressierung des Profinet IO-Devices erfolgt anhand der MAC-Adresse.

Zielsystem	Bibliothek/Ordner
MLC	RIL_ProfinetIO/POUs/DeviceReplacement
MTX	RIL_ProfinetIO/POUs/DeviceReplacement

Tab. 5-33: Referenztabelle

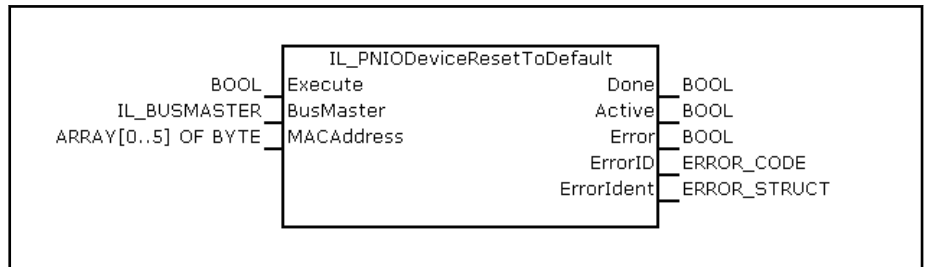


Abb. 5-19: Schnittstellenbeschreibung FB IL_PNIODeviceResetToDefault

I/O-Typ	Name	Typ	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Profinet IO-Controllers in der Reihenfolge des Konfigurationsbaumes. Ist nur ein Profinet IO-Controller vorhanden, besitzt dieser Controller immer den Wert IL_BUSMASTER_0 siehe auch Kap. 5.8.1 "IL_BUSMASTER" auf Seite 189
	MACAddress	ARRAY[0..5] OF BYTE	MAC-Adresse des Feldbusteilnehmers
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall (aus RIL_CommonTypes)
	ErrorIdent	ERROR_STRUCT	Detailldiagnose (ErrorTable PROFINETIO_TABLE)

Tab. 5-34: I/O-Schnittstelle des FB IL_PNIODeviceResetToDefault

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an Execute
MACAddress	ARRAY[0..5] OF BYTE	[0,0,0,0,0,0]		[0,0,0,0,0,0]	Steigende Flanke an Execute

Tab. 5-35: Min-, Max- und Default-Werte der IL_PNIODeviceResetToDefault-Eingänge

Funktionsbeschreibung

Der Funktionsbaustein **IL_PNIODeviceResetToDefault** setzt ein Profinet IO-Device auf den Auslieferungszustand (Factory-Default) zurück. Dabei wird unter anderem auch der Device-Name rückgesetzt bzw. gelöscht. Die Adressierung des Profinet IO-Device erfolgt über die MAC-Adresse. Die MAC-Adresse kann dem Funktionsbaustein **IL_PNIOGetLiveList** entnommen werden.

Der Dienst kann nur einmal pro Controller genutzt werden.

Fehlerbehandlung Der Funktionsbaustein **IL_PNIODeviceResetToDefault** nutzt die Fehlertabelle **PROFINETIO_TABLE** (ERROR_TABLE = 16#0200).
Der Funktionsbaustein kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Bemerkung
INPUT_RANGE_ERROR	16#0001	4	MAC-Adresse ist ungültig (0)
SYSTEM_ERROR	16#0101	internal use	Controller nicht in Konfigurationstabelle
COMMUNICATION_ERROR	16#0104	0	Zeitüberschreitung
STATE_MACHINE_ERROR	16#0105	0	Fehler in der State Machine
COMMUNICATION_ERROR	16#0003	Profinet IO-Controller-Diagnose - Siehe Kap. 5.9.2 "Allgemeine Fehlercodes " auf Seite 196 - Siehe Kap. 5.9.3 "Spezielle Fehlercodes, Profinet IO-Controller" auf Seite 198	
COMMUNICATION_ERROR	16#0004	<i>PROFINET IO Status</i> Siehe Kap. 5.9.4 "Spezielle Fehlercodes, Profinet IO-Status " auf Seite 199	

Tab. 5-36: Fehlercodes FB IL_PNIODeviceResetToDefault

5.4.5 IL_PNIODeviceSetIP

Kurzbeschreibung Der Funktionsbaustein **IL_PNIODeviceSetIP** setzt die IP-Einstellungen eines Profinet IO-Devices. Das Profinet IO-Device wird über seine MAC-Adresse adressiert.

Zielsystem	Bibliothek /Ordner
MLC	RIL_ProfinetIO/POUs/DeviceReplacement
MTX	RIL_ProfinetIO/POUs/DeviceReplacement

Tab. 5-37: Referenztablelle

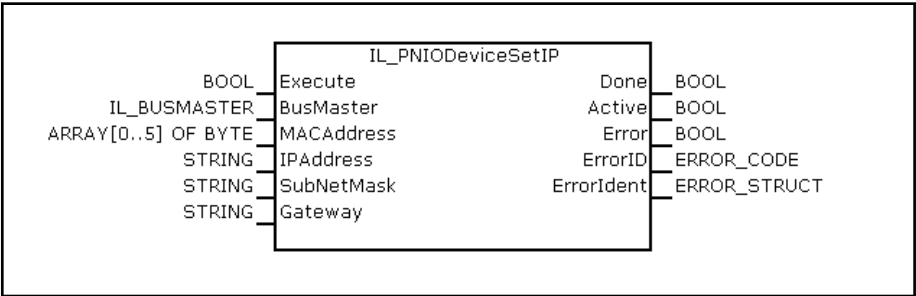


Abb. 5-20: Schnittstellenbeschreibung FB IL_PNIODeviceSetIP

I/O-Typ	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Profinet IO-Controllers in der Reihenfolge des Konfigurationsbaumes. Ist nur ein Profinet IO-Controller vorhanden, besitzt dieser Controller immer den Wert IL_BUSMASTER_0 siehe auch Kap. 5.8.1 "IL_BUSMASTER" auf Seite 189
	MACAddress	ARRAY[0..5] OF BYTE	MAC-Adresse des Feldbusteilnehmers
	IPAddress	STRING(16)	IP-Adresse des Feldbusteilnehmers
	SubNetMask	STRING(16)	Subnetmaske des Feldbusteilnehmers
	Gateway	STRING(16)	Gatewayadresse des Feldbusteilnehmers
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall (aus RIL_CommonTypes)
	ErrorIdent	ERROR_STRUCT	Detaildiagnose (ErrorTable PROFINETIO_TABLE)

Tab. 5-38: I/O-Schnittstelle des FB IL_PNIODDeviceSetIP

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an "Execute"
MACAddress	ARRAY[0..5] OF BYTE			[0,0,0,0,0,0]	Steigende Flanke an "Execute"
IPAddress	STRING	'0.0.0.0'	'255.255.255.255'	'0.0.0.0'	Steigende Flanke an "Execute"
SubNetMask	STRING	'0.0.0.0'	'255.255.255.255'	'0.0.0.0'	Steigende Flanke an "Execute"
Gateway	STRING	'0.0.0.0'	'255.255.255.255'	'0.0.0.0'	Steigende Flanke an "Execute"

Tab. 5-39: Min-, Max- und Default-Werte der IL_PNIODDeviceSetIP-Eingänge

Funktionsbeschreibung

Der Funktionsbaustein **IL_PNIODDeviceSetIP** setzt die IP-Adresse, die Subnetzmaske und die Gatewayadresse eines Profinet IO-Devices per DCP-Set Dienst. Die Übergabewerte müssen in der Form "aaA.bbB.ccC.ddD" (a, b, c, d sind optional) vorliegen und werden auf Plausibilität geprüft. Die Adressierung des Profinet IO-Devices erfolgt über die MAC-Adresse. Die MAC-Adresse kann dem Funktionsbaustein **IL_PNIODGetLiveList** entnommen werden. Der Dienst kann nur einmal pro Profinet IO-Controller genutzt werden.

Fehlerbehandlung

Der Funktionsbaustein **IL_PNIODDeviceSetIP** nutzt die Fehlertabelle **PROFINETIO_TABLE** (ERROR_TABLE = 16#0200).

Der Funktionsbaustein kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Bemerkung
INPUT_RANGE_ERROR	16#0001	4	MAC-Adresse ist ungültig (0)
INPUT_RANGE_ERROR	16#0001	5	IP-Adresse ist ungültig
INPUT_RANGE_ERROR	16#0001	6	Subnet-Mask ist ungültig
INPUT_RANGE_ERROR	16#0001	7	Gateway ist ungültig
SYSTEM_ERROR	16#0101	internal use	Profinet IO-Controller nicht in Konfigurationstabelle
COMMUNICATION_ERROR	16#0104	0	Zeitüberschreitung
STATE_MACHINE_ERROR	16#0105	0	Fehler in der State Machine
COMMUNICATION_ERROR	16#0003	Profinet IO-Controller-Diagnose - Siehe Kap. 5.9.2 "Allgemeine Fehlercodes " auf Seite 196 - Siehe Kap. 5.9.3 "Spezielle Fehlercodes, Profinet IO-Controller" auf Seite 198	
COMMUNICATION_ERROR	16#0004	Profinet IO-Status Siehe Kap. 5.9.4 "Spezielle Fehlercodes, Profinet IO-Status " auf Seite 199.	

Tab. 5-40: Fehlercodes FB IL_PNIODeviceSetIP

5.4.6 IL_PNIOGetLiveList

Kurzbeschreibung Der Funktionsbaustein **IL_PNIOGetLiveList** liefert eine Liste der aktuell erreichbaren Profinet IO-Teilnehmer.



Ein zyklischer Aufruf des Funktionsbausteins **IL_PNIOGetLiveList** kann zur Fehlfunktion von Profinet-Funktionen führen, z.B. >Geräte suchen< oder >Device Replacement<

Zielsystem	Bibliothek/Ordner
MLC	RIL_ProfinetIO/ POU's/ DeviceReplacement
MTX	RIL_ProfinetIO/ POU's/ DeviceReplacement

Tab. 5-41: Referenztablelle

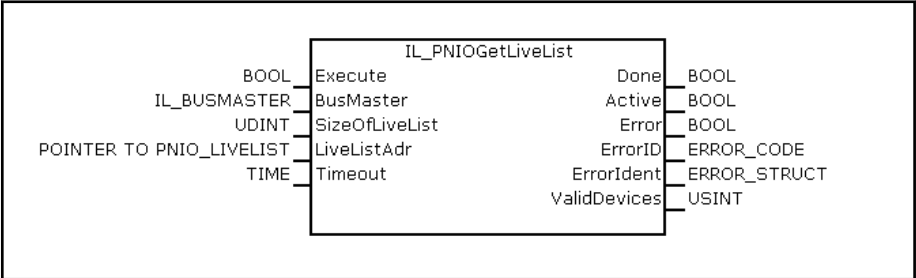


Abb. 5-21: Schnittstellenbeschreibung FB IL_PNIOGetLiveList

I/O-Typ	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Profinet IO-Controllers in der Reihenfolge des Konfigurationsbaumes. Ist nur ein Profinet IO-Controller vorhanden, besitzt dieser Controller immer den Wert IL_BUSMASTER_0 siehe auch Kap. 5.8.1 "IL_BUSMASTER" auf Seite 189
	SizeOfLiveList	UDINT	Maximale Größe des Datenbereichs von "LiveListAdr" in Byte. Muss ein ganzzahliges Vielfaches von SIZE-OF(PNIO_LIVELIST) sein
	LiveListAdr	PNIO_LIVELIST	Zeiger auf die Datenablage der Live-Liste. Dies sollte ein Zeiger auf ein Array vom Typ PNIO_LIVELIST sein siehe auch Kap. 5.8.13 "PNIO_LIVELIST" auf Seite 194
	Timeout	TIME	Timeout – maximale Wartezeit des Profinet IO-Controllers auf die Rückmeldung. T#0ms, Timeout-Überwachung ist inaktiv
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall (aus RIL_CommonTypes)
	ErrorIdent	ERROR_STRUCT	Detaildiagnose (ErrorTable PROFINETIO_TABLE)
	ValidDevices	USINT	Anzahl der gültigen Profinet IO-Devices

Tab. 5-42: I/O-Schnittstelle des FB IL_PNIOGetLiveList

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an "Execute"
SizeOfLiveList	UDINT	0	nicht def.	0	Steigende Flanke an "Execute"
LiveListAdr	POINTER TO PNIO_LIVELIST	nicht def.	nicht def.	0	Steigende Flanke an "Execute"
Timeout	TIME	T#0s	T#60s	T#20s	Steigende Flanke an "Execute"

Tab. 5-43: Min-, Max- und Default-Werte der IL_PNIOGetLiveList-Eingänge

Funktionsbeschreibung

Der Funktionsbaustein **IL_PNIOGetLiveList** löst einen DCP-IdentAll-Service aus und wartet die Timeout-Zeit auf Rückmeldungen der Profinet IO-Devices. Die Rückmeldungen werden in der Liste eingetragen.

Wenn nicht alle Profinet IO-Devices in der Liste eingetragen werden können, erfolgt eine Fehlermeldung.

Fehlerbehandlung Der Funktionsbaustein **IL_PNIOGetLiveList** nutzt die Fehlertabelle **PROFINE-TIO_TABLE** (ERROR_TABLE = 16#0200).
Der Funktionsbaustein kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Bemerkung
INPUT_RANGE_ERROR	16#0001	0	"LiveListAdr" ist nicht gesetzt
INPUT_RANGE_ERROR	16#0001	1	"SizeOfLiveList" ist nicht gesetzt
INPUT_RANGE_ERROR	16#0001	3	LiveListAdr: Übergebene Liste ist zu kurz
SYSTEM_ERROR	16#0101	internal use	Profinet IO-Controller nicht in Konfigurationstabelle
COMMUNICATION_ERROR	16#0104	0	Zeitüberschreitung
STATE_MACHINE_ERROR	16#0105	0	Fehler in der State Machine
COMMUNICATION_ERROR	16#0003	Profinet IO-Controller-Diagnose - Siehe Kap. 5.9.2 "Allgemeine Fehlercodes " auf Seite 196 - Siehe Kap. 5.9.3 "Spezielle Fehlercodes, Profinet IO-Controller" auf Seite 198	

Tab. 5-44: Fehlercodes FB IL_PNIOGetLiveList

5.4.7 IL_PNIOGetLogMsg

Kurzbeschreibung Der Funktionsbaustein holt einen Loggertext zum Gerätetausch aus dem PROFINET IO-Controller.
Es wird der jeweils älteste Text geholt. Jeder Text kann nur einmal geholt werden.

Zielsystem	Bibliothek /Ordner
MLC	RIL_ProfinetIO/ POU's/ DeviceReplacement
MTX	RIL_ProfinetIO/ POU's/ DeviceReplacement

Tab. 5-45: Referenztabelle

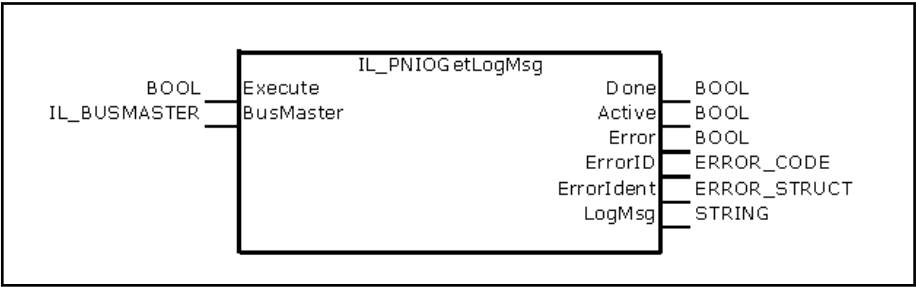


Abb. 5-22: Schnittstelle FB IL_PNIOGetLogMsg

	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Controllers in der Reihenfolge des Konfigurationsbaumes. Ist nur ein PROFINET IO- Controller vorhanden, besitzt dieser immer den Wert IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 189
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall (aus RIL_CommonTypes)
	ErrorIdent	ERROR_STRUCT	Detaildiagnose (ErrorTable PROFINETIO_TABLE)
	LogMsg	STRING(128)	Loggertext des Gerätetausches

Tab. 5-46: FB IL_PNIOGetLogMsg E/A Schnittstelle

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an Execute

Tab. 5-47: Min-/Max und Default-Werte der Eingänge IL_PNIOGetLogMsg

Funktionsbeschreibung Beim automatisierten Gerätetausch werden die Vorgänge in einem Logger protokolliert.

Mit diesem Baustein kann die jeweils älteste Protokollnachricht einmal gelesen werden.

Fehlerbehandlung Der Funktionsbaustein nutzt die **Fehlertabelle PROFINETIO_TABLE** (ERROR_TABLE = 16#0200).

Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Bemerkung
SYSTEM_ERROR	16#0101	internal use	Controller nicht in der Konfigurationstabelle
STATE_MACHINE_ERROR	16#0105	0	Fehler in der Zustandsmaschine

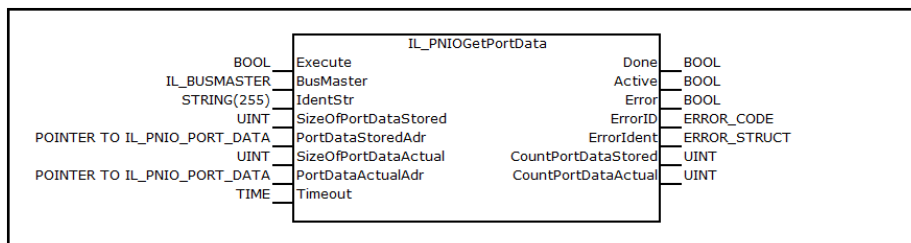
Tab. 5-48: Fehlercodes FB IL_PNIOGetLogMsg

5.4.8 IL_PNIOGetPortData

Kurzbeschreibung Der Funktionsbaustein **IL_PNIOGetPortData** liefert die Port-Daten ("Nachbar-Informationen") von einem Profinet IO-Device.

Zielsystem	Bibliothek/Ordner
MLC	RIL_ProfinetIO/POUs

Tab. 5-49: Referenztabelle

Abb. 5-23: Schnittstellenbeschreibung FB **IL_PNIOGetPortData**

I/O-Typ	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Profinet IO-Controllers in der Reihenfolge des Konfigurationsbaums. Ist nur ein Profinet IO-Controller vorhanden, besitzt dieser Controller immer den Wert IL_BUSMASTER_0 Siehe auch Kap. 5.8.1 "IL_BUSMASTER" auf Seite 189
	IdentStr	STRING	Identifikationsstring des Profinet IO-Device
	SizeOfPortDataStored	UINT	Größe des Arrays "PortDataStoredAdr" in Byte (z. B. SIZEOF() verwenden)
	PortDataStoredAdr	POINTER TO IL_PNIO_PORT_DATA	[optional] Pointer (z.B. ADR()) auf ein Array der gespeicherten Port-Daten aus "PNTopo_n.xml" (Gerätetauschfunktion des Profinet IO-Controllers); wenn diese Port-Daten nicht ausgelesen werden sollen, muss dieser Pointer auf "0" gesetzt werden Kap. 5.8.5 "IL_PNIO_PORT_DATA" auf Seite 191
	SizeOfPortDataActual	UINT	Größe des Arrays "PortDataActualAdr" in Byte (z. B. SIZEOF() verwenden)
	PortDataActualAdr	POINTER TO IL_PNIO_PORT_DATA	[optional] Pointer (z.B. ADR()) auf ein Array der aktuellen Port-Daten des Profinet IO-Device; wenn diese Port-Daten nicht ausgelesen werden sollen, muss dieser Pointer auf "0" gesetzt werden Kap. 5.8.5 "IL_PNIO_PORT_DATA" auf Seite 191
	Timeout	TIME	Timeout-Überwachung des Funktionsbausteins

I/O-Typ	Name	Typ	Kommentar
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibt die Diagnose im Fehlerfall (aus RIL_CommonTypes)
	ErrorIdent	ERROR_STRUCT	Detaildiagnose (Fehlertabelle PROFINE-TIO_TABLE) "Fehlerbehandlung" auf Seite 110
	CountPortDataStored	UINT	Anzahl der gültigen Elemente in "PortDataStoredAdr"
	CountPortDataActual	UINT	Anzahl der gültigen Elemente in "PortDataActualAdr"

Tab. 5-50: I/O-Schnittstelle des FB IL_PNIOGetPortData

Min-, Max- und Default-Werte: In der folgenden Tabelle sind die Min-, Max- und Default-Werte der Funktionsbausteineingänge aufgelistet.

Name	Typ	Min	Max	Default	Anwendung
Execute	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an "Execute"
IdentStr	STRING(255)	-	-	"	Steigende Flanke an "Execute"
SizeOfPortDataStored	UINT	0	n.def.	0	Steigende Flanke an "Execute"
PortDataStoreAdr	POINTER TO IL_PNIO_PORT_DATA	-	-	0	Steigende Flanke an "Execute"
SizeOfPortDataActual	UINT	0	n.def.	0	Steigende Flanke an "Execute"
PortDataActualAdr	POINTER TO IL_PNIO_PORT_DATA	-	-	0	Steigende Flanke an "Execute"
Timeout	TIME	T#0s	n.def.	T#2s	Steigende Flanke an "Execute"

Tab. 5-51: Min-, Max- und Default-Werte der IL_PNIOGetPortData-Eingänge

Funktionsbeschreibung

Der Funktionsbaustein **IL_PNIOGetPortData** liefert die Port-Daten ("Nachbar-Informationen") von einem Profinet IO-Device.

Um die Port-Daten aus einem PN-Device auslesen zu können, muss das PN-Device eine gültige IP-Adresse besitzen. Aus diesem Grund können die aktuellen Port-Daten nur aus einem konfigurierten PN-Device ausgelesen werden (bei diesem vergibt der Controller die konfigurierte IP-Adresse).

Es können die gespeicherten Port-Daten der Gerätetauschfunktion des Profinet IO-Controllers ausgelesen werden und/oder die aktuellen Port-Daten aus dem Gerät.

Bei den Port-Daten, die nicht ausgelesen werden sollen, ist der entsprechende Eingang auf "0" zu setzen ("PortDataStoredAdr"/"PortDataActualAdr").

Voraussetzung um die gespeicherten Port-Daten auszulesen ist, dass die Gerätetauschfunktion diese Port-Daten gültig gespeichert hat ("ReadyForReplace" ist TRUE, siehe Funktionsbaustein **IL_PNIODeviceReplacement**).

Fehlerbehandlung

Der Funktionsbaustein **IL_PNIOGetPortData** nutzt die Fehlertabelle PROFINETIO_TABLE (ERROR_TABLE = 16#0200).

Der Funktionsbaustein kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Bemerkungen
INPUT_RANGE_ERROR	16#0001	16#0000	"IdentStr" ist ungültig
INPUT_RANGE_ERROR	16#0001	16#0001	"PortDataStoredAdr" und "PortDataActualAdr" sind beide "0"
INPUT_RANGE_ERROR	16#0001	16#0002	"SizeOfPortDataStored" ist "0"
INPUT_RANGE_ERROR	16#0001	16#0003	"SizeOfPortDataActual" ist "0"
INPUT_RANGE_ERROR	16#0001	16#0101	profinet IO-Device nicht gefunden (Name/IP)
INPUT_RANGE_ERROR	16#0001	16#0102	Puffer zu klein
ACCESS_ERROR	16#0005	16#0100	Gespeicherte Daten stehen nicht zur Verfügung ("ReadyForReplace" ist nicht gesetzt)
ACCESS_ERROR	16#0005	internal use	Lesefehler zum aktuellen Profinet IO-Device (z. B. nicht erreichbar)
SYSTEM_ERROR	16#0101	internal use	Controller nicht in Konfigurationstabelle gefunden
COMMUNICATION_ERROR	16#0104	internal use	Zeitüberschreitung
STATE_MACHINE_ERROR	16#0105	internal use	Fehlerhafter Zustand (intern)

Tab. 5-52: Fehlercodes IL_PNIOGetPortData

5.4.9

IL_PNIOStoreTopology

Kurzbeschreibung

Der Funktionsbaustein startet das Generieren der aktuellen Netzwerktopologie-Datenbasis, wenn alle konfigurierten Devices erkannt wurden.

Die Topologie wird in einer XML-Datei abgelegt.

Zielsystem	Bibliothek /Ordner
MLC	RIL_ProfinetIO/ POU's/ DeviceReplacement
MTX	RIL_ProfinetIO/ POU's/ DeviceReplacement

Tab. 5-53: Referenztablelle

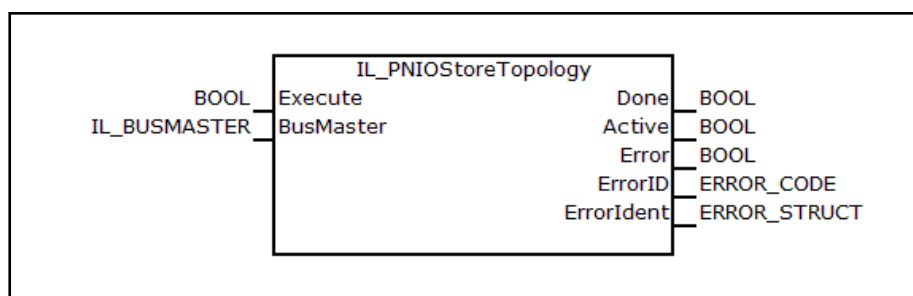


Abb. 5-24: Schnittstelle FB IL_PNIOSStoreTopology

	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Controllers in der Reihenfolge des Konfigurationsbaumes. Ist nur ein PROFINET IO- Controller vorhanden, besitzt dieser immer den Wert IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 189
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall (aus RIL_CommonTypes)
	ErrorIdent	ERROR_STRUCT	Detaildiagnose (ErrorTable PROFINETIO_TABLE)

Tab. 5-54: FB IL_PNIOSStoreTopology E/A Schnittstelle

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an Execute

Tab. 5-55: Min-/Max und Default-Werte der Eingänge IL_PNIOSStoreTopology

Funktionsbeschreibung	Der Funktionsbaustein startet das Generieren der Netzwerktopologie-Datenbasis. Vorraussetzung ist, dass alle Devices als fehlerfrei erkannt werden. Die Informationen werden als XML-Datei abgelegt. Der Dateiname und der Ablageort der XML-Datei können über die Konfigurationsdatei der Steuerung festgelegt werden.
Fehlerbehandlung	Der Funktionsbaustein nutzt die Fehlertabelle PROFINETIO_TABLE (ERROR_TABLE = 16#0200). Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Bemerkung
SYSTEM_ERROR	16#0101	internal use	Controller nicht in der Konfigurationstabelle
STATE_MACHINE_ERROR	16#0105	0	Fehler in der Zustandsmaschine
COMMUNICATION_ERROR	16#0003	<i>PROFINET IO Controller Diagnose</i> - Siehe Allgemeine Fehlercodes , Seite 196, - Siehe Spezielle Fehlercodes, Profinet IO-Controller, Seite 198.	

Tab. 5-56: Fehlercodes FB IL_PNIOStoreTopologie

5.4.10 IL_PNIORestoreTopology

Kurzbeschreibung Der Funktionsbaustein IL_PNIORestoreTopology versucht die gespeicherte Topologie des PNIO Controllers wiederherzustellen (z.B. Anwendungsfall "Kreuztausch" von PN Devices).

Der FB kann auch zur Überprüfung der Topologie-Information verwendet werden, d.h. ob die "Nachbarschafts-Informationen" von allen konfigurierten PN Devices ein komplett verbundenes Netzwerk bilden.

Zielsystem	Bibliothek /Ordner
MLC	RIL_ProfinetIO/ POU's/ DeviceReplacement
MTX	RIL_ProfinetIO/ POU's/ DeviceReplacement

Tab. 5-57: Referenztabelle

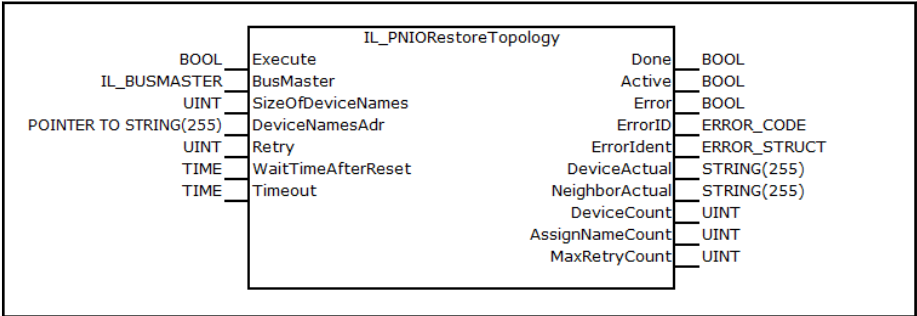


Abb. 5-25: Schnittstelle FB IL_PNIORestoreTopology

	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Controllers in der Reihenfolge des Konfigurationsbaumes. Ist nur ein PROFINET IO- Controller vorhanden, besitzt dieser immer den Wert IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 189
	SizeOfDeviceNames	UINT	Größe des Arrays DeviceNamesAdr in Byte (z.B. SIZE-OF())
	DeviceNamesAdr	POINTER TO STRING(255)	Zeiger auf ein Array von STRING(255) (z.B. ADR()). Hier werden die verarbeiteten DeviceNamen eingetragen. Die Größe dieses Arrays muss im Parameter ‚SizeOfDeviceNames‘ übergeben werden

	Name	Typ	Kommentar
	Retry	UINT	Anzahl Versuche für unterlagerte Basisfunktions-Aufrufe bis der FB einen Error meldet.
	WaitTimeAfterReset	TIME	Wartezeit bis ein PN Device nach einem Reset wieder erreichbar ist (z.B. Switches haben bis zu 50 sec).
	Timeout	TIME	Timeout für unterlagerte Basisfunktionen, T#0s = deaktiviert.
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall (aus RIL_CommonTypes)
	ErrorIdent	ERROR_STRUCT	Detaildiagnose (ErrorTable PROFINETIO_TABLE)
	DeviceActual	STRING(255)	DeviceName der aktuell geprüften Station
	NeighborActual	STRING(255)	DeviceName des Nachbarn, der aktuell geprüft wird
	DeviceCount	UINT	Anzahl verarbeiteter PN Devices (siehe DeviceNamesAdr)
	AssignNameCount	UINT	Anzahl ausgeführter "Benennungen" von PN Devices
	MaxRetryCount	UINT	Max Anzahl Ausführungen von unterlagerten Basisfunktionen

Tab. 5-58: FB IL_PNIORestoreTopology E/A Schnittstelle

Name	Typ	Min	Max	Default	Übernahme
Execute	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an Execute
SizeOfDeviceNames	UINT	0	-	0	Steigende Flanke an Execute
DeviceNamesAdr	POINTER TO STRING(255)	-	-	0	Kontinuierlich
Retry	UINT	0	-	5	Steigende Flanke an Execute
WaitTimeAfterReset	TIME	0	-	T#20s	Steigende Flanke an Execute
Timeout	TIME	0	-	T#3s	Steigende Flanke an Execute

Tab. 5-59: Min-/Max und Default-Werte der Eingänge IL_PNIORestoreTopology

Funktionsbeschreibung**Anwendungsfall:** Überprüfung der PROFINET Topologie-Informationen

Die Gerätetauschfunktion des PNIO Controllers speichert die Topologie-Information ("Nachbarschafts-Information" / "Port-Daten") des PROFINET Netzwerks automatisch ab.

Der FB IL_PNIORestoreTopology durchläuft beginnend beim Controller alle Geräte des PN Netzwerkes durch Auslesen der Port-Daten und überprüft auf diese Weise, ob die Nachbarschafts-Daten aller Geräte ein vollständig verbundenes Netzwerk bilden.

Wird beispielsweise ein Switch verwendet, der die Port-Informationen nicht unterstützt (Voraussetzung: LLDP – Link Layer Discovery Protocol), gehören die PN Devices "hinter" dem Switch nicht mehr zu einer vollständig verbundenen Topologie-Information. In diesem Fall meldet der FB einen Fehler, da nicht alle konfigurierten PN Devices mittels Nachbarschafts-Daten gefunden wurden. In gleicher Weise wäre hierbei dann auch die Gerätetauschfunktion des PNIO Controllers nur eingeschränkt funktionsfähig.

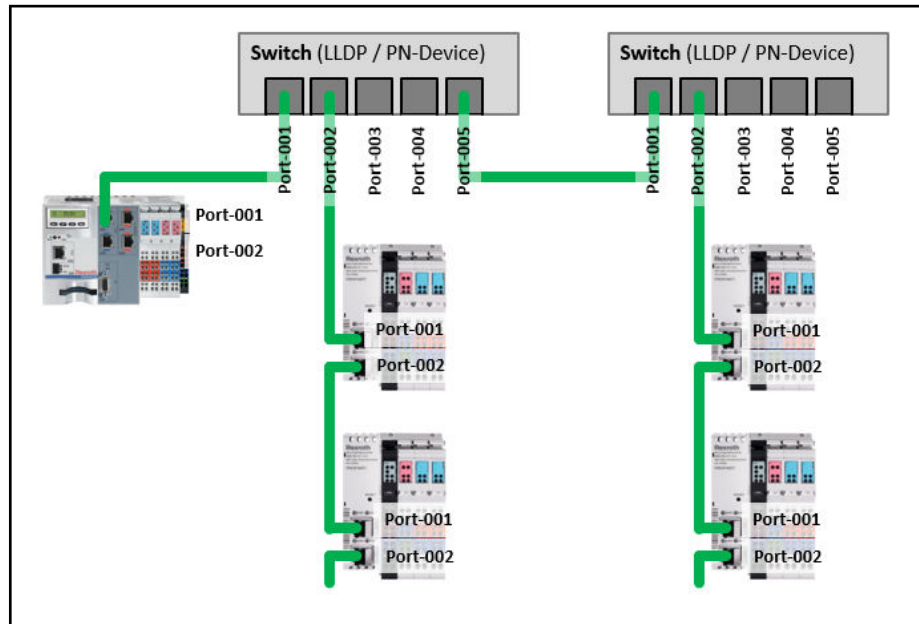


Abb. 5-26: Beispiel PROFINET Netzwerk mit Angabe der Port-Informationen

Anwendungsfall: Wiederherstellung der gespeicherten PROFINET Topologie

Der PNIO Controller überprüft beim Verbindungsaufbau nicht die Topologie bzw. nicht die "Stelle" an der ein bestimmtes PN Device verbunden ist.

Werden zwei identische PN Devices "kreuzgetauscht", um beispielsweise zu überprüfen, ob ein Fehler an dieser Station "mitwandert", behält jedes PN Device seinen gespeicherten Namen. D.h. die Topologie bzw. Reihenfolge der Devices würde damit geändert werden.

Der FB IL_PNIORestoreTopology durchläuft beginnend beim Controller alle Devices im Netzwerk mittels der Nachbarschafts-Informationen (Port-Daten) und erkennt die geänderte Topologie, hier am Beispiel die geänderten "StationsNamen". Der FB überprüft dann den Gerätetyp (VendorID u. DeviceID) und benennt die PN Devices dann wieder mit den gespeicherten "StationsNamen" um die Topologie wiederherzustellen.

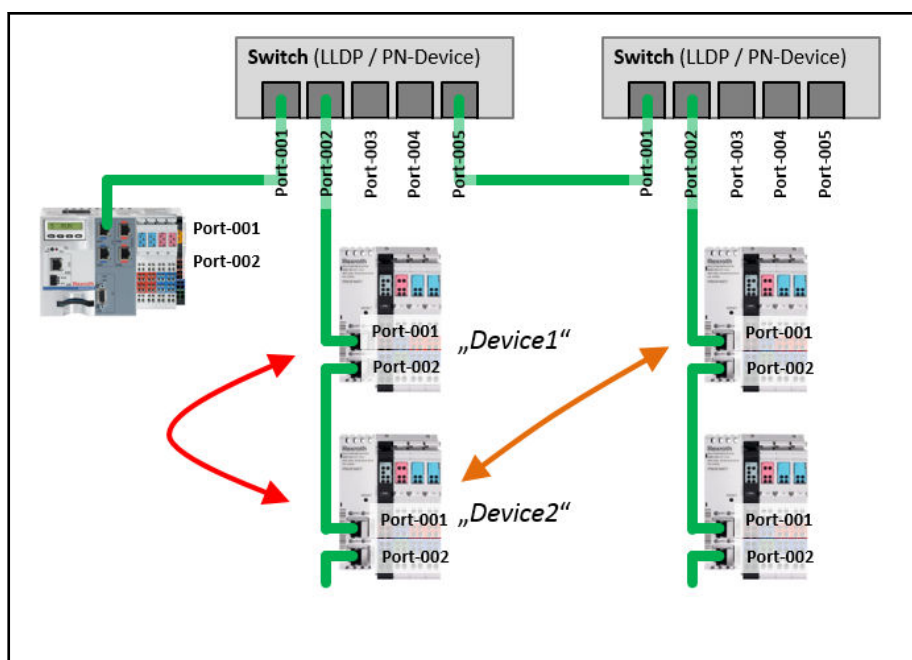


Abb. 5-27: Beispiel PROFINET Netzwerk, Anwendungsfall "Kreuztausch"

Stimmt der Gerätetyp nicht überein oder der Port-Steckplatz wurde geändert, meldet der FB IL_PNIORestoreTopology einen Topologie-Fehler. Details über Topologie-Änderungen können mit dem FB IL_PNIOCheckTopology ermittelt werden.

Hinweise zur Parametrierung und Anwendung des FBs

Voraussetzung um den FB IL_PNIORestoreTopology auszuführen ist, dass die Gerätetauschfunktion des PNIO Controllers die Topologie-Informationen des PROFINET Netzwerkes abgespeichert hat, siehe "ReadyForReplace"-Ausgang beim FB IL_PNIODeviceReplacement.

Es wird empfohlen während der Ausführung des FB IL_PNIORestoreTopology die Gerätetauschfunktion des PNIO Controllers zu deaktivieren (siehe FB IL_PNIODeviceReplacement), da intern genutzte Ressourcen sich gegenseitig blockieren können.

Bevor der FB IL_PNIORestoreTopology ein (Um-)Benennen eines PN Devices durchführt, wird ein "ResetToDefault" des PN Devices ausgeführt. Da PN Devices sehr unterschiedliche Hochlaufzeiten haben können, z.B. manche Switches über 50 Sekunden, kann dies am Eingang "WaitTimeAfterReset" eingestellt werden.

Zusätzlich kann am Eingang "Retry" eine Anzahl von Wiederholungen der unterlagerten Basisfunktionen ("ResetToDefault", "AssignName", "DCPGet") parametrierbar werden, um die "Robustheit" des FBs IL_PNIORestoreTopology zu erhöhen. Das Produkt aus "WaitTimeAfterReset" und "Retry" erhöht somit die Verfügbarkeit der Funktion, gleichzeitig allerdings auch die Ausführungszeit des FBs, z.B. auch die Zeit bis der FB aufgrund einer Topologie-Änderung einen Fehler meldet.

Voraussetzung, dass der Funktionsbaustein IL_PNIORestoreTopology ein PN-Device wieder richtig benennen kann bzw. die Topologie wiederherstellen kann, ist, dass die verwendeten PN-Devices im zyklischen Betrieb den DCP-Dienst "Reset to FactorySettings" (bzw. "Reset to Default") unterstützen.

Fehlerbehandlung

Der Funktionsbaustein nutzt die **Fehlertabelle PROFINETIO_TABLE** (ERROR_TABLE = 16#0200).

Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR	16#00000001	16#00000000	Eingang 'DeviceNamesAdr' ist 0
INPUT_INVALID_ERROR	16#00000001	16#00000001	Eingang 'SizeOfDeviceNames' ist 0
INPUT_RANGE_ERROR	16#00000001	16#00000003	'DeviceNamesAdr' zugewiesene Liste ist zu kurz
INPUT_INVALID_ERROR	16#00000001	x	siehe auch IL_PNIOGetPortData , Seite 107
ACCESS_ERROR	16#00000005	16#00000100	Gespeicherte Daten sind nicht komplett (ReadyForReplace ist nicht gesetzt)
ACCESS_ERROR	16#00000005	internal	Lesefehler, aktuelles Gerät (vielleicht nicht erreichbar)
INPUT_RANGE_ERROR	16#00000009	internal	Max. Anzahl an zugelassenen Geräten überschritten
ACCESS_ERROR	16#00000030	internal	Retry exceeded: access/com. error, Vorschlag: Retry/WaitTime/Timeout erhöhen
ACCESS_ERROR	16#00000031	internal	Nicht alle konfigurierten Geräte gefunden (vielleicht nicht erreichbar)
OTHER_ERROR	16#00000032	16#00000000	Topologiefehler: Ports getauscht (oder Nachbargerät fehlt)
OTHER_ERROR	16#00000032	16#00000001	Topologiefehler: unterschiedlicher Gerätetyp (VendorId / DeviceId)
OTHER_ERROR	16#00000032	16#00000002	Topologiefehler: Nachbargerät fehlt (oder evtl. Ports vertauscht)
SYSTEM_ERROR	16#00000101	internal	Controller im Konfigurationsbaum nicht gefunden
COMMUNICATION_ERROR	16#00000104	internal	Anwender-Timeout
STATE_MACHINE_ERROR	16#00000105	internal	State Machine Error (internal)
SYSTEM_ERROR	16#00000106	internal	Konfiguriertes Geräte nicht gefunden (internal)

Tab. 5-60: Fehlercodes FB [IL_PNIORestoreTopology](#)

5.5 Diagnose (Diagnostics)

5.5.1 IL_PNIOControllerState

Kurzbeschreibung Dieser Funktionsbaustein gibt eine einfache Diagnose des PROFINET IO Controllers zurück. Im Falle einer anstehenden Diagnose können über weitere FBs Details zu den Diagnosen ausgelesen werden (z.B. [IL_PNIOControllerStateDetails](#) auf Seite 118 bzw. [IL_PNIORemoteDeviceState](#) auf Seite 129).

Zielsystem	Bibliothek /Ordner
MLC	RIL_ProfinetIO/ POU's/ Diagnostics
MTX	RIL_ProfinetIO/ POU's/ Diagnostics

Tab. 5-61: Referenztable

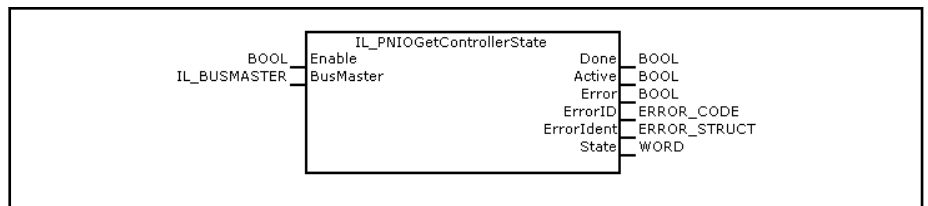


Abb. 5-28: Schnittstelle FB IL_PNIOControllerState

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (permanent, pegelgesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Controllers in der Reihenfolge des Konfigurationsbaumes. Ist nur ein PROFINET IO Controller vorhanden, besitzt dieser immer den Wert IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 189
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall (aus RIL_CommonTypes)
	ErrorIdent	ERROR_STRUCT	Detailldiagnose (ErrorTable PROFINETIO_TABLE)
	State	WORD	Status des PROFINET IO Controllers, siehe unten.

Tab. 5-62: FB IL_PNIOControllerState E/A Schnittstelle

Im Ausgang "State" ist eine Grobklassifizierung von Diagnosen binär kodiert. Die Bits sind in der RIL_FieldbusTypes.library als Globale Konstanten "FieldbusStates" enthalten.

Ausgang "State" (WORD)

BIT-Nr.	Bezeichnung	Beschreibung
0	IL_FBM_BUS_OFF	Feldbus nicht bereit
1	IL_FBM_MASTER_FAILURE	Fehler des Busmasters (bus master failure)
2	IL_FBM_SLAVE_NOT_REACHABLE	Ein oder mehrere FeldbusSlaves sind nicht erreichbar (slave not reachable)
3	IL_FBM_SLAVE_DIAGNOSIS	Ein oder mehrere FeldbusSlaves melden Diagnose (slave diag)

BIT-Nr.	Bezeichnung	Beschreibung
4	IL_FBM_SLAVE_ERROR	Ein oder mehrere FeldbusSlaves melden Fehler (slave error)
5	IL_FBM_TOPOLOGY_INVALID	Topologiedaten für Gerätetausch ungültig (topology data invalid) Dies ist ein Status der Topologiedaten der Gerätetauschfunktion des PNIO-Controllers. Bei einem SPS-Download oder einem SPS-Reset (kalt/ warm) löscht der PNIO Controller die gespeicherten Topologiedaten. Die Topologiedaten werden für die Gerätetauschfunktion benötigt. Sie werden automatisch (neu) erzeugt und gespeichert, wenn alle konfigurierten Geräte fehlerfrei kommunizieren und keine Diagnose ansteht.

Tab. 5-63: Elemente des Ausgangs State

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke von Enable

Tab. 5-64: Min-/Max und Default-Werte der Eingänge IL_PNIOControllerState

Funktionsbeschreibung Der Baustein holt sich die Statusinformationen über einen PROFINET Controller und gibt sie als Bitleiste in ‚State‘ zurück. Die Bearbeitung des Bausteins wird in einem SPS-Zyklus komplett durchgeführt.

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle **PROFINETIO_TABLE** (ERROR_TABLE = 16#0200).

Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Bemerkung
SYSTEM_ERROR	16#0101	internal use	Controller nicht in der Konfigurationstabelle
COMMUNICATION_ERROR	16#0003	PROFINET IO Controller Diagnose - Siehe Allgemeine Fehlercodes , Seite 196, - Siehe Spezielle Fehlercodes, Profinet IO-Controller, Seite 198.	

Tab. 5-65: Fehlercodes FB IL_PNIOControllerState

5.5.2 IL_PNIOControllerStateDetails

Kurzbeschreibung Der Funktionsbaustein ermittelt den detaillierten Status des Profinet-Controllerstacks. Er liefert Informationen über

- den Zustand des Kommunikationsstacks,
- den Zustand der Buskommunikation,
- Fehlerzähler,
- einen Überblick über konfigurierte, aktive und eine Diagnose meldende Slaves.

Zielsystem	Bibliothek /Ordner
MLC	RIL_ProfinetIO/ POU's/ Diagnostics
MTX	RIL_ProfinetIO/ POU's/ Diagnostics

Tab. 5-66: Referenztable

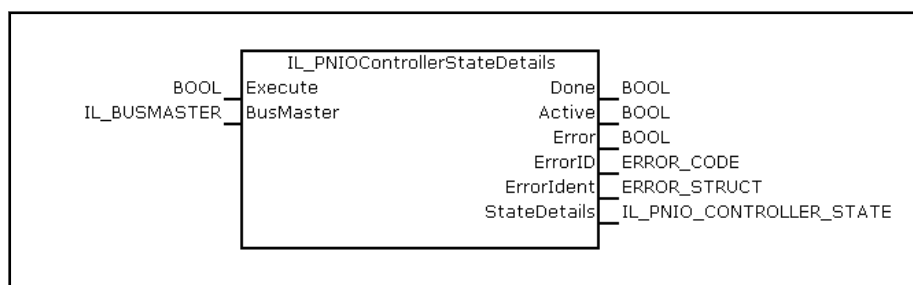


Abb. 5-29: Schnittstelle FB IL_PNIOControllerStateDetails

	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Controllers in der Reihenfolge des Konfigurationsbaumes. Ist nur ein PROFINET IO Controller vorhanden, besitzt dieser immer den Wert IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 189
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall (aus RIL_CommonTypes)
	ErrorIdent	ERROR_STRUCT	Detailldiagnose (ErrorTable PROFINETIO_TABLE)
	StateDetails	IL_PNIO_CONTROLLER_STATE	Detaillierte Statusinformationen des Controller-Stacks Siehe auch IL_PNIO_CONTROLLER_STATE auf Seite 190

Tab. 5-67: FB IL_PNIOControllerStateDetails E/A Schnittstelle

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an Execute

Tab. 5-68: Min-/Max und Default-Werte der Eingänge IL_PNIOControllerStateDetails

Funktionsbeschreibung Der Funktionsbaustein liefert eine aktuelle Statusinformation des PROFINET Controllers.

Insbesondere der Vergleich der Werte Anzahl der konfigurierten Devices (NumOfConfigSlaves), Anzahl der am PROFINET vorhandenen (NumOfActiveSlaves) und die Anzahl der Devices mit einer anstehenden Diagnosemeldung (NumOfDiagSlaves) ist für die Bewertung des Bussystems interessant.

Fehlerbehandlung Der Funktionsbaustein nutzt die **Fehlertabelle PROFINETIO_TABLE** (ERROR_TABLE = 16#0200).
Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Bemerkung
SYSTEM_ERROR	16#0101	internal use	Controller nicht in der Konfigurationstabelle
COMMUNICATION_ERROR	16#0003	<i>PROFINET IO Controller Diagnose</i> - Siehe Allgemeine Fehlercodes , Seite 196, - Siehe Spezielle Fehlercodes, Profinet IO-Controller, Seite 198.	

Tab. 5-69: Fehlercodes FB IL_PNIOControllerStateDetails

5.5.3 IL_PNIODiagnosis

Kurzbeschreibung Der Funktionsbaustein IL_PNIODiagnosis erstellt eine umfassende Diagnose eines Profinet IO Controllers und von dessen angeschlossenen Geräten.
Einfache Diagnosen werden zyklisch aufbereitet, Detaildiagnosen werden innerhalb des FBs azyklisch nacheinander ausgelesen.
Die Diagnosen werden sowohl in Datenstrukturen als auch in "Strings" zur Verfügung gestellt. Ein Auslesen von herstellerspezifischen Diagnosetexten aus der Gerätebeschreibungdatei (GSDML) ist ebenfalls möglich.

Zielsystem	Bibliothek /Ordner
MLC	RIL_ProfinetIO/POUs/Diagnostics
MTX	RIL_ProfinetIO/POUs/Diagnostics

Tab. 5-70: Referenztabelle

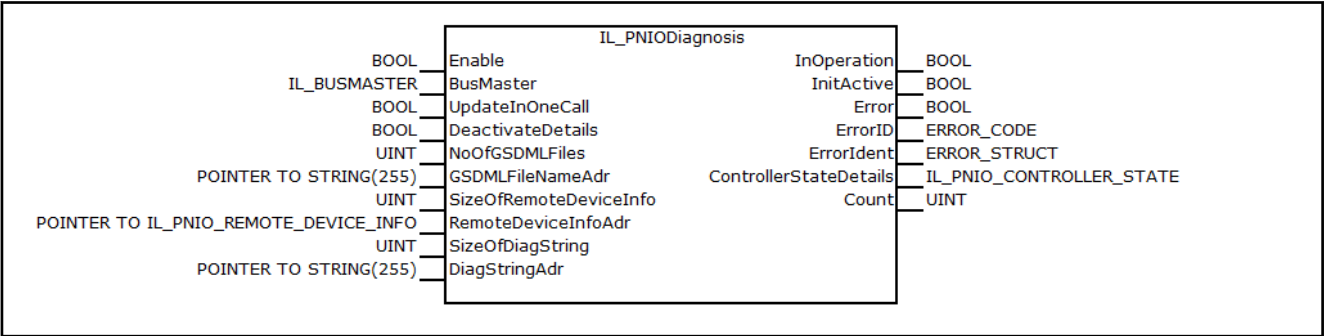


Abb. 5-30: Schnittstelle FB IL_PNIODiagnosis

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (permanent, pegelgesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Controllers in der Reihenfolge des Konfigurationsbaumes. Ist nur ein Profinet IO Controller vorhanden, besitzt dieser immer den Wert IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 189
	UpdateInOneCall	BOOL	Bei TRUE wird der "DeviceState" von allen Geräten in einem Aufruf aktualisiert, ansonsten in aufeinander folgenden Aufrufen (Laufzeit optimiert)
	DeactivateDetails	BOOL	Deaktiviert das azyklische Auslesen der detaillierten ‚DiagInfo‘ in RemoteDeviceInfo
	NoOfGSDMLFiles	UINT	[optional] Anzahl GSDML Dateien in GSDMLFileNameAdr
	GSDMLFileNameAdr	POINTER TO STRING(255)	[optional] Zeiger (ADR()) auf ein Array von STRING(255): Dateiname der GSDML-Dateien, inkl. Pfad (auf Flashkarte der Steuerung), wenn herstellerspezifische Diagnosetexte ausgelesen werden sollen
	SizeOfRemoteDeviceInfo	UINT	Größe des Array RemoteDeviceInfoAdr in Byte (SIZEOF())
	RemoteDeviceInfoAdr	POINTER TO IL_PNIO_REMOTE_DEVICE_INFO	Zeiger (ADR()) auf ein Array von IL_PNIO_REMOTE_DEVICE_INFO: hier wird die komplette Diagnoseinformation zu einem Gerät eingetragen Kap. 5.8.6 "IL_PNIO_REMOTE_DEVICE_INFO" auf Seite 191
	SizeOfDiagString	UINT	[optional] Größe des Array DiagStringAdr in Byte (SIZEOF())
	DiagStringAdr	POINTER TO STRING(255)	[optional] Zeiger (ADR()) auf ein Array von STRING(255): hier wird ein Diagnose-String zu jedem Gerät erstellt
VAR_OUTPUT	InOperation	BOOL	Zyklischer Bearbeitungszustand
	InitActive	BOOL	Initialisierung aktiv: bei GSDML-Dateien mit mehreren Tausend-Zeilen kann dies ein paar Sekunden dauern
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall (aus RIL_CommonTypes)
	ErrorIdent	ERROR_STRUCT	Detailldiagnose (ErrorTable PROFINETIO_TABLE)
	ControllerStateDetails	IL_PNIO_CONTROLLER_STATE	Detaillierte Statusinformationen des Controller-Stacks Siehe auch IL_PNIO_CONTROLLER_STATE auf Seite 190
	Count	UINT	Anzahl der vorhandenen Geräte (betrifft DiagStringAdr u. RemoteDeviceInfoAdr)

Tab. 5-71: FB IL_PNIODiagnosis E/A Schnittstelle

Name	Type	Min	Max	Default	Übernahme
Enable	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an Enable
UpdateInOneCall	BOOL			FALSE	Kontinuierlich
DeactivateDetails	BOOL	-	-	FALSE	Kontinuierlich
NoOfGSDMLFiles	UINT	0	n.def.	0	Steigende Flanke an Enable
GSDMLFileNameAdr	POINTER TO STRING(255)	-	-	0	Kontinuierlich
SizeOfRemoteDeviceInfo	UINT	0	n.def.	0	Steigende Flanke an Enable
RemoteDeviceInfoAdr	POINTER TO IL_PNIO_REMOTE_DEVICE_INFO	-	-	0	Kontinuierlich
SizeOfDiagString	UINT	0	n.def.	0	Kontinuierlich
DiagStringAdr	POINTER TO STRING(255)	-	-	0	Kontinuierlich

Tab. 5-72: Min-/Max und Default-Werte der Eingänge IL_PNIODiagnosis

Funktionsbeschreibung

Der Funktionsbaustein IL_PNIODiagnosis erstellt eine umfassende Diagnose eines Profinet IO Controllers und von dessen angeschlossenen Geräten. Der Controller wird über den Eingang "BusMaster" adressiert, die zugehörigen PN-Devices werden automatisch ausgelesen.

Der FB arbeitet zyklisch, d.h. einfache Diagnosen ("DeviceState") werden zyklisch aufbereitet, Detaildiagnosen werden azyklisch nacheinander ausgelesen ("DiagInfo").

Mit dem Eingang "DeactivateDetails" kann das azyklische Lesen der Detailinformationen auch deaktiviert werden, z.B. wenn andere azyklische Dienste bei einem Gerät ausgeführt werden sollen, die sonst blockiert werden könnten.

Die Eingänge "RemoteDeviceInfoAdr" und "DiagStringAdr" müssen (mindestens) so viele Array-Elemente haben wie PN-Devices konfiguriert sind. In beiden Arrays sind die PN-Devices entsprechend der Reihenfolge im Konfigurationsbaum aufgelistet.

	TestProg.arDiagStrings[INDEX]
0	ugw-micro-1 : is OK
1	### slc-0-gpnt2 : Diag Slot 0 SubSlot 0 Channel 1, ErrorType=216 TextId="T_ErrText0216" Value="Input 1 external test signal failure. Check for stuck-at-high or cabling."
2	### slc-0-gpnt : Diag Slot 0 SubSlot 0 Channel 1, ErrorType=216 TextId="T_ErrText0216" Value="Input 1 external test signal failure. Check for stuck-at-high or cabling."
3	il-pn-bk-fwxxx : is OK
4	### lx5-pn-device : not reachable
5	### il-pn-bk-fw330 : Diag Slot 2 SubSlot 0 Channel 32768, ErrorType=21008 Value="The specified device indicates a peripheral fault." TextId="ID0x5210"
6	
7	

Abb. 5-31: Beispiel: DiagString

Herstellerspezifische Diagnosetexte in der Gerätebeschreibung (GSDML):

Der Eingang "GSDMLFileNameAdr" kann optional verwendet werden, d.h. es kann auch keine oder eine Teilmenge der verwendeten GSDML-Dateien angegeben werden.

Einige Geräte haben auch keine spezifischen Diagnosetexte in der GSDML; hier ist dann das Element <ChannelDiagList> in der GSDML nicht enthalten und eine Verwendung der Datei am FB IL_PNIODiagnosis hat keinen Nutzen.

Am Eingang "GSDMLFileNameAdr" sind die Dateinamen inkl. Pfad (auf Flashkarte der Steuerung) anzugeben.

Die GSDML-Dateien müssen manuell auf der Flashkarte der Steuerung abgelegt werden. Bei folgendem Beispiel ist auf der USER-Partition ("/ata0b/") ein Unterordner GSDML:

```
arGSDMLFileName : ARRAY[0..2] OF STRING(255) := [
  '/ata0b/GSDML/GSDML-V2.31-MBS-MICRO3004-20150206.xml',
  '/ata0b/GSDML/GSDML-V2.1-Bosch Rexroth AG-SLC_0_GPNT-20101215.xml',
  '/ata0b/GSDML/GSDML-V2.3-Bosch Rexroth AG-R_IL PN BK DI8 DO4-20140915.xml'
];
```

Bei der Initialisierung des FB IL_PNIODiagnosis werden die angegebenen GSDML-Dateien den konfigurierten PN-Devices zugeordnet.

Diese Zuordnung erfolgt über die VendorID und die DeviceID.

Zudem wird in der Initialisierung auch bereits ein Offset auf die <ChannelDiagList> gespeichert.

Da GSDML-Dateien mehrere Tausend Zeilen haben können, kann das Auslesen der Informationen ggf. mehrere Sekunden dauern. Dies kann bei Bedarf manuell optimiert werden, in dem für die Diagnosetexte nicht benötigte Beschreibungen in der GSDML gelöscht werden.

Folgende Schlüsselwörter müssen in der GSDML vorhanden sein:

```
...
> could be deleted <
...
  <DeviceIdentity VendorID="0x011f" DeviceID="0x2403">
...
  </DeviceIdentity>
...
> could be deleted <
...
  <ChannelDiagList>
    <ChannelDiagItem ErrorType="0101">
      <Name TextId="T_ErrText0101" />
      <Help TextId="T_HelpChanSlot" />
    </ChannelDiagItem>
  </ChannelDiagList>
...
> could be deleted <
...
  <ExternalTextList>
    <PrimaryLanguage>
      <!--Identification-->
      <Text TextId="T_ErrText0101" Value="Internal Error:..." />
    </PrimaryLanguage>
    <Language xml:lang="de">
      <Text TextId="T_ErrText0101" Value="Interner Fehler:..." />
    </Language>
  </ExternalTextList>
```

Arbeitsweise des FBs:

Im Element <ChannelDiagList> wird der "ErrorType" gesucht, dieser entspricht dem "ChannelErrorType" in der Detaildiagnose PNIO_DIAGINFO.

Dem "ErrorType" ist eine "TextId" zugeordnet, zu der im Element <ExternalTextList> ein Beschreibungs- bzw. Diagnosetext hinterlegt ist.

Der FB verwendet die als erstes gefundene "TextId". Diese ist normalerweise in der PrimaryLanguage ("en"). Falls eine andere in der GSDML vorhandene Sprache, z.B. "de", bevorzugt wird, dann ist in der <ExternalTextList> einfach die Reihenfolge der Sprachen zu ändern, die "TextIds" sind identisch und der FB verwendet immer den zuerst gefundenen Eintrag.

Beispiel in ST Deklaration

```
VAR
fbIL_PNIODiagnosis: IL_PNIODiagnosis;
boEnable: BOOL := FALSE;
arRemoteDeviceInfo: ARRAY [0..127] OF IL_PNIO_REMOTE_DEVICE_INFO;
DiagStrings: ARRAY [0..127] OF STRING(255);
rControllerStateDetails: IL_PNIO_CONTROLLER_STATE;
CountOfDevices: UINT;
// ### Optional: name here the path of the GSDML files,
// which you have stored on the compact flash of the control
// and then assign the number of the files to the variable 'NoOfGSDMLFiles' (see below)
arGSDMLFileName : ARRAY[0..3] OF STRING(255) := [
// example: 'User-Partition' (ata0b) with folder '/GSDML/':
'/ata0b/GSDML/GSDML-V2.31-MBS-MICRO3004-20150206.xml',
'/ata0b/GSDML/GSDML-V2.1-Bosch Rexroth AG-SLC 0 GPNT-20101215.xml',
'/ata0b/GSDML/GSDML-V2.3-Bosch Rexroth AG-R IL PN BK DI8 DO4-20140915.xml' ,
'/ata0b/GSDML/GSDML-V2.3-Bosch_Rexroth_AG-S20_PN_BK_plus-20150910.xml'
];
NoOfGSDMLFiles : UINT := 0; // := SIZEOF(arGSDMLFileName) / SIZEOF(STRING(255));
END_VAR
```

Implementation

```
fbIL_PNIODiagnosis(
Enable:= boEnable,
BusMaster:= ,
UpdateInOneCall:= ,
DeactivateDetails:= ,
NoOfGSDMLFiles:= NoOfGSDMLFiles,
GSDMLFileNameAdr:= ADR(arGSDMLFileName),
SizeOfRemoteDeviceInfo:= SIZEOF(arRemoteDeviceInfo),
RemoteDeviceInfoAdr:= ADR(arRemoteDeviceInfo),
SizeOfDiagString:= SIZEOF(DiagStrings),
DiagStringAdr:= ADR(DiagStrings),
InOperation=> ,
Error=> ,
ErrorID=> ,
ErrorIdent=> ,
InitActive=> ,
ControllerStateDetails=> rControllerStateDetails,
Count=> CountOfDevices);
```

Fehlerbehandlung Der Funktionsbaustein nutzt die **Fehlertabelle PROFINETIO_TABLE** (ERROR_TABLE = 16#0200).

Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Bemerkung
INPUT_INVALID_ERROR	16#0001	16#0010	Eingang 'DiagString' und/oder 'RemoteDeviceInfo' zu kurz
INPUT_INVALID_ERROR	16#0001	16#0011	Eingang 'DiagStringAdr' ist 0
INPUT_INVALID_ERROR	16#0001	16#0013	Eingang 'RemoteDeviceInfoAdr' ist 0
INPUT_INVALID_ERROR	16#0001	16#0014	Eingang 'SizeOfRemoteDeviceInfo' ist 0
INPUT_INVALID_ERROR	16#0001	16#0015	Eingang 'GSDMLFileNameAdr' ist 0
ACCESS_ERROR	16#0005	Interne Verwendung	Interner Fehler

ErrorID	Additional1	Additional2	Bemerkung
INPUT_RANGE_ERROR	16#0009	Interne Verwendung	Maximale Anzahl der erlaubten konfigurierbaren Devices überschritten
RESOURCE_ERROR	16#0010	Index 0..(n-1) of GSDMLFile-NameAdr	GSDML 'File Open Error'
RESOURCE_ERROR	16#0011	Interne Verwendung	GSDML 'File Read Error'
RESOURCE_ERROR	16#0012	Interne Verwendung	GSDML 'File Close Error'
SYSTEM_ERROR	16#0101	Interne Verwendung	Controller nicht in der Konfigurationstabelle

Tab. 5-73: Fehlercodes FB IL_PNIODiagnosis

5.5.4 IL_PNIOGetConfigDeviceNameList

Kurzbeschreibung Der Funktionsbaustein liest die Namen aller konfigurierten PROFINETIO-Devices.

Der Anwender muss ein ARRAY OF STRING(255) zur Aufnahme der Device-Namen zur Verfügung stellen.

Diese Device-Namen können zur Beschaltung des Eingangs "IdentStr" bei anderen PNIO-Funktionsbausteinen benutzt werden.

Zielsystem	Bibliothek /Ordner
MLC	RIL_ProfinetIO/ POU's/ Diagnostics
MTX	RIL_ProfinetIO/ POU's/ Diagnostics

Tab. 5-74: Referenztable

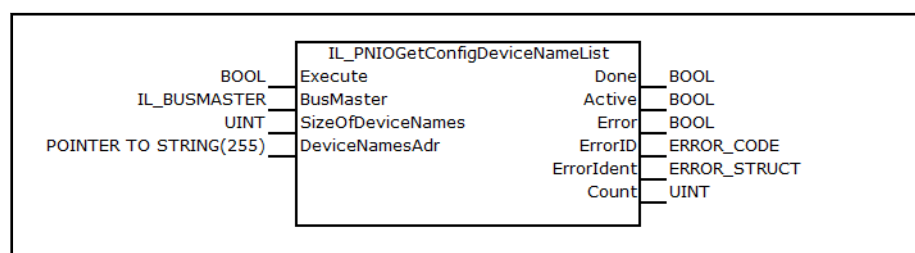


Abb. 5-32: IL_PNIOGetConfigDeviceNameList

	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Feldbus-Masters in der Reihenfolge des Konfigurationbaumes siehe auch IL_BUSMASTER auf Seite 189
	SizeOfDeviceNames	UINT	Größe des Arrays DeviceNamesAdr in Byte (SIZEOF())

	Name	Typ	Kommentar
	DeviceNamesAdr	POINTER TO STRING(255)	Zeiger (ADR()) auf ein Array von STRING(255). Hier werden die konfigurierten Device-Namen eingetragen. Die Größe dieses Arrays muss im Parameter ‚SizeOfDeviceNames‘ übergeben werden
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Detailldiagnose (ErrorTable PROFINETIO_TABLE)
	Count	UINT	Anzahl der gültigen Devicenamen in DeviceNamesAdr

Tab. 5-75: Funktionsbaustein IL_PNIOGetConfigDeviceNameList

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an Execute
SizeOfDeviceNames	UINT	0	n.def.	0	Steigende Flanke an Execute
DeviceNamesAdr	POINTER TO STRING(255)	-	-	0	Steigende Flanke an Execute

Tab. 5-76: Min-/Max und Default-Werte der Eingänge L_PNIOGetConfigDeviceNameList

Funktionsbeschreibung

Der Funktionsbaustein durchsucht die geladene IO-Konfiguration nach den Devices, die unterhalb des ausgewählten PROFINET-Controllers eingehängt sind.

Deaktivierte Devices werden nicht in die Liste aufgenommen. Diese Device-Namen können als Eingangsparameter ‚IdentStr‘ bei anderen Funktionsbausteinen verwendet werden.

Fehlerbehandlung

Der Funktionsbaustein nutzt die **Fehlertabelle PROFINETIO_TABLE** (ERROR_TABLE = 16#0200).

Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	
INPUT_RANGE_ERROR	16#0001	0	DeviceNamesAdr: Übergebene Liste ist zu kurz
INPUT_RANGE_ERROR	16#0001	1	DeviceNamesAdr nicht gesetzt
INPUT_RANGE_ERROR	16#0001	2	SizeOfDeviceNames nicht gesetzt
SYSTEM_ERROR	16#0101	internal use	Controller nicht in der Konfigurationstabelle

Tab. 5-77: Fehlercodes FB IL_PNIOGetConfigDeviceNameList

5.5.5 IL_PNIOGetControllerList

Kurzbeschreibung Der Funktionsbaustein IL_PNIOGetControllerList liefert die Anzahl der konfigurierten PROFINET IO-Controller.

Zielsystem	Bibliothek /Ordner
MLC	RIL_ProfinetIO/ POU's/ Diagnostics
MTX	RIL_ProfinetIO/ POU's/ Diagnostics

Tab. 5-78: Referenztable

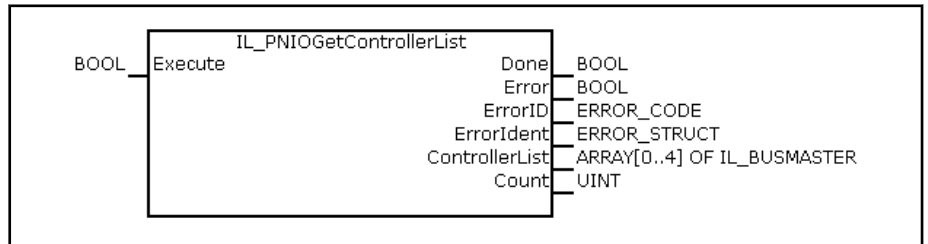


Abb. 5-33: IL_PNIOGetControllerList

	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	ControllerList	ARRAY[0..4] OF IL_BUSMASTER	Array vom Typ IL_BUSMASTER, das die konfigurierten BusMaster (PNIO Controller) ausgibt; siehe auch IL_BUSMASTER auf Seite 189
	Count	UINT	Anzahl der gültigen Elemente im Ausgang "ControllerList"

Tab. 5-79: Funktionsbaustein IL_PNIOGetControllerList

Name	Type	Min	Max	Default	Übernahme
Execute	BOOL	-	-	FALSE	Continuous

Tab. 5-80: Min-/Max und Default-Werte der Eingänge IL_PNIOGetControllerList

Funktionsbeschreibung

Der Funktionsbaustein IL_PNIOGetControllerList liefert die Anzahl der konfigurierten PROFINET IO Controller.

Der Ausgang "ControllerList" (vom Typ IL_BUSMASTER) kann an den Eingängen der sonstigen PROFINET-Diagnose FBs der RIL_ProfinetIO Bibliothek verwendet werden.

Fehlerbehandlung

Der Funktionsbaustein nutzt die **Fehlertabelle** PROFINETIO_TABLE (ERROR_TABLE = 16#0200).

Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	
SYSTEM_ERROR	16#0101	internal use	internal error (null pointer)

Tab. 5-81: Fehlercodes FB IL_PNIOGetControllerList

5.5.6 IL_PNIOGetDiagDeviceNameList

Kurzbeschreibung

Der Funktionsbaustein liest die Namen aller PROFINETIO-Devices, die eine Diagnose melden.

Der Anwender muss ein ARRAY OF STRING(255) zur Aufnahme der Device-Namen zur Verfügung stellen.

Zielsystem	Bibliothek /Ordner
MLC	RIL_ProfinetIO/ POU's/ Diagnostics
MTX	RIL_ProfinetIO/ POU's/ Diagnostics

Tab. 5-82: Referenztabelle

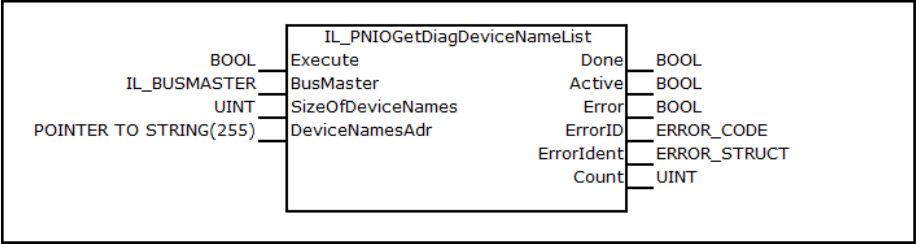


Abb. 5-34: IL_PNIOGetDiagDeviceNameList

	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Feldbus-Masters in der Reihenfolge des Konfigurationbaumes siehe auch IL_BUSMASTER auf Seite 189
	SizeOfDeviceNames	UINT	Größe des Arrays DeviceNamesAdr in Byte (SIZEOF())
	DeviceNamesAdr	POINTER TO STRING(255)	Zeiger (ADR()) auf ein Array von STRING(255). Hier werden die DevicesNames von denjenigen Devices eingetragen, die eine Diagnose melden. Die Größe dieses Arrays muss im Parameter ‚SizeOfDeviceNames‘ übergeben werden.
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers

	Name	Typ	Kommentar
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	Count	UINT	Anzahl der Devices mit aktiver Diagnose in DeviceName-sAdr

Tab. 5-83: Funktionsbaustein IL_PNIOGetDiagDeviceNameList

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an Execute
SizeOfDeviceNames	UINT	0	n.def.	0	Steigende Flanke an Execute
DeviceNamesAdr	POINTER TO STRING(255)	-	-	0	Steigende Flanke an Execute

Tab. 5-84: Min-/Max und Default-Werte der Eingänge IL_PNIOGetDiagDeviceNameList

Funktionsbeschreibung Aus der Konfigurationstabelle werden alle PROFINET-Devices ermittelt, die aktuell eine Diagnose melden.
Es wird der Device-Name übergeben.



Geräte mit anderen Fehlern tauchen in dieser Liste nicht auf.

Fehlerbehandlung Der Funktionsbaustein nutzt die **Fehlertabelle PROFINETIO_TABLE** (ERROR_TABLE = 16#0200).
Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	
INPUT_RANGE_ERROR	16#0001	0	DeviceNamesAdr: Übergebene Liste ist zu kurz
INPUT_RANGE_ERROR	16#0001	1	DeviceNamesAdr nicht gesetzt
INPUT_RANGE_ERROR	16#0001	2	SizeOfDeviceNames nicht gesetzt
SYSTEM_ERROR	16#0101	internal use	Controller nicht in der Konfigurationstabelle

Tab. 5-85: Fehlercodes FB IL_PNIOGetDiagDeviceNameList

5.5.7 IL_PNIORemoteDeviceState

Kurzbeschreibung Dieser Funktionsbaustein ermittelt den Grundstatus eines am Controller angeschlossenen Devices.

Zielsystem	Bibliothek /Ordner
MLC	RIL_ProfinetIO/ POU's/ Diagnostics
MTX	RIL_ProfinetIO/ POU's/ Diagnostics

Tab. 5-86: Referenztable

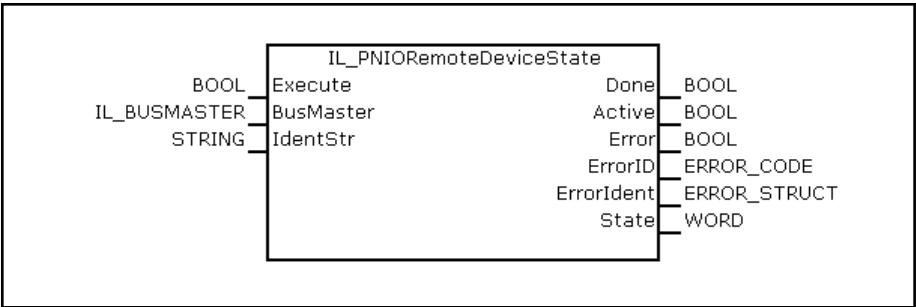


Abb. 5-35: IL_PNIORemoteDeviceState

	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Funktionsaktivierung; ein aktivierter Funktionsbaustein kann nicht unterbrochen werden.
	BusMaster	IL_BUSMASTER	Instanz des Controllers in der Reihenfolge des Konfigurationsbaumes. Erlaubte Werte 0..4; Default: 0 siehe auch IL_BUSMASTER auf Seite 189
	IdentStr	STRING	Stationsname des IO-Devices, auf das zugegriffen werden soll.
VAR_OUTPUT	Done	BOOL	Fertigmeldung (erfolgreich)
	Active	BOOL	Aktivitätsanzeige
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	State	WORD	Statusinformation des Slaves am Feldbus, siehe unten

Tab. 5-87: Funktionsbaustein IL_PNIORemoteDeviceState

Ausgang "State" (WORD)

BIT-Nr.	Bezeichnung	Beschreibung
0	IL_RBS_SLAVE_NOT_REACHABLE	Device nicht existent (bus slave state – slave not existent)
1	IL_RBS_SLAVE_NOT_READY	Device nicht bereit (bus slave state – slave not ready)
2	IL_RBS_SLAVE_CONFIG_ERROR	Device Konfigurationsfehler (bus slave state – slave configuration error)
3	IL_RBS_SLAVE_DIAGNOSIS	Device Diagnose verfügbar (bus slave state – slave diagnoses available)
4	IL_RBS_SLAVE_ERROR	Allgemeiner Device Fehler

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an Execute
IdentStr	STRING			''	Steigende Flanke an Execute

Tab. 5-88: Min-/Max und Default-Werte der Eingänge IL_PNIOControllerState

Funktionsbeschreibung Die bitcodierte Zustandsleiste des PROFINET-Devices wird ausgewertet. Für weiterführende Informationen kann der Dienst IL_PNIORemoteDeviceStateDetails verwendet werden. Zur Auswahl des Devices wird der PROFINET-Devicename in der Variablen ‚IdentStr‘ angegeben. Es können nur projektierte und aktive Devices betrachtet werden.

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle PROFINETIO_TABLE (ErrorIdent.Table = 16#0200). Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	
INPUT_RANGE_ERROR	16#0001	1	IdentStr ist nicht angegeben
SYSTEM_ERROR	16#0101	internal use	Controller nicht in der Konfigurationstabelle
INPUT_RANGE_ERROR	16#0106	0	Gerät IdentStr nicht konfiguriert

Tab. 5-89: Fehlercodes FB IL_PNIORemoteDeviceState

5.5.8 IL_PNIORemoteDeviceStateDetails

Kurzbeschreibung Der Funktionsbaustein ermittelt den detaillierten PROFINET IO-Status eines am Controller konfigurierten und angeschlossenen Devices.

Die PROFINET-Datenstrukturen werden in einer allgemeinen Diagnoseinformation zusammengefasst.

Zielsystem	Bibliothek /Ordner
MLC	RIL_ProfinetIO/ POU's/ Diagnostics
MTX	RIL_ProfinetIO/ POU's/ Diagnostics

Tab. 5-90: Referenztablelle

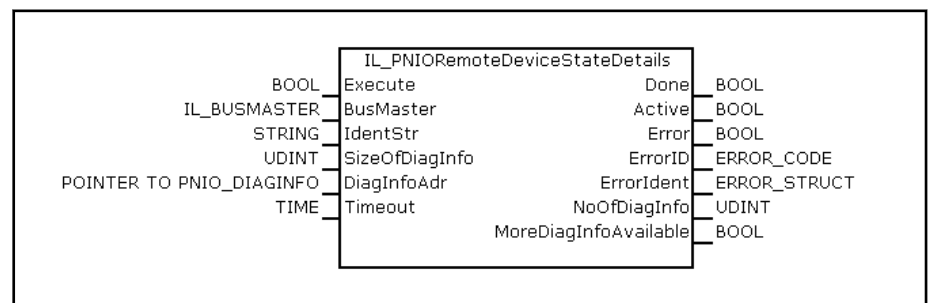


Abb. 5-36: Schnittstelle FB IL_PNIORemoteDeviceStateDetails

	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Controllers in der Reihenfolge des Konfigurationsbaumes. Ist nur ein PROFINET IO Controller vorhanden, besitzt dieser immer den Wert IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 189
	IdentStr	STRING	Identifikationsstring des Feldbusteilnehmers
	SizeOfDiagInfo	UDINT	Maximale Größe des Datenbereichs von ‚DiagInfoAdr‘. Muss ein ganzzahliges Vielfaches von SIZEOF(PNIO_DIAGINFO) sein.
	DiagInfoAdr	POINTER TO PNIO_DIAGINFO	Zeiger auf die Datenablage für die ausgewerteten Diagnosedaten des Devices. Siehe auch PNIO_DIAGINFO auf Seite 194
	Timeout	TIME	Timeout - Überwachung des FBs in ms. T#0ms = Timeout - Überwachung ist nicht aktiv
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall (aus RIL_CommonTypes)
	ErrorIdent	ERROR_STRUCT	Detaildiagnose (ErrorTable PROFINETIO_TABLE)
	NoOfDiagInfo	UDINT	Anzahl der in DiagInfoAdr eingetragenen Auswertungen
	MoreDiagInfoAvailable	BOOL	Es sind noch weitere Diagnose-Informationen vorhanden, die nicht in der bereitgestellten Datenstruktur Platz gefunden haben.

Tab. 5-91: FB IL_PNIORemoteDeviceStateDetails E/A Schnittstelle

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an Execute
IdentStr	STRING			''	Steigende Flanke an Execute
SizeOfDiagInfo	UDINT	0	n.def.	0	Steigende Flanke an Execute
DiagInfoAdr	POINTER TO PNIO_DIAGINFO			0	Steigende Flanke an Execute
Timeout	TIME	T#0ms	T#3600s	T#2s	Steigende Flanke an Execute

Tab. 5-92: Min-/Max und Default-Werte der Eingänge IL_PNIOControllerStateDetails

Funktionsbeschreibung Der Zustand des Devices wird durch azyklische Lesezugriffe ermittelt. Je nach Zustand können mehrere Zugriffe notwendig sein. Dies kann Auswirkungen auf den Durchsatz von anderen azyklischen Zugriffen auf dieses Device haben. Es können mehrere Diagnoseanzeigen gleichzeitig anstehen. Insbesondere bei Geräten, die eine kanalgranulare Diagnose (1 Diagnose pro IO-Punkt) liefern, können mehrere Diagnosen gleichzeitig anstehen. Der Ausgangsparameter ‚MoreDiagInfoAvailable‘ zeigt an, dass nicht alle anstehenden Diagnosemeldungen in den zur Verfügung gestellten Einträgen Platz gefunden haben.

Fehlerbehandlung Der Funktionsbaustein nutzt die **Fehlertabelle PROFINETIO_TABLE** (ERROR_TABLE = 16#0200). Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	
INPUT_RANGE_ERROR	16#0001	0	Parameterfehler
INPUT_RANGE_ERROR	16#0001	1	IdentStr ist nicht angegeben
INPUT_RANGE_ERROR	16#0001	2	DiagInfoAdr nicht gesetzt
INPUT_RANGE_ERROR	16#0001	3	SzoOfDiagInfo nicht gesetzt
INPUT_RANGE_ERROR	16#0001	4	SzoOfDiagInfo kein vielfaches der Größe von PNIO_DIAGINFO
SYSTEM_ERROR	16#0101	internal use	Controller nicht in der Konfigurationstabelle
COMMUNICATION_ERROR	16#0104	0	Zeitüberschreitung
COMMUNICATION_ERROR	16#0106	0	Device 'IdentStr' nicht konfiguriert
COMMUNICATION_ERROR	16#0003	PROFINET IO Controller Diagnose - Siehe Allgemeine Fehlercodes , Seite 196, - Siehe Spezielle Fehlercodes, Profinet IO-Controller, Seite 198.	

Tab. 5-93: Fehlercodes FB IL_PNIORemoteDeviceStateDetails

5.6 I&M (Identification and Maintenance) (IM Data)

5.6.1 IL_PNIO_IM0

Kurzbeschreibung Der Strukturtyp **IL_PNIO_IM0** dient als Datenklassifizierung von I&M0 (nur lesen).

Strukturelement	Datentyp	Beschreibung
ManufacturerID	WORD	Herstellerkennung
OrderID	STRING(21)	Teilenummer des Device
SerialNumber	STRING(17)	Seriennummer des Device
HardwareRevision	WORD	Hardware-Revision
SoftwareRevision	ARRAY[0..3] OF BYTE	Software-Revision
RevisionCounter	WORD	Revisionszähler

Strukturelement	Datentyp	Beschreibung
ProfileID	WORD	Profilnummer
ProfileSpecificType	WORD	Device-Typ
IM_Version	ARRAY[0..1] OF BYTE	Version des I&M-Profiles
IM_Supported	WORD	Unterstützung der I&M-Datensätze

Tab. 5-94: Struktur von IL_PNIO_IM0

5.6.2 IL_PNIO_IM1

Kurzbeschreibung Der Strukturtyp **IL_PNIO_IM1** dient als Datenklassifizierung von I&M1 (lesen/schreiben).

	Datentyp	Beschreibung	Min. Wert	Max. Wert	Defaultwert	Anwendung
TagFunction	STRING(33)	Funktion des Device	---	---	---	Steigende Flanke an „Execute“
TagLocation	STRING(23)	Ort des Device	---	---	---	

Tab. 5-95: Struktur von IL_PNIO_IM1

5.6.3 IL_PNIO_IM2

Kurzbeschreibung Der Strukturtyp **IL_PNIO_IM2** dient als Datenklassifizierung von I&M2 (lesen/schreiben).

	Datentyp	Beschreibung	Min. Wert	Max. Wert	Defaultwert	Anwendung
InstallationDate	STRING(17)	Installationsdaten des Device	---	---	---	Steigende Flanke an „Execute“
Reserved	ARRAY[0..37] OF BYTE	Reserviert	---	---	---	

Tab. 5-96: Struktur von IL_PNIO_IM2

5.6.4 IL_PNIO_IM3

Kurzbeschreibung Der Strukturtyp **IL_PNIO_IM3** dient als Datenklassifizierung von I&M3 (lesen/schreiben).

	Datentyp	Beschreibung	Min. Wert	Max. Wert	Defaultwert	Anwendung
Descriptor	STRING(55)	Beschreibung des Device	---	---	---	Steigende Flanke an „Execute“

Tab. 5-97: Struktur von IL_PNIO_IM3

5.6.5 IL_PNIO_IM4

Kurzbeschreibung Der Strukturtyp **IL_PNIO_IM4** dient als Datenklassifizierung von I&M4 (lesen/schreiben).

	Datentyp	Beschreibung	Min. Wert	Max. Wert	Defaultwert	Anwendung
Signature	STRING(55)		---	---	---	Steigende Flanke an „Execute“

Tab. 5-98: Struktur von IL_PNIO_IM4

5.6.6 IL_PNIO_IM_RWMODE

Kurzbeschreibung Der Aufzählungstyp **IL_PNIO_IM_RWMODE** dient der Auswahl des Lese-/Schreibmodus des Datensatzes für I&M0...I&M4.

Element	Wert	Beschreibung
IL_PNIO_READ_IM0	16#0000	I&M0-Datensatz lesen
IL_PNIO_READ_IM1	16#0001	I&M1-Datensatz lesen
IL_PNIO_READ_IM2	16#0002	I&M2-Datensatz lesen
IL_PNIO_READ_IM3	16#0003	I&M3-Datensatz lesen
IL_PNIO_READ_IM4	16#0004	I&M4-Datensatz lesen
IL_PNIO_READ_IMALL	16#0005	I&M0...I&M4-Datensatz lesen
IL_PNIO_WRITE_IM1	16#0006	I&M1-Datensatz schreiben
IL_PNIO_WRITE_IM2	16#0007	I&M2-Datensatz schreiben
IL_PNIO_WRITE_IM3	16#0008	I&M3-Datensatz schreiben
IL_PNIO_WRITE_IM4	16#0009	I&M4-Datensatz schreiben
IL_PNIO_WRITE_IMALL	16#000A	I&M1...I&M4-Datensatz schreiben

Tab. 5-99: Aufzählungstyp **IL_PNIO_IM_RWMODE**

5.6.7 IL_PNIOReadWriteIMData

Kurzbeschreibung Der Funktionsbaustein **IL_PNIOReadWriteIMData** kann I&M-Datensätze (Identifikation & Wartung) des ProfinetIO-Device mit verschiedenen Lese-/Schreibmodi lesen bzw. schreiben.

Zielsystem	Bibliothek / Ordner
MLC	RIL_ProfinetIO/ POU's/ IM Data
MTX	RIL_ProfinetIO/ POU's/ IM Data

Tab. 5-100: Referenztablelle

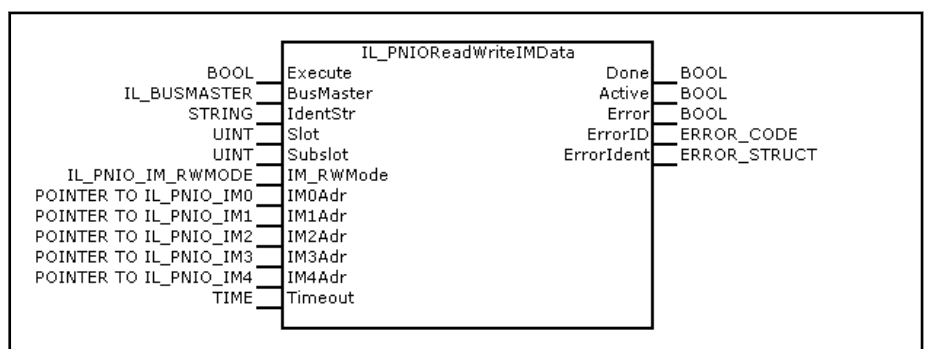


Abb. 5-37: Schnittstellenbeschreibung des Funktionsbausteins **IL_PNIOReadWriteIMData**

	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Controllers in der Reihenfolge des Konfigurationsbaumes. Ist nur ein PROFINET IO Controller vorhanden, besitzt dieser immer den Wert IL_BUSMASTER_0. Siehe auch IL_BUSMASTER auf Seite 189
	IdentStr	STRING	Identifikationsstring des PROFINET-Device.
	Slot	UINT	Adressierung des Modulslots. Der Slot adressiert ein logisches oder physikalisches Modul (Beispielsweise ein E/A Modul)
	SubSlot	UINT	Adressierung des Submoduls. Ein Modul kann sich aus mehreren logischen oder physikalischen Submodulen zusammensetzen. Es ist bei jedem Modul (Slot) mindestens das Submodul mit der Nummer 1 vorhanden.
	IM_RWMode	IL_PNIO_IM_RWMODE	Aufzählung für Lese-/Schreibmodus von IM-Daten Siehe IL_PNIO_IM_RWMODE , auf Seite 135
	IM0Adr	POINTER IL_PNIO_IM0	TO Adresse der IM0-Datenstruktur zum Lesen/Schreiben Siehe IL_PNIO_IM0 , auf Seite 133
	IM1Adr	POINTER IL_PNIO_IM1	TO Adresse der IM1-Datenstruktur zum Lesen/Schreiben Siehe IL_PNIO_IM1 , auf Seite 134
	IM2Adr	POINTER IL_PNIO_IM2	TO Adresse der IM2-Datenstruktur zum Lesen/Schreiben Siehe IL_PNIO_IM2 , auf Seite 134
	IM3Adr	POINTER IL_PNIO_IM3	TO Adresse der IM3-Datenstruktur zum Lesen/Schreiben Siehe IL_PNIO_IM3 , auf Seite 134
	IM4Adr	POINTER IL_PNIO_IM4	TO Adresse der IM4-Datenstruktur zum Lesen/Schreiben Siehe IL_PNIO_IM4 , auf Seite 134
	Timeout	TIME	Timeout-Überwachung des Funktionsbausteins
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall (aus RIL_CommonTypes)
	ErrorIdent	ERROR_STRUCT	Detailldiagnose (ErrorTable PROFINETIO_TABLE)

Tab. 5-101: FB IL_PNIOReadWriteIMData E/A-Schnittstelle

Min./max. und Default-Werte: In der folgenden Tabelle sind die Min-/Max- und Default-Werte der Funktionsbausteineingänge aufgelistet.

Name	Typ	Min. Wert	Max. Wert	Defaultwert	Anwendung
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an „Execute“
IdentStr	STRING	---	---	---	Steigende Flanke an „Execute“
Slot	UINT	0	16#7FFF	0	Steigende Flanke an „Execute“
SubSlot	UINT	1	16#8FFF	1	Steigende Flanke an „Execute“
IM_RWMode	IL_PNIO_IM_RWMODE	IL_PNIO_READ_IM0	IL_PNIO_WRITE_IMALL	IL_PNIO_READ_IM0	Steigende Flanke an „Execute“
IM0Adr	POINTER IL_PNIO_IM0 TO	nicht definiert	nicht definiert	0	entfällt
IM1Adr	POINTER IL_PNIO_IM1 TO	nicht definiert	nicht definiert	0	Steigende Flanke an „Execute“
IM2Adr	POINTER IL_PNIO_IM2 TO	nicht definiert	nicht definiert	0	Steigende Flanke an „Execute“
IM3Adr	POINTER IL_PNIO_IM3 TO	nicht definiert	nicht definiert	0	Steigende Flanke an „Execute“
IM4Adr	POINTER IL_PNIO_IM4 TO	nicht definiert	nicht definiert	0	Steigende Flanke an „Execute“
Timeout	TIME	T#0ms	T#3600s	T#2s	Steigende Flanke an „Execute“

Tab. 5-102: Min./Max. und Default-Werte der Eingänge L_PNIOReadWriteIMData

Funktionsbeschreibung

Der Funktionsbaustein **IL_PNIOReadWriteIMData** kann I&M-Datensätze (Identifikation & Wartung) des ProfinetIO-Device mit verschiedenen Lese-/Schreibmodi lesen bzw. schreiben. **IL_PNIOReadWriteIMData** greift auf I&M-Funktionen des Speicherindex 0xAFF0 – 0xAFF4 mit den folgenden Speicherindexattributen zu;

- I&M0 (erforderlich) nur lesen, Index AFF0
- I&M1 (optional) lesen/schreiben, Index AFF1
- I&M2 (optional) lesen/schreiben, Index AFF2
- I&M3 (optional) lesen/schreiben, Index AFF3
- I&M4 (optional) lesen/schreiben, Index AFF4

Das Device wird über den PROFINET Device-Namen in "IdentStr" adressiert. Es muss Teil der Konfiguration des Controllers sein. Der Datensatz auf dem PROFINET Device wird über die Kombination von Slot und Subslot adressiert. Slot und Subslot adressieren die verschiedenen logischen oder physikalischen Module eines Devices.

Der Betriebsmodus des Funktionsbausteins wird über "IM_RWMode" gewählt. Dies ermöglicht dem Funktionsbaustein die Ausführung von Lese-/Schreib-Operationen für den Speicherindex mit über "IM0Adr", "IM1Adr", "IM2Adr", "IM3Adr" und "IM4Adr" definierten Daten.

- Der Modus **IL_PNIO_READ_IMALL** liest erst IM0 und anschließend nacheinander alle unterstützten IM1...4 gemäß dem von IM0 identifizierten „IM_Supported“
- Der Modus **IL_PNIO_WRITE_IMALL** versucht nacheinander alle Datensätze IM1...4 zu schreiben
- In den Modi **IL_PNIO_READ_IM1...4** / **IL_PNIO_WRITE_IM1...4** versucht der Funktionsbaustein den Speicherindex zu schreiben oder zu lesen ohne zu prüfen, ob der definierte Datensatz unterstützt wird

"IM_Supported" ist eine Liste an Bits, die anzeigt, welche weiteren I&M-Datensätze verfügbar sind. Bit-0 : --; Bit-1: I&M1; Bit-2: I&M2; Bit-3: I&M3; Bit-4: I&M4;.... Bit-15: I&M15.

"IM_Version" des Device muss 1.x betragen. Andernfalls wird vom Funktionsbaustein der Fehler "I&M Version mismatch" angezeigt.

Der Funktionsbaustein zeigt den Fehler „Response header is not matching“ an, wenn die Antwort der angeforderten I&M nicht den Daten in der nachfolgenden Tabelle entspricht.

BlockHeader	I&M0	I&M1	I&M2	I&M3	I&M4
Typ	16#20	16#21	16#22	16#23	16#24

Die steigende Flanke am Eingang "Execute" startet den Vorgang. Ist der Vorgang erfolgreich beendet, wird der Ausgang "Done" auf TRUE gesetzt. Tritt ein Fehler auf, wird das "Error"-Bit gesetzt und detaillierte Informationen über den Fehler werden in "ErrorID" und "ErrorIdent" bereitgestellt.

Beispielprogramm

Dies ist das Beispielprojekt zum Lesen von I&M-Details von 0..4 durch einmaliges Ausführen vom Slave-Device. Das Profinet Slave-Device befindet sich an Slot 0 mit dem Stationsnamen "PNIO-device-1". Die steigende Flanke am Eingang "bExecute" führt den Vorgang aus und liest zuerst die Daten von I&M0 und anschließend die nachfolgenden unterstützen I&Ms.

Deklaration

```

PROGRAM ProgIL_PNIOReadWriteIMData
VAR
  fbIL_PNIOReadWriteIMData: IL_PNIOReadWriteIMData; // FB Instance
  bExecute:          BOOL; //Function block is running when TRUE
  strIdentString:    STRING := 'PNIO-device-1'; //Station name of PN BK
  uiSlot:            UINT := 16#00; //Addressing of the module slot.
                      //The slot addresses a logical or a physical module
  uiSubSlot:         UINT := 16#01; //Addressing of the submodule.
  enIW_RWMode:       IL_PNIO_IM_RWMode := IL_PNIO_IM_RWMode.IL_PNIO_READ_IMALL;
  stIM0Adr:          IL_PNIO_IM0; //Address of read/write IM0 data structure.
  stIM1Adr:          IL_PNIO_IM1; //Address of read/write IM1 data structure.
  stIM2Adr:          IL_PNIO_IM2; //Address of read/write IM2 data structure.
  stIM3Adr:          IL_PNIO_IM3; //Address of read/write IM3 data structure.
  stIM4Adr:          IL_PNIO_IM4; //Address of read/write IM4 data structure.
  tTimeout:          TIME := T#2S; //Timeout monitoring of the function block
  bDone:             BOOL; //Function Block is completed its task
  bActive:           BOOL; //Function Block is working on its actual task
  bError:            BOOL; //Indicates an error
  enErrorID:         RIL_COMMONTYPES.ERROR_CODE; //Class of error
  enErrorIdent:      RIL_COMMONTYPES.ERROR_STRUCT; //Detailed information about error
END_VAR

```

Implementierung

```

fbIL_PNIOReadWriteIMData(
  Execute:= bExecute,
  Done=> bDone,
  Active=> bActive,
  Error=> bError,
  ErrorID=> enErrorID,
  ErrorIdent=> enErrorIdent,
  BusMaster:= IL_BUSMASTER.IL_BUSMASTER_0,
  IdentStr:= strIdentString,

```

```

Slot:= uiSlot,
SubSlot:= uiSubSlot,
IM_RWMode:= enIW_RWMode,
IM0Adr:= ADR(stIM0Adr),
IM1Adr:= ADR(stIM1Adr),
IM2Adr:= ADR(stIM2Adr),
IM3Adr:= ADR(stIM3Adr),
IM4Adr:= ADR(stIM4Adr),
Timeout:= tTimeout);

```

fbIL_PNIOReadWriteIMData	IL_PNIOReadWriteIMData	
bExecute	BOOL	TRUE
strIdentString	STRING	'L25-PN'
uiSlot	UINT	0
uiSubSlot	UINT	1
enIW_RWMode	IL_PNIO_IM_RWMODE	IL_PNIO_READ_IMALL
stIM0Adr	IL_PNIO_IM0	
ManufacturerID	WORD	287
OrderID	STRING(21)	'CFL01.1-T*'
SerialNumber	STRING(17)	'6893150'
HardwareRevision	WORD	16
SoftwareRevision	ARRAY [0..3] OF BYTE	
SoftwareRevision[0]	BYTE	86
SoftwareRevision[1]	BYTE	14
SoftwareRevision[2]	BYTE	4
SoftwareRevision[3]	BYTE	0
RevisionCounter	WORD	0
ProfileID	WORD	0
ProfileSpecificType	WORD	0
IM_Version	ARRAY [0..1] OF BYTE	
IM_Version[0]	BYTE	1
IM_Version[1]	BYTE	1
IM_Supported	WORD	30
stIM1Adr	IL_PNIO_IM1	
TagFunction	STRING(33)	' '
TagLocation	STRING(23)	' '
stIM2Adr	IL_PNIO_IM2	
InstallationDate	STRING(17)	' '
Reserved	ARRAY [0..37] OF BYTE	
stIM3Adr	IL_PNIO_IM3	
Descriptor	STRING(55)	' '
stIM4Adr	IL_PNIO_IM4	
Signature	STRING(55)	' '
tTimeout	TIME	T#2s
bDone	BOOL	TRUE
bActive	BOOL	FALSE
bError	BOOL	FALSE
enErrorID	ERROR_CODE	NONE_ERROR
enErrorIdent	RIL_COMMONTYPES.ERROR_STRUCT	
Table	ERROR_TABLE	NO_TABLE_USED
Additional1	DWORD	16#00000000
Additional2	DWORD	16#00000000

Abb. 5-38: Online-Ausführen des FB

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle **PROFINETIO_TABLE** (ERROR_TABLE = 16#0200).

Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Bemerkungen
INPUT_INVALID_ERROR	16#0001	16#0000	IMxAdr ist nicht gesetzt
INPUT_INVALID_ERROR	16#0001	16#0002	IdentStr ist ungültig
INPUT_INVALID_ERROR	16#0001	16#0004	IM_RWMode ist ungültig
COMMUNICATION_ERROR	16#0003	PROFINET IO Controller-Diagnose - Allgemeine Fehlercodes , auf Seite 196 - Spezielle Fehlercodes, Profinet IO-Controller, auf Seite 198	
COMMUNICATION_ERROR	16#0004	PROFINET IO Status Spezielle Fehlercodes, Profinet IO-Status , auf Seite 199	
COMMUNICATION_ERROR	16#0005	16#0000	Keine Übereinstimmung für Antwort-Header
DEVICE_ERROR	16#0007	16#0000	I&M Version-Fehlzuordnung, Device I&M Major RevisionID muss 1 sein
SYSTEM_ERROR	16#0101	Interne Verwendung	Controller nicht in der Konfigurationstabelle gefunden
COMMUNICATION_ERROR	16#0104	16#0000	Zeitüberschreitung
STATE_MACHINE_ERROR	16#0105	16#0000 16#0001 16#0002	StateMachine-Fehler

Tab. 5-103: Fehlercodes IL_PNIOWriteIMData

5.7 Kommunikationsschnittstelle (IO-Link)

5.7.1 IL_IOL_DEVICE_STATUS

Kurzbeschreibung Die Aufzählung IL_IOL_DEVICE_STATUS wird verwendet, um Informationen zum Device-Zustand bereitzustellen (Diagnose).

Element	Wert	Beschreibung
STATUS_INVALID	16#FFFF	Status nicht verfügbar.
WORKING_PROPERLY	16#0000	Device funktioniert ordnungsgemäß.
MAINTENANCE_REQUIRED	16#0001	Obwohl die Prozessdaten gültig sind, zeigt die interne Diagnose, dass das Device demnächst nicht mehr richtig funktionieren wird.
OUT_OF_SPECIFICATION	16#0002	Obwohl die Prozessdaten gültig sind, zeigt die interne Diagnose, dass das Device außerhalb des festgelegten Messbereichs oder der festgelegten Umgebungsbedingungen arbeitet.
FUNCTIONAL_CHECK	16#0003	Die Prozessdaten sind aufgrund absichtlicher Manipulationen am Device zeitweise ungültig.
FAILURE	16#0004	Die Prozessdaten sind aufgrund einer Fehlfunktion im Device oder seiner Peripheriegeräte ungültig.
RESERVED	16#00FF	reserviert.

Tab. 5-104: Aufzählungstyp IL_IOL_DEVICE_STATUS

5.7.2 IL_IOL_DEVICE_STATUS_DETAILS

Kurzbeschreibung Diese Struktur wird verwendet, um Details zum Device-Status des IO-Link-Device anzuzeigen.

Strukturelement	Wert	Beschreibung
EventQualifier	IL_IOL_EVENT_QUALIFIER	Detaillierte Informationen zum Ereignis. IL_IOL_EVENT_QUALIFIER , auf Seite 141
EventCode	WORD	Der Eintrag EventCode enthält den Bezeichner eines tatsächlichen Ereignisses.

Tab. 5-105: Struktur von IL_IOL_DEVICE_STATUS_DETAILS

5.7.3 IL_IOL_EVENT_QUALIFIER

Kurzbeschreibung Diese Struktur wird verwendet, um detaillierte Ereignisinformationen des IO-Link-Device anzuzeigen.

Strukturelement	Datentyp	Beschreibung
Instance	IL_IOL_EVENT_QUALIFIER_INSTANCE	Zeigt die konkrete Quelle (Instanz) eines Ereignisses an IL_IOL_EVENT_QUALIFIER_INSTANCE , auf Seite 141
Source	IL_IOL_EVENT_QUALIFIER_SOURCE	Zeigt die Quelle des Ereignisses an IL_IOL_EVENT_QUALIFIER_SOURCE , auf Seite 142
QualifierType	IL_IOL_EVENT_QUALIFIER_TYPE	Zeigt die Ereigniskategorie an IL_IOL_EVENT_QUALIFIER_TYPE , auf Seite 142
Mode	IL_IOL_EVENT_QUALIFIER_MODE	Zeigt den Ereignismodus an IL_IOL_EVENT_QUALIFIER_MODE , auf Seite 141

Tab. 5-106: Struktur von IL_IOL_EVENT_QUALIFIER

5.7.4 IL_IOL_EVENT_QUALIFIER_INSTANCE

Kurzbeschreibung Die Aufzählung IL_IOL_EVENT_QUALIFIER_INSTANCE wird verwendet, um die konkrete Quelle (Instanz) eines Ereignisses anzuzeigen.

Element	Wert	Beschreibung
UNKNOWN	16#0000	unbekannt
RESERVED	16#0001	reserviert
APPLICATION	16#0004	Anwendung

Tab. 5-107: Aufzählungstyp IL_IOL_EVENT_QUALIFIER_INSTANCE

5.7.5 IL_IOL_EVENT_QUALIFIER_MODE

Kurzbeschreibung Die Aufzählung IL_IOL_EVENT_QUALIFIER_MODE wird verwendet, um den Ereignismodus anzuzeigen.

Element	Wert	Beschreibung
RESERVED	016#000	reserviert
SINGLE_SHOT	016#001	Einmaliges Ereignis
DISAPPEARS	016#002	Ereignis verschwindet
APPEARS	016#003	Ereignis tritt auf

Tab. 5-108: Aufzählungstyp IL_IOL_EVENT_QUALIFIER_MODE

5.7.6 IL_IOL_EVENT_QUALIFIER_SOURCE

Kurzbeschreibung Die Aufzählung IL_IOL_EVENT_QUALIFIER_SOURCE zeigt die Quelle des Ereignisses an.

Element	Wert	Beschreibung
DEVICE	16#0000	Device ist die Quelle. (Remote)
MASTER	16#0001	Master ist die Quelle. (Lokal)

Tab. 5-109: Aufzählungstyp IL_IOL_EVENT_QUALIFIER_SOURCE

5.7.7 IL_IOL_EVENT_QUALIFIER_TYPE

Kurzbeschreibung Die Aufzählung IL_IOL_EVENT_QUALIFIER_TYPE wird verwendet, um die Ereigniskategorie anzuzeigen.

Element	Wert	Beschreibung
RESERVED	016#000	reserviert
NOTIFICATION	016#001	Benachrichtigung
WARNING	016#002	Warnung
ERROR	016#003	Fehler

Tab. 5-110: Aufzählungstyp IL_IOL_EVENT_QUALIFIER_TYPE

5.7.8 IL_IOL_SERVICE

Kurzbeschreibung Die Aufzählung IL_IOL_SERVICE wird verwendet, um die IOL-Dienste auszuwählen.

Element	Wert	Beschreibung
CANCEL	16#0000	Abbrechen / Aufheben des IOL_CALL
IDLE	16#0001	Ungenutzte Sequenz
WRITE	16#0002	Daten schreiben
READ	16#0003	Daten lesen

Tab. 5-111: Aufzählungstyp IL_IOL_SERVICE

5.7.9 IL_IOL_BACKUP_MODE

Kurzbeschreibung Der Aufzählungstyp IL_IOL_BACKUP_MODE wählt den Modus zum Lesen und Speichern der Parameter aus dem IOL Gerät auf den Controller aus.

ENUM	Kontante	Beschreibung
NO_BLOCK	16#0000	Non-Block-Modus. Ein einzelner Parameter wird in diesem Modus eingelesen.
BLOCK	16#0001	Block-Modus. Mehrere Parameter werden in diesem Modus durch das Sperren von Schreibanfragen von anderen Quellen eingelesen um Datensicherheit zu gewährleisten.

Tab. 5-112: Aufzählungstyp IL_IOL_BACKUP_MODE

5.7.10 IL_IOL_RESTORE_MODE

Kurzbeschreibung Der Aufzählungstyp IL_IOL_RESTORE_MODE wird verwendet, um den Modus zum Wiederherstellen der Parameter aus dem Controller auf das IOL-Gerät auszuwählen.

ENUM	Kontante	Beschreibung
BLOCK_CHECK	16#00	Herunterladen von Parametern mit Parameter-Blocking und mit Gültigkeitsprüfung.
NO_BLOCK_CHECK	16#01	Herunterladen von Parametern ohne Parameter-Blocking und Online-Gültigkeitsprüfung.
NO_BLOCK_NO_CHECK	16#02	Herunterladen von Parametern ohne Parameter-Blocking und mit Gültigkeitsprüfung.

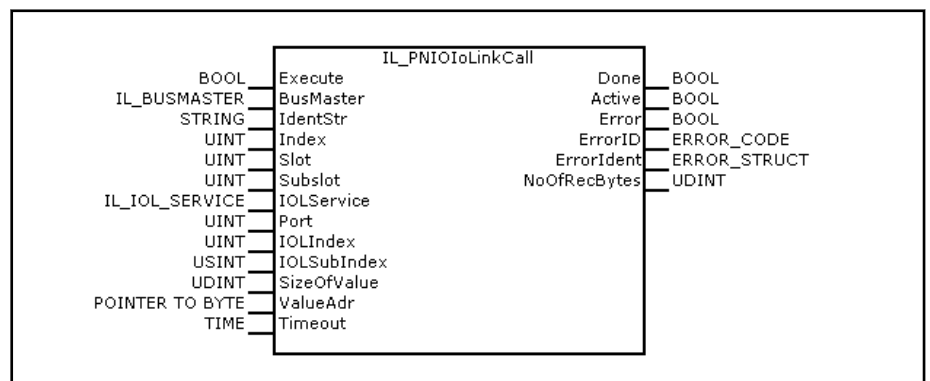
Tab. 5-113: Aufzählungstyp IL_IOL_RESTORE_MODE

5.7.11 IL_PNIOIoLinkCall

Kurzbeschreibung Der Funktionsbaustein **IL_PNIOIoLinkCall** ist für die Lese- und Schreibdienste der IOL-Datenobjekte über Profinet-IO-Link-Gateways zuständig.

Zielsystem	Bibliothek / Ordner
MLC	RIL_ProfinetIO/ POU's/ IO-Link
MTX	RIL_ProfinetIO/ POU's/ IO-Link

Tab. 5-114: Referenztablelle

Abb. 5-39: Schnittstellenbeschreibung des Funktionsbausteins **IL_PNIOIoLinkCall**:

	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Controllers in der Reihenfolge des Konfigurationsbaumes. Ist nur ein PROFINET IO Controller vorhanden, besitzt dieser immer den Wert IL_BUSMASTER_0. Siehe auch IL_BUSMASTER auf Seite 189
	IdentStr	STRING	Identifikationsstring des PROFINET-Device.
	Index	UINT	255 – Regulärer Indexaufruf 0...254 - Indexumlenkung (cap)
	Slot	UINT	Adressierung des Modulslots. Der Slot adressiert ein logisches oder physikalisches Modul (beispielsweise ein E/A Modul)

	Name	Typ	Kommentar
	SubSlot	UINT	Adressierung des Submoduls. Ein Modul kann sich aus mehreren logischen oder physikalischen Submodulen zusammensetzen. Es ist bei jedem Modul (Slot) mindestens das Submodul mit der Nummer 1 vorhanden.
	IOLService	IL_IOL_SERVICE	Aufzählung für IOL-Service. IL_IOL_SERVICE, auf Seite 142
	Port	UINT	1..63: IO-Link-Portnummer
	IOLIndex	UINT	0..32767 - IO-Link-Index, 65535 – Portfunktion
	IOLSubIndex	USINT	0..255: IO-Link-Subindex
	SizeOfValue	UDINT	Größe des ValueAdr-Arrays. Länge der Schreib- (max. 232 Byte) und Lesedaten
	ValueAdr	POINTER TO BYTE	Anfangsadresse des Anforderungs- oder Antwort-Datenarrays.
	Timeout	TIME	Timeout-Überwachung des Funktionsbausteins
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall (aus RIL_CommonTypes)
	ErrorIdent	ERROR_STRUCT	Detailldiagnose (ErrorTable PROFINETIO_TABLE)
	NoOfRecBytes	UDINT	Anzahl der als Antwort empfangenen Bytes.

Tab. 5-115: FB IL_PNIOIoLinkCall E/A-Schnittstelle

Min./max. und Default-Werte: In der folgenden Tabelle sind die Min-/Max- und Default-Werte der Funktionsbausteineingänge aufgelistet.

Name	Typ	Min. Wert	Max. Wert	Defaultwert	Anwendung
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an „Execute“
IdentStr	STRING			""	Steigende Flanke an „Execute“
Index	UINT			255	Steigende Flanke an „Execute“
Slot	UINT	0	32767	0	Steigende Flanke an „Execute“
SubSlot	UINT	1	36863	1	Steigende Flanke an „Execute“
IOLService	IL_IOL_SERVICE			IDLE	Steigende Flanke an „Execute“

Name	Typ	Min. Wert	Max. Wert	Defaultwert	Anwendung
Port	UINT	0	255	0	Steigende Flanke an „Execute“
IOLIndex	UINT			0	Steigende Flanke an „Execute“
IOLSubIndex	USINT			0	Steigende Flanke an „Execute“
SizeOfValue	UDINT	0	Schreiben – 232 Byte	0	Steigende Flanke an „Execute“
ValueAdr	POINTER TO BYTE			0	Steigende Flanke an „Execute“
Timeout	TIME	T#0ms		T#2s	Steigende Flanke an „Execute“

Tab. 5-116: Min./Max. und Default-Werte der Eingänge IL_PNIOIoLinkCall

Funktionsbeschreibung

Der Funktionsbaustein **IL_PNIOIoLinkCall** ist für die Lese- und Schreibdienste der IOL-Datenobjekte über Profinet-IO-Link-Gateways zuständig.

Der Lese- und Schreibdienst hängt von der am IOLService-Eingang getroffenen Auswahl ab. Die IOL-Datenobjekte können mithilfe von IOLIndex und IOLSubIndex direkt gelesen und geschrieben werden.

Die PDU des PNIO IOL-Aufrufs enthält PN-IO-Header, Aufrufs-Header, IOL-Header und Body.

Die Codierung der PDU der IOL Call-Anforderung ist in der nachstehenden Tabelle festgelegt.

Section	Content	Size	Coding	Definitions
DP-V1-Header	Function_Num	1 Octet	5FH	Fix ("Write Record")
	Slot_Number	1 Octet	x	Slot: IOL-M
	Index	1 Octet	255 0...254	Regular index (CALL) Index redirection (cap)
	Length	1 Octet	0...241	Length of data including header
Call -Header	Extended_Function_Num	1 Octet	08h	Indicates "CALL", fix
	Entity_Port	1 Octet	0 1...63 64...255	IOL-M Port number Reserved
	FI_Index	2 Octets	65098	IOL-Header is following
IOL-Header	Control	1 Octet	0...255	"Coding of Control" (9.3.3)
	IOL_Index	2 Octet	0...32767 65535	IOL-D data: Index Port function invocation
	IOL_Subindex	1 Octet	0...255	IOL-D data: Subindex or Port function
Body	IOL_Data_Object	232 Octets maximum		"Write" Data
NOTE The coding of the DP-V1-Header within this table is exemplary for PB-DP. The PN-IO-Header is characterized by API /Slot /Subslot /Index.				

Abb. 5-40: Codierung der PDU einer IOL_CALL-Anforderung

Der Benutzer muss alle Eingänge für den Funktionsbaustein bezüglich der IOL Call-PDU bereitstellen. PN-IO-Header, Extended_Fuction_Num & FI_Index werden intern geregelt. Während der Schreibenanforderung sind die im Array anzugebenden Daten am Eingang "ValueAdr" und die Anzahl der zu schreibenden Daten am Eingang "SizeOfValue" definiert. Während der Leseanforderung sollte der Benutzer darauf achten, dass "SizeOfValue" den Antwortdaten entspricht. Für den Antwortvorgang des Lesen und Schreibens

wird der Header überprüft, um die erfolgreiche Kommunikation zu bestätigen. Beim Lesevorgang werden die Antwortdaten nach "ValueAdr" kopiert und die Gesamtzahl der erhaltenen Datenbytes in "NoOfRecBytes" aktualisiert.

Die steigende Flanke am Eingang "Execute" bearbeitet den Vorgang und setzt Done auf True, wenn der Vorgang erfolgreich beendet ist.

Tritt ein Fehler auf, wird das Error-Bit gesetzt und detaillierte Informationen über den Fehler werden in "ErrorID" und "ErrorIdent." bereitgestellt.

Besonderheiten:

Parameter-/Werte-Anforderungen für Werte über 16 Bit und z.B. den Datentyp UINT müssen gewappt werden.

Beispielprogramm

1. In diesem Beispielprojekt wird der "Herstellername" des an Port 0 angeschlossenen Device gelesen.

Der Profinet-Buskoppler befindet sich an Slot 0.

Das Datenobjekt Herstellername wird dem IOL-Index 16#10 zugeordnet.

Die steigende Flanke am Eingang "bExecute" führt den Vorgang aus. Die gelesenen Daten des Device können "arValue" entnommen werden.

Die Länge des Herstellernamens wird unter "udiNoOfRecvBytes" angezeigt.

Deklaration

```
PROGRAM prgIL_PNIOIoLinkCall
VAR
    fbIL_PNIOIoLinkCall: IL_PNIOIoLinkCall;
    bExecute: BOOL := FALSE;
    uiIndex: UINT := 16#FF; //Index to read IOL data
    strIdentString: STRING := 'iolballuff'; //Station name of PN BK
    uiSlot: UINT := 16#01;
    uiSubSlot: UINT := 16#01;
    enIOLServiceRead: IL_IOL_SERVICE := IL_IOL_SERVICE.READ;
    uiPort: UINT := 16#01;
    uiIOLIndex: UINT := 16#10;
    usiIOLSubIndex: USINT := 16#00;
    arValue: ARRAY[1..232] OF BYTE;
    tTimeout: TIME := T#1S;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: RIL_COMMONTYPES.ERROR_CODE;
    enErrorIdent: RIL_COMMONTYPES.ERROR_STRUCT;
    udiNoOfRecvBytes: UDINT;
END_VAR
```

Implementierung

```
fbIL_PNIOIoLinkCall(
    Execute:= bExecute,
    BusMaster:= IL_BUSMASTER_IL_BUSMASTER_0,
    IdentStr:= strIdentString,
    Index:= uiIndex,
    Slot:= uiSlot,
    Subslot:= uiSubSlot,
    IOLService:= enIOLServiceRead,
    Port:= uiPort,
    IOLIndex:= uiIOLIndex,
    IOLSubIndex:= usiIOLSubIndex,
    SizeOfValue:= SIZEOF(arValue),
    ValueAdr:= ADR(arValue),
    Timeout:= tTimeout,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> enErrorIdent,
    NoOfRecBytes=> udiNoOfRecvBytes);
```

2. In diesem Beispiel werden die Daten mit den Werten 10 und 11 in den Index 16#40 geschrieben.

Deklaration

```
PROGRAM prgIL_PNIOIoLinkCall
VAR
    fbIL_PNIOIoLinkCall: IL_PNIOIoLinkCall;
    bExecute: BOOL := FALSE;
    uiIndex: UINT := 16#FF; //Index to read IOL data
    strIdentString: STRING := 'iolballuff'; //Station name of PN BK
    uiSlot: UINT := 16#01;
    uiSubSlot: UINT := 16#01;
    enIOLServiceWrite: IL_IOL_SERVICE := IL_IOL_SERVICE.WRITE;
    uiPort: UINT := 16#01;
    uiIOLIndex: UINT := 16#10;
    usiIOLSubIndex: USINT := 16#00;
    udiSizeOfWriteData: UDINT := 16#02;
    arValue: ARRAY[1..232] OF BYTE;
    tTimeout: TIME := T#1S;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: RIL_COMMONTYPES.ERROR_CODE;
    enErrorIdent: RIL_COMMONTYPES.ERROR_STRUCT;
    udiNoOfRecvBytes: UDINT;
END_VAR
```

Implementierung

```
fbIL_PNIOIoLinkCall(
    Execute:= bExecute,
    BusMaster:= IL_BUSMASTER.IL_BUSMASTER_0,
    IdentStr:= strIdentString,
    Index:= uiIndex,
    Slot:= uiSlot,
    Subslot:= uiSubSlot,
    IOLService:= enIOLServiceWrite,
    Port:= uiPort,
    IOLIndex:= uiIOLIndex,
    IOLSubIndex:= usiIOLSubIndex,
    SizeOfValue:= udiSizeOfWriteData,
    ValueAdr:= ADR(arValue),
    Timeout:= tTimeout,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> enErrorIdent,
    NoOfRecvBytes=> udiNoOfRecvBytes);
```



Prüfen Sie, ob das Gerät über denselben Index und kompatible Werte verfügt, bevor Sie das Beispielprojekt verwenden.

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle **PROFINETIO_TABLE** (ERROR_TABLE = 16#0200).

Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Bemerkungen
INPUT_RANGE_ERROR	16#0001	16#0000	ValueAdr ist nicht gesetzt
INPUT_RANGE_ERROR	16#0001	16#0001	SizeOfValue ist nicht gesetzt. SizeOfValue > 232 für Schreibanforderung
INPUT_RANGE_ERROR	16#0001	16#0002	IdentStr ist ungültig
INPUT_RANGE_ERROR	16#0001	16#0003	IOLService ist ungültig

ErrorID	Additional1	Additional2	Bemerkungen
COMMUNICATION_ERROR	16#0003	<i>PROFINET IO Controller-Diagnose</i> Allgemeine Fehlercodes , auf Seite 196 Spezielle Fehlercodes, Profinet IO-Controller , auf Seite 198	
COMMUNICATION_ERROR	16#0004	PROFINET IO Status Spezielle Fehlercodes, Profinet IO-Status , auf Seite 199	
COMMUNICATION_ERROR	16#0005	16#0000	Keine Übereinstimmung für Antwort-Header
SYSTEM_ERROR	16#0101	interne Verwendung	Controller nicht in der Konfigurationstabelle gefunden
COMMUNICATION_ERROR	16#0104	16#0000	Zeitüberschreitung
STATE_MACHINE_ERROR	16#0105	interne Verwendung	State Machine Fehler
Spezielle Fehlercodes, IO-Link Masters (IOL-M Fehler)			
COMMUNICATION_ERROR	16#3000	16#7000	IOL_CALL-Konflikt. Unerwartete Schreibanforderung statt Leseanforderung
COMMUNICATION_ERROR	16#3000	16#7001	Falscher IOL_CALL. Decodierfehler
COMMUNICATION_ERROR	16#3000	16#7002	Port blockiert. Port mit anderer Task beschäftigt
COMMUNICATION_ERROR	16#3000	16#8000	Zeitüberschreitung, wenn die Ports IOL-Ds oder IOL-M beschäftigt sind
COMMUNICATION_ERROR	16#3000	16#8001	Falscher Index. IOL_Index 32767 ... 65535
COMMUNICATION_ERROR	16#3000	16#8002	Falsche Port-Adresse. Port-Adresse über dem festgelegten Maximum
COMMUNICATION_ERROR	16#3000	16#8003	Falsche Portfunktion. Portfunktion wird nicht unterstützt
Spezielle Fehlercodes, IO Link Device (anwendungsbezogene Fehler)			
COMMUNICATION_ERROR	16#3001	16#8000	Device-Anwendungsfehler – keine Details
COMMUNICATION_ERROR	16#3001	16#8011	Index nicht vorhanden
COMMUNICATION_ERROR	16#3001	16#8012	SubIndex nicht vorhanden
COMMUNICATION_ERROR	16#3001	16#8020	Dienst zeitweise nicht verfügbar
COMMUNICATION_ERROR	16#3001	16#8021	Dienst zeitweise nicht verfügbar – Lokale Steuerung
COMMUNICATION_ERROR	16#3001	16#8022	Dienst zeitweise nicht verfügbar – Device-Steuerung
COMMUNICATION_ERROR	16#3001	16#8023	Zugriff verweigert
COMMUNICATION_ERROR	16#3001	16#8030	Parameterwert außerhalb des Bereichs
COMMUNICATION_ERROR	16#3001	16#8031	Parameterwert über dem Grenzwert
COMMUNICATION_ERROR	16#3001	16#8032	Parameterwert unter dem Grenzwert
COMMUNICATION_ERROR	16#3001	16#8033	Parameterlänge überschritten
COMMUNICATION_ERROR	16#3001	16#8034	Parameterlänge unterschritten
COMMUNICATION_ERROR	16#3001	16#8035	Funktion nicht verfügbar
COMMUNICATION_ERROR	16#3001	16#8036	Funktion zeitweise nicht verfügbar
COMMUNICATION_ERROR	16#3001	16#8040	Ungültiger Parametersatz
COMMUNICATION_ERROR	16#3001	16#8041	Inkonsistenter Parametersatz

ErrorID	Additional1	Additional2	Bemerkungen
COMMUNICATION_ERROR	16#3001	16#8082	Anwendung nicht bereit
COMMUNICATION_ERROR	16#3001	16#8001..16#81FF	Herstellerspezifisch

Tab. 5-117: Fehlercodes PNIO_IO-Link

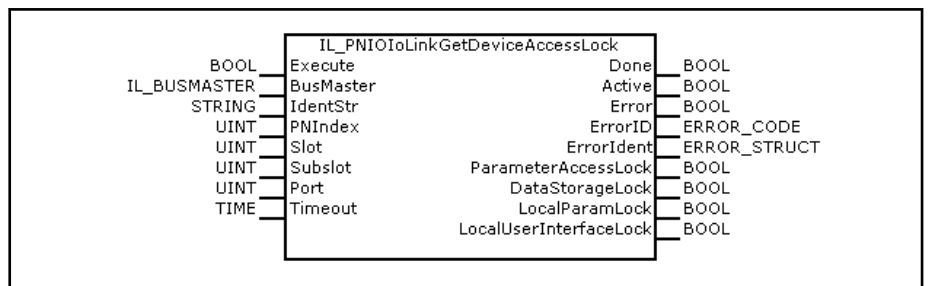
5.7.12 IL_PNIOIoLinkGetDeviceAccessLock

Kurzbeschreibung Der Funktionsbaustein **IL_PNIOIoLinkGetDeviceAccessLock** liest den Status des IO-Link-Deviceparameters Device Access Locks des IO-Link-Device, das an den Profinet-Koppler angeschlossen ist. Dieser Parameter enthält die folgenden Sperrkategorien:

- Parameter-Schreibzugriff (optional)
- Datenspeicherung (erforderlich, wenn das Device Datenspeicherung unterstützt)
- Lokale Parametrierung (optional)
- Bedienung einer lokalen Benutzerschnittstelle (optional)

Zielsystem	Bibliothek / Ordner
MLC	RIL_ProfinetIO/ POU's/ IO-Link
MTX	RIL_ProfinetIO/ POU's/ IO-Link

Tab. 5-118: Referenztafel

Abb. 5-41: Schnittstellenbeschreibung des Funktionsbausteins **IL_PNIOIoLinkGetDeviceAccessLock**

	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Controllers in der Reihenfolge des Konfigurationsbaumes. Ist nur ein PROFINET IO Controller vorhanden, besitzt dieser immer den Wert IL_BUSMASTER_0. Siehe auch IL_BUSMASTER auf Seite 189
	IdentStr	STRING	Identifikationsstring des PROFINET-Device.
	PNIndex	UINT	255 – Regulärer Indexaufruf 0...254 - Indexumlenkung (cap)
	Slot	UINT	Adressierung des Modulslots. Der Slot adressiert ein logisches oder physikalisches Modul (beispielsweise ein E/A Modul)

	Name	Typ	Kommentar
	SubSlot	UINT	Adressierung des Submoduls. Ein Modul kann sich aus mehreren logischen oder physikalischen Submodulen zusammensetzen. Es ist bei jedem Modul (Slot) mindestens das Submodul mit der Nummer 1 vorhanden.
	Port	UINT	1..63: IO-Link-Portnummer
	Timeout	TIME	Timeout-Überwachung des Funktionsbausteins
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall (aus RIL_CommonTypes)
	ErrorIdent	ERROR_STRUCT	Detailldiagnose (ErrorTable PROFINETIO_TABLE)
	ParameterAccess-Lock	BOOL	Sperrstatus des Parameter-Schreibzugriffs FALSE: nicht gesperrt, TRUE: gesperrt
	DataStorageLock	BOOL	Sperrstatus der Datenspeicherung FALSE: nicht gesperrt, TRUE: gesperrt
	LocalParamLock	BOOL	Sperrstatus der lokalen Parametrierung FALSE: nicht gesperrt, TRUE: gesperrt
	LocalUserInterface-Lock	BOOL	Sperrstatus der lokalen Benutzerschnittstelle FALSE: nicht gesperrt, TRUE: gesperrt

Tab. 5-119: FB IL_PNIOIoLinkGetDeviceAccessLock E/A-Schnittstelle

Min./max. und Default-Werte: In der folgenden Tabelle sind die Min-/Max- und Default-Werte der Funktionsbausteineingänge aufgelistet.

Name	Typ	Min. Wert	Max. Wert	Defaultwert	Anwendung
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an „Execute“
IdentStr	STRING	---	---	---	Steigende Flanke an „Execute“
PNIndex	UINT	0	16#ffff	0	Steigende Flanke an „Execute“
Slot	UINT	0	16#7fff	0	Steigende Flanke an „Execute“
SubSlot	UINT	1	16#8fff	1	Steigende Flanke an „Execute“
Port	UINT	0	255	0	Steigende Flanke an „Execute“
Timeout	TIME	T#0ms	----	T#2s	Steigende Flanke an „Execute“

Tab. 5-120: Min./Max. und Default-Werte der Eingänge IL_PNIOIoLinkGetDeviceAccessLock

Funktionsbeschreibung Der Funktionsbaustein **IL_PNIOIoLinkGetDeviceAccessLock** liest den Status des IO-Link-Geräteparameters Device Access Locks, der an einen bestimmten IO-Link-Port angeschlossen ist. Dieser Funktionsbaustein liest den IO-Link-Index 16#000C. Dieser Parameter enthält die folgenden Sperrkategorien. Die Unterstützung dieser Kategorien ist geräteabhängig:

- Parameter-Schreibzugriff (optional)
- Datenspeicherung (erforderlich, wenn das Device Datenspeicherung unterstützt)
- Lokale Parametrierung (optional)
- Bedienung einer lokalen Benutzerschnittstelle (optional)

Wenn eine Kategorie des Parameters vom angeschlossenen Device nicht unterstützt wird, wird der Status dieses Parameters standardmäßig als FALSE angezeigt. Die folgende Tabelle spezifiziert die Sperrmöglichkeiten für Devices.

Das Device wird über den PROFINET Device-Namen in "IdentStr" und die Portnummer in "Port" adressiert. Es muss Teil der Konfiguration des Controllers sein. Der Datensatz auf dem PROFINET Device wird über die Kombination von Slot und Subslot sowie den PNIndex adressiert.

Die steigende Flanke am Eingang Execute startet den Vorgang. Ist der Vorgang erfolgreich beendet, wird der Ausgang Done auf TRUE gesetzt. Tritt ein Fehler auf, wird das Error-Bit gesetzt und detaillierte Informationen über den Fehler werden in ErrorID und ErrorIdent bereitgestellt.

Sperrmöglichkeiten des Device

Bit	Kategorie	Definition
0	Parameter-Schreibzugriff (optional)	0: nicht gesperrt (Standardeinstellung) 1: gesperrt
1	Datenspeicherung (erforderlich, wenn das Device Datenspeicherung unterstützt)	0: nicht gesperrt (Standardeinstellung) 1: gesperrt (siehe Hinweis)
2	Lokale Parametrierung (optional)	0: nicht gesperrt (Standardeinstellung) 1: gesperrt
3	Bedienung einer lokalen Benutzerschnittstelle (optional)	0: nicht gesperrt (Standardeinstellung) 1: gesperrt
4-15	reserviert	
Hinweis: Der Master liest die Parameter State_Properties/ State of Data Storage vor jeder Aktion		

Beispielprogramm In diesem Beispielprojekt wird der Status der Device-Zugriffssperren vom IO-Link-Device gelesen, das an Port 5 angeschlossen ist.

Der Profinet-Buskoppler befindet sich an Slot 1 mit "iolballuff" als Stationsname.

Die steigende Flanke am Eingang bExecute führt den Vorgang aus und liest die Daten von "ParameterAccessLock", "DataStorageLock", "LocalParamLock" und "LocalUserInterfaceLock" des Device.

Deklaration

```
PROGRAM ProgIL_PNIOIoLinkGetDeviceAccessLock
VAR
  fb_PNIOIoLinkGetDeviceAccessLock: IL_PNIOIoLinkGetDeviceAccessLock;
  bExecute: BOOL; //Function block is running when TRUE
  uiIndex: UINT := 16#FF; //Index to read IOL data
  strIdentString: STRING := 'iolballuff'; //Station name of PN BK
  uiSlot: UINT := 16#01; //Addressing of the module slot.
  //The slot addresses a logical or a physical module
  uiSubSlot: UINT := 16#01; //Addressing of the submodule.
  uiPort: UINT := 16#06; //IO-Link Port Number
  tTimeout: TIME := T#2S; //Timeout monitoring of the function block
  bDone: BOOL; //Function Block is completed its task
  bActive: BOOL; //Function Block is working on its actual task
  bError: BOOL; //Indicates an error
  enErrorID: RIL_COMMONTYPES.ERROR_CODE; //Class of error
  enErrorIdent: RIL_COMMONTYPES.ERROR_STRUCT; //Detailed information about error
  bParameterAccessLock: BOOL; //Parameter write access lock status
  bDataStorageLock: BOOL; //Data storage lock status
  bLocalParamLock: BOOL; //Local parameterisation lock status
  bLocalUserInterfaceLock: BOOL; //Local user interface lock status
END_VAR
```

Implementierung

```
fb_PNIOIoLinkGetDeviceAccessLock(
  Execute:= bExecute,
  Done=> bDone,
  Active=> bActive,
  Error=> bError,
  ErrorID=> enErrorID,
  ErrorIdent=> enErrorIdent,
  BusMaster:= IL_BUSMASTER.IL_BUSMASTER_0,
  IdentStr:= strIdentString,
  PNIndex:= uiIndex,
  Slot:= uiSlot,
  Subslot:= uiSubSlot,
  Port:= uiPort,
  Timeout:= tTimeout,
  ParameterAccessLock=> bParameterAccessLock,
  DataStorageLock=> bDataStorageLock,
  LocalParamLock=> bLocalParamLock,
  LocalUserInterfaceLock=> bLocalUserInterfaceLock);
```

```
fb_PNIOIoLinkGetDeviceAccessLock(
  Execute TRUE := bExecute TRUE,
  Done TRUE => bDone TRUE,
  Active FALSE => bActive FALSE,
  Error FALSE => bError FALSE,
  ErrorID NONE_ERROR => enErrorID NONE_ERROR,
  ErrorIdent => enErrorIdent,
  BusMaster IL_BUSMASTER := IL_BUSMASTER.IL_BUSMASTER_0,
  IdentStr 'iolballuff' := strIdentString 'iolballuff',
  PNIndex 255 := uiIndex 255,
  Slot 1 := uiSlot 1,
  Subslot 1 := uiSubSlot 1,
  Port 6 := uiPort 6,
  Timeout T#2s := tTimeout T#2s,
  ParameterAccessLock FALSE => bParameterAccessLock FALSE,
  DataStorageLock TRUE => bDataStorageLock TRUE,
  LocalParamLock FALSE => bLocalParamLock FALSE,
  LocalUserInterfaceLock FALSE => bLocalUserInterfaceLock FALSE); RETURN
```

Abb. 5-42: Online-Ausführen des FB

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle PROFINETIO_TABLE (ERROR_TABLE = 16#0200).

Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Bemerkungen
INPUT_RANGE_ERROR	16#0001	16#0000	ValueAdr ist nicht gesetzt
INPUT_RANGE_ERROR	16#0001	16#0001	SizeOfValue ist nicht gesetzt. SizeOfValue > 232 für Schreibanforderung
INPUT_RANGE_ERROR	16#0001	16#0002	IdentStr ist ungültig
INPUT_RANGE_ERROR	16#0001	16#0003	IOLService ist ungültig
COMMUNICATION_ERROR	16#0003	PROFINET IO Controller-Diagnose • Allgemeine Fehlercodes , auf Seite 196 • Spezielle Fehlercodes, Profinet IO-Controller , auf Seite 198	
COMMUNICATION_ERROR	16#0004	PROFINET IO Status • Spezielle Fehlercodes, Profinet IO-Status , auf Seite 199	
COMMUNICATION_ERROR	16#0005	16#0000	Keine Übereinstimmung für Antwort-Header
SYSTEM_ERROR	16#0101	interne Verwendung	Controller nicht in der Konfigurationstabelle gefunden
COMMUNICATION_ERROR	16#0104	16#0000	Zeitüberschreitung
STATE_MACHINE_ERROR	16#0105	interne Verwendung	State Machine Fehler
Spezielle Fehlercodes, IO-Link Masters (IOL-M Fehler)			
COMMUNICATION_ERROR	16#3000	16#7000	IOL_CALL-Konflikt. Unerwartete Schreibanforderung statt Leseanforderung
COMMUNICATION_ERROR	16#3000	16#7001	Falscher IOL_CALL. Decodierfehler
COMMUNICATION_ERROR	16#3000	16#7002	Port blockiert. Port mit anderer Task beschäftigt
COMMUNICATION_ERROR	16#3000	16#8000	Zeitüberschreitung, wenn die Ports IOL-Ds oder IOL-M beschäftigt sind
COMMUNICATION_ERROR	16#3000	16#8001	Falscher Index. IOL_Index 32767 ... 65535
COMMUNICATION_ERROR	16#3000	16#8002	Falsche Port-Adresse. Port-Adresse über dem festgelegten Maximum
COMMUNICATION_ERROR	16#3000	16#8003	Falsche Portfunktion. Portfunktion wird nicht unterstützt
Spezielle Fehlercodes, IO Link Device (anwendungsbezogene Fehler)			
COMMUNICATION_ERROR	16#3001	16#8000	Device-Anwendungsfehler – keine Details
COMMUNICATION_ERROR	16#3001	16#8011	Index nicht vorhanden
COMMUNICATION_ERROR	16#3001	16#8012	SubIndex nicht vorhanden
COMMUNICATION_ERROR	16#3001	16#8020	Dienst zeitweise nicht verfügbar
COMMUNICATION_ERROR	16#3001	16#8021	Dienst zeitweise nicht verfügbar – Lokale Steuerung
COMMUNICATION_ERROR	16#3001	16#8022	Dienst zeitweise nicht verfügbar – Device-Steuerung
COMMUNICATION_ERROR	16#3001	16#8023	Zugriff verweigert
COMMUNICATION_ERROR	16#3001	16#8030	Parameterwert außerhalb des Bereichs
COMMUNICATION_ERROR	16#3001	16#8031	Parameterwert über dem Grenzwert
COMMUNICATION_ERROR	16#3001	16#8032	Parameterwert unter dem Grenzwert

ErrorID	Additional1	Additional2	Bemerkungen
COMMUNICATION_ERROR	16#3001	16#8033	Parameterlänge überschritten
COMMUNICATION_ERROR	16#3001	16#8034	Parameterlänge unterschritten
COMMUNICATION_ERROR	16#3001	16#8035	Funktion nicht verfügbar
COMMUNICATION_ERROR	16#3001	16#8036	Funktion zeitweise nicht verfügbar
COMMUNICATION_ERROR	16#3001	16#8040	Ungültiger Parametersatz
COMMUNICATION_ERROR	16#3001	16#8041	Inkonsistenter Parametersatz
COMMUNICATION_ERROR	16#3001	16#8082	Anwendung nicht bereit
COMMUNICATION_ERROR	16#3001	16#8001..16#81FF	Herstellerspezifisch

Tab. 5-121: Fehlercodes PNIO_IO-Link

5.7.13 IL_PNIOIoLinkIdent

Kurzbeschreibung Der Funktionsbaustein **L_PNIOIoLinkIdent** wird verwendet, um grundlegende Informationen zur Identifizierung eines IO-Link-Device zu lesen, das an einen bestimmten IO-Link-Port von Profinet angeschlossen ist.

Zielsystem	Bibliothek / Ordner
MLC	RIL_ProfinetIO/ POU's/ IO-Link
MTX	RIL_ProfinetIO/ POU's/ IO-Link

Tab. 5-122: Referenztablelle

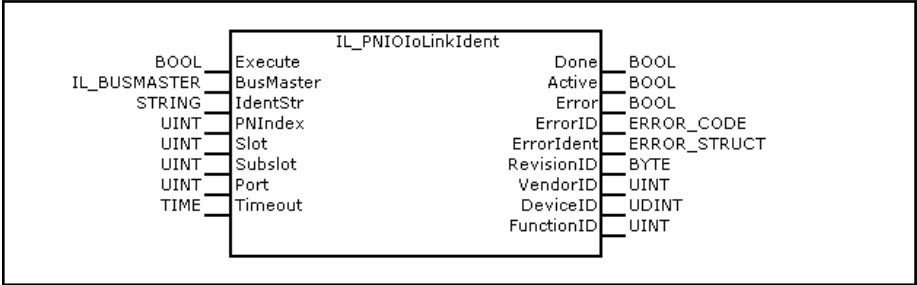


Abb. 5-43: Schnittstellenbeschreibung des Funktionsbausteins IL_PNIOIoLinkIdent

	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Controllers in der Reihenfolge des Konfigurationsbaumes. Ist nur ein PROFINET IO Controller vorhanden, besitzt dieser immer den Wert IL_BUSMASTER_0. Siehe auch IL_BUSMASTER auf Seite 189
	IdentStr	STRING	Identifikationsstring des PROFINET-Device.
	PNIndex	UINT	255 – Regulärer Indexaufruf 0...254 - Indexumlenkung (cap)

	Name	Typ	Kommentar
	Slot	UINT	Adressierung des Modulslots. Der Slot adressiert ein logisches oder physikalisches Modul (beispielsweise ein E/A Modul)
	SubSlot	UINT	Adressierung des Submoduls. Ein Modul kann sich aus mehreren logischen oder physikalischen Submodulen zusammensetzen. Es ist bei jedem Modul (Slot) mindestens das Submodul mit der Nummer 1 vorhanden.
	Port	UINT	1..63: IO-Link-Portnummer
	Timeout	TIME	Timeout-Überwachung des Funktionsbausteins
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall (aus RIL_CommonTypes)
	ErrorIdent	ERROR_STRUCT	Detailldiagnose (ErrorTable PROFINETIO_TABLE)
	RevisionID	BYTE	ID der für die Implementierung verwendeten Protokollversion. (XXXX YYYY of byte) das ‚X‘ gibt die MajorRevisionID und das ‚Y‘ die MinorRevisionId an.
	VendorID	UINT	Eindeutige Herstellerkennung.
	DeviceID	UDINT	Eindeutige von einem Hersteller vergebene Device-Kennung.
	FunctionID	UINT	reserviert (zukünftige Verwendung).

Tab. 5-123: FB IL_PNIOIoLinkIdent E/A-Schnittstelle

Min./max. und Default-Werte: In der folgenden Tabelle sind die Min-/Max- und Default-Werte der Funktionsbausteineingänge aufgelistet.

Name	Typ	Min. Wert	Max. Wert	Defaultwert	Anwendung
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an „Execute“
IdentStr	STRING	---		„	Steigende Flanke an „Execute“
PNIndex	UINT	0	16#ffff	255	Steigende Flanke an „Execute“
Slot	UINT	0	16#7fff	0	Steigende Flanke an „Execute“
SubSlot	UINT	1	16#8fff	1	Steigende Flanke an „Execute“

Name	Typ	Min. Wert	Max. Wert	Defaultwert	Anwendung
Port	UINT	0	255	0	Steigende Flanke an „Execute“
Timeout	TIME	T#0ms	----	T#5s	Steigende Flanke an „Execute“

Tab. 5-124: Min./Max. und Default-Werte der Eingänge IL_PNIOIoLinkIdent

Funktionsbeschreibung

Der Funktionsbaustein **IL_PNIOIoLinkIdent** wird verwendet, um grundlegende Informationen zur Identifizierung eines IO-Link-Device zu lesen, das an einen bestimmten IO-Link-Port angeschlossen ist.

Das Device wird über den PROFINET Device-Namen in "IdentStr" adressiert. Es muss Teil der Konfiguration des Controllers sein. Der Datensatz auf dem PROFINET Device wird über die Kombination von Slot und Subslot sowie den PNIndex adressiert.

Der Funktionsbaustein liest die direkten Parameter von Seite 1 eines IOL-Device. Seite 1 geht von 0x00 bis 0x0F. Sie umfasst das Kommunikationsprotokoll und die Identifizierungsparameter. Der Funktionsbaustein liest nur Identifizierungsparameter von Seite 1 und ihren subIndex-Bereich von 0x04 bis 0x0D.

Beispielprogramm

Dieses Beispielprojekt ist so konfiguriert, dass die Identifizierungsparameter des Device BNI IOL-801-000-Z036 gelesen werden, das an Port 6 des Profinet-Gateways angeschlossen ist. Die steigende Flanke am Eingang "bExecute" führt den Lesevorgang am Device aus. Die Identifizierungsdaten werden an den Ausgangsattributen angezeigt.

Deklaration

```

PROGRAM ProgIL_PNIOIoLinkIdent
VAR
    fbIL_PNIOIoLinkIdent: IL_PNIOIoLinkIdent;
    bExecute:          BOOL; //Function block is running when TRUE
    uiIndex:           UINT := 16#FF; //Index to read IOL data
    strIdentString:    STRING := 'iolballuff'; //Station name of PN BK
    uiSlot:            UINT := 16#01; //Addressing of the module slot.
                        //The slot addresses a logical or a physical module
    uiSubSlot:         UINT := 16#01; //Addressing of the submodule.
    uiPort:            UINT := 16#06; //IO-Link Port Number
    tTimeout:          TIME := T#2S; //Timeout monitoring of the function block
    bDone:             BOOL; //Function Block is completed its task
    bActive:           BOOL; //Function Block is working on its actual task
    bError:            BOOL; //Indicates an error
    enErrorID:         RIL_COMMONTYPES.ERROR_CODE; //Class of error
    enErrorIdent:      RIL_COMMONTYPES.ERROR_STRUCT; //Detailed information about error
    byRevisionID:      BYTE; // ID of the used protocol version for implementation.
    // (XXXX YYYY of byte) the 'X' gives RevisionIDMajor and 'Y' gives RevisionIdMinor.
    uiVendorID:        UINT; // Unique vendor identification.
    udiDeviceID:       UDINT; //Unique Device identification allocated BY a vendor.
    uiFunctionID:      UINT; //Reserved (Engineering shall set both octets to "16#00")
END_VAR

```

Implementierung

```

fbIL_PNIOIoLinkIdent(
    Execute:= bExecute,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> enErrorIdent,
    BusMaster:= IL_BUSMASTER.IL_BUSMASTER_0,
    IdentStr:= strIdentString,
    PNIndex:= uiIndex,
    Slot:= uiSlot,
    Subslot:= uiSubSlot,
    Port:= uiPort,

```



```

Timeout:= tTimeout,
RevisionID=> byRevisionID,
VendorID=> uiVendorID,
DeviceID=> udiDeviceID,
FunctionID=> uiFunctionID);

```

Expression	Type	Value
* fbIL_PNIOIoLinkIdent	IL_PNIOIoLinkIdent	
bExecute	BOOL	TRUE
uiIndex	UINT	16#00FF
strIdentString	STRING	'Iolballuff'
uiSlot	UINT	16#0001
uiSubSlot	UINT	16#0001
uiPort	UINT	16#0006
tTimeout	TIME	T#2s
bDone	BOOL	TRUE
bActive	BOOL	FALSE
bError	BOOL	FALSE
enErrorID	ERROR_CODE	NONE_ERROR
* enErrorIdent	RIL_COMMONTYPES.ERROR_STRUCT	
byRevisionID	BYTE	16#11
uiVendorID	UINT	16#0378
udiDeviceID	UDINT	16#00050A02
uiFunctionID	UINT	16#0000

Abb. 5-44: Online-Ausführen des FB

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle **PROFINETIO_TABLE** (ERROR_TABLE = 16#0200).

Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Bemerkungen
INPUT_RANGE_ERROR	16#0001	16#0000	ValueAdr ist nicht gesetzt
INPUT_RANGE_ERROR	16#0001	16#0001	SizeOfValue ist nicht gesetzt. SizeOfValue > 232 für Schreibanforderung
INPUT_RANGE_ERROR	16#0001	16#0002	IdentStr ist ungültig
INPUT_RANGE_ERROR	16#0001	16#0003	IOLService ist ungültig
COMMUNICATION_ERROR	16#0003	PROFINET IO Controller-Diagnose Allgemeine Fehlercodes , auf Seite 196 Spezielle Fehlercodes, Profinet IO-Controller , auf Seite 198	
COMMUNICATION_ERROR	16#0004	PROFINET IO Status Spezielle Fehlercodes, Profinet IO-Status , auf Seite 199	
COMMUNICATION_ERROR	16#0005	16#0000	Keine Übereinstimmung für Antwort-Header
SYSTEM_ERROR	16#0101	interne Verwendung	Controller nicht in der Konfigurationstabelle gefunden
COMMUNICATION_ERROR	16#0104	16#0000	Zeitüberschreitung
STATE_MACHINE_ERROR	16#0105	interne Verwendung	State Machine Fehler
Spezielle Fehlercodes, IO-Link Masters (IOL-M Fehler)			
COMMUNICATION_ERROR	16#3000	16#7000	IOL_CALL-Konflikt. Unerwartete Schreibanforderung statt Leseanforderung
COMMUNICATION_ERROR	16#3000	16#7001	Falscher IOL_CALL. Decodierfehler

ErrorID	Additional1	Additional2	Bemerkungen
COMMUNICATION_ERROR	16#3000	16#7002	Port blockiert. Port mit anderer Task beschäftigt
COMMUNICATION_ERROR	16#3000	16#8000	Zeitüberschreitung, wenn die Ports IOL-Ds oder IOL-M beschäftigt sind
COMMUNICATION_ERROR	16#3000	16#8001	Falscher Index. IOL_Index 32767 ... 65535
COMMUNICATION_ERROR	16#3000	16#8002	Falsche Port-Adresse. Port-Adresse über dem festgelegten Maximum
COMMUNICATION_ERROR	16#3000	16#8003	Falsche Portfunktion. Portfunktion wird nicht unterstützt
Spezielle Fehlercodes, IO Link Device (anwendungsbezogene Fehler)			
COMMUNICATION_ERROR	16#3001	16#8000	Device-Anwendungsfehler – keine Details
COMMUNICATION_ERROR	16#3001	16#8011	Index nicht vorhanden
COMMUNICATION_ERROR	16#3001	16#8012	SubIndex nicht vorhanden
COMMUNICATION_ERROR	16#3001	16#8020	Dienst zeitweise nicht verfügbar
COMMUNICATION_ERROR	16#3001	16#8021	Dienst zeitweise nicht verfügbar – Lokale Steuerung
COMMUNICATION_ERROR	16#3001	16#8022	Dienst zeitweise nicht verfügbar – Device-Steuerung
COMMUNICATION_ERROR	16#3001	16#8023	Zugriff verweigert
COMMUNICATION_ERROR	16#3001	16#8030	Parameterwert außerhalb des Bereichs
COMMUNICATION_ERROR	16#3001	16#8031	Parameterwert über dem Grenzwert
COMMUNICATION_ERROR	16#3001	16#8032	Parameterwert unter dem Grenzwert
COMMUNICATION_ERROR	16#3001	16#8033	Parameterlänge überschritten
COMMUNICATION_ERROR	16#3001	16#8034	Parameterlänge unterschritten
COMMUNICATION_ERROR	16#3001	16#8035	Funktion nicht verfügbar
COMMUNICATION_ERROR	16#3001	16#8036	Funktion zeitweise nicht verfügbar
COMMUNICATION_ERROR	16#3001	16#8040	Ungültiger Parametersatz
COMMUNICATION_ERROR	16#3001	16#8041	Inkonsistenter Parametersatz
COMMUNICATION_ERROR	16#3001	16#8082	Anwendung nicht bereit
COMMUNICATION_ERROR	16#3001	16#8001..16#81FF	Herstellerspezifisch

Tab. 5-125: Fehlercodes PNIO_IO-Link

5.7.14 IL_PNIOIoLinkIdentDetails

Kurzbeschreibung Der Funktionsbaustein **IL_PNIOIoLinkIdentDetails** wird verwendet, um Identifizierungsparameter und gerätespezifische Informationen eines IO-Link-Device zu lesen, das an einen bestimmten IO-Link-Port von Profinet angeschlossen ist.

Zielsystem	Bibliothek / Ordner
MLC	RIL_ProfinetIO/ POU's/ IO-Link
MTX	RIL_ProfinetIO/ POU's/ IO-Link

Tab. 5-126: Referenztable

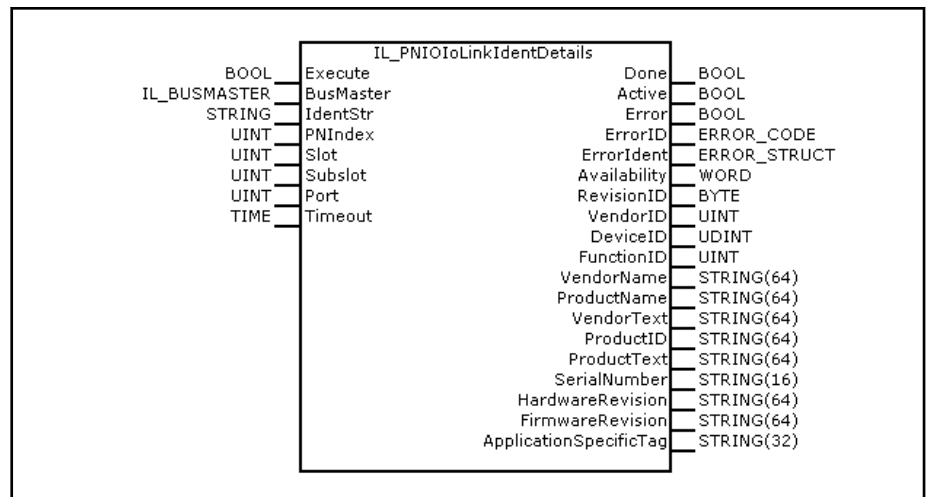


Abb. 5-45: Schnittstellenbeschreibung des Funktionsbausteins **IL_PNIOIoLinkIdentDetails**

	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Controllers in der Reihenfolge des Konfigurationsbaumes. Ist nur ein PROFINET IO Controller vorhanden, besitzt dieser immer den Wert IL_BUSMASTER_0. Siehe auch IL_BUSMASTER auf Seite 189
	IdentStr	STRING	Identifikationsstring des PROFINET-Device.
	PNIndex	UINT	255 – Regulärer Indexaufruf 0...254 - Indexumlenkung (cap)
	Slot	UINT	Adressierung des Modulslots. Der Slot adressiert ein logisches oder physikalisches Modul (beispielsweise ein E/A Modul)
	SubSlot	UINT	Adressierung des Submoduls. Ein Modul kann sich aus mehreren logischen oder physikalischen Submodulen zusammensetzen. Es ist bei jedem Modul (Slot) mindestens das Submodul mit der Nummer 1 vorhanden.
	Port	UINT	1..63: IO-Link-Portnummer
	Timeout	TIME	Timeout-Überwachung des Funktionsbausteins
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall (aus RIL_CommonTypes)
	ErrorIdent	ERROR_STRUCT	Detalldiagnose (ErrorTable PROFINETIO_TABLE)

	Name	Typ	Kommentar
	RevisionID	BYTE	ID der für die Implementierung verwendeten Protokollversion. (XXXX YYYY of byte) das ‚X‘ gibt die MajorRevisionID und das ‚Y‘ die MinorRevisionId an
	VendorID	UINT	Eindeutige Herstellerkennung
	DeviceID	UDINT	Eindeutige von einem Hersteller vergebene Device-Kennung
	FunctionID	UINT	reserviert (zukünftige Verwendung)
	VendorName	STRING(64)	Name des Herstellers
	ProductName	STRING(64)	Name des Produkts
	VendorText	STRING(64)	Herstellertext
	ProductID	STRING(64)	Eindeutige vom Hersteller vergebene Produktkennung
	ProductText	STRING(64)	Produkttext
	SerialNumber	STRING(16)	Herstellerspezifische Seriennummer
	HardwareRevision	STRING(64)	Hardware-Revision des Device
	FirmwareRevision	STRING(64)	Firmware-Revision des Device
	ApplicationSpecific-Tag	STRING(32)	Kann als "Tag Function" (Rolle des Device) oder als "Tag Location" (Ort des Device) dienen

Tab. 5-127: FB IL_PNIOIoLinkIdentDetails E/A-Schnittstelle

Min./max. und Default-Werte: In der folgenden Tabelle sind die Min-/Max- und Default-Werte der Funktionsbausteineingänge aufgelistet.

Name	Typ	Min. Wert	Max. Wert	Defaultwert	Anwendung
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an „Execute“
IdentStr	STRING	---		„	Steigende Flanke an „Execute“
PNIndex	UINT	0	16#ffff	255	Steigende Flanke an „Execute“
Slot	UINT	0	16#7fff	0	Steigende Flanke an „Execute“
SubSlot	UINT	1	16#8fff	1	Steigende Flanke an „Execute“
Port	UINT	0	255	0	Steigende Flanke an „Execute“
Timeout	TIME	T#0ms	----	T#5s	Steigende Flanke an „Execute“

Tab. 5-128: Min./Max. und Default-Werte der Eingänge IL_PNIOIoLinkIdentDetails

Funktionsbeschreibung	Der Funktionsbaustein IL_PNIOIoLinkIdentDetails wird verwendet, um Identifizierungsparameter und gerätespezifische Informationen eines IO-Link-Device zu lesen, das an einen bestimmten IO-Link-Port angeschlossen ist. Das Device wird über den PROFINET Device-Namen in "IdentStr" adressiert. Es muss Teil der Konfiguration des Controllers sein. Der Datensatz auf dem PROFINET Device wird über die Kombination von Slot und Subslot sowie den PNIndex adressiert. Der Funktionsbaustein liest die direkten Parameter von Seite 1 eines IOL-Device. Seite 1 geht von 16#00 bis 16#0F. Sie umfasst das Kommunikationsprotokoll und die Identifizierungsparameter sowie Parameter zur Anwendungssteuerung. Der Funktionsbaustein liest die Identifizierungsparameter von Seite 1 und ihren subIndex-Bereich von 16#04 bis 16#0D sowie die Indices 16#10(VendorName), 16#11(VendorText), 16#12(Product-Name), 16#13(ProductID), 16#14(ProductText), 16#15(SerialNumber), 16#16(HardwareRevision), 16#17(FirmwareRevision) und 16#18(ApplicationSpecificTag). Bei Nichtverfügbarkeit von Optionen wird standardmäßig ein leerer String " " als Parameterwert angezeigt. Die Variable „Availability“ bestätigt die am Ausgang vorhandenen gültigen Daten Bit für Bit wie folgt: Availability Bit0 = TRUE => Index 16#00 (Direkter Parameter Seite 1) ist verfügbar. Availability Bit1 = TRUE => Index 16#10 (Herstellernamen) ist verfügbar. Availability Bit2 = TRUE => Index 16#11 (Herstellertext) ist verfügbar. Availability Bit3 = TRUE => Index 16#12 (Produktname) ist verfügbar. Availability Bit4 = TRUE => Index 16#13 (Produkt-ID) ist verfügbar. Availability Bit5 = TRUE => Index 16#14 (Produkttext) ist verfügbar. Availability Bit6 = TRUE => Index 16#15 (Seriennummer) ist verfügbar. Availability Bit7 = TRUE => Index 16#16 (Hardware-Revision) ist verfügbar. Availability Bit8 = TRUE => Index 16#17 (Firmware-Revision) ist verfügbar. Availability Bit9 = TRUE => Index 16#18 (anwendungsspezifischer Tag) ist verfügbar.
Beispielprogramm	Dieses Beispielprojekt ist so konfiguriert, dass die Identifizierungsparameter des Device BNI IOL-801-000-Z036 gelesen werden, das an Port 6 des Profinet-Gateways angeschlossen ist. Die steigende Flanke am Eingang "bExecute" führt den Lesevorgang am Device aus. Die Identifizierungsdaten werden an den Ausgangsattributen angezeigt.

Deklaration

```

PROGRAM ProgIL_PNIOIoLinkIdentDetails
VAR
  fbIL_PNIOIoLinkIdentDetails: IL_PNIOIoLinkIdentDetails;
  bExecute:          BOOL; //Function block is running when TRUE
  uiIndex:           UINT := 16#FF; //Index to read IOL data
  strIdentString:    STRING := 'iolballuff'; //Station name of PN BK
  uiSlot:            UINT := 16#01; //Addressing of the module slot.
                      // The slot addresses a logical or a physical module
  uiSubSlot:         UINT := 16#01; //Addressing of the submodule.
  uiPort:            UINT := 16#06; //IO-Link Port Number
  tTimeout:          TIME := T#2S; //Timeout monitoring of the function block
  bDone:             BOOL; //Function Block is completed its task
  bActive:           BOOL; //Function Block is working on its actual task
  bError:            BOOL; //Indicates an error
  enErrorID:         RIL_COMMONTYPES.ERROR_CODE; //Class of error
  enErrorIdent:      RIL_COMMONTYPES.ERROR_STRUCT; //Detailed information about error
  wAvailable:        WORD;

```

RIL_ProfinetIO.library

```

byRevisionID:          BYTE; // ID of the used protocol version for implementation. (XXXX YYYY of
                        //byte) the 'X' gives RevisionIDMajor and 'Y' gives RevisionIdMinor.
uiVendorID:            UINT; // Unique vendor identification.
udiDeviceID:          UDINT; //Unique Device identification allocated BY a vendor.
uiFunctionID:          UINT; //Reserved (Engineering shall set both octets to "16#00")
strVendorName:         STRING; // Name of the Vendor.
strProductName:        STRING; // Name of the Product.
strVendorText:         STRING; // Vendor text.
strProductID:          STRING; // Unique Product identification allocated by a vendor.
strProductText:        STRING; // Product Text.
strSerialNumber:       STRING; // Vendor specific serial number.
strHardwareRevision:   STRING; // Hardware revision of the Device.
strFirmwareRevision:   STRING; // Firmware revision of the Device.
strApplicationSpecificTag : STRING ; //It can serve as a "tag function" (role of the Device)
                        // or a "tag location" (location of the Device).

```

END_VAR

Implementierung

```

fbIL_PNIOIoLinkIdentDetails(
    Execute:= bExecute ,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> enErrorIdent,
    BusMaster:= IL_BUSMASTER_IL_BUSMASTER_0,
    IdentStr:= strIdentString,
    PNIndex:= uiIndex,
    Slot:= uiSlot,
    Subslot:= uiSubSlot,
    Port:= uiPort,
    Timeout:= tTimeout,
    Available=> wAvailable,
    RevisionID=> byRevisionID,
    VendorID=> uiVendorID,
    DeviceID=> udiDeviceID,
    FunctionID=> uiFunctionID,
    VendorName=> strVendorName,
    ProductName=> strProductName,
    VendorText=> strVendorText,
    ProductID=> strProductID,
    ProductText=> strProductText,
    SerialNumber=> strSerialNumber,
    HardwareRevision=> strHardwareRevision,
    FirmwareRevision=> strFirmwareRevision,
    ApplicationSpecificTag=> strApplicationSpecificTag);

```

Expression	Type	Value
* fbIL_PNIOIoLinkIdentDetails	IL_PNIOIoLinkIdentDetails	
bExecute	BOOL	TRUE
uiIndex	UINT	16#00FF
strIdentString	STRING	'iolballuff'
uiSlot	UINT	16#0001
uiSubSlot	UINT	16#0001
uiPort	UINT	16#0006
tTimeout	TIME	T#2s
bDone	BOOL	TRUE
bActive	BOOL	FALSE
bError	BOOL	FALSE
enErrorID	ERROR_CODE	NONE_ERROR
* enErrorIdent	RIL_COMMONTYPES.ERROR_STRUCT	
byRevisionID	BYTE	16#11
uiVendorID	UINT	16#0378
udiDeviceID	UDINT	16#00050A02
uiFunctionID	UINT	16#0000
strVendorName	STRING	'BALLUFF'
strProductName	STRING	'BNI IOL-801-000-Z036'
strVendorText	STRING	'www.balluff.com'
strProductID	STRING	'BNI007F'
strProductText	STRING	'Smart Light 3 segment'
strSerialNumber	STRING	''
strHardwareRevision	STRING	'1'
strFirmwareRevision	STRING	'1.1 Build: 2013/05/24 10:53:30 Rev:1268 IOL: 961'
strApplicationSpecificTag	STRING	''

Abb. 5-46: Online-Ausführen des FB



Unter Umständen stellen nicht alle IO-Link-Devices die vollständigen Identifizierungsdaten bereit. Die verfügbaren Daten angeschlossener Devices werden verarbeitet und an Ausgangsattributen angezeigt.

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle **PROFINETIO_TABLE** (ERROR_TABLE = 16#0200).

Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Bemerkungen
INPUT_RANGE_ERROR	16#0001	16#0000	ValueAdr ist nicht gesetzt
INPUT_RANGE_ERROR	16#0001	16#0001	SizeOfValue ist nicht gesetzt. SizeOfValue > 232 für Schreibanforderung
INPUT_RANGE_ERROR	16#0001	16#0002	IdentStr ist ungültig
INPUT_RANGE_ERROR	16#0001	16#0003	IOLService ist ungültig
COMMUNICATION_ERROR	16#0003	PROFINET IO Controller-Diagnose Allgemeine Fehlercodes , auf Seite 196 Spezielle Fehlercodes, Profinet IO-Controller, auf Seite 198	
COMMUNICATION_ERROR	16#0004	PROFINET IO Status Spezielle Fehlercodes, Profinet IO-Status , auf Seite 199	
COMMUNICATION_ERROR	16#0005	16#0000	Keine Übereinstimmung für Antwort-Header
SYSTEM_ERROR	16#0101	interne Verwendung	Controller nicht in der Konfigurationstabelle gefunden
COMMUNICATION_ERROR	16#0104	16#0000	Zeitüberschreitung
STATE_MACHINE_ERROR	16#0105	interne Verwendung	State Machine Fehler

ErrorID	Additional1	Additional2	Bemerkungen
Spezielle Fehlercodes, IO-Link Masters (IOL-M Fehler)			
COMMUNICATION_ERROR	16#3000	16#7000	IOL_CALL-Konflikt. Unerwartete Schreibenanforderung statt Leseanforderung
COMMUNICATION_ERROR	16#3000	16#7001	Falscher IOL_CALL. Decodierfehler
COMMUNICATION_ERROR	16#3000	16#7002	Port blockiert. Port mit anderer Task beschäftigt
COMMUNICATION_ERROR	16#3000	16#8000	Zeitüberschreitung, wenn die Ports IOL-Ds oder IOL-M beschäftigt sind
COMMUNICATION_ERROR	16#3000	16#8001	Falscher Index. IOL_Index 32767 ... 65535
COMMUNICATION_ERROR	16#3000	16#8002	Falsche Port-Adresse. Port-Adresse über dem festgelegten Maximum
COMMUNICATION_ERROR	16#3000	16#8003	Falsche Portfunktion. Portfunktion wird nicht unterstützt
Spezielle Fehlercodes, IO Link Device (anwendungsbezogene Fehler)			
COMMUNICATION_ERROR	16#3001	16#8000	Device-Anwendungsfehler – keine Details
COMMUNICATION_ERROR	16#3001	16#8011	Index nicht vorhanden
COMMUNICATION_ERROR	16#3001	16#8012	SubIndex nicht vorhanden
COMMUNICATION_ERROR	16#3001	16#8020	Dienst zeitweise nicht verfügbar
COMMUNICATION_ERROR	16#3001	16#8021	Dienst zeitweise nicht verfügbar – Lokale Steuerung
COMMUNICATION_ERROR	16#3001	16#8022	Dienst zeitweise nicht verfügbar – Device-Steuerung
COMMUNICATION_ERROR	16#3001	16#8023	Zugriff verweigert
COMMUNICATION_ERROR	16#3001	16#8030	Parameterwert außerhalb des Bereichs
COMMUNICATION_ERROR	16#3001	16#8031	Parameterwert über dem Grenzwert
COMMUNICATION_ERROR	16#3001	16#8032	Parameterwert unter dem Grenzwert
COMMUNICATION_ERROR	16#3001	16#8033	Parameterlänge überschritten
COMMUNICATION_ERROR	16#3001	16#8034	Parameterlänge unterschritten
COMMUNICATION_ERROR	16#3001	16#8035	Funktion nicht verfügbar
COMMUNICATION_ERROR	16#3001	16#8036	Funktion zeitweise nicht verfügbar
COMMUNICATION_ERROR	16#3001	16#8040	Ungültiger Parametersatz
COMMUNICATION_ERROR	16#3001	16#8041	Inkonsistenter Parametersatz
COMMUNICATION_ERROR	16#3001	16#8082	Anwendung nicht bereit
COMMUNICATION_ERROR	16#3001	16#8001..16#81FF	Herstellerspezifisch

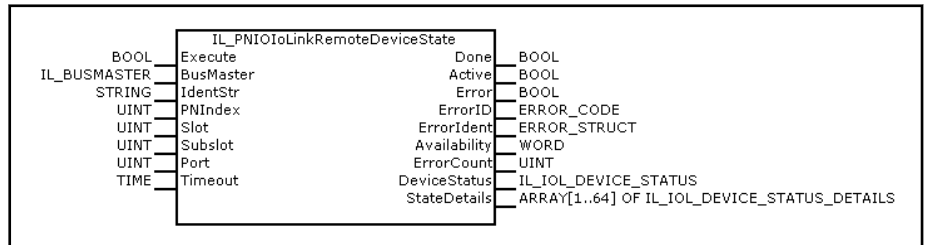
Tab. 5-129: Fehlercodes PNIO_IO-Link

5.7.15 IL_PNIOIoLinkRemoteDeviceState

Kurzbeschreibung Der Funktionsbaustein **IL_PNIOIoLinkRemoteDeviceState** liest den aktuellen Diagnosestatus des IO-Link-Device, das an einen bestimmten IO-Link-Port auf Profinet angeschlossen ist.

Zielsystem	Bibliothek / Ordner
MLC	RIL_ProfinetIO/ POU's/ IO-Link
MTX	RIL_ProfinetIO/ POU's/ IO-Link

Tab. 5-130: Referenztable

Abb. 5-47: Schnittstellenbeschreibung des Funktionsbausteins **IL_PNIOIoLinkRemoteDeviceState**

	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Controllers in der Reihenfolge des Konfigurationsbaumes. Ist nur ein PROFINET IO Controller vorhanden, besitzt dieser immer den Wert IL_BUSMASTER_0. Siehe auch IL_BUSMASTER auf Seite 189
	IdentStr	STRING	Identifikationsstring des PROFINET-Device.
	PNIndex	UINT	255 – Regulärer Indexaufruf 0...254 - Indexumlenkung (cap)
	Slot	UINT	Adressierung des Modulslots. Der Slot adressiert ein logisches oder physikalisches Modul (beispielsweise ein E/A Modul)
	SubSlot	UINT	Adressierung des Submoduls. Ein Modul kann sich aus mehreren logischen oder physikalischen Submodulen zusammensetzen. Es ist bei jedem Modul (Slot) mindestens das Submodul mit der Nummer 1 vorhanden.
	Port	UINT	1..63: IO-Link-Portnummer
	Timeout	TIME	Timeout-Überwachung des Funktionsbausteins
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall (aus RIL_CommonTypes)
	ErrorIdent	ERROR_STRUCT	Detailldiagnose (ErrorTable PROFINETIO_TABLE)

	Name	Typ	Kommentar
	DeviceStatus	IL_IOL_DEVICE_STATUS	Informationen zum Device-Zustand (Diagnose). Aufzählung für den Device-Status IL_IOL_DEVICE_STATUS , auf Seite 140
	StateDetails	ARRAY[1..64] OF IL_IOL_DEVICE_STATUS_DETAILS	Informationen zu aktuell anstehenden Ereignissen im Device. Struktur für DeviceStatusDetails IL_IOL_DEVICE_STATUS_DETAILS , auf Seite 140

Tab. 5-131: FB IL_PNIOIoLinkRemoteDeviceState E/A-Schnittstelle

Min./max. und Default-Werte: In der folgenden Tabelle sind die Min-/Max- und Default-Werte der Funktionsbausteineingänge aufgelistet.

Name	Typ	Min. Wert	Max. Wert	Defaultwert	Anwendung
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an „Execute“
IdentStr	STRING	---		„	Steigende Flanke an „Execute“
PNIndex	UINT	0	16#ffff	255	Steigende Flanke an „Execute“
Slot	UINT	0	16#7fff	0	Steigende Flanke an „Execute“
SubSlot	UINT	1	16#8fff	1	Steigende Flanke an „Execute“
Port	UINT	0	255	0	Steigende Flanke an „Execute“
Timeout	TIME	T#0ms	----	T#2s	Steigende Flanke an „Execute“

Tab. 5-132: Min./Max. und Default-Werte der Eingänge IL_PNIOIoLinkRemoteDeviceState

Funktionsbeschreibung

Der Funktionsbaustein **IL_PNIOIoLinkRemoteDeviceState** liest den aktuellen Diagnosestatus des IO-Link-Device, das an einen bestimmten IO-Link-Port angeschlossen ist.

Das Device wird über den PROFINET Device-Namen in "IdentStr" adressiert. Es muss Teil der Konfiguration des Controllers sein. Der Datensatz auf dem PROFINET Device wird über die Kombination von Slot und Subslot sowie den "PNIndex" adressiert. Der Funktionsbaustein verwendet Index 0x0020, 0x0024 und 0x0025, um auf "ErrorCount", "DeviceStatus" bzw. "DeviceStatusDetails" zuzugreifen.

Der Parameter "ErrorCount" bietet Informationen zu Fehlern, die seit dem Einschalten oder Zurücksetzen in der Device-Anwendung aufgetreten sind. Der Parameter DeviceStatusDetails bietet Informationen zu aktuell anstehenden Ereignissen im Device. Device-Statusparameter sind für einige Devices optional. Für nicht verfügbare Parameter werden die Standardwerte wie folgt zugeordnet

- ErrorCount: 0xFFFF (Standardwert)
- DeviceStatus: STATUS_INVALID (Standardwert)

- StateDetails: Leer (Standardwert).

Die Variable "Availability" bestätigt die am Ausgang vorhandenen gültigen Daten Bit für Bit wie folgt:

- Availability Bit0 = TRUE => Index 0x20 (Fehlerzahl) ist verfügbar.
- Availability Bit1 = TRUE => Index 0x24 (Device-Status) ist verfügbar.
- Availability Bit2 = TRUE => Index 0x25 (Status-Details) ist verfügbar.

Beispielprogramm

Dieses Beispielprojekt ist so konfiguriert, dass die Identifizierungsparameter des Device BFS 33M gelesen werden, das an Port 2 des Profinet-Gateways angeschlossen ist. Die steigende Flanke am Eingang "bExecute" führt den Lesevorgang der Device-Details des Device durch. "ErrorCount" und "DeviceState" werden an den Ausgangsattributen angezeigt.

Deklaration

```
PROGRAM ProgIL_PNIOIoLinkRemoteDeviceState
VAR
    fbIL_PNIOIoLinkRemoteDeviceState: IL_PNIOIoLinkRemoteDeviceState;
    bExecute:      BOOL; //Function block is running when TRUE
    uiIndex:       UINT := 16#FF; //Index to read IOL data
    strIdentString: STRING := 'test'; //Station name of PN BK
    uiSlot:        UINT := 16#01; //Addressing of the module slot.
                    // The slot addresses a logical or a physical module
    uiSubSlot:     UINT := 16#01; //Addressing of the submodule
    uiPort:        UINT := 16#02; //IO-Link Port Number
    tTimeout:      TIME := T#2S; //Timeout monitoring of the function block
    bDone:         BOOL; //Function Block is completed its task
    bActive:       BOOL; //Function Block is working on its actual task
    bError:        BOOL; //Indicates an error
    enErrorID:     RIL_COMMONTYPES.ERROR CODE; //Class of error
    enErrorIdent:  RIL_COMMONTYPES.ERROR STRUCT; //Detailed information about error
    ErrorCount:    UINT := 0 ; //Information on errors occurred in the Device application
                    //since power-on or reset
    DeviceStatus:  IL_IOL_DEVICE_STATUS; //Information about the Device condition (diagnosis)
    StateDetails:  ARRAY[1..64] OF IL_IOL_DEVICE_STATUS_DETAILS ;
                    //Information about currently pending Events in the Device
END_VAR
```

Implementierung

```
fbIL_PNIOIoLinkRemoteDeviceState(
    Execute:= bExecute,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> enErrorIdent,
    BusMaster:= IL_BUSMASTER_IL_BUSMASTER_0,
    IdentStr:= strIdentString,
    PNIndex:= uiIndex,
    Slot:= uiSlot,
    Subslot:= uiSubSlot,
    Port:= uiPort,
    Timeout:= tTimeout,
    ErrorCount=> ErrorCount,
    DeviceStatus=> DeviceStatus,
    StateDetails=> StateDetails);
```

Expression	Type	Value
* fbIL_PNIOIoLinkRemoteDeviceState	IL_PNIOIoLinkRemoteDeviceState	
bExecute	BOOL	TRUE
uiIndex	UINT	16#00FF
strIdentString	STRING	'test'
uiSlot	UINT	16#0001
uiSubSlot	UINT	16#0001
uiPort	UINT	16#0002
tTimeout	TIME	T#2s
bDone	BOOL	TRUE
bActive	BOOL	FALSE
bError	BOOL	FALSE
enErrorID	ERROR_CODE	NONE_ERROR
* enErrorIdent	RIL_COMMONTYPES.ERROR_STRUCT	
ErrorCount	UINT	16#FFFF
DeviceStatus	IL_IOL_DEVICE_STATUS	WORKING_PROPERLY
StateDetails	ARRAY [1..64] OF IL_IOL_DEVICE_STATUS_DETAILS	
StateDetails[1]	IL_IOL_DEVICE_STATUS_DETAILS	
EventCode	WORD	16#0000
EventQualifier	IL_IOL_EVENT_QUALIFIER	
Instance	IL_IOL_EVENT_QUALIFIER_INSTANCE	UNKNOWN
Source	IL_IOL_EVENT_QUALIFIER_SOURCE	DEVICE
QualifierType	IL_IOL_EVENT_QUALIFIER_TYPE	RESERVED
Mode	IL_IOL_EVENT_QUALIFIER_MODE	RESERVED
StateDetails[2]	IL_IOL_DEVICE_STATUS_DETAILS	
StateDetails[3]	IL_IOL_DEVICE_STATUS_DETAILS	

Abb. 5-48: Online-Ausführen des FB

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle **PROFINETIO_TABLE** (ERROR_TABLE = 16#0200).

Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Bemerkungen
INPUT_RANGE_ERROR	16#0001	16#0000	ValueAdr ist nicht gesetzt
INPUT_RANGE_ERROR	16#0001	16#0001	SizeOfValue ist nicht gesetzt. SizeOfValue > 232 für Schreibanforderung
INPUT_RANGE_ERROR	16#0001	16#0002	IdentStr ist ungültig
INPUT_RANGE_ERROR	16#0001	16#0003	IOLService ist ungültig
COMMUNICATION_ERROR	16#0003	PROFINET IO Controller-Diagnose Allgemeine Fehlercodes , auf Seite 196 Spezielle Fehlercodes, Profinet IO-Controller, auf Seite 198	
COMMUNICATION_ERROR	16#0004	PROFINET IO Status Spezielle Fehlercodes, Profinet IO-Status , auf Seite 199	
COMMUNICATION_ERROR	16#0005	16#0000	Keine Übereinstimmung für Antwort-Header
SYSTEM_ERROR	16#0101	interne Verwendung	Controller nicht in der Konfigurationstabelle gefunden
COMMUNICATION_ERROR	16#0104	16#0000	Zeitüberschreitung
STATE_MACHINE_ERROR	16#0105	interne Verwendung	State Machine Fehler
Spezielle Fehlercodes, IO-Link Masters (IOL-M Fehler)			
COMMUNICATION_ERROR	16#3000	16#7000	IOL_CALL-Konflikt. Unerwartete Schreibanforderung statt Leseanforderung

ErrorID	Additional1	Additional2	Bemerkungen
COMMUNICATION_ERROR	16#3000	16#7001	Falscher IOL_CALL. Decodierfehler
COMMUNICATION_ERROR	16#3000	16#7002	Port blockiert. Port mit anderer Task beschäftigt
COMMUNICATION_ERROR	16#3000	16#8000	Zeitüberschreitung, wenn die Ports IOL-Ds oder IOL-M beschäftigt sind
COMMUNICATION_ERROR	16#3000	16#8001	Falscher Index. IOL_Index 32767 ... 65535
COMMUNICATION_ERROR	16#3000	16#8002	Falsche Port-Adresse. Port-Adresse über dem festgelegten Maximum
COMMUNICATION_ERROR	16#3000	16#8003	Falsche Portfunktion. Portfunktion wird nicht unterstützt
Spezielle Fehlercodes, IO Link Device (anwendungsbezogene Fehler)			
COMMUNICATION_ERROR	16#3001	16#8000	Device-Anwendungsfehler – keine Details
COMMUNICATION_ERROR	16#3001	16#8011	Index nicht vorhanden
COMMUNICATION_ERROR	16#3001	16#8012	SubIndex nicht vorhanden
COMMUNICATION_ERROR	16#3001	16#8020	Dienst zeitweise nicht verfügbar
COMMUNICATION_ERROR	16#3001	16#8021	Dienst zeitweise nicht verfügbar – Lokale Steuerung
COMMUNICATION_ERROR	16#3001	16#8022	Dienst zeitweise nicht verfügbar – Device-Steuerung
COMMUNICATION_ERROR	16#3001	16#8023	Zugriff verweigert
COMMUNICATION_ERROR	16#3001	16#8030	Parameterwert außerhalb des Bereichs
COMMUNICATION_ERROR	16#3001	16#8031	Parameterwert über dem Grenzwert
COMMUNICATION_ERROR	16#3001	16#8032	Parameterwert unter dem Grenzwert
COMMUNICATION_ERROR	16#3001	16#8033	Parameterlänge überschritten
COMMUNICATION_ERROR	16#3001	16#8034	Parameterlänge unterschritten
COMMUNICATION_ERROR	16#3001	16#8035	Funktion nicht verfügbar
COMMUNICATION_ERROR	16#3001	16#8036	Funktion zeitweise nicht verfügbar
COMMUNICATION_ERROR	16#3001	16#8040	Ungültiger Parametersatz
COMMUNICATION_ERROR	16#3001	16#8041	Inkonsistenter Parametersatz
COMMUNICATION_ERROR	16#3001	16#8082	Anwendung nicht bereit
COMMUNICATION_ERROR	16#3001	16#8001..16#81FF	Herstellerspezifisch

Tab. 5-133: Fehlercodes PNIO_IO-Link

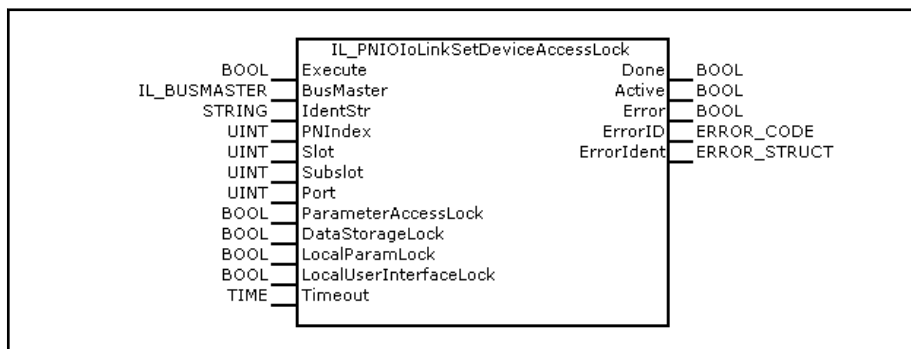
5.7.16 IL_PNIOIoLinkSetDeviceAccessLock

Kurzbeschreibung Der Funktionsbaustein **IL_PNIOIoLinkSetDeviceAccessLock** ermöglicht die Steuerung des Device-Verhaltens durch Überschreiben des Parameters Device Access Locks des IO-Link-Device, das an den Profinet-Koppler angeschlossen ist. Dieser Parameter enthält die folgenden Sperrkategorien:

- Parameter-Schreibzugriff (optional)
- Datenspeicherung (erforderlich, wenn das Device Datenspeicherung unterstützt)
- Lokale Parametrierung (optional)
- Bedienung einer lokalen Benutzerschnittstelle (optional)

Zielsystem	Bibliothek / Ordner
MLC	RIL_ProfinetIO/ POU's/ IO-Link
MTX	RIL_ProfinetIO/ POU's/ IO-Link

Tab. 5-134: Referenztable

Abb. 5-49: Schnittstellenbeschreibung des Funktionsbausteins **IL_PNIOIoLink-SetDeviceAccessLock**

	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Controllers in der Reihenfolge des Konfigurationsbaumes. Ist nur ein PROFINET IO Controller vorhanden, besitzt dieser immer den Wert IL_BUSMASTER_0. Siehe auch IL_BUSMASTER auf Seite 189
	IdentStr	STRING	Identifikationsstring des PROFINET-Device.
	PNIndex	UINT	255 – Regulärer Indexaufruf 0...254 - Indexumlenkung (cap)
	Slot	UINT	Adressierung des Modulslots. Der Slot adressiert ein logisches oder physikalisches Modul (beispielsweise ein E/A Modul)
	SubSlot	UINT	Adressierung des Submoduls. Ein Modul kann sich aus mehreren logischen oder physikalischen Submodulen zusammensetzen. Es ist bei jedem Modul (Slot) mindestens das Submodul mit der Nummer 1 vorhanden.
	Port	UINT	1..63: IO-Link-Portnummer
	ParameterAccessLock	BOOL	Sperrstatus des Parameter-Schreibzugriffs FALSE: nicht gesperrt, TRUE: gesperrt
	DataStorageLock	BOOL	Sperrstatus der Datenspeicherung FALSE: nicht gesperrt, TRUE: gesperrt
	LocalParamLock	BOOL	Sperrstatus der lokalen Parametrierung FALSE: nicht gesperrt, TRUE: gesperrt
	LocalUserInterfaceLock	BOOL	Sperrstatus der lokalen Benutzerschnittstelle FALSE: nicht gesperrt, TRUE: gesperrt

	Name	Typ	Kommentar
	Timeout	TIME	Timeout-Überwachung des Funktionsbausteins
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten ungültig
	Fehler	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall (aus RIL_CommonTypes)
	ErrorIdent	ERROR_STRUCT	Detaildiagnose (ErrorTable PROFINETIO_TABLE)

Tab. 5-135: FB IL_PNIOIoLinkSetDeviceAccessLock E/A-Schnittstelle

Min./max. und Default-Werte: In der folgenden Tabelle sind die Min-/Max- und Default-Werte der Funktionsbausteineingänge aufgelistet.

Name	Typ	Min. Wert	Max. Wert	Defaultwert	Anwendung
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an „Execute“
IdentStr	STRING	---	---	---	Steigende Flanke an „Execute“
PNIndex	UINT	0	16#ffff	0	Steigende Flanke an „Execute“
Slot	UINT	0	16#7fff	0	Steigende Flanke an „Execute“
SubSlot	UINT	1	16#8fff	1	Steigende Flanke an „Execute“
Port	UINT	0	255	0	Steigende Flanke an „Execute“
Timeout	TIME	T#0ms	---	T#2s	Steigende Flanke an „Execute“
ParameterAccessLock	BOOL	FALSE	TRUE	FALSE	Steigende Flanke an „Execute“
DataStorageLock	BOOL	FALSE	TRUE	FALSE	Steigende Flanke an „Execute“
LocalParamLock	BOOL	FALSE	TRUE	FALSE	Steigende Flanke an „Execute“
LocalUserInterfaceLock	BOOL	FALSE	TRUE	FALSE	Steigende Flanke an „Execute“

Tab. 5-136: Min./Max. und Default-Werte der Eingänge IL_PNIOIoLinkSetDeviceAccessLock

Funktionsbeschreibung

Der Funktionsbaustein **IL_PNIOIoLinkSetDeviceAccessLock** ermöglicht die Steuerung des Device-Verhaltens durch Überschreiben des Parameters Device Access Locks, der an einen bestimmten IO-Link-Port angeschlossen ist. Dieser Funktionsbaustein schreibt den IO-Link-Index 16#000C über Sub-In-

dex 0. Dieser Parameter enthält die folgenden Sperrkategorien. Die Unterstützung dieser Kategorien ist geräteabhängig:

- Parameter-Schreibzugriff (optional)
- Datenspeicherung (erforderlich, wenn das Device Datenspeicherung unterstützt)
- Lokale Parametrierung (optional)
- Bedienung einer lokalen Benutzerschnittstelle (optional)

Wenn eine Kategorie dieses Parameters von einem angeschlossenen Device nicht unterstützt wird, meldet der Funktionsbaustein einen Fehler. Die folgende Tabelle spezifiziert die Sperrmöglichkeiten für Devices.

Das Device wird über den PROFINET Device-Namen in "IdentStr" und die Portnummer in "Port" adressiert. Es muss Teil der Konfiguration des Controllers sein. Der Datensatz auf dem PROFINET Device wird über die Kombination von Slot und Subslot sowie den PNIndex adressiert.

Die steigende Flanke am Eingang Execute startet den Vorgang. Ist der Vorgang erfolgreich beendet, wird der Ausgang Done auf TRUE gesetzt. Tritt ein Fehler auf, wird das Error-Bit gesetzt und detaillierte Informationen über den Fehler werden in ErrorID und ErrorIdent bereitgestellt.

Sperrmöglichkeiten des Device

Bit	Kategorie	Definition
0	Parameter-Schreibzugriff (optional)	0: nicht gesperrt (Standardeinstellung) 1: gesperrt
1	Datenspeicherung (erforderlich, wenn das Device Datenspeicherung unterstützt)	0: nicht gesperrt (Standardeinstellung) 1: gesperrt (siehe Hinweis)
2	Lokale Parametrierung (optional)	0: nicht gesperrt (Standardeinstellung) 1: gesperrt
3	Bedienung einer lokalen Benutzerschnittstelle (optional)	0: nicht gesperrt (Standardeinstellung) 1: gesperrt
4-15	reserviert	
Hinweis: Der Master liest die Parameter State_Properties/ State of Data Storage vor jeder Aktion		

Beispielprogramm

In diesem Beispielprojekt wird DataStorageLock von Device Access Locks auf das IO-Link-Device überschrieben, das an Port 5 angeschlossen ist.

Der Profinet-Buskoppler befindet sich an Slot 1 mit "iolballuff" als Stationsname.

Die steigende Flanke am Eingang "bExecute" führt den Vorgang aus und Daten von "ParameterAccessLock", "DataStorageLock", "LocalParamLock" und "LocalUserInterfaceLock" werden auf das Device geschrieben.

Deklaration

```
PROGRAM ProgIL_PNIOIoLinkSetDeviceAccessLock
VAR
```



```

fb_PNIOIoLinkSetDeviceAccessLock: IL_PNIOIoLinkSetDeviceAccessLock; // FB Instance
bExecute: BOOL; //Function block is running when TRUE
uiIndex: UINT := 16#FF; //Index to read IOL data
strIdentString: STRING := 'iolballuff'; //Station name of PN BK
uiSlot: UINT := 16#01; //Addressing of the module slot.
//The slot addresses a logical or a physical module
uiSubSlot: UINT := 16#01; //Addressing of the submodule.
uiPort: UINT := 16#06; //IO-Link Port Number
tTimeout: TIME := T#2S; //Timeout monitoring of the function block
bDone: BOOL; //Function Block is completed its task
bActive: BOOL; //Function Block is working on its actual task
bError: BOOL; //Indicates an error
enErrorID: RIL_COMMONTYPES.ERROR_CODE; //Class of error
enErrorIdent: RIL_COMMONTYPES.ERROR_STRUCT; //Detailed information about error
bParameterAccessLock: BOOL; //Parameter write access lock
bDataStorageLock: BOOL; //Data storage lock
bLocalParamLock: BOOL; //Local parameterisation lock
bLocalUserInterfaceLock: BOOL; //Local user interface lock
END_VAR

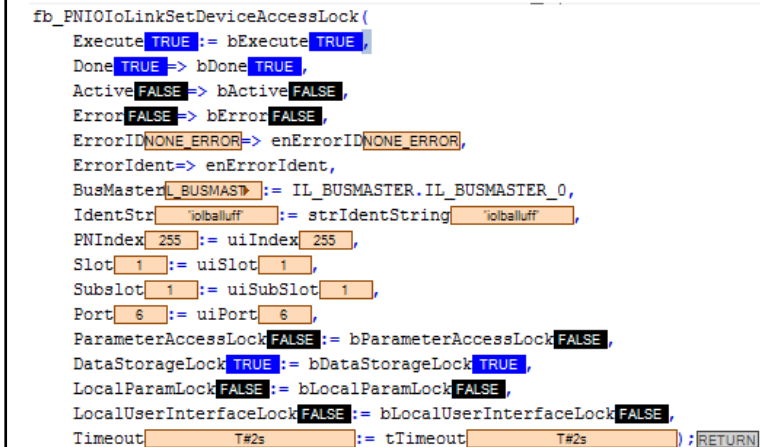
```

Implementierung

```

fb_PNIOIoLinkSetDeviceAccessLock(
    Execute:= bExecute,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> enErrorIdent,
    BusMaster:= IL_BUSMASTER.IL_BUSMASTER_0,
    IdentStr:= strIdentString,
    PNIndex:= uiIndex,
    Slot:= uiSlot,
    Subslot:= uiSubSlot,
    Port:= uiPort,
    ParameterAccessLock:= bParameterAccessLock,
    DataStorageLock:= bDataStorageLock,
    LocalParamLock:= bLocalParamLock,
    LocalUserInterfaceLock:= bLocalUserInterfaceLock,
    Timeout:= tTimeout);

```



```

fb_PNIOIoLinkSetDeviceAccessLock(
    Execute TRUE := bExecute TRUE,
    Done TRUE => bDone TRUE,
    Active FALSE => bActive FALSE,
    Error FALSE => bError FALSE,
    ErrorID NONE_ERROR => enErrorID NONE_ERROR,
    ErrorIdent => enErrorIdent,
    BusMaster IL_BUSMASTER := IL_BUSMASTER.IL_BUSMASTER_0,
    IdentStr 'iolballuff' := strIdentString 'iolballuff',
    PNIndex 255 := uiIndex 255,
    Slot 1 := uiSlot 1,
    Subslot 1 := uiSubSlot 1,
    Port 6 := uiPort 6,
    ParameterAccessLock FALSE := bParameterAccessLock FALSE,
    DataStorageLock TRUE := bDataStorageLock TRUE,
    LocalParamLock FALSE := bLocalParamLock FALSE,
    LocalUserInterfaceLock FALSE := bLocalUserInterfaceLock FALSE,
    Timeout T#2s := tTimeout T#2s); RETURN

```

Abb. 5-50: Online-Ausführen des FB

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle PROFINETIO_TABLE (ERROR_TABLE = 16#0200).

Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Bemerkungen
INPUT_RANGE_ERROR	16#0001	16#0000	ValueAdr ist nicht gesetzt
INPUT_RANGE_ERROR	16#0001	16#0001	SizeOfValue ist nicht gesetzt. SizeOfValue > 232 für Schreibenforderung
INPUT_RANGE_ERROR	16#0001	16#0002	IdentStr ist ungültig
INPUT_RANGE_ERROR	16#0001	16#0003	IOLService ist ungültig
COMMUNICATION_ERROR	16#0003	PROFINET IO Controller-Diagnose Allgemeine Fehlercodes , auf Seite 196 Spezielle Fehlercodes, Profinet IO-Controller , auf Seite 198	
COMMUNICATION_ERROR	16#0004	PROFINET IO Status Spezielle Fehlercodes, Profinet IO-Status , auf Seite 199	
COMMUNICATION_ERROR	16#0005	16#0000	Keine Übereinstimmung für Antwort-Header
SYSTEM_ERROR	16#0101	interne Verwendung	Controller nicht in der Konfigurationstabelle gefunden
COMMUNICATION_ERROR	16#0104	16#0000	Zeitüberschreitung
STATE_MACHINE_ERROR	16#0105	interne Verwendung	State Machine Fehler
Spezielle Fehlercodes, IO-Link Masters (IOL-M Fehler)			
COMMUNICATION_ERROR	16#3000	16#7000	IOL_CALL-Konflikt. Unerwartete Schreibenforderung statt Leseanforderung
COMMUNICATION_ERROR	16#3000	16#7001	Falscher IOL_CALL. Decodierfehler
COMMUNICATION_ERROR	16#3000	16#7002	Port blockiert. Port mit anderer Task beschäftigt
COMMUNICATION_ERROR	16#3000	16#8000	Zeitüberschreitung, wenn die Ports IOL-Ds oder IOL-M beschäftigt sind
COMMUNICATION_ERROR	16#3000	16#8001	Falscher Index. IOL_Index 32767 ... 65535
COMMUNICATION_ERROR	16#3000	16#8002	Falsche Port-Adresse. Port-Adresse über dem festgelegten Maximum
COMMUNICATION_ERROR	16#3000	16#8003	Falsche Portfunktion. Portfunktion wird nicht unterstützt
Spezielle Fehlercodes, IO Link Device (anwendungsbezogene Fehler)			
COMMUNICATION_ERROR	16#3001	16#8000	Device-Anwendungsfehler – keine Details
COMMUNICATION_ERROR	16#3001	16#8011	Index nicht vorhanden
COMMUNICATION_ERROR	16#3001	16#8012	SubIndex nicht vorhanden
COMMUNICATION_ERROR	16#3001	16#8020	Dienst zeitweise nicht verfügbar
COMMUNICATION_ERROR	16#3001	16#8021	Dienst zeitweise nicht verfügbar – Lokale Steuerung
COMMUNICATION_ERROR	16#3001	16#8022	Dienst zeitweise nicht verfügbar – Device-Steuerung
COMMUNICATION_ERROR	16#3001	16#8023	Zugriff verweigert
COMMUNICATION_ERROR	16#3001	16#8030	Parameterwert außerhalb des Bereichs
COMMUNICATION_ERROR	16#3001	16#8031	Parameterwert über dem Grenzwert
COMMUNICATION_ERROR	16#3001	16#8032	Parameterwert unter dem Grenzwert
COMMUNICATION_ERROR	16#3001	16#8033	Parameterlänge überschritten

ErrorID	Additional1	Additional2	Bemerkungen
COMMUNICATION_ERROR	16#3001	16#8034	Parameterlänge unterschritten
COMMUNICATION_ERROR	16#3001	16#8035	Funktion nicht verfügbar
COMMUNICATION_ERROR	16#3001	16#8036	Funktion zeitweise nicht verfügbar
COMMUNICATION_ERROR	16#3001	16#8040	Ungültiger Parametersatz
COMMUNICATION_ERROR	16#3001	16#8041	Inkonsistenter Parametersatz
COMMUNICATION_ERROR	16#3001	16#8082	Anwendung nicht bereit
COMMUNICATION_ERROR	16#3001	16#8001..16#81FF	Herstellerspezifisch

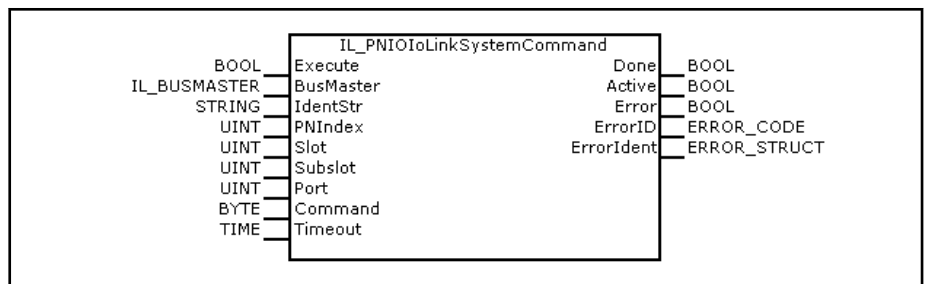
Tab. 5-137: Fehlercodes PNIO_IO-Link

5.7.17 IL_PNIOIoLinkSystemCommand

Kurzbeschreibung Der Funktionsbaustein **IL_PNIOIoLinkSystemCommand** wird verwendet, um den Systembefehl auf ein IO-Link-Device zu schreiben, das an einen bestimmten IO-Link-Port von Profinet angeschlossen ist.

Zielsystem	Bibliothek / Ordner
MLC	RIL_ProfinetIO/ POU's/ IO-Link
MTX	RIL_ProfinetIO/ POU's/ IO-Link

Tab. 5-138: Referenztablelle

Abb. 5-51: Schnittstellenbeschreibung des Funktionsbausteins **IL_PNIOIoLinkSystemCommand**

	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Controllers in der Reihenfolge des Konfigurationsbaumes. Ist nur ein PROFINET IO Controller vorhanden, besitzt dieser immer den Wert IL_BUSMASTER_0. Siehe auch IL_BUSMASTER auf Seite 189
	IdentStr	STRING	Identifikationsstring des PROFINET-Device.
	PNIndex	UINT	255 – Regulärer Indexaufruf 0...254 - Indexumlenkung (cap)
	Slot	UINT	Adressierung des Modulslots. Der Slot adressiert ein logisches oder physikalisches Modul (beispielsweise ein E/A Modul)

	Name	Typ	Kommentar
	SubSlot	UINT	Adressierung des Submoduls. Ein Modul kann sich aus mehreren logischen oder physikalischen Submodulen zusammensetzen. Es ist bei jedem Modul (Slot) mindestens das Submodul mit der Nummer 1 vorhanden.
	Port	UINT	1..63: IO-Link-Portnummer
	Command	BYTE	Systembefehl
	Timeout	TIME	Timeout-Überwachung des Funktionsbausteins
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten ungültig
	Fehler	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall (aus RIL_CommonTypes)
	ErrorIdent	ERROR_STRUCT	Detailldiagnose (ErrorTable PROFINETIO_TABLE)

Tab. 5-139: FB IL_PNIOIoLinkSystemCommand E/A-Schnittstelle

Min./max. und Default-Werte: In der folgenden Tabelle sind die Min-/Max- und Default-Werte der Funktionsbausteineingänge aufgelistet.

Name	Typ	Min. Wert	Max. Wert	Defaultwert	Anwendung
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an „Execute“
IdentStr	STRING	---		‘ ‘	Steigende Flanke an „Execute“
PNIndex	UINT	0	16#ffff	255	Steigende Flanke an „Execute“
Slot	UINT	0	16#7fff	0	Steigende Flanke an „Execute“
SubSlot	UINT	1	16#8fff	1	Steigende Flanke an „Execute“
Port	UINT	0	255	0	Steigende Flanke an „Execute“
Command	BYTE	0	255	0	Steigende Flanke an „Execute“
Timeout	TIME	T#0ms	----	T#2s	Steigende Flanke an „Execute“

Tab. 5-140: Min./Max. und Standardwerte der Eingänge IL_PNIOIoLinkSystem-Command

Funktionsbeschreibung

Der Funktionsbaustein **IL_PNIOIoLinkSystemCommand** soll den Systembefehl auf ein IO-Link-Device schreiben, das an einen bestimmten IO-Link-Port angeschlossen ist. Das Device wird über den PROFINET Device-Namen in "IdentStr" adressiert. Es muss Teil der Konfiguration des Controllers sein.

Der Datensatz auf dem PROFINET Device wird über die Kombination von Slot und Subslot sowie den PNIndex adressiert. Devices, die ISDU unterstützen, sollen den ISDU-Index 0x0002 verwenden, um den Systembefehl zu empfangen.

Ein Systembefehl soll innerhalb von weniger als 5 s ausgeführt werden, um die ISDU-Timing-Anforderungen zu erfüllen. Es gibt außerdem herstellerspezifische Systembefehle.

Die Codierung von Systembefehlen ist in nachstehender Tabelle angegeben.

Command (hex)	Command (dec)	Command name	M/O	Definition
0x00	0	Reserved		
0x01	1	ParamUploadStart	O	Start parameter upload
0x02	2	ParamUploadEnd	O	Stop parameter upload
0x03	3	ParamDownloadStart	O	Start parameter download
0x04	4	ParamDownloadEnd	O	Stop parameter download
0x05	5	ParamDownloadStore	O	Finalize parameterization and start Data Storage
0x06	6	ParamBreak	O	Cancel all Param commands
0x07 to 0x3F	7 to 63	Reserved		
0x40 to 0x7F	64 to 127	Reserved for profiles		
0x80	128	Device reset	O	
0x81	129	Application reset	O	
0x82	130	Restore factory settings	O	
0x83 to 0x9F	131 to 159	Reserved		
0xA0 to 0xFF	160 to 255	Vendor specific		
NOTE See 10.3				
Key M = mandatory; O = optional				

Abb. 5-52: Codierung der Systembefehle

Device reset: Diese Funktion ermöglicht es einem Device einen "Warmstart" durchzuführen. Sie ist besonders nützlich, wenn ein Device auf einen Anfangszustand, zum Beispiel den Zustand beim Einschalten, zurückgesetzt werden muss.

Application reset: Diese Funktion ermöglicht es einem Device die technologiespezifische Anwendung zurückzusetzen. Sie ist besonders nützlich, wenn eine technologiespezifische Anwendung ohne Unterbrechung der Kommunikation und ohne Abschaltzyklus auf einen vordefinierten Betriebszustand eingestellt werden muss.

Factory settings: Diese Funktion ermöglicht es einem Device die Parameter des ursprünglichen Lieferzustands wiederherzustellen.

Beispielprogramm

In diesem Beispielprojekt wird der FB IL_PNIOIoLinkSystemCommand so konfiguriert und ausgeführt, dass die Fabrikeinstellungen für das IO-Link-Device, das an Port 5 angeschlossen ist, wiederhergestellt werden.

Der Profinet-Buskoppler wird an Slot 1 mit 'iolballuff' als Stationsname bereitgestellt.

Die steigende Flanke am Eingang "bExecute" führt den Vorgang aus. Dadurch wird der Systembefehl Restore factory settings - 16#82 auf das Gerät geschrieben, das am Eingang "ByCommand" festgelegt ist.

Deklaration

```

PROGRAM ProgIL_PNIOIoLinkSystemCommand
VAR
    fbIL_PNIOIoLinkSystemCommand: IL_PNIOIoLinkSystemCommand;
    bExecute:          BOOL; //Function block is running when TRUE
    uiIndex:           UINT := 16#FF; //Index to read IOL data
    strIdentString: STRING := 'iolballuff'; //Station name of PN BK

```

```
uiSlot:          UINT := 16#01; //Addressing of the module slot.
//The slot addresses a logical or a physical module
uiSubSlot:       UINT := 16#01; //Addressing of the submodule.
uiPort:          UINT := 16#06; //IO-Link Port Number
ByCommand:       BYTE :=0; //System command.
tTimeout:        TIME := T#2S; //Timeout monitoring of the function block
bDone:           BOOL; //Function Block is completed its task
bActive:         BOOL; //Function Block is working on its actual task
bError:          BOOL; //Indicates an error
enErrorID:       RIL_COMMONTYPES.ERROR_CODE; //Class of error
enErrorIdent:    RIL_COMMONTYPES.ERROR_STRUCT; //Detailed information about error
END_VAR
```

Implementierung

```
fbIL_PNIOIoLinkSystemCommand(
  Execute:= bExecute,
  Done=> bDone,
  Active=> bActive,
  Error=> bError,
  ErrorID=> enErrorID,
  ErrorIdent=> enErrorIdent,
  BusMaster:= IL_BUSMASTER.IL_BUSMASTER_0,
  IdentStr:= strIdentString,
  PNIndex:= uiIndex,
  Slot:= uiSlot,
  SubSlot:= uiSubSlot,
  Port:= uiPort,
  Command:= ByCommand,
  Timeout:= tTimeout);
```

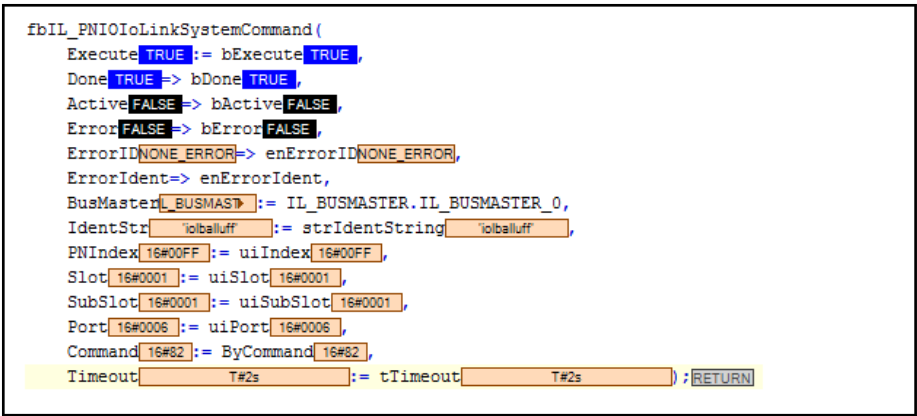


Abb. 5-53: Online-Ausführen des FB

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle PROFINETIO_TABLE (ERROR_TABLE = 16#0200). Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Bemerkungen
INPUT_RANGE_ERROR	16#0001	16#0000	ValueAdr ist nicht gesetzt
INPUT_RANGE_ERROR	16#0001	16#0001	SizeOfValue ist nicht gesetzt. SizeOfValue > 232 für Schreibenanforderung
INPUT_RANGE_ERROR	16#0001	16#0002	IdentStr ist ungültig
INPUT_RANGE_ERROR	16#0001	16#0003	IOLService ist ungültig
COMMUNICATION_ERROR	16#0003	PROFINET IO Controller-Diagnose - Allgemeine Fehlercodes , auf Seite 196 - Spezielle Fehlercodes, Profinet IO-Controller, auf Seite 198	

ErrorID	Additional1	Additional2	Bemerkungen
COMMUNICATION_ERROR	16#0004	PROFINET IO Status Spezielle Fehlercodes, Profinet IO-Status , auf Seite 199	
COMMUNICATION_ERROR	16#0005	16#0000	Keine Übereinstimmung für Antwort-Header
SYSTEM_ERROR	16#0101	interne Verwendung	Controller nicht in der Konfigurationstabelle gefunden
COMMUNICATION_ERROR	16#0104	16#0000	Zeitüberschreitung
STATE_MACHINE_ERROR	16#0105	interne Verwendung	State Machine Fehler
Spezielle Fehlercodes, IO-Link Masters (IOL-M Fehler)			
COMMUNICATION_ERROR	16#3000	16#7000	IOL_CALL-Konflikt. Unerwartete Schreibanforderung statt Leseanforderung
COMMUNICATION_ERROR	16#3000	16#7001	Falscher IOL_CALL. Decodierfehler
COMMUNICATION_ERROR	16#3000	16#7002	Port blockiert. Port mit anderer Task beschäftigt
COMMUNICATION_ERROR	16#3000	16#8000	Zeitüberschreitung, wenn die Ports IOL-Ds oder IOL-M beschäftigt sind
COMMUNICATION_ERROR	16#3000	16#8001	Falscher Index. IOL_Index 32767 ... 65535
COMMUNICATION_ERROR	16#3000	16#8002	Falsche Port-Adresse. Port-Adresse über dem festgelegten Maximum
COMMUNICATION_ERROR	16#3000	16#8003	Falsche Portfunktion. Portfunktion wird nicht unterstützt
Spezielle Fehlercodes, IO Link Device (anwendungsbezogene Fehler)			
COMMUNICATION_ERROR	16#3001	16#8000	Device-Anwendungsfehler – keine Details
COMMUNICATION_ERROR	16#3001	16#8011	Index nicht vorhanden
COMMUNICATION_ERROR	16#3001	16#8012	SubIndex nicht vorhanden
COMMUNICATION_ERROR	16#3001	16#8020	Dienst zeitweise nicht verfügbar
COMMUNICATION_ERROR	16#3001	16#8021	Dienst zeitweise nicht verfügbar – Lokale Steuerung
COMMUNICATION_ERROR	16#3001	16#8022	Dienst zeitweise nicht verfügbar – Device-Steuerung
COMMUNICATION_ERROR	16#3001	16#8023	Zugriff verweigert
COMMUNICATION_ERROR	16#3001	16#8030	Parameterwert außerhalb des Bereichs
COMMUNICATION_ERROR	16#3001	16#8031	Parameterwert über dem Grenzwert
COMMUNICATION_ERROR	16#3001	16#8032	Parameterwert unter dem Grenzwert
COMMUNICATION_ERROR	16#3001	16#8033	Parameterlänge überschritten
COMMUNICATION_ERROR	16#3001	16#8034	Parameterlänge unterschritten
COMMUNICATION_ERROR	16#3001	16#8035	Funktion nicht verfügbar
COMMUNICATION_ERROR	16#3001	16#8036	Funktion zeitweise nicht verfügbar
COMMUNICATION_ERROR	16#3001	16#8040	Ungültiger Parametersatz
COMMUNICATION_ERROR	16#3001	16#8041	Inkonsistenter Parametersatz
COMMUNICATION_ERROR	16#3001	16#8082	Anwendung nicht bereit
COMMUNICATION_ERROR	16#3001	16#8001..16#81FF	Herstellerspezifisch

Tab. 5-141: Fehlercodes PNIO_IO-Link

5.7.18 IL_PNIOIoLinkBackupDevice

Kurzbeschreibung Der Funktionsbaustein **IL_PNIOIoLinkBackupDevice** dient zum Sichern der Konfiguration und Parameter eines (an einem Profinet-IO-Link-Gateway betriebenen) IOL-Devices in eine Datei.

Zielsystem	Bibliothek/Ordner
MLC	RIL_ProfinetIO/ POU's/ IO-Link
MTX	RIL_ProfinetIO/ POU's/ IO-Link

Tab. 5-142: Referenztablelle

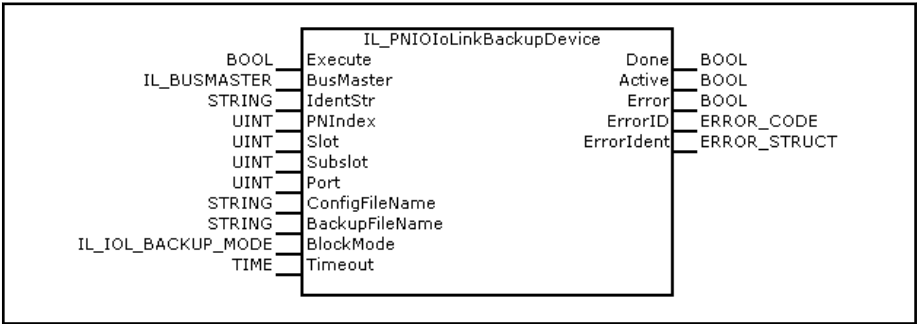


Abb. 5-54: Schnittstellenbeschreibung des Funktionsbausteins **IL_PNIOIoLink-BackupDevice**

	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Controllers in der Reihenfolge des Konfigurationsbaums. Ist nur ein Profinet I/O-Controller vorhanden, besitzt dieser immer den Wert IL_BUSMASTER_0. Siehe auch IL_BUSMASTER auf Seite 189
	IdentStr	STRING	Identifikationsstring des Profinet-Devices.
	PNIndex	UINT	255 – Regulärer Indexaufruf 0...254 - Indexumlenkung (cap)
	Slot	UINT	Adressierung des Modulslots. Der Slot adressiert ein logisches oder physikalisches Modul (beispielsweise ein E/A Modul)
	SubSlot	UINT	Adressierung des Submoduls. Ein Modul kann sich aus mehreren logischen oder physikalischen Submodulen zusammensetzen. Es ist bei jedem Modul (Slot) mindestens das Submodul mit der Nummer 1 vorhanden.
	Port	UINT	1..63: IO-Link, Portnummer
	ConfigFileName	STRING	ConfigFile mit Index und Subindex, um die Parameter aus dem Gerät zu lesen.
	BackupFileName	STRING	Sicherungsdatei, die vom Funktionsbaustein erstellt/überschrieben wird.

	Name	Typ	Kommentar
	BlockMode	IL_IOL_BACKUP_MODE	Aufzählung für den IOL-Block-Modus. Siehe IL_IOL_BACKUP_MODE , auf Seite 142
	Timeout	TIME	Timeout-Überwachung des Funktionsbausteins
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibt die Diagnose im Fehlerfall (aus RIL_Common-Types)
	ErrorIdent	ERROR_STRUCT	Detaildiagnose (Fehlertabelle PROFINETIO_TABLE)

Tab. 5-143: E/A-Schnittstelle des Funktionsbausteins *IL_PNIOIoLinkBackupDevice*

Min./Max. und Default-Werte: In der folgenden Tabelle sind die Min./Max.- und Default-Werte der Funktionsbausteineingänge aufgelistet.

Name	Typ	Min. Wert	Max. Wert	Defaultwert	Anwendung
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an "Execute"
IdentStr	STRING	---	---	""	Steigende Flanke an "Execute"
PNIndex	UINT	16#0	16#FFFF	16#FF	Steigende Flanke an "Execute"
Slot	UINT	16#0	16#7FFF	16#0	Steigende Flanke an "Execute"
SubSlot	UINT	16#1	16#8FFF	16#1	Steigende Flanke an "Execute"
Port	UINT	16#0	16#FF	16#0	Steigende Flanke an "Execute"
ConfigFileName	STRING	---	---	""	Steigende Flanke an "Execute"
BackupFileName	STRING	---	---	""	Steigende Flanke an "Execute"
BlockMode	IL_IOL_BLOCK_MODE	NO_BLOCK	BLOCK	NO_BLOCK	Steigende Flanke an "Execute"
Timeout	TIME	T#0ms		T#2s	Steigende Flanke an "Execute"

Tab. 5-144: Min.-, Max.- und Default-Werte der *IL_PNIOIoLinkBackupDevice*-Eingänge

Funktionsbeschreibung

Der Funktionsbaustein **IL_PNIOIoLinkBackupDevice** dient zum Sichern der Konfiguration und der Parameter eines (an einem Profinet-IO-Link-Gateway betriebenen) IOL-Devices in eine Datei. Der Funktionsbaustein liest die in der Konfigurationsdatei "ConfigFileName" aufgeführten Parameter aus dem De-

vice aus und speichert die Daten in der Datei "BackupFileName". Die Konfigurations- und Sicherungsdateien werden im Flash-Speicher des Controllers abgelegt.

Der Funktionsbaustein erstellt bei jeder Ausführung eine neue Datei "BackupFileName". Ist die Datei bereits vorhanden, wird sie überschrieben.

Index, Subindex	16#0001,16#05
Index, Subindex	16#0020,16#02
Index, Subindex	16#0022,16#00
...	...
...	...
...	...
Index, Subindex	16#0042,16#01
Basisstruktur der ConfigFile	Beispiel der ConfigFile

Der Funktionsbaustein lädt die Parameter im Block/No-Block-Modus, wie am Eingang "BlockMode" festgelegt, hoch.

1. **NO_BLOCK**: Hochladen von Parametern ohne Parameter-Blocking. Parameter werden einzeln gelesen.
2. **BLOCK**: Hochladen von Parametern mit Parameter-Blocking. Bevor die Daten hochgeladen werden, wird der Befehl "ParamUploadStart" ausgeführt. Nachdem die Daten gesichert worden sind, wird der Befehl "ParamUploadEnd" ausgeführt. In diesem Fall kann der Anwender die Schreib Anfragen anderer Quellen unterbinden um die Datenintegrität zu gewährleisten.

Bei steigender Flanke am Eingang "bExecute", liest der Funktionsbaustein die Parameter aus dem Gerät aus und speichert die Daten mit dem ausgewählten Modus in die Sicherungsdatei.

VendorName:	VendorName: XYZ
VendorID:	VendorID: XYZ
ProductName:	ProductName: XYZ
DeviceID:	DeviceID: XYZ
RevisionID:	RevisionID: XYZ
[PARAMETER_VALUES]	[PARAMETER_VALUES]
Index,SubIndex,DataSize,Data	16#0002,16#01,16#01,16#1B
Index,SubIndex,DataSize,Data1,Data2,Data3	16#0005,16#00,16#03,16#28,16#2F,16#43
...	...
...	...
Index,SubIndex,DataSize,Data	16#0020,16#03,16#01,16#08
Basissicherungsdatei	Beispiel der Sicherungsdatei

Beispielprogramm

In diesem Beispielprojekt wird das Konfigurieren beschrieben, um die in "ConfigFileName" festgelegten Parameter zu sichern und diese in der Datei mit dem Namen "BackupFileName" zu speichern.

Die Identifizierungsparameter des Geräts BNI IOL-801-000-Z036 werden mit Port 6 des Profinet-Gateways verbunden. Bei steigender Flanke am Eingang "bExecute", führt der Funktionsbaustein die Sicherung aus dem Gerät durch. Das Ergebnis wird an den Ausgangsattributen dargestellt. Die Konfigurations-

datei muss auf der User-Partition des Dateisystems der Steuerung abgelegt werden. Zum Übertragen der Datei verwenden Sie den Geräte-Explorer in IndraWorks. Durch einen Rechtsklick auf das Symbol der Steuerung öffnet sich das folgende Menü.

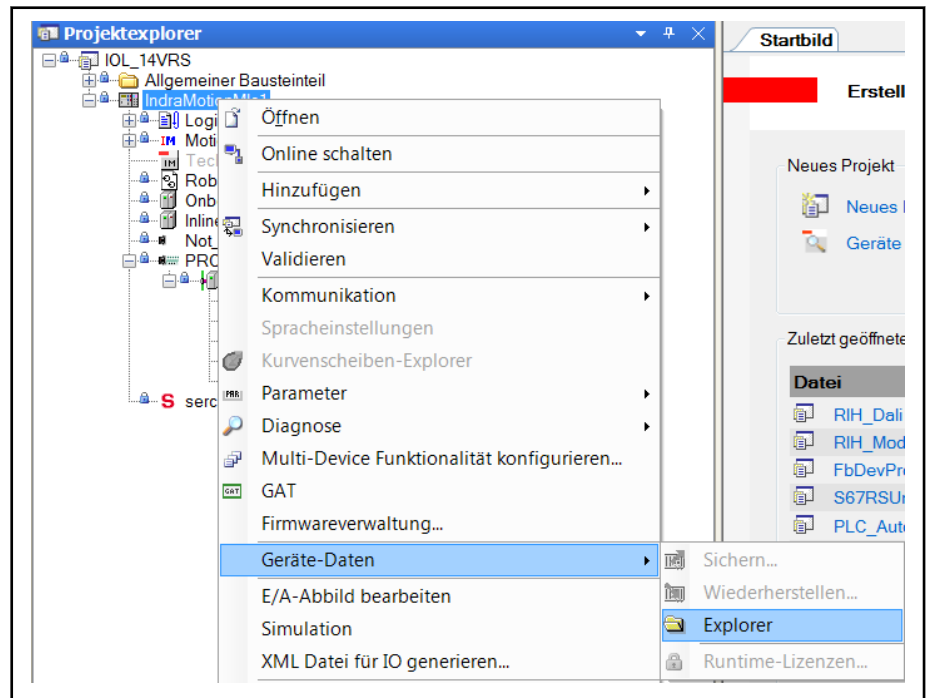


Abb. 5-55: Auswahl des Geräte-Explorers

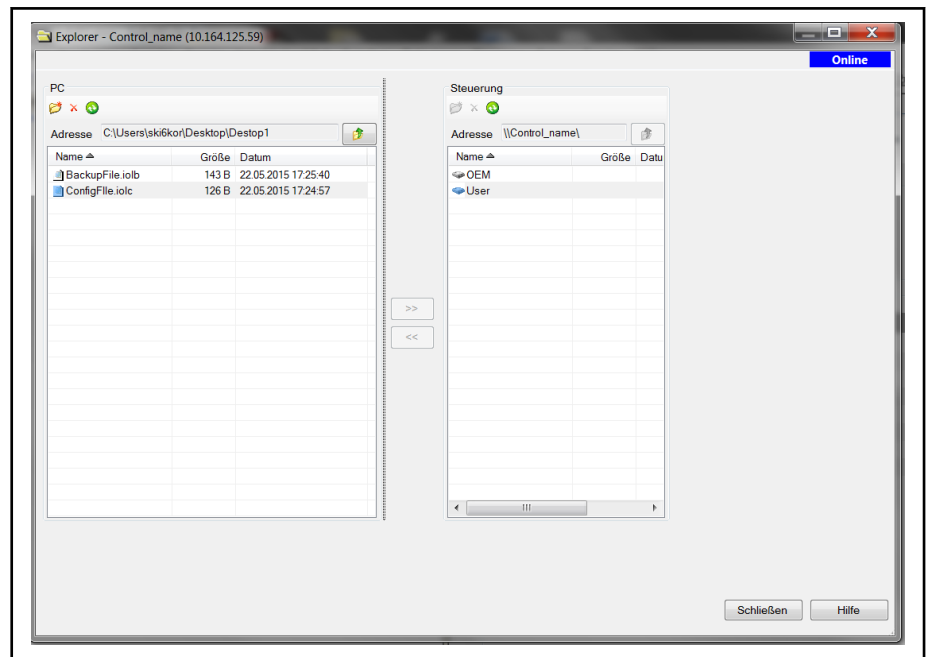


Abb. 5-56: Auswahl der User-Partition

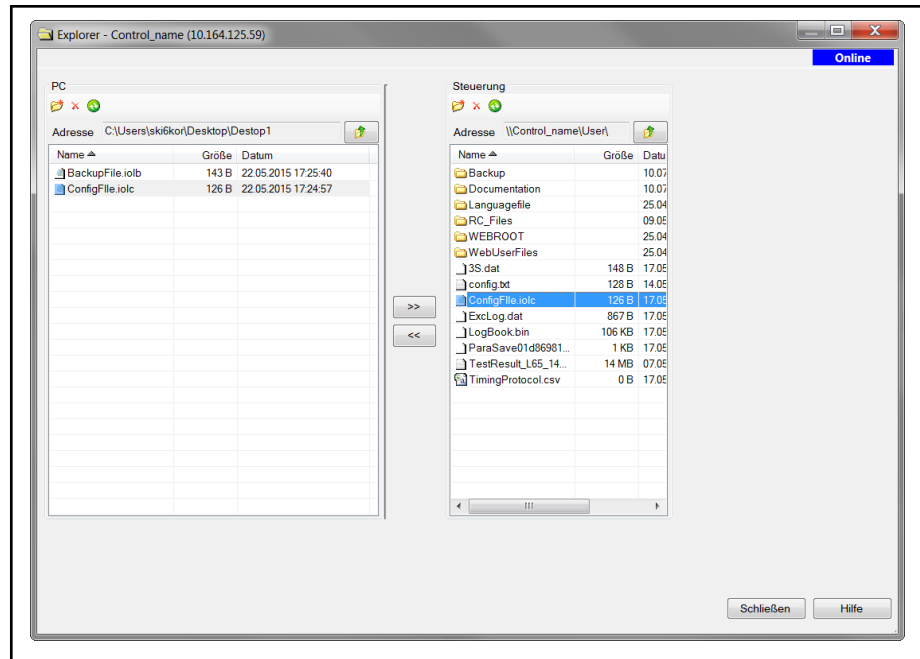


Abb. 5-57: Auswahl der Konfigurationsdatei

Deklaration

```

PROGRAM ProgIL_PNIOIoLinkBackupDevice
VAR
    fbIL_PNIOIoLinkBackupDevice: IL_PNIOIoLinkBackupDevice; // FB Instance

    bExecute:          BOOL; //Function block is running when TRUE
    uiIndex:           UINT := 255; // PN Index
    strIdentString:    STRING := 'iolballuff'; //Station name of PN BK
    uiSlot:            UINT := 16#01; //Addressing of the module slot.
                                //The slot addresses a logical or a physical module
    uiSubSlot:         UINT := 16#01; //Addressing of the submodule.
    uiPort:            UINT := 16#06; //IO-Link Port Number
    strConfigFileName: STRING:= 'ConfigFile'; // Configuration file name
    strBackupFileName: STRING:= 'BackupFile'; // Backup file name
    enBlockMode:       IL_IOL_RESTORE_MODE := IL_IOL_BACKUP_MODE.BLOCK; // Backup Mode
    tTimeout:          TIME := T#3S; //Timeout monitoring of the function block
    bDone:             BOOL; //Function Block is completed its task
    bActive:           BOOL; //Function Block is working on its actual task
    bError:            BOOL; //Indicates an error
    enErrorID:         RIL_COMMONTYPES.ERROR_CODE; //Class of error
    enErrorIdent:      RIL_COMMONTYPES.ERROR_STRUCT; //Detailed information about error
END_VAR

```

Implementierung

```

fbIL_PNIOIoLinkBackupDevice (
    Execute:= bExecute,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> enErrorIdent,
    BusMaster:= IL_BUSMASTER.IL_BUSMASTER_0,
    IdentStr:= strIdentString,
    PNIndex:= uiIndex,
    Slot:= uiSlot,
    SubSlot:= uiSubSlot,
    Port:= uiPort,
    ConfigFileName:= strConfigFileName,
    BackUpFileName:= strBackupFileName,
    BlockMode:= enBlockMode,
    Timeout:= tTimeout);

```

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle **PROFINETIO_TABLE** (ERROR_TABLE = 16#0200).

Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Bemerkungen
INPUT_RANGE_ERROR	16#0001	16#0002	IdentStr ist ungültig
COMMUNICATION_ERROR	16#0003	PROFINET IO Controller-Diagnose • Allgemeine Fehlercodes , Seite 196 • Spezielle Fehlercodes, Profinet IO-Controller, Seite 198	
COMMUNICATION_ERROR	16#0004	PROFINET IO Status • Spezielle Fehlercodes, Profinet IO-Status , Seite 199	
SYSTEM_ERROR	16#0101	internal use	Controller nicht in der Konfigurationstabelle gefunden
COMMUNICATION_ERROR	16#0104	16#0000	Zeitüberschreitung
COMMUNICATION_ERROR	16#3001	16#8011	Index nicht vorhanden
COMMUNICATION_ERROR	16#3001	16#8012	SubIndex nicht vorhanden
COMMUNICATION_ERROR	16#3001	16#8023	Zugriff verweigert, BlockMode wird durch das Device nicht unterstützt
INPUT_INVALID_ERROR	16#3002	16#0000	"BlockMode"-Eingang ist ungültig
INPUT_INVALID_ERROR	16#3002	16#0001	"ConfigFileName" ist ungültig
INPUT_INVALID_ERROR	16#3002	16#0002	"BackupFileName" ist ungültig
ACCESS_ERROR	16#3003	16#0000	"Configuration file" konnte nicht gelesen werden
ACCESS_ERROR	16#3003	16#0001	"Configuration file" konnte nicht geöffnet werden
ACCESS_ERROR	16#3003	16#0002	Das Format des "Configuration file" ist ungültig
ACCESS_ERROR	16#3003	16#0003	"Backup file" konnte nicht geöffnet werden
ACCESS_ERROR	16#3003	16#0004	Schreiben/Lesen des "Backup file" nicht möglich

Tab. 5-145: Fehlercodes IL_PNIOIoLinkBackupDevice

5.7.19 IL_PNIOIoLinkRestoreDevice

Kurzbeschreibung Der Funktionsbaustein **IL_PNIOIoLinkRestoreDevice** dient zum Wiederherstellen der Parameter eines IOL-Geräts aus der Sicherungsdatei über den Profinet-IO-Link-Gateway.

Zielsystem	Bibliothek/Ordner
MLC	RIL_ProfinetIO/ POU's/ IO-Link
MTX	RIL_ProfinetIO/ POU's/ IO-Link

Tab. 5-146: Referenztablelle

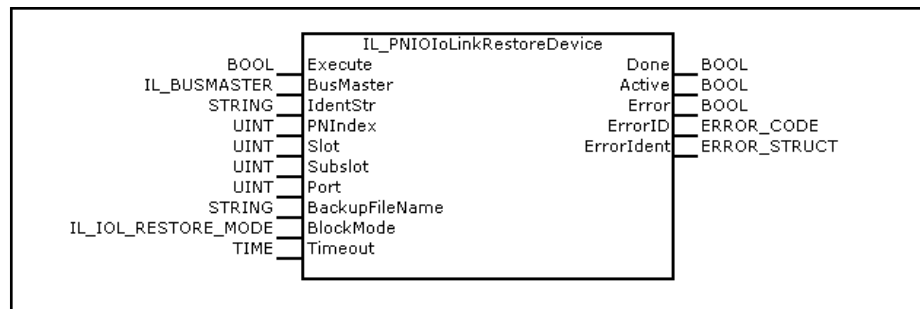


Abb. 5-58: Schnittstellenbeschreibung des Funktionsbausteins **IL_PNIOIoLinkRestoreDevice**

	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Controllers in der Reihenfolge des Konfigurationsbaums. Ist nur ein Profinet I/O-Controller vorhanden, besitzt dieser immer den Wert IL_BUSMASTER_0. Siehe auch IL_BUSMASTER auf Seite 189
	IdentStr	STRING	Identifikationsstring des Profinet-Geräts
	PNIndex	UINT	255 – Regulärer Indexaufruf 0...254 - Indexumlenkung (cap)
	Slot	UINT	Adressierung des Modulslots. Der Slot adressiert ein logisches oder physikalisches Modul (beispielsweise ein E/A Modul)
	SubSlot	UINT	Adressierung des Submoduls. Ein Modul kann sich aus mehreren logischen oder physikalischen Submodulen zusammensetzen. Es ist bei jedem Modul (Slot) mindestens das Submodul mit der Nummer 1 vorhanden.
	Port	UINT	1..63: IO-Link, Portnummer
	BackupFileName	STRING	Sicherungsdatei von Funktionsbaustein erstellt/überschrieben.
	BlockMode	IL_IOL_RESTORE_MODE	Aufzählung für den IOL-Blockparametermodus. Siehe IL_IOL_RESTORE_MODE, auf Seite 142
	Timeout	TIME	Timeout-Überwachung des Funktionsbausteins
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibt die Diagnose im Fehlerfall (aus RIL_CommonTypes)
	ErrorIdent	ERROR_STRUCT	Detaildiagnose (Fehlertabelle PROFINETIO_TABLE)

Tab. 5-147: E/A-Schnittstelle des Funktionsbausteins **IL_PNIOIoLinkRestoreDevice**

Min./max. und Default-Werte: In der folgenden Tabelle sind die Min./Max.- und Default-Werte der Funktionsbausteineingänge aufgelistet.

Name	Typ	Min. Wert	Max. Wert	Defaultwert	Anwendung
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an „Execute“
IdentStr	STRING	---	---	---	Steigende Flanke an „Execute“
PNIndex	UINT	16#0	16#FFFF	16#FF	Steigende Flanke an „Execute“
Slot	UINT	16#0	16#7FFF	16#0	Steigende Flanke an „Execute“
SubSlot	UINT	16#1	16#8FFF	16#1	Steigende Flanke an „Execute“
Port	UINT	16#0	16#FF	16#0	Steigende Flanke an „Execute“
BackupFileName	STRING	---	---	---	Steigende Flanke an „Execute“
BlockMode	IL_IOL_RESTORE_MODE	BLOCK_CHECK	NO_BLOCK_NO_CHECK	BLOCK_CHECK	Steigende Flanke an „Execute“
Timeout	TIME	T#0ms	---	T#2s	Steigende Flanke an „Execute“

Tab. 5-148: Min.-, Max.- und Default-Werte der IL_PNIOIoLinkRestoreDevice-Eingänge

Funktionsbeschreibung

Der Funktionsbaustein **IL_PNIOIoLinkRestoreDevice** dient zum Wiederherstellen der Parameter eines IOL-Geräts aus der Sicherungsdatei über den Profinet-IO-Link-Gateway. Der Funktionsbaustein liest die Parameter aus der Sicherungsdatei des IOL-Geräts, die im CF-Speicher des Controllers abliegen. Wie in "IL_IOL_RESTORE_MODE" festgelegt, können die Parameter in drei verschiedenen Modi wiederhergestellt werden.

1. **BLOCK_CHECK:** Herunterladen von Parametern mit Parameter-Blocking und mit Gültigkeitsüberprüfung. Vor dem Wiederherstellen der Daten, führen sie den Befehl "ParamDownloadStart" aus. Nach dem Wiederherstellen der Daten, führen sie den Befehl "ParamDownloadStore" aus. In diesem Fall ist das Schreiben von Parametern aus anderen Quellen gesperrt und der Befehl zur Gültigkeitsprüfung wird nach dem Schreiben des letzten Parameters gegeben.
2. **NO_BLOCK_CHECK:** Herunterladen von Parametern ohne Parameter-Blocking und mit Online-Gültigkeitsprüfung. Nach dem Wiederherstellen der Daten, wird der Befehl "ParamDownloadStore" ausgeführt, der die Validitätsprüfung nach dem letzten Parameterschreiben auslöst.
3. **NO_BLOCK_NO_CHECK:** Herunterladen von Parametern ohne Parameter-Blocking und Gültigkeitsprüfung. Nur einzelne Parameter werden geschrieben. Die Gültigkeitsprüfung des Parameters wird intern vom Gerät bei jedem Parameterschreiben durchgeführt.

Die Variable am Eingang "BackupFileName" sollte nur aus dem Dateinamen ohne Erweiterung bestehen.

Bei steigender Flanke an "Execute" liest der Funktionsbaustein alle Parameter aus der Sicherungsdatei aus und schreibt die Parameterwerte mittels des gewählten Wiederherstellungsmoduls auf das Gerät.

Beispielprogramm

In diesem Beispielprojekt wird das Wiederherstellen der IOL-Geräteparameter von der - in der CF-Karte im IndraControl-Gerät gespeicherten - Sicherungsdatei beschrieben.

Definieren Sie den Wiederherstellungsmodus in "enBlockMode". Legen Sie den Namen der Sicherungsdatei in "strFileName" fest. Geben Sie den Dateinamen ohne Erweiterung aus, z.B. "Backupfile". Bei steigender Flanke am "bExecute"-Eingang wird der Vorgang ausgeführt, indem die - in der Sicherungsdatei vorhandenen Parameter - in dem in "uiPort" festgelegten Gerät wiederhergestellt werden.

Deklaration

```
PROGRAM ProgIL_PNIOIoLinkRestoreDevice
VAR
  fbIL_PNIOIoLinkRestoreDevice: IL_PNIOIoLinkRestoreDevice; // FB Instance

  bExecute:      BOOL; //Function block is running when TRUE
  uiIndex:       UINT := 255; // PN Index
  strIdentString: STRING := 'iolballuff'; //Station name of PN BK
  uiSlot:        UINT := 16#01; //Addressing of the module slot.
                                //The slot addresses a logical or a physical module
  uiSubSlot:     UINT := 16#01; //Addressing of the submodule.
  uiPort:        UINT := 16#06; //IO-Link Port Number
  strFileName:   STRING:= 'BackupFile'; // Backup file name
  enBlockMode:   IL_IOL_RESTORE_MODE := IL_IOL_RESTORE_MODE.BLOCK_CHECK; // Backup Mode
  tTimeout:      TIME := T#3S; //Timeout monitoring of the function block
  bDone:         BOOL; //Function Block is completed its task
  bActive:       BOOL; //Function Block is working on its actual task
  bError:        BOOL; //Indicates an error
  enErrorID:     RIL_COMMONTYPES.ERROR_CODE; //Class of error
  enErrorIdent:  RIL_COMMONTYPES.ERROR_STRUCT; //Detailed information about error
END_VAR
```

Implementierung

```
fbIL_PNIOIoLinkRestoreDevice(
  Execute:= bExecute,
  Done=> bDone,
  Active=> bActive,
  Error=> bError,
  ErrorID=> enErrorID,
  ErrorIdent=> enErrorIdent,
  BusMaster:= IL_BUSMASTER.IL_BUSMASTER_0,
  IdentStr:= strIdentString,
  PNIndex:= uiIndex,
  Slot:= uiSlot,
  SubSlot:= uiSubSlot,
  Port:= uiPort,
  BackUpFileName:= strFileName,
  BlockMode:= enBlockMode,
  Timeout:= tTimeout);
```

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle PROFINETIO_TABLE (ERROR_TABLE = 16#0200).

Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Bemerkungen
INPUT_RANGE_ERROR	16#0001	16#0002	IdentStr ist ungültig
COMMUNICATION_ERROR	16#0003	PROFINET IO Controller Diagnosis - Allgemeine Fehlercodes , Seite 196 - Spezielle Fehlercodes, Profinet IO-Controller, Seite 198	

ErrorID	Additional1	Additional2	Bemerkungen
COMMUNICATION_ERROR	16#0004	PROFINET IO Status	Spezielle Fehlercodes, Profinet IO-Status , Seite 199
SYSTEM_ERROR	16#0101	internal use	Controller nicht in der Konfigurationstabelle gefunden
COMMUNICATION_ERROR	16#0104	16#0000	Zeitüberschreitung
COMMUNICATION_ERROR	16#3001	16#8011	Index nicht vorhanden
COMMUNICATION_ERROR	16#3001	16#8012	SubIndex nicht vorhanden
COMMUNICATION_ERROR	16#3001	16#8023	Zugriff verweigert/ DataStorage gesperrt/ BlockMode wird durch das Device nicht unterstützt
INPUT_INVALID_ERROR	16#3002	16#0000	"BlockMode"-Eingang ist ungültig
INPUT_INVALID_ERROR	16#3002	16#0002	"BackupFileName" ist ungültig
ACCESS_ERROR	16#3003	16#0002	"Backup file" Format ist ungültig
ACCESS_ERROR	16#3003	16#0003	"Backup file" konnte nicht geöffnet werden
ACCESS_ERROR	16#3003	16#0004	"Backup file" schreiben/lesen nicht möglich

Tab. 5-149: Error codes IL_PNIOIoLinkBackupDevice

5.8 Datentypen (Data Types)

5.8.1 IL_BUSMASTER

Kurzbeschreibung

Der Aufzählungstyp IL_BUSMASTER aus der RIL_FieldbusTypes.library dient dazu, den jeweiligen PROFINET IO-Controller auszuwählen. In einem MLC-System können neben der onboard Realtime-Ethernet-Schnittstelle über Funktionsmodule weitere Controller vorhanden sein.

Die PROFINET IO-Controller werden anhand ihrer aufsteigenden Reihenfolge in der Konfiguration unterschieden.

Die Indizierung der Controller geschieht dabei in der Reihenfolge

- Onboard-Master,
- Funktionsmodul 1,
- Funktionsmodul 2,
- Funktionsmodul 3,
- Funktionsmodul 4.



D.h. ist beispielsweise der Onboard-Master kein PROFINET IO-Controller und ist auf dem FM1 ein PROFINET IO-Controller konfiguriert, so erhält der Controller auf dem FM1 den Wert IL_BUSMASTER_0.

Element	Wert	Beschreibung
IL_BUSMASTER_0	0	Erster PN-Controller
IL_BUSMASTER_1	1	Zweiter PN-Controller
IL_BUSMASTER_2	2	Dritter PN-Controller
IL_BUSMASTER_3	3	Vierter PN-Controller
IL_BUSMASTER_4	4	Fünfter PN-Controller

Tab. 5-150: Aufzählungstyp IL_BUSMASTER

5.8.2 IL_FBUS_COMMUNICATION_STATE

Kurzbeschreibung Der Aufzählungstyp **IL_FBUS_COMMUNICATION_STATE** dient dazu, den Status der Feldbus-Kommunikation darzustellen.

Element	Wert	Beschreibung
COMM_STATE_NOT_CONFIGURED	16#00000001	Kommunikation ist nicht konfiguriert
COMM_STATE_STOP	16#00000002	Kommunikation befindet sich im Stopp-Zustand
COMM_STATE_IDLE	16#00000003	Kommunikation befindet sich im Idle-Zustand
COMM_STATE_OPERATE	16#00000004	Kommunikation im zyklischen Betrieb

Tab. 5-151: Aufzählungstyp **IL_FBUS_COMMUNICATION_STATE**

5.8.3 IL_FBUS_SLAVE_STATE

Der Aufzählungstyp **IL_FBUS_SLAVE_STATE** dient dazu, den Status der Kommunikationsbeziehungen zu den Slaves darzustellen, d. h. der Status zeigt an, ob der Feldbus-Master eine aktive Verbindung zu allen konfigurierten Slaves besitzt.

Element	Wert	Beschreibung
SLAVE_STATE_OK	16#00000001	Kommunikation zu allen aktiven Slaves vorhanden
SLAVE_STATE_FAILED	16#00000002	Kommunikationsfehler mindestens zu einem Slave

Tab. 5-152: Aufzählungstyp **IL_FBUS_SLAVE_STATE**

5.8.4 IL_PNIO_CONTROLLER_STATE

Kurzbeschreibung Die Datenstruktur **IL_PNIO_CONTROLLER_STATE** beinhaltet Diagnoseinformationen über

- den Zustand des Kommunikationsstacks,
- den Zustand der Buskommunikation,
- Fehlerzähler,
- einen Überblick über konfigurierte, aktive und eine Diagnose meldende Slaves.

Strukturelement	Typ	Beschreibung
CommunicationCOS	DWORD	Status des Kommunikationsstacks, siehe unten
CommunicationState	IL_FBUS_COMMUNICATION_STATE	Status der Kommunikation Siehe IL_FBUS_COMMUNICATION_STATE , Seite 190.
Version	UINT	Version der Datenstruktur der Diagnoseinfo
Watchdog	UINT	Watchdog Timeout zwischen PLC und Feldbusanbindung in ms
ErrorCount	UDINT	Fehlerzähler. Zählt die Anzahl der aufgetretenen Kommunikationsfehler (sowohl interne Fehler als auch Kommunikationsfehler)
SlaveState	IL_FBUS_SLAVE_STATE	Zeigt an, ob der PROFINET IO Controller eine aktive Verbindung zu allen konfigurierten Slaves besitzt. Siehe IL_FBUS_SLAVE_STATE , Seite 190.
NumOfConfigSlaves	UDINT	Anzahl an konfigurierten Slaves

Strukturelement	Typ	Beschreibung
NumOfActiveSlaves	UDINT	Anzahl aktiver Slaves, d.h. Slaves, zu denen der Master eine Kommunikationsverbindung aufgebaut hat.
NumOfDiagSlaves	UDINT	Anzahl an Slaves mit aktiver Diagnose

Tab. 5-153: Datenstruktur IL_PNIO_CONTROLLER_STATE

Strukturelement CommunicationCOS

Bit-Nr.	Bezeichnung	Beschreibung
0	COMM_COS_READY	Protokollstack ist gestartet und wartet auf Konfiguration
1	COMM_COS_RUN	Protokollstack ist konfiguriert und wartet auf Busverbindung
2	COMM_COS_BUS_ON	Protokollstack darf auf dem Bus kommunizieren
3	COMM_COS_CONFIG_LOCKED	Buskonfiguration ist verriegelt und kann nicht verändert werden
4	COMM_COS_CONFIG_NEW	Neue Buskonfiguration ist vorhanden, aber noch nicht initialisiert
5	COMM_COS_RESTART_REQUIRED	Neustart der Kommunikation notwendig
6	COMM_COS_RESTART_REQUIRED_ENABLE	Neustart der Kommunikation möglich
		übrige mit FALSE belegt

Im Ausgang CommunicationCOS ist der Status des Kommunikationsstacks binär kodiert.

Die Bits sind als Defines in der Library enthalten (PNIO_Defines).

5.8.5 IL_PNIO_PORT_DATA

Kurzbeschreibung Die Datenstruktur **IL_PNIO_PORT_DATA** enthält Informationen miteinander verbundener Ports benachbarter Profinet IO-Devices.

Strukturelement	Wert	Beschreibung
Deviceldx	UINT	interne Verwendung
PeerPortIdx	UINT	interne Verwendung
OwnPortIDStr	STRING(15)	Name des eigenen Ports
PeerPortIDStr	STRING(15)	Name des entfernten Ports, der mit dem eigenen Port verbunden ist (Port-Name des Nachbarn)
PeerChassisIDStr	STRING(255)	Name des entfernten Profinet IO-Device, das mit dem eigenen Profinet IO-Device verbunden ist (Device-Name des Nachbarn)
PeerMACAddress	ARRAY[0..5] OF BYTE	MAC-Adresse des entfernten Profinet IO-Device

Tab. 5-154: Struktur IL_PNIO_PORT_DATA

5.8.6 IL_PNIO_REMOTE_DEVICE_INFO

Kurzbeschreibung Die Datenstruktur enthält eine Zusammenfassung der PROFINET-Status- und Diagnoseinformationen eines Remote-Devices.

Strukturelement	Typ	Beschreibung
DeviceName	STRING(255)	StationName ("IdentStr")
DeviceID	UINT	Device ID
VendorID	UINT	Vendor ID
DeviceState	IL_PNIO_REMOTE_DEVICE_STATE	Gerätezustand Kap. 5.8.7 "IL_PNIO_REMOTE_DEVICE_STATE" auf Seite 192
DiagInfo	PNIO_DIAGINFO	Detaillierte Gerätediagnose Kap. 5.8.11 "PNIO_DIAGINFO" auf Seite 194

Tab. 5-155: Struktur IL_PNIO_REMOTE_DEVICE_INFO

5.8.7 IL_PNIO_REMOTE_DEVICE_STATE

Kurzbeschreibung Die Datenstruktur enthält eine Zusammenfassung des vereinfachten Gerätestatus eines Remote-Devices.

Strukturelement	Typ	Beschreibung
RemoteDeviceNotReachable	BIT	0.0, Gerät nicht erreichbar
RemoteDeviceNotReady	BIT	0.1, Gerät nicht bereit
RemoteDeviceConfigError	BIT	0.2, Geräte-Konfigurationsfehler
RemoteDeviceDiagnosis	BIT	0.3, Geräte-Diagnose
RemoteDeviceError	BIT	0.4, Geräte-Fehler
Reserved05	BIT	0.5
Reserved06	BIT	0.6
Reserved07	BIT	0.7
Reserved10	BIT	1.0
...	BIT	...
Reserved17	BIT	1.7

Tab. 5-156: Struktur IL_PNIO_REMOTE_DEVICE_STATE

5.8.8 IL_PNIO_TOPO_STATE

Kurzbeschreibung Die Datenstruktur IL_PNIO_TOPO_STATE enthält Angaben zur Topologie eines Profinet IO-Device.

Verglichen wird die gespeicherte Topologie mit der aktuell vorhandenen.

Strukturelement	Typ	Beschreibung
OwnPortChanged	Bit	0.0 (16#0001) eigene(s) Port(s) geändert bzw. vertauscht
NeighborDiff_1	Bit	0.1 (16#0002) ein Nachbar nicht wie erwartet
NeighborDiff_2m	Bit	0.2 (16#0004) zwei oder mehrere Nachbarn nicht wie erwartet
NeighborMiss_1	Bit	0.3 (16#0008) ein erwarteter Nachbar nicht vorhanden
NeighborMiss_2m	Bit	0.4 (16#0010) zwei oder mehr Nachbarn nicht vorhanden
ReadError	Bit	0.5 (16#0020) Lesefehler, Profinet IO-Device nicht erreichbar
Reserved06	BIT	0.6 (16#0040)

Strukturelement	Typ	Beschreibung
...		
Reserved17	BIT	1.7 (16#8000)

Tab. 5-157: Struktur IL_PNIO_TOPO_STATE

5.8.9 IL_PNIO_TOPO_STATUS

Kurzbeschreibung Die Datenstruktur **IL_PNIO_TOPO_STATUS** enthält Angaben zum Status eines Profinet IO-Device.

Strukturelement	Wert	Beschreibung
Device	PNIO_LiveList	enthält die Information zu einem Profinet IO-Device Kap. 5.8.13 "PNIO_LIVELIST" auf Seite 194
TopoState	IL_PNIO_TOPO_STATE	enthält Angaben zur Umgebung eines Profinet IO-Device (Differenz zwischen gespeicherter Topologie und aktueller) Kap. 5.8.8 "IL_PNIO_TOPO_STATE" auf Seite 192
NoOfPorts	USINT	Anzahl der Ports
NoOfNeighborsStored	USINT	Anzahl der gespeicherten Nachbarn (PNTopo_n.xml)
NoOfNeighborsActual	USINT	Anzahl der aktuell gelesenen Nachbarn
NoOfNeighborsDiff	USINT	Anzahl der Nachbarn, bei denen es Unterschiede zwischen der gespeicherten und der gelesenen Topologie gibt

Tab. 5-158: Struktur IL_PNIO_TOPO_STATUS

5.8.10 PNIO_CHANNELERROR

Kurzbeschreibung Der Aufzählungstyp **PNIO_CHANNELERROR** bildet die **ChannelErrorType** gemäß der IEC 61158-6-10 V23 6.2.7.2.

Zusätzlich können noch herstellerspezifische Werte vorkommen.

Element	Wert	Beschreibung
ChanErr_Unknown	0	Unbekannter Fehler
ChanErr_ShortCircuit	1	Kurzschluss
ChanErr_Undervoltage	2	Zu geringe Spannung
ChanErr_Overvoltage	3	Zu hohe Spannung
ChanErr_Overtemperature	4	Übertemperatur
ChanErr_Overload	5	Überlast
ChanErr_LineBreak	6	Kabelbruch
ChanErr_UpperLimit	7	Wertebereich Überschreitung
ChanErr_LowerLimit	8	Wertebereich Unterschreitung
ChanErr_Error	9	Allg. Fehler
ChanErr_SimulationActive	10	Simulationsbetrieb

Tab. 5-159: Aufzählungstyp PNIO_CHANNELERROR

5.8.11 PNIO_DIAGINFO

Kurzbeschreibung Die Datenstruktur enthält eine Zusammenfassung der PROFINET-Status- und Diagnoseinformationen. Sie besteht aus der Art (DiagType), dem Ort (Slot, Subslot, Channel) und ggf. einem Fehlertyp des Ereignisses.

Strukturelement	Typ	Beschreibung
DiagType	PNIO_DIAGTYPE	Art des Ereignisses
Slot	UINT	Slot auf dem ein Ereignis aufgetreten ist.
Subslot	UINT	Subslot auf dem ein Ereignis aufgetreten ist.
Channel	UINT	Kanal auf dem ein Ereignis aufgetreten ist, 16#FFFF falls kein Kanal vorhanden. Unter Kanal ist hierbei der einzelne IO-Punkt zu verstehen, der eine Diagnose (z.B. Kabelbruch) liefern kann.
ChannelErrorType	PNIO_CHANNELERROR	Ereignis bzw. Diagnose
ExtErrorType	UINT	Erweiterter, vendorspezifischer Fehlertyp
AddDiagInfo	UDINT	Zusätzliche Diagnose-Information

Tab. 5-160: Struktur PNIO_DIAGINFO

5.8.12 PNIO_DIAGTYPE

Kurzbeschreibung Der Aufzählungstyp PNIO_DIAGTYPE dient dazu, die dignostizierbaren Ereignisse zu kennzeichnen.



Bei PROFINET-Devices dürfen überzählige Module vorhanden sein. Diese Module erzeugen keine Diagnosemeldung.

Element	Wert	Beschreibung
NO_DIAGNOSIS	0	Kein Ereignis/Diagnose
DIAGNOSIS_APPEARS	1	Diagnose liegt vor. PNIO_CHANNELERROR auswerten
MODULE_MISSING	2	Das durch Slot/Subslot bezeichnete Modul ist konfiguriert aber am Device nicht vorhanden
MODULE_WRONG	3	Das durch Slot/Subslot bezeichnete Modul stimmt nicht mit dem konfigurierten Modul überein
SUBMODULE_STATE	4	Das durch Slot/Subslot bezeichnete Modul liefert den in ChannelErrorType abgelegten Status

Tab. 5-161: Aufzählungstyp PNIO_DIAGTYPE

5.8.13 PNIO_LIVELIST

Kurzbeschreibung Die Datenstruktur enthält die Information über ein Profinet IO-Device.

Strukturelement	Typ	Beschreibung
IPAddress	STRING(16)	IP Adresse des Profinet IO-Devices
SubNetMask	STRING(16)	Subnetmaske des Profinet IO-Devices
Gateway	STRING(16)	Gatewayadresse des Profinet IO-Devices
DeviceId	UINT	Device ID des Profinet IO-Devices
VendorId	UINT	Vendor ID des Profinet IO-Devices

Strukturelement	Typ	Beschreibung
MacAddress	ARRAY [0..5] OF BYTE	MAC Adresse des Profinet IO-Devices
DeviceName	STRING(240)	StationsName des Profinet IO-Devices
DeviceTypeName	STRING(240)	TypName des Profinet IO-Devices
DeviceRole	WORD	Rolle des Profinet IO-Devices, siehe unten

Tab. 5-162: Datenstruktur PNIO_LIVELIST

Strukturelement DeviceRole

Bit-Nr.	Bezeichnung	Beschreibung
0	PNIO Device	Das Gerät realisiert die angegebene Funktion
1	PNIO Controller	
2	PNIO Multidevice	
3	PNIO Supervisor	
4...7	reserviert	---

Der Wert von "DeviceRole" ist binär kodiert.

5.9 PROFINETIO_TABLE 16#200

5.9.1 Allgemeines

PROFINETIO_TABLE - ERROR_TABLE, Tabellennummer 16#0200

Typdefinition

```

TYPE ERROR_STRUCT :
  STRUCT
    Table      :ERROR_TABLE;
    Additional1 :DWORD;
    Additional2 :DWORD;
  END STRUCT
END_TYPE

```

Die Tabelle enthält die Fehlernummern, die von den Funktionsbausteinen nachstehender Bibliotheken ausgegeben werden:

- RIL_ProfiNetIO - Bibliothek für Profinet IO-Controller (BusMaster),
- RIL_ProfiNetIODevice - Bibliothek für Profinet IO-Devices (BusSlave).

Die Bausteine selbst geben nachstehende Fehlermeldungen aus:

ErrorID	Additional1	Additional2	Bemerkung
INPUT_RANGE_ERROR	16#0001	xxx	Funktionsbausteinabhängig
COMMUNICATION_ERROR	16#0003	<i>Profinet IO Controller Diagnose</i> • Allgemeine Fehlercodes , Seite 196, • Spezielle Fehlercodes, Profinet IO-Controller, Seite 198.	
COMMUNICATION_ERROR	16#0004	<i>Profinet IO Status</i> • Spezielle Fehlercodes, Profinet IO-Status , Seite 199.	
STATE_MACHINE_ERROR	16#0005	16#0000	Keine Übereinstimmung für Antwort-Header
INPUT_INVALID_ERROR	16#0005	16#000x	IMxAdr ist nicht gesetzt (x - I&M-Nummer)
INPUT_INVALID_ERROR	16#0006	16#0000	Ungültiger RW-Mode

ErrorID	Additional1	Additional2	Bemerkung
DEVICE_ERROR	16#0007	16#0000	Ungültige I&M-Version (die Device-I&M-Version muss 1.1 sein)
SYSTEM_ERROR	16#0101	internal use	Controller nicht in der Konfigurationstabelle
COMMUNICATION_ERROR	16#0104	0	Zeitüberschreitung
STATE_MACHINE_ERROR	16#0105	xxx	Fehler in der Zustandsmaschine
INPUT_RANGE_ERROR	16#0106	0	Gerät 'IdentStr' nicht konfiguriert
COMMUNICATION_ERROR	16#3000	<i>Profinet IO IO-Link Master (IOL-M-Fehler)</i> Spezielle Fehlercodes, Profinet IO IO-Link Master , Seite 209	
COMMUNICATION_ERROR	16#3001	<i>Profinet IO IO-Link Device (anwendungsbezogene Fehler)</i> Spezielle Fehlercodes, Profinet IO IO-Link Device (anwendungsbezogene Fehler) , Seite 209	
		Profinet IO Backup/ Restore	
INPUT_INVALID_ERROR	16#3002	16#0000	"BlockMode"-Eingang ist ungültig
INPUT_INVALID_ERROR	16#3002	16#0001	"ConfigFileName" ist ungültig
INPUT_INVALID_ERROR	16#3002	16#0002	"BackupFileName" ist ungültig
ACCESS_ERROR	16#3003	16#0000	"Configuration file" konnte nicht gelesen werden
ACCESS_ERROR	16#3003	16#0001	"Configuration file" konnte nicht geöffnet werden
ACCESS_ERROR	16#3003	16#0002	"Configuration file" / "Backup file" Format ist ungültig
ACCESS_ERROR	16#3003	16#0003	"Backup file" konnte nicht geöffnet werden
ACCESS_ERROR	16#3003	16#0004	"Backup file" schreiben/lesen nicht möglich

Tab. 5-163: Fehlercodes Profinet IO (Controller), Auswahl

5.9.2 Allgemeine Fehlercodes

Die dargestellten Fehlercodes gelten für alle BusMaster und BusSlaves, sie sind zusätzlich feldbusunabhängig.

Die Fehlercodes werden als **ERROR_STRUCT, Additional2**

bei **ERROR_STRUCT, Additional1= 16#0003** gemeldet.

Additional2	
16#800A0001	Ungültiger Zeiger (NULL) an Treiber übergeben
16#800A0002	Keine Steckkarte mit dem angegebenen Namen/Index verfügbar
16#800A0003	Kein Kanal mit dem angegebenen Index verfügbar
16#800A0004	Ungültiges Handel an Treiber übergeben
16#800A0005	Ungültiger Parameter
16#800A0006	Ungültiger Befehl
16#800A0007	Ungültige Puffergröße
16#800A0008	Ungültige Zugriffsgröße
16#800A0009	Funktion fehlgeschlagen

Additional2	
16#800A000A	Datei konnte nicht geöffnet werden
16#800A000B	Dateigröße ist gleich Null
16#800A000C	Unzureichende Speicherkapazität um Datei zu laden
16#800A000D	Vergleich der Dateiprüfsummen fehlgeschlagen
16#800A000E	Fehler beim Lesen von Datei
16#800A000F	Ungültiger Dateityp
16#800A0010	Ungültiger Dateiname
16#800A0011	Treiberfunktion nicht vorhanden
16#800A0012	Angegebener Puffer ist zu klein
16#800A0013	Speicher kann nicht abgebildet werden
16#800A0014	Keine weiteren Einträge vorhanden
16#800B0001	Treiber ist nicht initialisiert
16#800B0002	Init-Statusfehler des Treibers
16#800B0003	Treiber Statuslesefehler
16#800B0004	Befehl ist aktiv auf Gerät
16#800B0005	Allgemeiner Fehler bei Download
16#800B0006	Falsche Treiberversion
16#800B0030	CIFx Treiber nicht in Betrieb
16#800B0031	Gerät konnte nicht initialisiert werden
16#800B0032	Kanal konnte nicht initialisiert werden (xOpenChannel wurde nicht aufgerufen)
16#800B0033	IOControl Aufruf fehlgeschlagen
16#800B0034	Treiber konnte nicht geöffnet werden
16#800C0010	Auf den Dual-Port Speicher konnte nicht zugegriffen werden (Speicherkarte nicht gefunden)
16#800C0011	Gerät nicht bereit (Ready-Flag fehlgeschlagen)
16#800C0012	Gerät nicht betriebsbereit (Running-Flag fehlgeschlagen)
16#800C0013	Watchdog-Test fehlgeschlagen
16#800C0015	Handshake-Flag Fehler
16#800C0016	Mailbox für gesendete Mails ist voll
16#800C0017	Timeout für Senden der Datenpakete
16#800C0018	Timeout für Empfangen der Datenpakete
16#800C0019	Kein Datenpaket verfügbar
16#800C001A	Zu wenig Speicherplatz in Mailbox
16#800C0020	Befehl-Timeout zurücksetzen
16#800C0021	COM-Flag nicht gesetzt
16#800C0022	EA-Datenaustausch fehlgeschlagen
16#800C0023	EA-Datenaustausch Timeout

Additional2	
16#800C0024	Unbekannter EA-Datenaustauschmodus
16#800C0025	Gerätefunktion fehlgeschlagen
16#800C0026	DPM-Größe size unterscheidet sich von Konfiguration
16#800C0027	Unbekannter Statusmodus
16#800C0028	Ausgangsport wird bereits verwendet
16#800C0029	Timeout Konfigurationsverriegelung
16#800C002A	Konfigurationsentriegelung Timeout
16#800C002B	HOST-Status Timeout setzen
16#800C002C	HOST-Status Timeout löschen
16#800C002D	Timeout während Kanalinitialisierung
16#800C002E	Bus EIN Timeout setzen
16#800C002F	Bus AUS Timeout setzen

Tab. 5-164: Allgemeine Fehlercodes

5.9.3 Spezielle Fehlercodes, Profinet IO-Controller

Die dargestellten Fehlercodes gelten für Profinet IO-Controller.

Die Fehlercodes werden als **ERROR_STRUCT, Additional2**,
bei **ERROR_STRUCT, Additional1= 16#0003**, gemeldet.

Additional2	
16#C000001A	Azyklischer Kanal bereits belegt Es kann nur ein azyklischer Dienst (DCP-Dienste oder Read-Implicit-Dienst) ausgeführt werden, d.h. eine neue azyklische Anfrage kann erst dann gestartet werden, wenn die zuvor gestartete Anfrage beendet ist. Ist ein Gerät nicht erreichbar, ist der azyklische Kanal erst nach 7 sec wieder verfügbar (interner Timeout). Betroffene Funktionsbausteine: alle FBs bzgl. Acyclic Communication, Backup&Restore, Device Replacement, IO-Link, IM-Data, sowie der FB IL_PNIORemoteDeviceStateDetails.
16#C0000142	Verbindungs-Timeout
16#C0000145	Es ist kein Ethernetkabel gesteckt.
16#C0000180	Der Feldbus ist im Zustand OFF.
16#C00C0038	Das angeforderte EA-Gerät hat nicht auf die DCP-Anforderung geantwortet. Das Gerät ist unter Umständen nicht vorhanden.
16#C00C0056	Dienst nicht verfügbar; Device ist nicht im IO-Datenaustausch.
16#C00C0060	Der azyklische Dienst wurde von der Gegenstation (Device) negativ quittiert.
16#C00C0061	Der azyklische Dienst schlug fehl. Die RPC-Schicht hat einen Fehler im Empfangspaket entdeckt.
16#C00C0062	Der azyklische Dienst schlug fehl. Es ist ein interner Fehler aufgetreten.
16#C0140041	Im Devicenamen sind nicht erlaubte Zeichen

Tab. 5-165: Spezielle Fehlercodes des PROFINET IO-Controllers

5.9.4 Spezielle Fehlercodes, Profinet IO-Status

Die dargestellten Fehlercodes sind die speziellen Fehlercodes des Profinet IO-Status.

Diese Codierungen sind der IEC 61158-6-10 Kap 6.2.5, Coding section related to PNIOStatus entnommen.

Die Fehlercodes werden als **ERROR_STRUCT**, **Additional2** bei **ERROR_STRUCT**, **Additional1= 16#0004** gemeldet.

Codierung von Profinet IO-Status

Additional2-Maske	Bedeutung	
16#FF000000	ErrorCode	Fehlercode für negative Responses
16#00FF0000	ErrorDecode	Auswahl der weiteren Decodierung
16#0000FF00	ErrorCode1	Fehlercodes 1
16#000000FF	ErrorCode2	Fehlercodes 2

Beispiel

Der Funktionsbaustein IL_PNIOReadRecord meldet den Fehler ErrorIdent.Additional1 = 16#00000004 (≙ Profinet IO Status) und ErrorIdent.Additional2 = 16#DE80B000. Dieser Additional2-Fehlercode wird dann wie folgt ausgewertet:

- 16#DExxxxxx : Es liegt ein Fehler bei einem Lesezugriff vor, siehe "[ErrorCode](#)" auf Seite 199
- 16#xx80xxxx : Fehler bei einem Read/Write-Dienst "PNIORW", siehe "[ErrorDecode](#)" auf Seite 200
- 16#xxxxB0xx : Bit 4-7 = 11 (dezimal) → Bit 0-3 = 0 (dezimal) → "invalid index", siehe "[ErrorCode1/2 für ErrorCode PNIORW](#)" auf Seite 200, d.h. der Index beim Lesezugriff wurde vom PN-Device nicht akzeptiert; am Funktionsbaustein ist zu überprüfen, ob der Eingang "Index" richtig gesetzt wurde
- 16#xxxxxx00 : ErrorCode2 hat in diesem Beispiel keine Bedeutung

ErrorCode Auswahl der Werte für **ErrorCode**. Es sind weitere Werte definiert, die in diesem Zusammenhang jedoch nicht auftreten werden.

ErrorCode	Bedeutung	
16#20 - 16#3F	Herstellerspezifisch	Durch Maschinenhersteller festgelegt
16#81	PNIO	Wird für alle Fehler verwendet, die nicht durch andere Domänen abgedeckt sind
16#CF	RTA error	In RTA Fehler PDUs verwendet. Verwendet in Alarm Fehlertelegrammen (Low Level)
16#DA	AlarmAck	In RTA Daten PDUs verwendet. Verwendet in Alarm Datentelegrammen (High Level)
16#DB	IODConnect-Res	RPC Connect Response wird verwendet, um Fehler die beim Verarbeiten eines Connect Requests auftreten, anzuzeigen

ErrorCode	Bedeutung	
16#DC	IODRelease-Res	RPC Release Response wird verwendet, um Fehler die beim Verarbeiten eines Release Requests auftreten, anzuzeigen
16#DD	IODControlRes	RPC Control Response wird verwendet, um Fehler die beim Verarbeiten eines Control Requests auftreten, anzuzeigen
16#DE	IODReadRes	RPC Read Response wird verwendet, um Fehler die beim Verarbeiten eines Read Requests auftreten, anzuzeigen
16#DF	IODWriteRes	RPC Write Response wird verwendet, um Fehler die beim Verarbeiten einer Write Response auftreten, anzuzeigen

ErrorDecode Auswahl der Werte für **ErrorDecode**. Es sind weitere Wert definiert, die in diesem Zusammenhang jedoch nicht auftreten werden.

ErrorDecode	Bedeutung	
16#80	PNIORW	Für Read und Write Dienste, siehe " ErrorCode1/2 für ErrorCode PNIORW " auf Seite 200
16#81	PNIO	Für andere Dienste, siehe " ErrorCode1/2 für ErrorDecode PNIO " auf Seite 201
16#82	Herstellerspezifisch	Durch Maschinenhersteller festgelegt

ErrorCode1/2 für ErrorCode "PNIORW"

Bedeutung ErrorCode1 für ErrorDecode "PNIORW"

ErrorClass (dezimal) Bit7 – 4	Bedeutung	ErrorCode (dezimal) Bit3 – 0
0 to 9 (0x0 to 0x9)	Not specified	Reserviert
10 (0xA)	Application	0 = Lesefehler 1 = Schreibfehler 2 = Modulausfall 3,4,5,6 = nicht definiert 7 = beschäftigt 8 = Versionskonflikt 9 = Feature wird nicht unterstützt 10 = Benutzerspezifisch 1 15 = Benutzerspezifisch 6

ErrorClass (dezimal) Bit7 – 4	Bedeutung	ErrorCode (dezimal) Bit3 – 0
11 (0xB)	Access	0 = Ungültiger Index 1 = Schreiblängenfehler 2 = Ungültiger Slot / Sub-slot 3 = Typenkonflikt 4 = Ungültiger Bereich / API 5 = Statuskonflikt 6 = Zugang verweigert 7 = Ungültiger Bereich 8 = Ungültiger Parameter 9 = Ungültiger Typ 10 = Backup 11 = Benutzerspezifisch 7 ... 15 = Benutzerspezifisch 11
12 (0xC)	Ressource	0 = Lesebeschränkungskonflikt 1 = Schreibbeschränkungskonflikt 2 = Ressource beschäftigt 3 = Ressource nicht verfügbar 4,5,6,7 = nicht angegeben 8 = Benutzerspezifisch 12 15 = Benutzerspezifisch 19
13 to 15 (0xD to 0xF)	Benutzerspezifisch	Benutzerspezifisch

Tab. 5-166: Auswahl der Codierung von *PNIOStatus.ErrorCode*

Die Bedeutung von ErrorCode2 ist in diesem Fall anwenderspezifisch und wird nicht weiter ausgeführt.

ErrorCode1/2 für ErrorDecode
"PNIO"

Bedeutung ErrorCode1/2 für ErrorDecode "PNIO"

ErrorC-ode1	Beschreibung	ErrorC-ode2	Beschreibung
16#01	Connect Parameter Error. ARBlockReq fehlerhaft	16#00 16#0D	- Fehler in einem der Parametersätze
16#02	Connect Parameter Error. IOCRBlockReq fehlerhaft	16#00 16#1C	- Fehler in einem der Parametersätze
16#03	Connect Parameter Error. ExpectedSubmoduleBlockReq fehlerhaft	16#00 16#10	- Fehler in einem der Parametersätze

16#04	Connect Parameter Error. AlarmCRBlockReq fehlerhaft	16#00 16#0F	-	Fehler in einem der Parametersätze
16#05	Connect Parameter Error. PrmServerBlockReq fehlerhaft	16#00 16#08	-	Fehler in einem der Parametersätze
16#06	Connect Parameter Error. MCRBlockReq fehlerhaft	16#00 16#08	-	Fehler in einem der Parametersätze
16#07	Connect Parameter Error. ARRPCBlockReq fehlerhaft	16#00 16#04	-	Fehler in einem der Parametersätze
16#08	Read Write Record Parameter Error Faulty Record	16#00 16#0C	-	Fehler in einem der Parametersätze
16#09	Connect Parameter Error IRInfoBlock fehlerhaft	16#00 16#05	-	Fehler in einem der Parametersätze
16#0A	Connect Parameter Error SRInfoBlock fehlerhaft	16#00 16#05	-	Fehler in einem der Parametersätze
16#0B	Connect Parameter Error ARFSUBlock fehlerhaft	16#00 16#05	-	Fehler in einem der Parametersätze
16#14	IODControl Parameter Error. ControlBlockConnect fehlerhaft	16#00 16#09	-	Fehler in einem der Parametersätze
16#15	IODControl Parameter Error. ControlBlockPlug fehlerhaft	16#00 16#09	-	Fehler in einem der Parametersätze
16#16	IOXControl Parameter Error. ControlBlock fehlerhaft nachdem Verbindung hergestellt wurde	16#00 16#07	-	Fehler in einem der Parametersätze
16#17	IOXControl Parameter Error. ControlBlock Steckeralarm fehlerhaft	16#00 16#07	-	Fehler in einem der Parametersätze
16#18	IODControl Parameter Error. ControlBlockPrmBegin fehlerhaft	16#00 16#09	-	Fehler in einem der Parametersätze
16#19	IODControl Parameter Error SubmoduleListBlock fehlerhaft	16#00 16#07	-	Fehler in einem der Parametersätze
16#28	Release Parameter Error. ReleaseBlock fehlerhaft	16#00 16#07	-	Fehler in einem der Parametersätze

16#32	Response Parameter Error. ARBlockRes fehlerhaft	16#00 16#08	- Fehler in einem der Parametersätze
16#33	Response Parameter Error. IOCRBlockRes fehlerhaft	16#00 16#06	Fehler in einem der Parametersätze
16#34	Response Parameter Error. AlarmCRBlockRes fehlerhaft	16#00 16#06	- Fehler in einem der Parametersätze
16#35	Response Parameter Error. ModuleDiffBlock fehlerhaft	16#00 16#0D	- Fehler in einem der Parametersätze
16#36	Response Parameter Error. ARRPCBlockRe fehlerhaft	16#00 16#04	- Fehler in einem der Parametersätze
16#37	Response Parameter Error ARServerBlockRes fehlerhaft	16#00 16#05	- Fehler in einem der Parametersätze
16#3C	AlarmAck Fehlercodes	16#00 16#01	Warnungstyp nicht unterstützt Falscher Untermodulstatus
16#3D	CMDEV	16#00 16#01 16#02 16#FF	- Statuskonflikt Ressource Status maschinenspezifisch
16#3E	CMCTL	16#00 16#01 16#02	Statuskonflikt Timeout Keine Daten gesendet
16#3F	CTLDINA	16#00 16#01 16#02 16#03 16#04 16#05 16#06	Kein DCP aktiv DNS Unknown_RealStation-Name DCP No_RealStationName DCP Multiple_RealStation-Name DCP No_StationName No_IP_Addr DCP_Set_Error

16#40	CMRPC	16#00	ArgsLength ungültig
		16#01	Unbekannte Parametersätze
		16#02	IOCR fehlt
		16#03	Falsches AlarmCRBlock Ergebnis
		16#04	Keine AR Ressourcen mehr
		16#05	AR UUID unbekannt
		16#06	Statuskonflikt
		16#07	Kein Provider, Consumer oder AlarmResources mehr
		16#08	Kein Speicherplatz mehr
		16#09	PDev bereits in Besitz
		16#0A	ARset Statuskonflikt beim Herstellen der Verbindung
		16#0B	ARset Parameterkonflikt beim Herstellen der Verbindung
16#41	ALPMI	16#00	Ungültiger Status
		16#01	Falsche ACK-PDU
16#42	ALPMR	16#00	Ungültiger Status
		16#01	Falsche Notification PDU
16#43	LMPM	16#00	- Fehler bei Ethernet schalten
		16#FF	
16#44	MAC	16#00	-
		16#FF	-

16#45	RPC	16#01	CLRPC_ERR_REJECTED: EPM oder Server hat den Ruf abgelehnt
		16#02	CLRPC_ERR_FAULTED: Server war im Fehlerzustand während des Aufrufs
		16#03	CLRPC_ERR_TIMEOUT: EPM oder Server antwortet nicht
		16#04	CLRPC_ERR_IN_ARGS; Sammelaufruf oder "ndr_data" zu groß
		16#05	CLRPC_ERR_OUT_ARGS: Server Server schickt mehr als "alloc_len" zurück
		16#06	CLRPC_ERR_DECODE: Ergebnis des EPM Lookup konnte nicht decodiert werden
		16#07	CLRPC_ERR_PNIO_OUT_ARGS: Out-Args nicht "PN IO signature", zu kurz oder inkonsistent
		16#08	CLRPC_ERR_PNIO_APP_TIMEOUT: RPC Aufruf wurde beendet nach einem Timeout der RPC-Anwendung
16#46	APMR	16#00	Ungültiger Status
		16#01	LMPM meldet einen Fehler
16#47	APMS	16#00	Ungültiger Status
		16#01	LMPM meldet einen Fehler
		16#02	Timeout
16#48	CPM	16#00	Ungültiger Status
16#49	PPM	16#00	Ungültiger Status
16#4A	DCPUCS	16#00	Ungültiger Status
		16#01	LMPM meldet einen Fehler
		16#02	Timeout
16#4B	DCPUCR	16#00	Ungültiger Status
		16#01	LMPM meldet einen Fehler
16#4C	DCPMCS	16#00	Ungültiger Status
		16#01	LMPM meldet einen Fehler
16#4D	DCPMCR	16#00	Ungültiger Status
		16#01	LMPM meldet einen Fehler
16#4E	FSPM	16#00	- FAL Serviceprotokoll Maschinenfehler
		16#FF	

16#64	CTLSM	16#00 16#01	Ungültiger Status CTLSM meldet einen Fehler
16#65	CTLRDI	16#00 16#01	Ungültiger Status CTLRDI meldet einen Fehler
16#66	CTLRDR	16#00 16#01	Ungültiger Status CTLRDR meldet einen Fehler
16#67	CTLWRI	16#00 16#01	Ungültiger Status CTLWRI meldet einen Fehler
16#68	CTLWRR	16#00 16#01	Ungültiger Status CTLWRR meldet einen Fehler
16#69	CTLIO	16#00 16#01	Ungültiger Status CTLIO meldet einen Fehler
16#6A	CTLSU	16#00 16#01 16#02 16#03 16#04 16#05	Ungültiger Status AR Hinzufügen Provider oder Consumer fehlgeschlagen AR alarm-open fehlgeschlagen AR alarm-ack-send AR alarm-send AR alarm-ind
16#6B	CTLRPC	16#00 16#01	Ungültiger Status CTLRPC meldet einen Fehler
16#6C	CTLPBE	16#00 16#01	Ungültiger Status CTLPBE meldet einen Fehler
16#C8	CMSM	16#00 16#01	Ungültiger Status CMSM meldet einen Fehler
16#CA	CMRDR	16#00 16#01	Ungültiger Status CMRDR meldet einen Fehler
16#CC	CMWRR	16#00 16#01 16#02	Ungültiger Status AR ist nicht im Status Primary (Schreiben nicht erlaubt) CMWRR meldet einen Fehler
16#CD	CMIO	16#00 16#01	Ungültiger Status CMIO meldet einen Fehler

16#CE	CMSU	16#00	Ungültiger Status
		16#01	AR Hinzufügen Provider
		16#02	oder Consumer fehlgeschla-
		16#03	gen
		16#04	AR alarm-open fehlgeschla-
		16#05	gen
16#D0	CMINA		AR alarm-send
			AR alam-ack-send
			AR alarm-ind
16#D0	CMINA	16#00	Ungültiger Status
		16#01	CMINA meldet einen Fehler
16#D1	CMPBE	16#00	Ungültiger Status
		16#01	CMPBE meldet einen Feh-
16#D2	CMDMC		ler
16#D2	CMDMC	16#00	Ungültiger Status
		16#01	CMDMC meldet einen Feh-
16#D2	CMDMC		ler

16#FD	Used by RTA for protocol error (RTA_ERR_CLS_PROTOCOL)	16#01	Fehler bei der Koordinierung der Sequenznummern (RTA_ERR_CODE_SEQ)
		16#02	Instanz geschlossen (RTA_ERR_ABORT)
		16#03	AR Kein Speicherplatz mehr (RTA_ERR_ABORT)
		16#04	AR Hinzufügen Provider oder Consumer fehlgeschlagen (RTA_ERR_ABORT)
		16#05	AR Consumer DHT / WDT abgelaufen (RTA_ERR_ABORT)
		16#06	AR cmi timeout (RTA_ERR_ABORT)
		16#07	AR alarm-open fehlgeschlagen (RTA_ERR_ABORT)
		16#08	AR alarm-send.cnf(-) (RTA_ERR_ABORT)
		16#09	AR alarm-ack- send.cnf(-) (RTA_ERR_ABORT)
		16#0A	AR Alarmdaten zu lang (RTA_ERR_ABORT)
		16#0B	AR alarm.ind(err) (RTA_ERR_ABORT)
		16#0C	AR rpc-client call.cnf(-) (RTA_ERR_ABORT)
		16#0D	AR abort.req (RTA_ERR_ABORT)
		16#0E	AR Wiederdurchführung bricht bestehendes (RTA_ERR_ABORT) ab
		16#0F	AR release.ind empfangen (RTA_ERR_ABORT)
		16#10	AR Gerät deaktiviert (RTA_ERR_ABORT)
		16#11	AR entfernt (RTA_ERR_ABORT)
		16#12	AR Protokollverletzung (RTA_ERR_ABORT)
		16#13	AR Fehler bei Namensauflösung (RTA_ERR_ABORT)
		16#14	AR RPC-Bindungsfehler (RTA_ERR_ABORT)
		16#15	AR RPC-Verbindungsfehler (RTA_ERR_ABORT)
		16#16	AR RPC-Lesefehler (RTA_ERR_ABORT)
		16#17	AR RPC-Schreibfehler (RTA_ERR_ABORT)
		16#18	AR RPC-Control-Fehler (RTA_ERR_ABORT)

16#FF	Benutzerspezifisch	16#00 16#FE 16#FF	- Benutzerspezifisch Empfohlen für "Benutzerab- bruch" ohne weitere Infor- mationen

Tab. 5-167: Auswahl der Codierung von *PNIOStatus.ErrorCode*

5.9.5 Spezielle Fehlercodes, Profinet IO IO-Link Master

Die dargestellten Fehlercodes gelten für IO-Link Master.

Die Fehlercodes werden als **ERROR_STRUCT, Additional2**
bei **ERROR_STRUCT, Additional1= 16#3000** gemeldet.

Spezielle Fehler des IO-Link Master

ERROR_CODE	Additional2	
COMMUNICATION_ERROR	16#7000	IOL_CALL-Konflikt. Unerwartete Schreibanforderung statt Leseanforderung
COMMUNICATION_ERROR	16#7001	Falscher IOL_CALL. Decodierfehler
COMMUNICATION_ERROR	16#7002	Port blockiert. Port mit anderer Task beschäftigt
COMMUNICATION_ERROR	16#8000	Zeitüberschreitung, wenn die Ports IOL-Ds oder IOL-M beschäftigt sind
COMMUNICATION_ERROR	16#8001	Falscher Index. IOL_Index 32767 ... 65535
COMMUNICATION_ERROR	16#8002	Falsche Port-Adresse. Port-Adresse über dem festgelegten Maximum
COMMUNICATION_ERROR	16#8003	Falsche Portfunktion. Portfunktion wird nicht unterstützt

Tab. 5-168: Fehlercodes *PNIO_IO-Link Master*

5.9.6 Spezielle Fehlercodes, Profinet IO IO-Link Device (anwendungsbezogene Fehler)

Die dargestellten Fehlercodes gelten für Profinet IO IO-Link Device.

Die Fehlercodes werden als **ERROR_STRUCT, Additional2**
bei **ERROR_STRUCT, Additional1= 16#3001** gemeldet.

Spezielle Fehler des Profinet IO IO-Link Device (anwendungsbezogene Fehler)

ERROR_CODE	Additional2	
COMMUNICATION_ERROR	16#8000	Device-Anwendungsfehler – keine Details
COMMUNICATION_ERROR	16#8011	Index nicht vorhanden
COMMUNICATION_ERROR	16#8012	SubIndex nicht vorhanden
COMMUNICATION_ERROR	16#8020	Dienst zeitweise nicht verfügbar
COMMUNICATION_ERROR	16#8021	Dienst zeitweise nicht verfügbar – Lokale Steuerung
COMMUNICATION_ERROR	16#8022	Dienst zeitweise nicht verfügbar – Device-Steuerung
COMMUNICATION_ERROR	16#8023	Zugriff verweigert
COMMUNICATION_ERROR	16#8030	Parameterwert außerhalb des Bereichs
COMMUNICATION_ERROR	16#8031	Parameterwert über dem Grenzwert

ERROR_CODE	Additional2	
COMMUNICATION_ERROR	16#8032	Parameterwert unter dem Grenzwert
COMMUNICATION_ERROR	16#8033	Parameterlänge überschritten
COMMUNICATION_ERROR	16#8034	Parameterlänge unterschritten
COMMUNICATION_ERROR	16#8035	Funktion nicht verfügbar
COMMUNICATION_ERROR	16#8036	Funktion zeitweise nicht verfügbar
COMMUNICATION_ERROR	16#8040	Ungültiger Parametersatz
COMMUNICATION_ERROR	16#8041	Inkonsistenter Parametersatz
COMMUNICATION_ERROR	16#8082	Anwendung nicht bereit
COMMUNICATION_ERROR	16#8001..16#81FF	Herstellerspezifisch

Tab. 5-169: Fehlercodes Profinet IO IO-Link Device

5.10 Beispielprogramme

Beispielprogramme zur Diagnose des Profinet IO Controller stehen als GAT Plug-In exemplarisch zur Verfügung, z. B. das Beispielprogramm "PNIODiagLogger" kann auch ohne GAT mit der Steuerung eingesetzt werden.

Die GAT Plug-Ins werden mit IndraWorks installiert (mit 'x' = Nummer der IndraWorks Installation):

C:\ProgramData\Rexroth\IndraWorks\'x'\GatWizard\Plugins\

GAT Plug-Ins bestehen aus:

- *.iwx Datei, die unterhalb einer Application importiert werden kann
- kurze Dokumentation (*.pdf, überwiegend nur in Englisch)
- Konfigurationsdatei *Config.txt, die nur vom GAT Wizard benötigt wird

Für weitere Informationen siehe Dokumentation "IndraLogic XLC IndraMotion MLC 14VRS Generic Application Template" (DOK-XLCMLC-TF*GAT**V14-APRS-DE-P).

6 RIL_ProfinetIODevice.library

6.1 Einführung und Übersicht

Die Steuerungen können als Feldbuslave betrieben werden. Im Falle von PROFINET IO nennt man diesen speziellen Feldbuslave PROFINET IO-Device. Die vorliegende Bibliothek beschreibt die Schnittstellen dieses PROFINET IO-Devices zum IEC-Anwenderprogramm.

Bei der Realisierung der Bibliothek wird auf die RIL_UTILITIES aufgesetzt. Es werden die Methoden der Basisfunktionsbausteine (Basisklassen) IL_EnableDoneBase und IL_ExecuteDoneBase verwendet und erweitert oder überschrieben.

Anwendungsgebiete und Funktionsbausteine von RIL_ProfinetIODevice

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine	Beschreibung
Diagnose (Diagnostics)	
IL_PNIIODeviceState auf Seite 211	zur Ermittlung des Status eines Profinet-Devices
IL_PNIIODeviceStateDetails auf Seite 213 <i>unter Verwendung des Datentyps</i> IL_PNIO_DEVICE_STATE auf Seite 216	zur Ermittlung des Status des Profinet-Devicestacks
IL_PNIIODeviceStateDetailsXMAC auf Seite 214 <i>unter Verwendung der Datentypen</i> IL_PNIO_DEVICE_XMAC auf Seite 217 IL_PNIO_DEVICE_XMAC_PORT auf Seite 218	Status des Profinet 2-port Switchs

Tab. 6-1: RIL_ProfinetIODevice-Bibliothek

Adressierung der Devices

Adressierung: Die PROFINET IO-Devices werden Anhand ihrer aufsteigenden Reihenfolge in der Konfiguration unterschieden.

Die Zählung beginnt mit dem 0. Device, d.h. wenn nur ein Device konfiguriert ist, ist seine Device-Instanz immer 0.

Siehe auch [IL_BUSSLAVE auf Seite 218](#).

PROFINETIO_TABLE 16#200

Die [PROFINETIO_TABLE 16#200 auf Seite 219](#) enthält die Fehlernummern, die von den Funktionsbausteinen der Bibliothek ausgegeben werden.

6.2 Diagnosen (POUs/Diagnosis)

6.2.1 IL_PNIIODeviceState

Kurzbeschreibung

Dieser Funktionsbaustein gibt eine einfache Diagnose des PROFINET IO Devices zurück. Im Falle einer anstehenden Diagnose kann über weitere FBs Details zu den Diagnosen ausgelesen werden (z.B. [IL_PNIIODeviceStateDetails](#)).

Zielsystem	Bibliothek /Ordner
MLC	RIL_ProfinetIODevice/POUs/Diagnosis
MTX	RIL_ProfinetIODevice/POUs/Diagnosis

Tab. 6-2: Referenztablelle

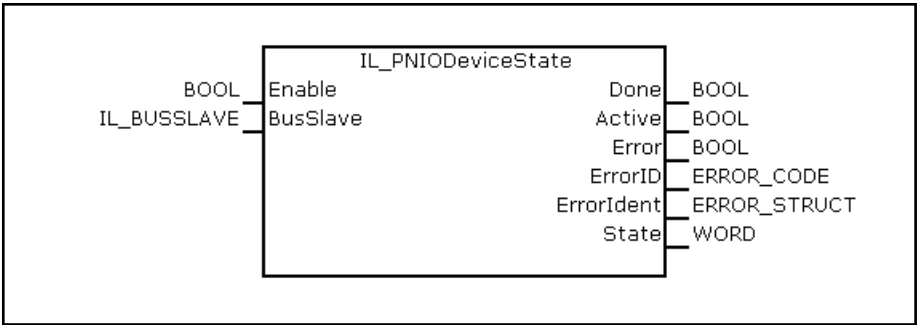


Abb. 6-1: IL_PNIIODeviceState

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (permanent, pegelgesteuert)
	BusSlave	IL_BUSSLAVE	Instanz des Feldbus-Slaves in der Reihenfolge des Konfigurationbaumes siehe auch IL_BUSSLAVE auf Seite 218
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	State	WORD	Statusinformation des lokalen Feldbus-Slaves

Tab. 6-3: Funktionsbaustein IL_PNIIODeviceState

Im Ausgang "State" ist eine Grobklassifizierung von Diagnosen binär kodiert. Die Bits sind in der RIL_FieldbusTypes.library als Globale Konstanten "FieldbusStates" enthalten.

Ausgang "State"

Bit-Nr.	Bezeichnung	Beschreibung
0	IL_FBS_BUS_OFF	Keine Feldbus Kommunikation
2	IL_FBS_CONFIG_ERROR	Fehlerhafte Konfiguration durch den Feldbus Master
4	IL_FBS_SLAVE_ERROR	Fehler des Gerätes
5	IL_FBS_UNUSED_MODULE_INFO	Information: nicht benutzte IO Module; die IO Module werden vom Controller nicht bedient.

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
BusSlave	IL_BUSSLAVE	IL_BUSSLAVE_0	IL_BUSSLAVE_4	IL_BUSSLAVE_0	Steigende Flanke von Enable

Tab. 6-4: Min-/Max und Default-Werte der Eingänge IL_PNIIODeviceState

Funktionsbeschreibung Der Baustein holt sich die Statusinformationen über das PROFINET Device und gibt sie als Bitleiste in ‚State‘ zurück. Die Bearbeitung des Bausteins wird in einem SPS-Zyklus komplett durchgeführt.

Fehlerbehandlung Der Funktionsbaustein nutzt die **Fehlertabelle PROFINETIO_TABLE** (ERROR_TABLE = 16#0200). Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	
SYSTEM_ERROR	16#0101	internal use	Device nicht in der Konfigurationstabelle gefunden

Tab. 6-5: Fehlercodes FB IL_PNIODeviceState

6.2.2 IL_PNIODeviceStateDetails

Kurzbeschreibung Der Funktionsbaustein ermittelt den detaillierten Status des PROFINET Device-Stacks. Er liefert Informationen zu

- dem Zustand des Kommunikationsstacks
- dem Zustand der Buskommunikation
- den letzten Fehlerwerten

Zielsystem	Bibliothek /Ordner
MLC	RIL_ProfinetIODevice/ POU's/ Diagnosis
MTX	RIL_ProfinetIODevice/ POU's/ Diagnosis

Tab. 6-6: Referenztable

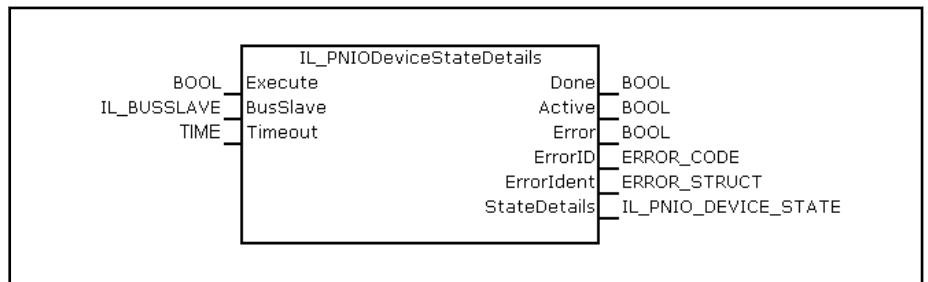


Abb. 6-2: IL_PNIODeviceStateDetails

	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusSlave	IL_BUSSLAVE	Instanz des Feldbus-Slaves in der Reihenfolge des Konfigurationbaumes siehe auch IL_BUSSLAVE auf Seite 218
	Timeout	TIME	Timeout - Überwachung des FBs in ms. T#0ms = Timeout - Überwachung ist nicht aktiv
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers

	Name	Typ	Kommentar
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	StateDetails	IL_PNIO_DEVICE_STATE	Detaillierte Statusinformationen des Device-Stacks IL_PNIO_DEVICE_STATE auf Seite 216

Tab. 6-7: Funktionsbaustein IL_PNIODeviceStateDetails

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusSlave	IL_BUSSLAVE	IL_BUSSLAVE_0	IL_BUSSLAVE_4	IL_BUSSLAVE_0	Steigende Flanke von Execute
Timeout	TIME	T#0ms	T#3600s	T#2s	Steigende Flanke an Execute

Tab. 6-8: Min-/Max und Default-Werte der Eingänge IL_PNIODeviceStateDetails

Funktionsbeschreibung Der Funktionsbaustein ermittelt bei Aktivierung einmalig den aktuellen Stand des Status des Devices.

Fehlerbehandlung Der Funktionsbaustein nutzt die **Fehlertabelle PROFINETIO_TABLE** (ERROR_TABLE = 16#0200). Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	
SYSTEM_ERROR	16#0101	internal use	Device nicht in der Konfigurationstabelle
COMMUNICATION_ERROR	16#0104	0	Zeitüberschreitung
STATE_MACHINE_ERROR	16#0105	0	Fehler in der Statemaschine
COMMUNICATION_ERROR	16#0003	PROFINET IO-Device-Diagnose - Siehe Allgemeine Fehlercodes, Seite 220. - Siehe Spezielle Fehlercodes, PROFINET IO-Device, Seite 222.	

Tab. 6-9: Fehlercodes Funktionsbaustein IL_PNIODeviceStateDetails

6.2.3 IL_PNIODeviceStateDetailsXMAC

Kurzbeschreibung Der Funktionsbaustein ermittelt den detaillierten Status des Profinet Switches auf dem netX[©] Kommunikations-Controller. Er liefert für jeden Ethernet-Port Informationen zu

- seinen Verbindungen (Connectivity) und
- zu seinem Statistikzähler.

Zielsystem	Bibliothek /Ordner
MLC	RIL_ProfinetIODevice/ POU's/ Diagnosis
MTX	RIL_ProfinetIODevice/ POU's/ Diagnosis

Tab. 6-10: Referenztable

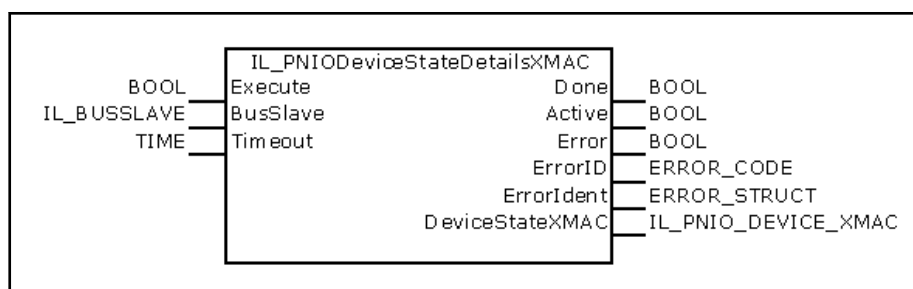


Abb. 6-3: IL_PNIOGetDeviceXMACState

	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusSlave	IL_BUSSLAVE	Instanz des Feldbus-Slaves in der Reihenfolge des Konfigurationbaumes IL_BUSSLAVE auf Seite 218
	Timeout	TIME	Timeout - Überwachung des FBs in ms. T#0ms = Timeout - Überwachung ist nicht aktiv
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	DeviceStateXMAC	IL_PNIO_DEVICE_XMAC	Detaillierte Statusinformationen der beiden Ports des Kommunikationscontrollers IL_PNIO_DEVICE_XMAC auf Seite 217

Tab. 6-11: Funktionsbaustein IL_PNIOGetDeviceXMACState

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusSlave	IL_BUSSLAVE	IL_BUSSLAVE_0	IL_BUSSLAVE_4	IL_BUSSLAVE_0	Steigende Flanke von Execute
Timeout	TIME	T#0ms	T#3600s	T#2s	Steigende Flanke an Execute

Tab. 6-12: Min-/Max- und Default-Werte der Eingänge IL_PNIODeviceStateDetailsXMAC

Funktionsbeschreibung

Der Funktionsbaustein ermittelt bei Aktivierung einmalig die aktuellen Statistik-Informationen der beiden Ethernet Ports des Kommunikationscontrollers.

Fehlerbehandlung

Der Funktionsbaustein nutzt die **Fehlertabelle PROFINETIO_TABLE** (ERROR_TABLE = 16#0200). Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	
SYSTEM_ERROR	16#0101	internal use	Device nicht in der Konfigurationstabelle
COMMUNICATION_ERROR	16#0104	0	Zeitüberschreitung
STATE_MACHINE_ERROR	16#0105	0	Fehler in der Zustandsmaschine
COMMUNICATION_ERROR	16#0003	PROFINET IO-Device-Diagnose - Siehe Allgemeine Fehlercodes, Seite 220. - Siehe Spezielle Fehlercodes, PROFINET IO-Device, Seite 222.	

Tab. 6-13: Fehlercodes Funktionsbaustein IL_PNIODeviceStateDetailsXMAC

6.3 Datentypen (DataTypes)

6.3.1 IL_PNIO_DEVICE_STATE

Kurzbeschreibung Die Datenstruktur IL_PNIO_DEVICE_STATE beinhaltet die Statusinformationen des PNIO Device Stacks.

Strukturelement	Typ	Beschreibung
PnsState	UDINT	Status der PROFINET IO Device-Task
LastRslt	UDINT	Letzter Errorcode; Die Bedeutung des Fehlers ist der Fehlertabelle zu entnehmen.
LinkState	UDINT	Link Status der Netzwerkverbindung
ConfigState	UDINT	Configurationstatus
CommunicationState	UDINT	Netzwerkstatus des Kommunikationskanals
CommunicationError	UDINT	Aktueller Fehlercode des Kommunikationskanals; Die Bedeutung des Fehlers ist der Fehlertabelle zu entnehmen.

Tab. 6-14: Datenstruktur IL_PNIO_DEVICE_STATE

Die Strukturelemente können folgende Werte annehmen:

PnsState Das Element ‚PnsState‘ der Datenstruktur IL_PNIO_DEVICE_STATE beschreibt bitkodiert den Status des PNIO Devices. Es können mehrere Bits gleichzeitig gesetzt sein.

Bit	Wert	
7	16#00000080	zyklischer Datenaustausch des Stacks
6	16#00000040	Freigabe zum zyklischen Datenaustausch erteilt
5	16#00000020	Module 0 an Submodule1 konfiguriert
4	16#00000010	Module 0 ist konfiguriert
3	16#00000008	Mindestens eine API (Application ProzessIdentifier) ist angelegt
2	16#00000004	reserviert
1	16#00000002	Der Profinet Stack ist gestartet
0	16#00000001	Die Device Information ist gesetzt

LinkState Das Element ‚LinkState‘ der Datenstruktur IL_PNIO_DEVICE_STATE beschreibt den Status der Netzwerkverbindung.

	Wert
0	keine Information vorhanden
1	physikalische Verbindung aufgebaut
2	langsame (10Mbit) Geschwindigkeit
3	Keine physikalische Verbindung vorhanden

ConfigState	Wert
0	Nicht konfiguriert
1	Über Datenbank konfiguriert
2	Konfigurationsdatenbank fehlerhaft
3	Durch die Applikation konfiguriert
4	Konfiguration durch die Applikation findet statt
5	Fehler bei der Konfiguration durch die Applikation
6	Konfiguration mit Warmstart-Parametern
7	Konfiguration über Warmstart-Parameter findet statt
8	Fehler bei Konfiguration mit Warmstart-Parametern

CommunicationState Das Element ‚CommunicationState‘ der Datenstruktur IL_PNIO_DEVICE_STATE beschreibt den Zustand der Kommunikation des PNIO Devices.

Wert	Bezeichnung
0	UNKNOWN Kommunikationzustand ist nicht bekannt
1	OFFLINE Keine PN-Kommunikation
2	STOP Keine PN-Kommunikation möglich oder erlaubt
3	IDLE Reserviert
4	OPERATE PN-Kommunikation ist erlaubt (kein Rückschluss auf zyklische Kommunikation möglich)

6.3.2 IL_PNIO_DEVICE_XMAC

Kurzbeschreibung Die Struktur enthält die Status- und Statistik-Information des unterlagerten Profinet 2-Port-Switches.

Für jedes vorhandene Port wird eine Struktur vom Typ [IL_PNIO_DEVICE_XMAC_PORT](#) auf Seite 218 abgelegt.

Type	Bezeichnung	Beschreibung
ARRAY [0..1] of IL_PNIO_DEVICE_XMAC_PORT	XmacPort[]	Status der Profinet IO Device Task

Tab. 6-15: Datenstruktur IL_PNIO_DEVICE_XMAC

Zuordnung der Steckerbezeichnungen zu den Ports mit Index 0 und 1 im Feld aXmacDiag[]:

Steuerung	Port 0	Port 1
CFL01_1_TP	X7E1	X7E2

6.3.3 IL_PNIO_DEVICE_XMAC_PORT

Kurzbeschreibung Die Struktur enthält die Status- und Statistik-Information eines Switch-Ports.

Bezeichnung	Typ	
FramesTransmittedOk	UDINT	Anzahl der erfolgreich gesendeten Telegramme
SingleCollisionFrames	UDINT	Anzahl der Telegramme mit einer einzigen Kollision
MultipleCollisionFrames	UDINT	Anzahl der Telegramme mit mehrfacher Kollision
LateCollisions	UDINT	Anzahl der 'späten Kollisionen' (nach mehr als 512 Bitzeiten)
LinkDownDuringTransmission	UDINT	Anzahl der beim Herunterfahren der Verbindung erkannten Telegramme
UtxUnderflowDuringTransmission	UDINT	Anzahl der FIFO-Leerläufe beim Senden
TxFatalErrors	UDINT	Anzahl der fatalen Sende-Fehler
FramesReceivedOk	UDINT	Anzahl der erfolgreich empfangenen Telegramme
FrameCheckSequenceErrors	UDINT	Anzahl der richtig zusammengesetzten Telegramme mit einem Blockchecksummenfehler
AlignmentErrors	UDINT	Anzahl der fehlerhaft zusammengesetzten Telegramme mit einem Blockchecksummenfehler
FrameTooLongErrors	UDINT	Anzahl der empfangenen Telegramme größer als zulässige Telegrammgröße
RuntFramesReceived	UDINT	Anzahl der Telegramme zwischen 42 und 63 Bytes mit einer gültigen Checksumme
CollisionFragmentsReceived	UDINT	Anzahl der Telegramme unter 64Bytes mit einem Checksummenfehler
FramesDroppedDueLowResource	UDINT	Anzahl wegen Ressourcenmangels verworfener Telegramme
FramesDroppedDueUrxOverflow	UDINT	Anzahl wegen FIFO-Überlaufs verworfener Telegramme
RxFatalErrors	UDINT	Anzahl fataler Empfangsfehler

Tab. 6-16: Struktur IL_PNIO_DEVICE_XMAC_PORT

6.4 IL_BUSSLAVE

Kurzbeschreibung Der Aufzählungstyp IL_BUSSLAVE aus der RIL_FeldbusTypes.library dient dazu, den jeweiligen FeldbusSlave auszuwählen. In einem MLC-System können neben der onboard Realtime-Ethernet-Schnittstelle über Funktionsmodule weitere Devices vorhanden sein.

Die Geräte werden anhand ihrer aufsteigenden Reihenfolge in der Konfiguration unterschieden. Die Indizierung der Geräte geschieht dabei in der Reihenfolge

- Onboard
- Funktionsmodul 1
- Funktionsmodul 2
- Funktionsmodul 3
- Funktionsmodul 4

Element	Wert	Beschreibung
IL_BUSSLAVE_0	0	Erstes Gerät
IL_BUSSLAVE_1	1	Zweites Gerät
IL_BUSSLAVE_2	2	Drittes Gerät
IL_BUSSLAVE_3	3	Viertes Gerät
IL_BUSSLAVE_4	4	Fünftes Gerät

Tab. 6-17: Elemente des Aufzählungstyps IL_BUSSLAVE

Beispiel zur Adressierung der verschiedenen Instanzen des PROFINET IO Devices auf einer Steuerung:

Konfiguriert ist:	Adressierung mit:
Onboard: PROFINET IO Device	IL_BUSSLAVE_0
FM1: PROFINET IO Controller	IL_BUSMASTER_0
FM2: PROFINET IO Device	IL_BUSSLAVE_1
FM3: PROFINET IO Device	IL_BUSSLAVE_2

Tab. 6-18: Aufzählungstyp IL_BUSSLAVE, Beispiel

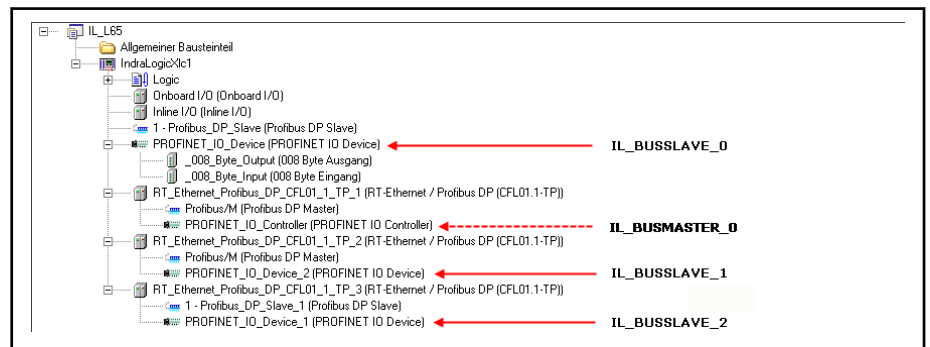


Abb. 6-4: Aufzählungstyp IL_BUSSLAVE, Beispiel

6.5 PROFINETIO_TABLE 16#200

6.5.1 Allgemeines

PROFINETIO_TABLE - ERROR_TABLE, Tabellenummer 16#0200

Typdefinition

```

TYPE ERROR_STRUCT :
    STRUCT
        Table          :ERROR_TABLE;
        Additional1    :DWORD;
        Additional2    :DWORD;
    END_STRUCT
END_TYPE

```

Die Tabelle enthält die Fehlernummern, die von den Funktionsbausteinen nachstehender Bibliotheken ausgegeben werden:

- RIL_ProfiNetIO - Bibliothek für PROFINET IO-Controller (BusMaster),
- RIL_ProfiNetIODevice - Bibliothek für PROFINET IO-Devices (BusSlave).

Die Bausteine selbst geben nachstehende Fehlermeldungen aus:

ErrorID	Additional1	Additional2	
COMMUNICATION_ERROR	16#0003	PROFINET IO-Device-Diagnose • Siehe Allgemeine Fehlercodes, Seite 220. • Siehe Spezielle Fehlercodes, PROFINET IO-Device , Seite 222.	
SYSTEM_ERROR	16#0101	internal use	Device nicht in der Konfigurationstabelle
COMMUNICATION_ERROR	16#0104	0	Zeitüberschreitung
STATE_MACHINE_ERROR	16#0105	xxx	Fehler in der Zustandsmaschine

Tab. 6-19: Fehlercodes ProfiNet IO (Device), Auswahl

6.5.2 Allgemeine Fehlercodes

Die dargestellten Fehlercodes gelten für alle BusMaster und BusSlaves, sie sind zusätzlich feldbusunabhängig.

Die Fehlercodes werden als **ERROR_STRUCT, Additional2**
bei **ERROR_STRUCT, Additional1= 16#0003** gemeldet.

Additional2	
16#800A0001	Ungültiger Zeiger (NULL) an Treiber übergeben
16#800A0002	Keine Steckkarte mit dem angegebenen Namen/Index verfügbar
16#800A0003	Kein Kanal mit dem angegebenen Index verfügbar
16#800A0004	Ungültiges Handel an Treiber übergeben
16#800A0005	Ungültiger Parameter
16#800A0006	Ungültiger Befehl
16#800A0007	Ungültige Puffergröße
16#800A0008	Ungültige Zugriffsgröße
16#800A0009	Funktion fehlgeschlagen
16#800A000A	Datei konnte nicht geöffnet werden
16#800A000B	Dateigröße ist gleich Null
16#800A000C	Unzureichende Speicherkapazität um Datei zu laden
16#800A000D	Vergleich der Dateiprüfsummen fehlgeschlagen
16#800A000E	Fehler beim Lesen von Datei
16#800A000F	Ungültiger Dateityp
16#800A0010	Ungültiger Dateiname
16#800A0011	Treiberfunktion nicht vorhanden
16#800A0012	Angegebener Puffer ist zu klein
16#800A0013	Speicher kann nicht abgebildet werden
16#800A0014	Keine weiteren Einträge vorhanden
16#800B0001	Treiber ist nicht initialisiert
16#800B0002	Init-Statusfehler des Treibers
16#800B0003	Treiber Statuslesefehler

Additional2	
16#800B0004	Befehl ist aktiv auf Gerät
16#800B0005	Allgemeiner Fehler bei Download
16#800B0006	Falsche Treiberversion
16#800B0030	CIFx Treiber nicht in Betrieb
16#800B0031	Gerät konnte nicht initialisiert werden
16#800B0032	Kanal konnte nicht initialisiert werden (xOpenChannel wurde nicht aufgerufen)
16#800B0033	IOControl Aufruf fehlgeschlagen
16#800B0034	Treiber konnte nicht geöffnet werden
16#800C0010	Auf den Dual-Port Speicher konnte nicht zugegriffen werden (Speicherkarte nicht gefunden)
16#800C0011	Gerät nicht bereit (Ready-Flag fehlgeschlagen)
16#800C0012	Gerät nicht betriebsbereit (Running-Flag fehlgeschlagen)
16#800C0013	Watchdog-Test fehlgeschlagen
16#800C0015	Handshake-Flag Fehler
16#800C0016	Mailbox für gesendete Mails ist voll
16#800C0017	Timeout für Senden der Datenpakete
16#800C0018	Timeout für Empfangen der Datenpakete
16#800C0019	Kein Datenpaket verfügbar
16#800C001A	Zu wenig Speicherplatz in Mailbox
16#800C0020	Befehl-Timeout zurücksetzen
16#800C0021	COM-Flag nicht gesetzt
16#800C0022	EA-Datenaustausch fehlgeschlagen
16#800C0023	EA-Datenaustausch Timeout
16#800C0024	Unbekannter EA-Datenaustauschmodus
16#800C0025	Gerätfunktion fehlgeschlagen
16#800C0026	DPM-Größe size unterscheidet sich von Konfiguration
16#800C0027	Unbekannter Statusmodus
16#800C0028	Ausgangsport wird bereits verwendet
16#800C0029	Timeout Konfigurationsverriegelung
16#800C002A	Konfigurationsentriegelung Timeout
16#800C002B	HOST-Status Timeout setzen
16#800C002C	HOST-Status Timeout löschen
16#800C002D	Timeout während Kanalinitialisierung
16#800C002E	Bus EIN Timeout setzen
16#800C002F	Bus AUS Timeout setzen

Tab. 6-20: Allgemeine Fehlercodes

6.5.3 Spezielle Fehlercodes, PROFINET IO-Device

Die dargestellten Fehlercodes gelten für PROFINET IO-Devices.

Die Fehlercodes werden als **ERROR_STRUCT, Additional2**,
bei **ERROR_STRUCT, Additional1= 16#0003**, gemeldet.

Additional2	
16#C0300001	Ungültiges Kommando
16#C0300004	PROFINET IO-Device Setup fehlerhaft

Tab. 6-21: *Spezielle Fehlercodes eines PROFINET IO-Devices*

7 RIL_EtherNetIP.library

7.1 Einführung und Übersicht

Die Bibliothek **RIL_EtherNetIP.library** ist die Schnittstelle vom SPS-Programm zu den Ethernet/IP Teilnehmern. Über diese Schnittstelle können die Ethernet/IP Teilnehmer diagnostiziert werden.

Des Weiteren werden Funktionsbausteine beschrieben, welche azyklisch einzelne Daten oder ganze Datenblöcke lesen und schreiben können.

Es werden die Methoden IL_EnableDoneBase und IL_ExecuteDoneBase verwendet und erweitert oder überschrieben.

Zielsysteme Die Bibliothek kann auf folgenden Systemen eingesetzt werden:

Zielsystem	Bibliothek
MLC	RIL_EtherNetIP
MTX	RIL_EtherNetIP

Tab. 7-1: Referenztablelle

Anwendungsgebiete und Funktionsbausteine von RIL_EtherNetIP.library

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine (POUs)	Beschreibung
Azyklische Dienste (Acyclic_Communication)	
IL_ENIPGetAttributeSingle, Seite 224	liest einen "Class-Instance-Attribute"-Wert
IL_ENIPGetIdentityObjAllAttributes, Seite 226	liest alle Pflichtattribute des "Identity Objects" eines Adapters
IL_ENIPSendUnconnectedMessage, Seite 228	kann einen beliebigen azyklischen CIP-Service ausführen
IL_ENIPSetAttributeSingle, Seite 230	schreibt einen "Class-Instance-Attribute"-Wert
Sichern und Wiederherstellen (Backup Restore)	
IL_ENIPAdapterBackup, Seite 233	dient zum Sichern der Konfiguration und der Parameter eines Ethernet/IP Adapter-Gerätes auf eine Datei mittels azyklischen Lesens
IL_ENIPAdapterRestore, Seite 236	dient zum Wiederherstellen der Parameter eines Ethernet/IP Adapter-Gerätes mittels azyklischen Schreibens aus der Sicherungsdatei
Diagnose (Diagnosis)	
IL_ENIPRemoteAdapterState, Seite 239	liest die Einfachdiagnose des Ethernet/IP Adapters am Scanner
IL_ENIPRemoteAdapterStateDetails, Seite 241	liest die Detaildiagnose des Ethernet/IP Adapters am Scanner
IL_ENIPScannerState, Seite 243	liest die Einfachdiagnose des Ethernet/IP Scanners
IL_ENIPScannerStateDetails, Seite 245	liest die Detaildiagnose des Ethernet/IP Scanners
Mit Datentypen:	
IL_ENIP_IDENTITY_ATTRIBUTES, Seite 247	enthält die Pflichtattribute des "Identity Objects", die mittels "Get-Attribute-All" gelesen werden
IL_ENIP_REMOTE_ADAPTER_STATE_DETAILS, Seite 248	erlaubt die Anzeige der Bits 4...7 "Detailstatus" eines Ethernet/IP Adapters (entspricht Attribut 5 des "Identity Objects") als Text

IL_ENIP_SCANNER_STATE, Seite 249	Struktur, beinhaltet Diagnoseinformationen über den Zustand des Kommunikationsstacks, den Zustand der Buskommunikation, den Fehlerzähler, sowie einen Überblick über konfigurierte, aktive und eine Diagnose meldende Slaves
Mit Datentypen aus RIL_FieldbusTypes	
IL_FBUS_COMMUNICATION_STATE, Seite 190	Status der Feldbus-Kommunikation
IL_FBUS_SLAVE_STATE, Seite 190	Status der Kommunikationsbeziehungen zu einem Adapter (Slave)

Tab. 7-2: RIL_EthernetIP-Bibliothek

Auswahl des Ethernet/IP Scanners

Adressierung:

Die Ethernet/IP Scanner werden anhand ihrer aufsteigenden Reihenfolge in der Konfiguration unterschieden.

Die Zählung beginnt mit "0". Wenn nur ein Scanner konfiguriert ist, ist seine Scanner-Instanz immer "0" (siehe auch [IL_BUSMASTER, Seite 246](#)).

Adressierung der IP-Adresse des Ethernet/IP Adapters

IP-Adresse des Adapters, auf den zugegriffen werden soll.

7.2 Azyklische Dienste (Acyclic_Communication)

7.2.1 IL_ENIPGetAttributeSingle

Kurzbeschreibung Mit dem Funktionsbaustein IL_ENIPGetAttributeSingle kann ein einzelnes Attribut ausgelesen werden.

Zielsystem	Bibliothek/Ordner
MLC	RIL_EtherNetIP/POUs/Acyclic_Communication
MTX	RIL_EtherNetIP/POUs/Acyclic_Communication

Tab. 7-3: Referenztabelle

Schnittstellenbeschreibung

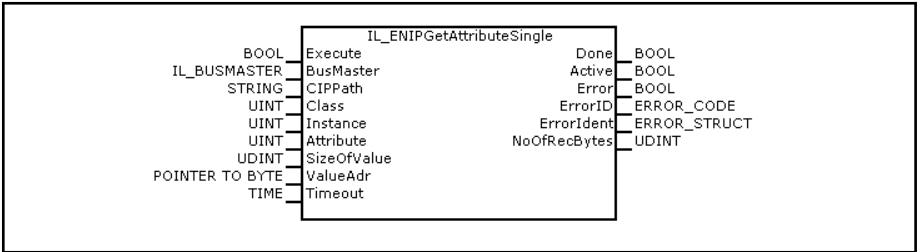


Abb. 7-1: Schnittstellenbeschreibung FB IL_ENIPGetAttributeSingle

I/O-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Ethernet/IP Scanners in der Reihenfolge des Konfigurationsbaumes. Ist nur ein Ethernet/IP Scanner vorhanden, besitzt dieser Scanner immer den Wert IL_BUSMASTER_0 siehe Kap. 7.5.1 "IL_BUSMASTER" auf Seite 246
	CIPPath	STRING	Adresse des CIP-Teilnehmers als String, in der Form „aaa.bbb.ccc.ddd“
	Class	UINT	Klassen Code
	Instance	UINT	Instanz Nummer
	Attribute	UINT	Attribut ID
	SizeOfValue	UDINT	Größe des mit "ValueAdr" adressierten Datenbereichs
	ValueAdr	POINTER TO BYTE	Anfangsadresse des Datenbereichs
	Timeout	TIME	Timeout - Überwachung des FBs in ms. T#0ms = Timeout - Überwachung ist nicht aktiv
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	NoOfRecBytes	UDINT	Anzahl der empfangenen Bytes im Datenbereich

Tab. 7-4: I/O-Schnittstelle des FB IL_ENIPGetAttributeSingle

Die Eingangsvariablen können die in der folgenden Tabelle dargestellten Werte annehmen.

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an "Execute"
CIPPath	STRING	Zeichenfolge	Zeichenfolge	„“	Steigende Flanke an "Execute"
Class	UINT	16#01	16#FF	16#01	Steigende Flanke an "Execute"
Instance	UINT	16#00	16#FF	16#00	Steigende Flanke an "Execute"
Attribute	UINT	16#01	16#FF	16#01	Steigende Flanke an "Execute"
SizeOfValue	UDINT	16#0000	16#01C2	16#0000	Steigende Flanke an "Execute"

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
ValueAdr	POINTER TO BYTE	n.def.	(450 Byte)	16#0000	Steigende Flanke an "Execute"
Timeout	TIME	T#0ms	n.def.	T#2s	Steigende Flanke an "Execute"

Tab. 7-5: Min-, Max- und Default-Werte der IL_ENIPGetAttributeSingle-Eingänge

Funktionsbeschreibung Mit dem Funktionsbaustein **IL_ENIPGetAttributeSingle** kann ein einzelnes Attribut ausgelesen werden.

Zuerst wird das Gerät (Adapter) über die IP-Adresse adressiert.

Dann erfolgt der Zugriff über "Class", "Instance" und "Attribute" auf einen ganz bestimmten "Wert" (oder eine Einstellung) des Adapters.

Die Attribute kann man sich baumartig-strukturiert vorstellen. Allerdings gibt es auch Klassen-Attribute. In diesem Fall entfällt die Angabe der "Instance", bzw. die "Instance"-Nummer ist das Attribut.

Die Adressierung ist herstellerspezifisch. In der Regel werden die folgenden Standard-Objekte unterstützt:

Identity Object	(Class Code: 0x01)
Message Router Object	(Class Code: 0x02)
Assembly Object	(Class Code: 0x04)
Connection Object	(Class Code: 0x05)
Connection Manager Object	(Class Code: 0x06)
TCP/IP Interface Object	(Class Code: 0xF5)
Ethernet Link Object	(Class Code: 0xF6)

Weitere Informationen siehe Hersteller-Dokumentation und CIP-Spezifikation

Fehlerbehandlung Der Funktionsbaustein nutzt die **Fehlertabelle ETHERNET_IP_TABLE** (ERROR_TABLE = 16#0151), siehe [Kap. 7.5.7 "Auszug aus der Fehlertabelle ETHERNET_IP_TABLE \(ERROR_TABLE = 16#0151\)"](#) auf Seite 250.

7.2.2 IL_ENIPGetIdentityObjAllAttributes

Kurzbeschreibung Mit dem Funktionsbaustein **IL_ENIPGetIdentityObjAllAttributes** werden aus dem "Identity Object" des Adapters alle Pflicht-Attribute ausgelesen (d. h. unterlagert wird der CIP-Service "Get-Attribute-All" ausgeführt).

Zielsystem	Bibliothek/Ordner
MLC ab 14V06	RIL_EtherNetIP/POUs/Acyclic_Communication
MTX ab 14V06	RIL_EtherNetIP/POUs/Acyclic_Communication

Tab. 7-6: Referenztablelle

Schnittstellenbeschreibung

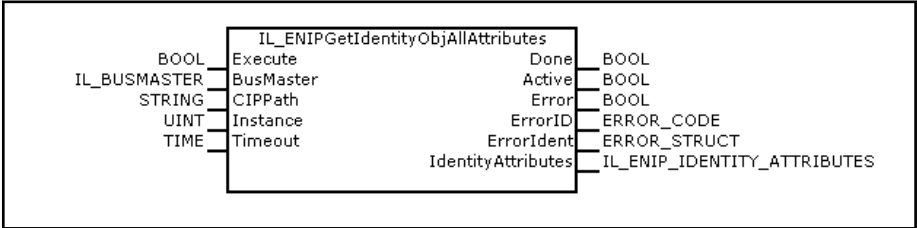


Abb. 7-2: Schnittstellenbeschreibung FB IL_ENIPGetIdentityObjAllAttributes

I/O-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Ethernet/IP Scanners in der Reihenfolge des Konfigurationsbaumes. Ist nur ein Ethernet/IP Scanner vorhanden, besitzt dieser Scanner immer den Wert IL_BUSMASTER_0 siehe Kap. 7.5.1 "IL_BUSMASTER" auf Seite 246
	CIPPath	STRING	Adresse des CIP-Teilnehmers als String, in der Form „aaa.bbb.ccc.ddd“
	Instance	UINT	Instanz Nummer, in der Regel hat ein Adapter nur ein "Identity Object" (Instance = 1)
	Timeout	TIME	Timeout - Überwachung des FBs in ms. T#0ms = Timeout - Überwachung ist nicht aktiv
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	IdentityAttributes	IL_ENIP_IDENTITY_ATTRIBUTES	Pflichtattribute des "Identity Objects" Siehe Kap. 7.5.2 "IL_ENIP_IDENTITY_ATTRIBUTES" auf Seite 247

Tab. 7-7: I/O-Schnittstelle des FB IL_ENIPGetIdentityObjAllAttributes

Die Eingangsvariablen können die in der folgenden Tabelle dargestellten Werte annehmen.

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an "Execute"
CIPPath	STRING	Zeichenfolge	Zeichenfolge	„“	Steigende Flanke an "Execute"
Instance	UINT	0	255	1	Steigende Flanke an "Execute"
Timeout	TIME	T#0ms	n.def.	T#2s	Steigende Flanke an "Execute"

Tab. 7-8: Min-, Max- und Default-Werte der IL_ENIPGetIdentityObjAllAttributes-Eingänge

Funktionsbeschreibung

Mit dem Funktionsbaustein **IL_ENIPGetIdentityObjAllAttributes** werden aus dem "Identity Object" des Adapters alle Pflicht-Attribute ausgelesen (d. h. unterlagert wird der CIP-Service "Get-Attribute-All" ausgeführt).

Zuerst wird das Gerät (Adapter) über die IP-Adresse adressiert.

In der Regel hat ein Adapter-Gerät eine Instanz des "Identity Object" (Instance = 1).

Die Pflicht-Attribute werden mittels "Get-Attribute-All" aus dem "Identity Object" ausgelesen.

Fehlerbehandlung Der Funktionsbaustein nutzt die **Fehlertabelle ETHERNET_IP_TABLE** (ERROR_TABLE = 16#0151), siehe [Kap. 7.5.7 "Auszug aus der Fehlertabelle ETHERNET_IP_TABLE \(ERROR_TABLE = 16#0151\)"](#) auf Seite 250.

7.2.3 IL_ENIPSendUnconnectedMessage

Kurzbeschreibung Mit dem Funktionsbaustein **IL_ENIPSendUnconnectedMessage** kann ein beliebiger azyklischer Service ausgeführt werden. Zum Beispiel nutzen die Funktionsbausteine [Kap. 7.2.1 "IL_ENIPGetAttributeSingle"](#) auf Seite 224 und [Kap. 7.2.4 "IL_ENIPSetAttributeSingle"](#) auf Seite 230 unterlagert auch diese Funktion.

Zielsystem	Bibliothek/Ordner
MLC ab 14V06	RIL_EtherNetIP/POUs/Acyclic_Communication
MTX ab 14V06	RIL_EtherNetIP/POUs/Acyclic_Communication

Tab. 7-9: Referenztabelle

Schnittstellenbeschreibung

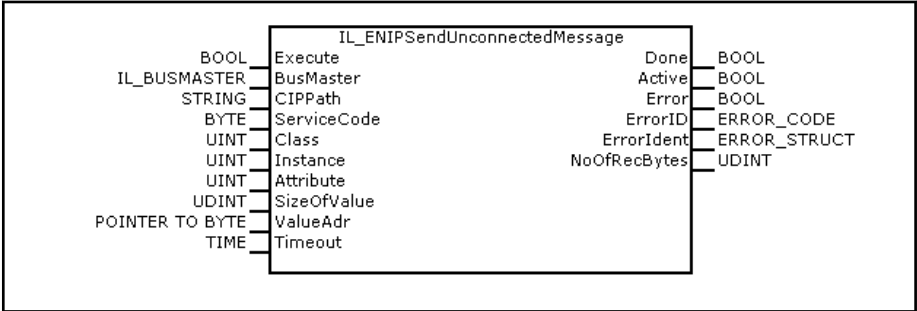


Abb. 7-3: Schnittstellenbeschreibung FB IL_ENIPSendUnconnectedMessage

I/O-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Ethernet/IP Scanners in der Reihenfolge des Konfigurationsbaumes. Ist nur ein Ethernet/IP Scanner vorhanden, besitzt dieser Scanner immer den Wert IL_BUSMASTER_0 siehe Kap. 7.5.1 "IL_BUSMASTER" auf Seite 246
	CIPPath	STRING	Adresse des CIP Teilnehmers als String, in der Form "aaa.bbb.ccc.ddd"
	Service Code	BYTE	CIP-Service-Code
	Class	UINT	Klassen-Code
	Instance	UINT	Instanz-Nummer
	Attribute	UINT	Attribut-ID
	SizeOfValue	UDINT	Größe des mit ValueAdr adressierten Datenbereichs
	ValueAdr	POINTER TO BYTE	Anfangsadresse des Datenbereichs
	Timeout	TIME	Timeout - Überwachung des FBs in ms. T#0ms = Timeout - Überwachung ist nicht aktiv
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	NoOfRecBytes	UDINT	Anzahl der empfangenen Bytes im Datenbereich

Tab. 7-10: I/O-Schnittstelle des FB IL_ENIPSendUnconnectedMessage

Die Eingangsvariablen können die in der folgenden Tabelle dargestellten Werte annehmen.

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an "Execute"
CIPPath	STRING	Zeichenfolge	Zeichenfolge	„“	Steigende Flanke an "Execute"
Service Code	BYTE	1	255	0	Steigende Flanke an "Execute"
Class	UINT	16#01	16#FF	16#01	Steigende Flanke an "Execute"
Instance	UINT	16#00	16#FF	16#00	Steigende Flanke an "Execute"

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Attribute	UINT	16#01	16#FF	16#01	Steigende Flanke an "Execute"
SizeOfValue	UDINT	16#0000	16#01C2	16#0000	Steigende Flanke an "Execute"
ValueAdr	POINTER TO BYTE	n.def.	(450 Byte)	16#0000	Steigende Flanke an "Execute"
Timeout	TIME	T#0ms	n.def.	T#2s	Steigende Flanke an "Execute"

Tab. 7-11: Min-, Max- und Default-Werte der IL_ENIPSendUnconnectedMessage-Eingänge

Funktionsbeschreibung Mit dem Funktionsbaustein **IL_ENIPSendUnconnectedMessage** kann ein beliebiger CIP-Service ausgeführt werden. Es kann ein Standard CIP-Service-Code oder auch ein Hersteller- oder Geräte-spezifischer CIP-Service ausgeführt werden.

Zuerst wird das Gerät (Adapter) über die IP-Adresse adressiert.

Dann erfolgt der Zugriff über "Class", "Instance" und "Attribute" auf einen ganz bestimmten "Wert" (oder eine Einstellung) des Adapters. Zum Beispiel gibt es folgende Standard-Klassen:

Identity Object	(Class Code: 0x01)
Message Router Object	(Class Code: 0x02)
Assembly Object	(Class Code: 0x04)
Connection Object	(Class Code: 0x05)
Connection Manager Object	(Class Code: 0x06)
TCP/IP Interface Object	(Class Code: 0xF5)
Ethernet Link Object	(Class Code: 0xF6)

Geräte-spezifische Details sind in der Hersteller-Dokumentation nachzuschlagen

Fehlerbehandlung Der Funktionsbaustein nutzt die **Fehlertabelle ETHERNET_IP_TABLE** (ERROR_TABLE = 16#0151), siehe [Kap. 7.5.7 "Auszug aus der Fehlertabelle ETHERNET_IP_TABLE \(ERROR_TABLE = 16#0151\)"](#) auf Seite 250.

7.2.4 IL_ENIPSetAttributeSingle

Kurzbeschreibung Mit dem Funktionsbaustein **IL_ENIPSetAttributeSingle** kann ein einzelnes Attribut gesetzt oder geschrieben werden.

Zielsystem	Bibliothek/Ordner
MLC	RIL_EtherNetIP/POUs/Acyclic_Communication
MTX	RIL_EtherNetIP/POUs/Acyclic_Communication

Tab. 7-12: Referenztable

Schnittstellenbeschreibung

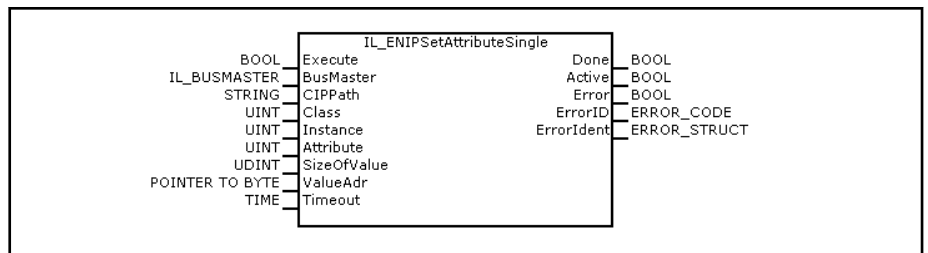


Abb. 7-4: Schnittstellenbeschreibung FB IL_ENIPSetAttributeSingle

I/O-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Scanners in der Reihenfolge des Konfigurationsbaumes. Ist nur ein Ethernet/IP Scanner vorhanden, besitzt dieser Scanner immer den Wert IL_BUSMASTER_0 siehe Kap. 7.5.1 "IL_BUSMASTER" auf Seite 246
	CIPPath	STRING	Adresse des CIP-Teilnehmers als String, in der Form "aaa.bbb.ccc.ddd"
	Class	UINT	Klassen-Code
	Instance	UINT	Instanz-Nummer
	Attribute	UINT	Attribute-ID
	SizeOfValue	UDINT	Größe des mit ValueAdr adressierten Datenbereichs
	ValueAdr	POINTER TO BYTE	Anfangsadresse des Datenbereichs
	Timeout	TIME	Timeout - Überwachung des FBs in ms. T#0ms = Timeout - Überwachung ist nicht aktiv
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers

Tab. 7-13: I/O-Schnittstelle des FB IL_ENIPSetAttributeSingle

Die Eingangsvariablen können die in der folgenden Tabelle dargestellten Werte annehmen.

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an "Execute"
CIPPath	STRING	Zeichenfolge	Zeichenfolge	„ “	Steigende Flanke an "Execute"

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Class	UINT	16#01	16#FF	16#01	Steigende Flanke an "Execute"
Instance	UINT	16#00	16#FF	16#00	Steigende Flanke an "Execute"
Attribute	UINT	16#01	16#FF	16#01	Steigende Flanke an "Execute"
SizeOfValue	UDINT	16#0000	16#01C2	16#0000	Steigende Flanke an "Execute"
ValueAdr	POINTER TO BYTE	n.def.	(450 Byte)	16#0000	Steigende Flanke an "Execute"
Timeout	TIME	T#0ms	n.def.	T#2s	Steigende Flanke an "Execute"

Tab. 7-14: Min-, Max- und Default-Werte der IL_ENIPSetAttributeSingle-Eingänge

Funktionsbeschreibung

Mit dem Funktionsbaustein **IL_ENIPSetAttributeSingle** kann ein einzelnes Attribut gesetzt oder geschrieben werden.

Zuerst wird das Gerät (Adapter) über die IP-Adresse adressiert.

Dann erfolgt der Zugriff über "Class", "Instance" und "Attribute" auf einen ganz bestimmten "Wert" (oder eine Einstellung) des Adapters.

Die Attribute kann man sich baumartig-strukturiert vorstellen. Allerdings gibt es auch Klassen-Attribute. In diesem Fall entfällt die Angabe der "Instance" bzw. die "Instance"-Nummer ist das Attribut.

Die Adressierung ist herstellerspezifisch. In der Regel werden die folgenden Standard-Objekte unterstützt:

Identity Object	(Class Code: 0x01)
Message Router Object	(Class Code: 0x02)
Assembly Object	(Class Code: 0x04)
Connection Object	(Class Code: 0x05)
Connection Manager Object	(Class Code: 0x06)
TCP/IP Interface Object	(Class Code: 0xF5)
Ethernet Link Object	(Class Code: 0xF6)

Weitere Informationen siehe Hersteller-Dokumentation und CIP-Spezifikation

Fehlerbehandlung

Der Funktionsbaustein nutzt die **Fehlertabelle ETHERNET_IP_TABLE** (ERROR_TABLE = 16#0151), siehe [Kap. 7.5.7 "Auszug aus der Fehlertabelle ETHERNET_IP_TABLE \(ERROR_TABLE = 16#0151\)"](#) auf Seite 250.

7.3 Sichern und Wiederherstellen (Backup Restore)

7.3.1 IL_ENIPAdapterBackup

Kurzbeschreibung Der Funktionsbaustein **IL_ENIPAdapterBackup** dient zum Sichern der Konfiguration und der Parameter eines Ethernet/IP Adapter-Gerätes auf eine Datei mittels azyklischen Lesens (siehe [Kap. 7.2.1 "IL_ENIPGetAttributeSingle" auf Seite 224](#)).

Zielsystem	Bibliothek/Ordner
MLC	RIL_ProfinetIO/POUs/Backup Restore

Tab. 7-15: Referenztable

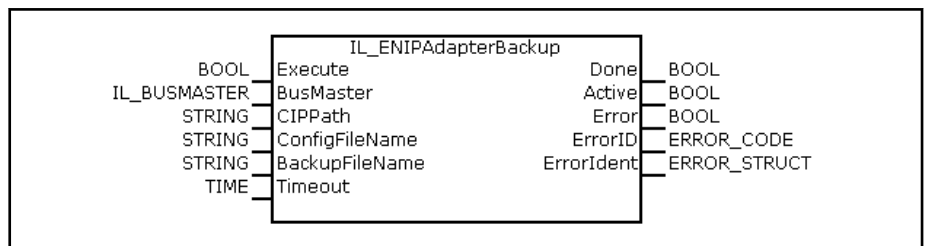


Abb. 7-5: Schnittstellenbeschreibung FB IL_ENIPAdapterBackup

I/O-Typ	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Scanners in der Reihenfolge des Konfigurationsbaumes Ist nur ein Ethernet/IP Scanner vorhanden, besitzt dieser Scanner immer den Wert IL_BUSMASTER_0 Siehe auch Kap. 7.5.1 "IL_BUSMASTER" auf Seite 246
	CIPPath	STRING	Adresse des CIP-Teilnehmers als String, in der Form "aaa.bbb.ccc.ddd"
	ConfigFileName	STRING	ConfigFile mit "Class", "Instance" und "Attribute" der Parameter, die aus dem Gerät zu lesen sind. Name der Konfigurationsdatei ohne Extension (*.enc)
	BackupFileName	STRING	Name der Sicherungsdatei ohne Extension (*.enb)
	Timeout	TIME	Timeout-Überwachung des Funktionsbausteins
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibt die Diagnose im Fehlerfall (aus RIL_Common-Types)
	ErrorIdent	ERROR_STRUCT	Detailldiagnose (Fehlertabelle PROFINETIO_TABLE) siehe "Fehlerbehandlung" auf Seite 236

Tab. 7-16: I/O-Schnittstelle des FB IL_ENIPAdapterBackup

Min-, Max- und Default-Werte: In der folgenden Tabelle sind die Min-, Max- und Default-Werte der Funktionsbausteineingänge aufgelistet.

Name	Typ	Min. Wert	Max. Wert	Defaultwert	Anwendung
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an "Execute"
Timeout	TIME	T#0ms		T#2s	Steigende Flanke an "Execute"

Tab. 7-17: Min-/Max- und Default-Werte der IL_ENIPAdapterBackup-Eingänge

Funktionsbeschreibung

Der Funktionsbaustein **IL_ENIPAdapterBackup** dient zum Sichern der Konfiguration und der Parameter eines Ethernet/IP Adapter-Gerätes auf eine Datei nach azyklischem Lesen (siehe [Kap. 7.2.1 "IL_ENIPGetAttributeSingle" auf Seite 224](#)) aus dem Adapter.

Der Funktionsbaustein liest die in der Konfigurationsdatei "ConfigFileName" aufgeführten Parameter aus dem Adapter aus und speichert die Daten in der Datei "BackupFileName".

Die Konfigurations- und Sicherungsdateien werden auf der Compact-Flash-Karte der Steuerung (USER-Partition, /ata0b/) abgelegt.

Der Funktionsbaustein erstellt bei jeder Ausführung eine neue Datei. Ist eine Datei bereits vorhanden, wird die Datei überschrieben.

Die Konfigurationsdatei ist im Textformat gespeichert und hat die Extension "*.enc".

Basisstruktur der ConfigFile-Datei	Beispiel der ConfigFile-Datei
Class, Instance, Attribute	16#00C7, 16#0002, 16#0001
Class, Instance, Attribute	16#00C7, 16#0003, 16#0001
Class, Instance, Attribute	16#00C7, 16#0004, 16#0001
...	...
...	...
...	...
Class, Instance, Attribute	16#00C7, 16#0008, 16#0001

Bei steigender Flanke am Eingang "Execute" liest der Funktionsbaustein die Parameter aus dem Adapter aus und speichert sie in die Sicherungsdatei.

Basissicherungsdatei	Beispiel der Sicherungsdatei
Class, Instance, Attribute, DataSize, Data	16#00C7, 16#0002, 16#0001, 16#0001, 16#58
Class, Instance, Attribute, DataSize, Data1, Data2	16#00C7, 16#0003, 16#0001, 16#0002, 16#20, 16#82
...	...
...	...
Class, Instance, Attribute, DataSize, Data1, Data2	16#00C7, 16#0004, 16#0001, 16#0002, 16#45, 16#56

Beispielprogramm

Dieses Beispiel beschreibt das Erstellen des Backups der ENIP Adapter-Parameter zur Ablage auf der Compact-Flash-Karte in der IndraControl-Steuerung.

- Festlegen des Konfigurationsdateinamens "strConfigFileName"
- Die Konfigurationsdatei ist auf der Compact-Flash-Karte (USER-Partition, /ata0b/) der Steuerung zu speichern

Schritte zur Erzeugung und Ablage der Konfigurationsdatei

1. Notepad öffnen und Backup-Parameterinformationen eingeben.

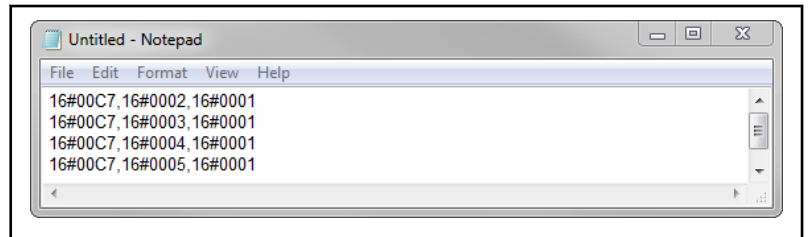


Abb. 7-6: Konfigurationsdatei

2. Konfigurationsdatei als "ConfigFile.enc" speichern.
3. Rechte-Maus-Klick auf die Steuerung im Projektextplorer und navigieren zu **Gerätedaten** ► **Explorer**.

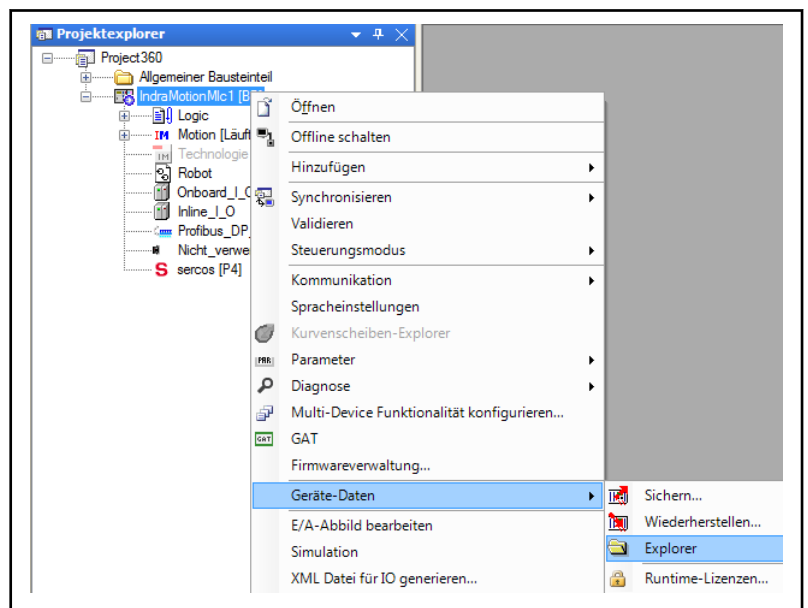


Abb. 7-7: Kontextmenü der Steuerung

4. Navigieren zur User-Partition auf der Compact-Flash-Karte der Steuerung.

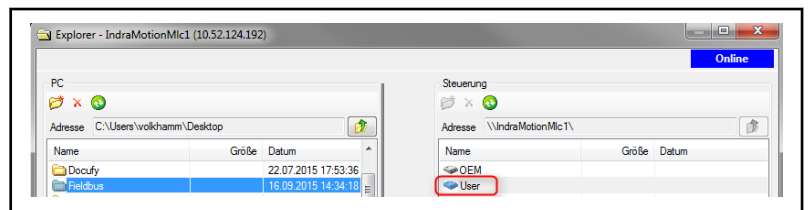


Abb. 7-8: User-Partition

5. Übertragung der Konfigurationsdatei zur User-Partition.

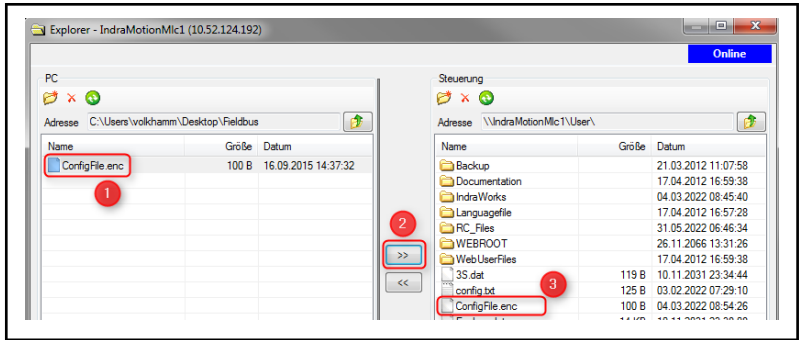


Abb. 7-9: Übertragung in drei Schritten

- Festlegen des Sicherungsdateinamens "strBackupFileName"
- Eingabe des Dateinamen ohne Extension, z. B. "Backupfile"
- Mit steigender Flanke an Eingang "bExecute" beginnt die Bearbeitung:
 - Lesen der durch die Konfigurationsdatei vorgegebenen Parameter vom durch "CIPPath" adressierten Adapter
 - Ablegen der Daten in der Sicherungsdatei

Deklaration

```
PROGRAM ProgIL_ENIPAdapterBackup
VAR
    fbIL_ENIPAdapterBackup: IL_ENIPAdapterBackup;
    bExecute: BOOL;           // FB running when TRUE
    strCIPPath: STRING := '192.168.2.2'; // IP addr. of ENIP adapter
    strConfigFileName: STRING:= 'ConfigFile'; // Config file name
    strBackupFileName: STRING:= 'BackupFile'; // Backup file name
    tTimeout: TIME := T#2S; // Timeout monitoring of FB
    bDone: BOOL;             // FB completed its task
    bActive: BOOL;           // FB working on actual task
    bError: BOOL;            // Indicates an error
    enErrorID: RIL_COMMONTYPES.ERROR_CODE; // Class of error
    enErrorIdent: RIL_COMMONTYPES.ERROR_STRUCT; // Detailed info on error
END_VAR
```

Implementierung

```
fbIL_ENIPAdapterBackup(
    Execute:= bExecute,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> enErrorIdent,
    BusMaster:= IL_BUSMASTER.IL_BUSMASTER_0,
    CIPPath:= strCIPPath,
    ConfigFileName:= strConfigFileName,
    BackupFileName:= strBackupFileName,
    Timeout:= tTimeout);
```

Fehlerbehandlung Der Funktionsbaustein nutzt die **Fehlertabelle ETHERNET_IP_TABLE** (ERROR_TABLE = 16#0151), siehe [Kap. 7.5.7 "Auszug aus der Fehlertabelle ETHERNET_IP_TABLE \(ERROR_TABLE = 16#0151\)"](#) auf Seite 250.

7.3.2 IL_ENIPAdapterRestore

Kurzbeschreibung Der Funktionsbaustein **IL_ENIPAdapterRestore** dient zum Wiederherstellen der Parameter eines Ethernet/IP Adapter-Gerätes mittels azyklischen Schreibens aus der Sicherungsdatei ([Kap. 7.2.4 "IL_ENIPSetAttributeSingle" auf Seite 230](#)).

Zielsystem	Bibliothek/Ordner
MLC	RIL_EthernetIP/POUs/Backup Restore

Tab. 7-18: Referenztablelle

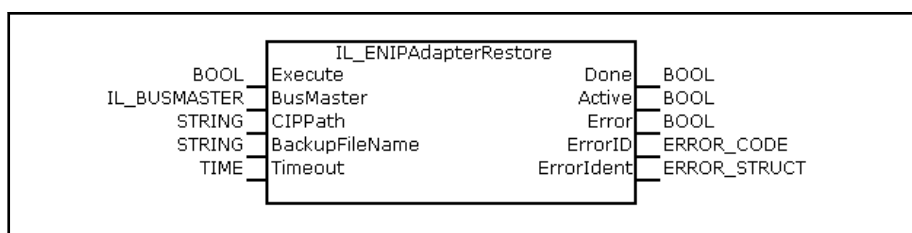


Abb. 7-10: Schnittstellenbeschreibung FB IL_ENIPAdapterRestore

I/O-Typ	Name	Typ	Kommentar
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Scanners in der Reihenfolge des Konfigurationsbaumes. Ist nur ein Ethernet/IP Scanner vorhanden, besitzt dieser Scanner immer den Wert IL_BUSMASTER_0 Siehe auch Kap. 7.5.1 "IL_BUSMASTER" auf Seite 246
	CIPPath	STRING	Adresse des CIP-Teilnehmers als String, in der Form "aaa.bbb.ccc.ddd"
	BackupFileName	STRING	Name der Sicherungsdatei ohne Extension (*.enb)
	Timeout	TIME	Timeout-Überwachung des Funktionsbausteins
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibt die Diagnose im Fehlerfall (aus RIL_Common-Types)
	ErrorIdent	ERROR_STRUCT	Detailldiagnose (Fehlertabelle PROFINETIO_TABLE) "Fehlerbehandlung" auf Seite 239

Tab. 7-19: I/O-Schnittstelle des FB IL_ENIPAdapterRestore

Min-, Max- und Default-Werte: In der folgenden Tabelle sind die Min-, Max- und Default-Werte der Funktionsbausteineingänge aufgelistet.

Name	Typ	Min. Wert	Max. Wert	Defaultwert	Anwendung
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an "Execute"
CIPPath	STRING	Character string	Character string	' '	Steigende Flanke an "Execute"

Name	Typ	Min. Wert	Max. Wert	Defaultwert	Anwendung
BackupFileName	STRING	Character string	Character string	' '	Steigende Flanke an "Execute"
Timeout	TIME	T#0ms	--	T#2s	Steigende Flanke an "Execute"

Tab. 7-20: Min-, Max- und Default-Werte der IL_ENIPAdapterRestore-Eingänge

Funktionsbeschreibung

Der Funktionsbaustein **IL_ENIPAdapterRestore** dient zum Wiederherstellen der Parameter eines Ethernet/IP Adapter-Geräts mittels azyklischen Schreibens aus der Sicherungsdatei ([Kap. 7.2.4 "IL_ENIPSetAttributeSingle" auf Seite 230](#)).

Der Funktionsbaustein liest die Parameter aus der Sicherungsdatei des Ethernet/IP Adapter-Gerätes, die auf der Compact-Flash-Karte der Steuerung abliegt.

Der Adapter wird über "CIPPath" adressiert.

"Class", "Instance" und "Attribute" werden genutzt, um einen einzelnen "Wert" des Adapters zu adressieren.

Bei steigender Flanke an "Execute" liest der Funktionsbaustein alle Parameter aus der Sicherungsdatei aus und schreibt die Parameterwerte auf den Adapter auf die durch "Slot", "SubSlot" und "Index" spezifizierten Stellen.

Nachfolgende Tabelle enthält die Basisstruktur und ein Beispiel einer Sicherungsdatei.

Basisstruktur der Sicherungsdatei	Beispiel der Sicherungsdatei
Class, Instance, Attribute, DataSize, Data	16#00C7, 16#0002, 16#0001, 16#0001, 16#58
Class, Instance, Attribute, DataSize, Data1, Data2	16#00C7, 16#0003, 16#0001, 16#0002, 16#20, 16#82
...	...
...	...
...	...
Class, Instance, Attribute, DataSize, Data1, Data2	16#00C7, 16#0004, 16#0001, 16#0002, 16#45, 16#56



Die Variable am Eingang "BackupFileName" sollte nur aus dem Dateinamen ohne Extension bestehen.

Die Extension (*.enb) wird intern verwaltet.

Beispielprogramm

In diesem Beispielprojekt wird das Wiederherstellen der ENIP Adapter-Geräteparameter von der – in der Compact-Flash-Karte im IndraControl-Gerät gespeicherten – Sicherungsdatei beschrieben.

Vorzugeben sind die IP-Adresse des ENIP-Adapters "CIPPath" und der Dateiname der Sicherungsdatei ohne Extension.

Bei steigender Flanke am Eingang "bExecute" wird der Vorgang ausgeführt: Lesen der Parameter von der Sicherungsdatei und Schreiben auf den durch "CIPPath" adressierten Adapter.

Deklaration

```
PROGRAM ProgIL_ENIPAdapterRestore
VAR
    fbIL_ENIPAdapterRestore: IL_ENIPAdapterRestore;
    bExecute: BOOL // FB running when TRUE
```

```

strCIPPath: STRING := '192.168.2.2';           // IP addr. of ENIP adapter
strFileName: STRING:= 'BackupFile';           // Backup file name
tTimeout: TIME := T#2S;                       // Timeout monitoring of FB
bDone: BOOL;                                  // FB completed its task
bActive: BOOL;                                // FB working on actual task
bError: BOOL;                                 // Indicates an error
enErrorID: RIL_COMMONTYPES.ERROR_CODE;       // Class of error
enErrorIdent: RIL_COMMONTYPES.ERROR_STRUCT;   // Detailed info on error
END_VAR

```

Implementierung

```

fbIL_ENIPAdapterRestore(
  Execute:= bExecute,
  Done=> bDone,
  Active=> bActive,
  Error=> bError,
  ErrorID=> enErrorID,
  ErrorIdent=> enErrorIdent,
  BusMaster:= IL_BUSMASTER.IL_BUSMASTER_0,
  CIPPath:= strCIPPath,
  BackUpFileName:= strFileName,
  Timeout:= tTimeout);

```

Fehlerbehandlung Der Funktionsbaustein **IL_ENIPAdapterRestore** nutzt die **Fehlertabelle ETHERNET_IP_TABLE** (ERROR_TABLE = 16#0151), siehe [Kap. 7.5.7 "Auszug aus der Fehlertabelle ETHERNET_IP_TABLE \(ERROR_TABLE = 16#0151\)" auf Seite 250.](#)

7.4 Diagnose (Diagnosis)

7.4.1 IL_ENIPRemoteAdapterState

Kurzbeschreibung Der Funktionsbaustein **IL_ENIPRemoteAdapterState** liefert die Einfachdiagnose eines Adapters.

Zielsystem	Bibliothek/Ordner
MLC	RIL_EtherNetIP/POUs/Diagnosis
MTX	RIL_EtherNetIP/POUs/Diagnosis

Tab. 7-21: Referenztable

Schnittstellenbeschreibung

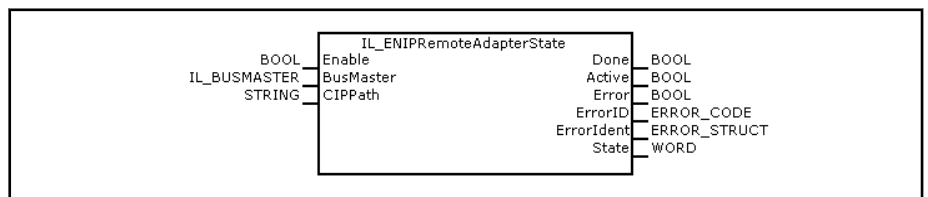


Abb. 7-11: Schnittstellenbeschreibung FB IL_ENIPRemoteAdapterState

I/O-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (permanent, pegelgesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Ethernet/IP Scanners in der Reihenfolge des Konfigurationsbaumes. Ist nur ein Ethernet/IP Scanner vorhanden, besitzt dieser Scanner immer den Wert IL_BUSMASTER_0 siehe Kap. 7.5.1 "IL_BUSMASTER" auf Seite 246
	CIPPath	STRING	Adresse des CIP-Teilnehmers als String in der Form "aaa.bbb.ccc.ddd" z. B. IP-Adr: "192.168.15.8"

I/O-Typ	Name	Datentyp	Beschreibung
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	State	WORD	Statusinformation des Adapters am Feldbus (siehe "Ausgang State" auf Seite 240)

Tab. 7-22: I/O-Schnittstelle des FB IL_ENIPRemoteAdapterState

Ausgang "State"

Im Ausgang "State" ist eine Grobklassifizierung von Diagnosen binär kodiert.

Die Bits sind in der RIL_FeldbusTypes.library als Globale Konstanten "FieldbusStates" enthalten.

Bit-Nr.	Bezeichnung	Beschreibung
0	IL_RBS_SLAVE_NOT_REACHABLE	Teilnehmer existiert nicht oder ist nicht erreichbar
1	IL_RBS_SLAVE_NOT_READY	Teilnehmer nicht bereit
2	IL_RBS_SLAVE_CONFIG_ERROR	Teilnehmer meldet Konfigurationsfehler
3	IL_RBS_SLAVE_DIAGNOSIS	Teilnehmer meldet verfügbare Diagnose
4	IL_RBS_SLAVE_ERROR	Allgemeiner Fehler beim Teilnehmer
5 ... 15	--- keine ---	reserviert

Tab. 7-23: Bitbelegung des Ausganges "State"

Die Eingangsvariablen können die in der folgenden Tabelle dargestellten Werte annehmen.

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an "Enable"
CIPPath	STRING	0.0.0.0	255.255.255.255	0.0.0.0	Steigende Flanke an "Enable"

Tab. 7-24: Min-, Max- und Default-Werte der IL_ENIPRemoteAdapterState-Eingänge

Funktionsbeschreibung

Der Funktionsbaustein ermittelt bei Aktivierung den aktuellen Status eines Adapters.

Diese Einfachdiagnose dient dazu, bei einer aufgetretenen Busmaster-Diagnose mit einer Slave-Diagnose, bzw. einem Adapter-Fehler, diesen Fehler des Adapters abzufragen.

Die Diagnoseinformationen sind als Datenstruktur mit definierten Bits auszugeben.

Fehlerbehandlung Der Funktionsbaustein nutzt die **Fehlertabelle ETHERNET_IP_TABLE** (ERROR_TABLE = 16#0151), siehe [Kap. 7.5.7 "Auszug aus der Fehlertabelle ETHERNET_IP_TABLE \(ERROR_TABLE = 16#0151\)"](#) auf Seite 250.

7.4.2 IL_ENIPRemoteAdapterStateDetails

Kurzbeschreibung Der Funktionsbaustein **IL_ENIPRemoteAdapterStateDetails** liefert die Detaildiagnose eines Adapters. Hierbei wird vom Adapter aus dem "Identity Object" das Attribut 5 "Status" ausgelesen.

Zielsystem	Bibliothek/Ordner
MLC ab 14V06	RIL_EtherNetIP/POUs/Diagnosis
MTX ab 14V06	RIL_EtherNetIP/POUs/Diagnosis

Tab. 7-25: Referenztable

Schnittstellenbeschreibung

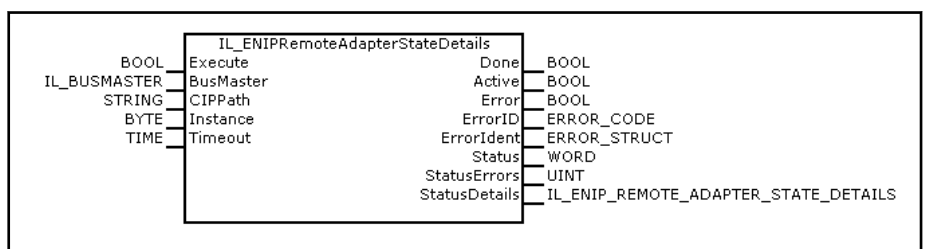


Abb. 7-12: Schnittstellenbeschreibung FB IL_ENIPRemoteAdapterStateDetails

I/O-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Ethernet/IP Scanners in der Reihenfolge des Konfigurationsbaumes. Ist nur ein Ethernet/IP Scanner vorhanden, besitzt dieser Scanner immer den Wert IL_BUSMASTER_0 siehe Kap. 7.5.1 "IL_BUSMASTER" auf Seite 246
	CIPPath	STRING	Adresse des CIP-Teilnehmers als String. In der Form "aaa.bbb.ccc.ddd" z. B. IP-Adr: "192.168.15.8"
	Instance	BYTE	Instanz-Nummer, in der Regel hat ein Adapter nur ein "Identity Object" (Instance = 1)
	Timeout	TIME	Timeout - Überwachung des FBs in ms. T#0ms = Timeout - Überwachung ist nicht aktiv

I/O-Typ	Name	Datentyp	Beschreibung
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig.
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	Status	WORD	Detaillierter Status des Ethernet/IP Adapters (entspricht Attribut 5 des "Identity Objects") siehe Tab. 7-28 "Beschreibung des Status-Attributs des Identity Objects" auf Seite 242
	StatusErrors	UINT	Error-Bits (Bit 8 -11) aus "Status"
	StatusDetails	IL_ENIP_REMOTE_ADAPTER_STATE_DETAILS	Detailstatus (Bit 4 - 7) aus "Status" siehe Kap. 7.5.3 "IL_ENIP_REMOTE_ADAPTER_STATE_DETAILS" auf Seite 248

Tab. 7-26: I/O-Schnittstelle des FB IL_ENIPRemoteAdapterStateDetails

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an "Execute"
CIPPath	STRING	„0.0.0.0“	"255.255.255.255"	"0.0.0.0"	Steigende Flanke an "Execute"
Instance	BYTE	0	255	1	Steigende Flanke an "Execute"
Timeout	TIME	T#0ms	T#3600s	T#2s	Steigende Flanke an "Execute"

Tab. 7-27: Min-, Max- und Default-Werte der IL_ENIPRemoteAdapterStateDetails-Eingänge

Funktionsbeschreibung Der Funktionsbaustein **IL_ENIPRemoteAdapterStateDetails** ermittelt bei Aktivierung einmalig den erweiterten Status eines Adapters. Hierbei wird vom Adapter aus dem "Identity Object" das Attribut 5 "Status" ausgelesen:

Bit	Benennung	Beschreibung
0	Owned	TRUE: der Adapter (oder ein Objekt im Adapter) wird von einem Scanner belegt
1	-	Reserviert
2	Configured	TRUE: die Applikation im Adapter ist konfiguriert (d. h. nicht mehr im Auslieferungszustand)
3	-	Reserviert

Bit	Benennung	Beschreibung
4-7	Extended Device Status	Siehe unten, alternativ ist auch eine herstellerspezifische Definition möglich
8	Minor Recoverable Fault	TRUE: es liegt ein leichter, behebbarer Fehler im Gerät vor; das Gerät ist weiterhin funktionsfähig
9	Minor Unrecoverable Fault	TRUE: es liegt ein leichter, nicht behebbarer Fehler im Gerät vor; das Gerät ist weiterhin funktionsfähig
10	Major Recoverable Fault	TRUE: es liegt ein schwerwiegender, behebbarer Fehler im Gerät vor; das Gerät ist im Fehlerzustand
11	Major Unrecoverable Fault	TRUE: es liegt ein schwerwiegender, nicht behebbarer Fehler im Gerät vor, das Gerät ist im Fehlerzustand
12-15	Extended Device Status 2	Reserviert oder herstellerspezifische Umsetzung

Tab. 7-28: Beschreibung des "Status"-Attributs des "Identity Objects"

Wert	Beschreibung
0	Selbsttest oder unbekannt
1	Firmware-Update aktiv
2	Mindestens eine I/O-Verbindung im Fehlerzustand
3	Keine I/O-Verbindung eingerichtet
4	Gespeicherte Konfiguration korrupt
5	Schwerwiegender Fehler: Bit 10 oder 11 im "Status" ist TRUE
6	Mindestens eine I/O-Verbindung ist aktiv
7	Mindestens eine I/O-Verbindung ist eingerichtet, alle sind im "Idle"-Zustand (ungültige Daten)
8-9	Reserviert
10-15	Herstellerspezifisch

Tab. 7-29: "Extended Device Status" (Bits 4-7) im "Status"-Attribut

Fehlerbehandlung Der Funktionsbaustein nutzt die **Fehlertabelle ETHERNET_IP_TABLE** (ERROR_TABLE = 16#0151), siehe [Kap. 7.5.7 "Auszug aus der Fehlertabelle ETHERNET_IP_TABLE \(ERROR_TABLE = 16#0151\)"](#) auf Seite 250.

7.4.3 IL_ENIPScannerState

Kurzbeschreibung Der Funktionsbaustein **IL_ENIPScannerState** liefert die Einfachdiagnose des Ethernet/IP Scanners.

Diese Einfachdiagnose dient dazu, in einem Polling zyklisch abgefragt zu werden. Erst wenn dieser Funktionsbaustein eine Diagnose meldet, sollte mit dem komplexeren Funktionsbaustein **IL_ENIPRemoteAdapterStateDetails** (siehe [Kap. 7.4.4 "IL_ENIPScannerStateDetails"](#) auf Seite 245) eine Detaildiagnose geholt werden.

Zielsystem	Bibliothek/Ordner
MLC	RIL_EtherNetIP/POUs/Diagnosis
MTX	RIL_EtherNetIP/POUs/Diagnosis

Tab. 7-30: Referenztablelle

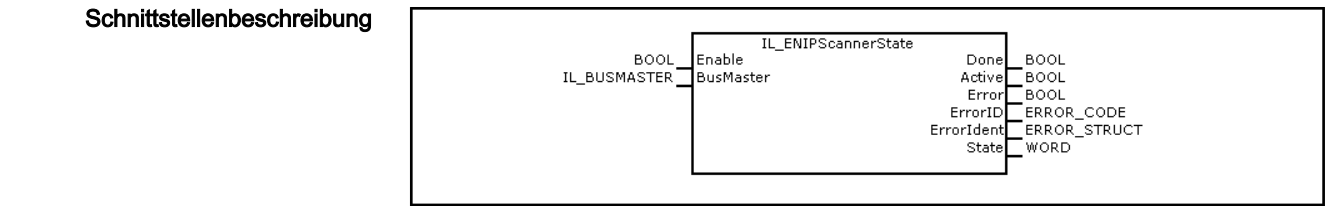


Abb. 7-13: Schnittstellenbeschreibung FB IL_ENIPScannerState

I/O-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (permanent, zustandsgesteuert)
	BusMaster	IL_BUSMASTER	Scanner-ID. Wenn mehrere Scanner gleichzeitig betrieben werden, dann werden diese Scanner durch die Variable "BusMaster" unterschieden siehe Kap. 7.5.1 "IL_BUSMASTER" auf Seite 246
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	ERROR_CODE aus RIL_CommonTypes. Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCT	Fehlerstruktur aus ErrorTable ETHERNET_IP_TABLE
	State	WORD	Status des Ethernet/IP Scanners (siehe "Ausgang State" auf Seite 244)

Tab. 7-31: I/O-Schnittstelle des FB IL_ENIPScannerState

Ausgang "State" Im Ausgang "State" ist eine Grobklassifizierung von Diagnosen binär kodiert. Die Bits sind in der RIL_FeldbusTypes.library als Globale Konstanten "FieldbusStates" enthalten.

BIT-Nr.	Name	Beschreibung
0	IL_FBM_BUS_OFF	Feldbus nicht bereit
1	IL_FBM_MASTER_FAILURE	Fehler des Busmasters
2	IL_FBM_SLAVE NOT_REACHABLE	Ein oder mehrere Feldbusteilnehmer nicht erreichbar
3	IL_FBM_SLAVE_DIAGNOSIS	Ein oder mehrere Feldbusteilnehmer melden Diagnose
4	IL_FBM_SLAVE_ERROR	Ein oder mehrere Feldbusteilnehmer melden Fehler
5 ... 15	--- keine ---	reserviert

Tab. 7-32: Bitbelegung des Ausgangs "State"

Die Eingangsvariablen können die in der folgenden Tabelle dargestellten Werte annehmen.

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an "Enable"

Tab. 7-33: Min-, Max- und Default-Werte der IL_ENIPScannerState-Eingänge

- Funktionsbeschreibung** Mit dem Funktionsbaustein **IL_ENIPScannerState** kann die Einfachdiagnose des Scanners gelesen werden.
- Die Einfachdiagnose kann zyklisch abgefragt werden. Erst, wenn der Funktionsbaustein eine Diagnose meldet, sollte mit dem komplexeren Funktionsbaustein **IL_ENIPRemoteAdapterStateDetails** eine Detaildiagnose geholt werden.
- Die Diagnoseinformationen werden als Bitleiste mit definierten Bits ausgegeben.
- Der "State"-Ausgang zeigt den Zustand des Master an.
- Im Falle einer fehlerfreien Kommunikation wird der Wert 16#0000 am Ausgang "State" zurückgemeldet.
- Ist der Ausgang "State" ungleich 16#0000, kann entsprechend der definierten Ausgangsbits im Anwenderprogramm eine Fehlerreaktion eingeleitet werden.
- Fehlerbehandlung** Der Funktionsbaustein nutzt die **Fehlertabelle ETHERNET_IP_TABLE** (ERROR_TABLE = 16#0151), siehe [Kap. 7.5.7 "Auszug aus der Fehlertabelle ETHERNET_IP_TABLE \(ERROR_TABLE = 16#0151\)" auf Seite 250](#).

7.4.4 IL_ENIPScannerStateDetails

- Kurzbeschreibung** Mit dem Funktionsbaustein **IL_ENIPScannerStateDetails** kann die folgende Detaildiagnose des Scanners gelesen werden:
- Zustand des Kommunikationsstacks
 - Zustand der Buskommunikation
 - Fehlerzähler
 - Überblick über konfigurierte, aktive und eine Diagnose meldende Adapter

Zielsystem	Bibliothek/Ordner
MLC	RIL_EtherNetIP/POUs/Diagnosis
MTX	RIL_EtherNetIP/POUs/Diagnosis

Tab. 7-34: Referenztable

Schnittstellenbeschreibung

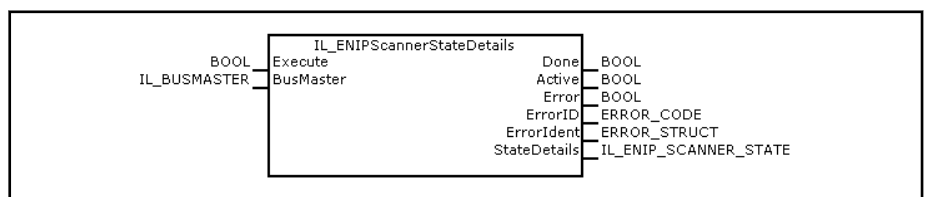


Abb. 7-14: Schnittstellenbeschreibung FB IL_ENIPScannerStateDetails

I/O-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des Ethernet/IP Scanners in der Reihenfolge des Konfigurationsbaumes. Ist nur ein Ethernet/IP Scanner vorhanden, besitzt dieser Scanner immer den Wert IL_BUSMASTER_0 siehe Kap. 7.5.1 "IL_BUSMASTER" auf Seite 246

I/O-Typ	Name	Datentyp	Beschreibung
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	StateDetails	IL_ENIP_SCANNER_STATE	Datenstruktur des Scanner-Status siehe Kap. 7.5.4 "IL_ENIP_SCANNER_STATE" auf Seite 249

Tab. 7-35: I/O-Schnittstelle des FB `IL_ENIPScannerStateDetails`

Die Eingangsvariablen können die in der folgenden Tabelle dargestellten Werte annehmen.

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an "Execute"

Tab. 7-36: Min-, Max- und Default-Werte der `IL_ENIPScannerStateDetails`-Eingänge

Funktionsbeschreibung Mit dem Funktionsbaustein `IL_ENIPScannerStateDetails` kann die folgende Detaildiagnose des Scanners gelesen werden:

- Zustand des Kommunikationsstacks
- Zustand der Buskommunikation
- Fehlerzähler und
- Überblick über konfigurierte, aktive und eine Diagnose meldende Adapter

Dieser Funktionsbaustein sollte aufgerufen werden, wenn bei der Einfachdiagnose ein Fehler angezeigt wurde.

Fehlerbehandlung Der Funktionsbaustein nutzt die **Fehlertabelle** `ETHERNET_IP_TABLE` (`ERROR_TABLE = 16#0151`), siehe [Kap. 7.5.7 "Auszug aus der Fehlertabelle ETHERNET_IP_TABLE \(ERROR_TABLE = 16#0151\)" auf Seite 250](#).

7.5 Datentypen (DataTypes)

7.5.1 IL_BUSMASTER

Kurzbeschreibung Der Aufzählungstyp `IL_BUSMASTER` aus der `RIL_FeldbusTypes.library` dient dazu, den jeweiligen Ethernet/IP Scanner auszuwählen. In einem MLC-System können neben der onboard Realtime-Ethernet-Schnittstelle über Funktionsmodule weitere Scanner vorhanden sein.

Die Ethernet/IP Scanner werden anhand ihrer aufsteigenden Reihenfolge in der Konfiguration unterschieden.

Die Indizierung der Controller geschieht dabei in folgender Reihenfolge:

- Onboard-Master

- Funktionsmodul 1
- Funktionsmodul 2
- Funktionsmodul 3
- Funktionsmodul 4



Ist beispielsweise der Onboard-Master kein Ethernet/IP Scanner und ist auf dem Funktionsmodul 1 ein Ethernet/IP Scanner konfiguriert, dann erhält der Controller auf dem Funktionsmodul 1 den Wert IL_BUSMASTER_0.

Element	Wert	Beschreibung
IL_BUSMASTER_0	0	Erster Ethernet/IP Scanner
IL_BUSMASTER_1	1	Zweiter Ethernet/IP Scanner
IL_BUSMASTER_2	2	Dritter Ethernet/IP Scanner
IL_BUSMASTER_3	3	Vierter Ethernet/IP Scanner
IL_BUSMASTER_4	4	Fünfter Ethernet/IP Scanner

Tab. 7-37: Aufzählungstyp IL_BUSMASTER

7.5.2 IL_ENIP_IDENTITY_ATTRIBUTES

Die Struktur **IL_ENIP_IDENTITY_ATTRIBUTES** enthält die Pflichtattribute des "Identity Objects", die mittels "Get-Attribute-All" gelesen werden.

Vendor ID	UINT	Identifizierung des Herstellers, z. B. Bosch Rexroth = 287
Device Type	UINT	Identifizierung des Geräte-Typs (aus CIP-Spezifikation)
Product Code	UINT	Identifizierung eines bestimmten Produkts (herstellerspezifisch)
Major Revision	BYTE	Major Revision
Minor Revision	BYTE	Minor Revision
Status	WORD	Status (siehe nachfolgende Tabellen)
Serial Number	UDINT	Seriennummer
Product Name	STRING(32)	Produktname

Tab. 7-38: Pflicht-Attribute des "Identity Object"

Bit	Benennung	Beschreibung
0	Owned	TRUE: der Adapter (oder ein Objekt im Adapter) wird von einem Scanner belegt
1	-	Reserviert
2	Configured	TRUE: die Applikation im Adapter ist konfiguriert (d. h. nicht mehr im Auslieferungszustand)
3	-	Reserviert
4-7	Extended Device Status	Siehe unten, alternativ ist auch eine herstellerspezifische Definition möglich
8	Minor Recoverable Fault	TRUE: es liegt ein leichter, behebbarer Fehler im Gerät vor; das Gerät ist weiterhin funktionsfähig
9	Minor Unrecoverable Fault	TRUE: es liegt ein leichter, nicht behebbarer Fehler im Gerät vor; das Gerät ist weiterhin funktionsfähig

Bit	Benennung	Beschreibung
10	Major Recoverable Fault	TRUE: es liegt ein schwerwiegender, behebbarer Fehler im Gerät vor; das Gerät ist im Fehlerzustand
11	Major Unrecoverable Fault	TRUE: es liegt ein schwerwiegender, nicht behebbarer Fehler im Gerät vor, das Gerät ist im Fehlerzustand
12-15	Extended Device Status 2	Reserviert oder herstellerspezifische Umsetzung

Tab. 7-39: Beschreibung des "Status"-Attributs des "Identity Objects"

Wert	Beschreibung
0	Selbsttest oder unbekannt
1	Firmware-Update aktiv
2	Mindestens eine I/O-Verbindung im Fehlerzustand
3	Keine I/O-Verbindung eingerichtet
4	Gespeicherte Konfiguration korrupt
5	Schwerwiegender Fehler: Bit 10 oder 11 im "Status" ist TRUE
6	Mindestens eine I/O-Verbindung ist aktiv
7	Mindestens eine I/O-Verbindung ist eingerichtet, alle sind im "Idle"-Zustand (ungültige Daten)
8-9	Reserviert
10-15	Herstellerspezifisch

Tab. 7-40: "Extended Device Status" (Bits 4-7) im "Status"-Attribut

7.5.3 IL_ENIP_REMOTE_ADAPTER_STATE_DETAILS

Der Aufzählungsdatentyp **IL_ENIP_REMOTE_ADAPTER_STATE_DETAILS** erlaubt die Anzeige der Bits 4...7 "Detailstatus" eines Ethernet/IP Adapters (entspricht Attribut 5 des "Identity Objects") als Text.

Name	Wert	Beschreibung
SelfTestingOrUnknown	0	Selbsttest oder unbekannt
FwUpdateInProgress	1	Firmwareupdate läuft
AtLeastOneFaultedIoConnection	2	Mindestens eine fehlerhafte IO-Verbindung
NoIoConnectionEstablished	3	Keine IO-Verbindung aufgebaut
NonVolatileConfigurationBad	4	
MajorFault	5	
AtLeastOneIoConnectionInRun	6	Mindestens eine IO-Verbindung im Zustand "Run"
AtLeastOneIoConnectionInIdle	7	Mindestens eine IO-Verbindung im Zustand "Idle"
Reserved_8	8	reserviert
Reserved_9	9	reserviert
Vendor_Specific_10	10	herstellerspezifisch_10
Vendor_Specific_11	11	herstellerspezifisch_11
Vendor_Specific_12	12	herstellerspezifisch_12

Name	Wert	Beschreibung
Vendor_Specific_13	13	herstellerspezifisch_13
Vendor_Specific_14	14	herstellerspezifisch_14
Vendor_Specific_15	15	herstellerspezifisch_15

Tab. 7-41: Detailstatus (Bit 4 - 7) aus "Status".

7.5.4 IL_ENIP_SCANNER_STATE

Kurzbeschreibung Die Datenstruktur **IL_ENIP_SCANNER_STATE** beinhaltet Diagnoseinformationen über:

- Zustand des Kommunikationsstacks
- Zustand der Buskommunikation
- Fehlerzähler
- Überblick über konfigurierte, aktive und eine Diagnose meldende Slaves

Strukturelement	Typ	Beschreibung
CommunicationCOS	DWORD	Status des Kommunikationsstacks, siehe unten
CommunicationState	IL_FBUS_COMMUNICATION_STATE	Status der Kommunikation siehe Kap. 5.8.2 "IL_FBUS_COMMUNICATION_STATE" auf Seite 190
Version	UINT	Version der Datenstruktur der Diagnoseinfo
Watchdog	UINT	Watchdog-Timeout zwischen PLC und Feldbusanbindung in ms
ErrorCount	UDINT	Fehlerzähler. Zählt die Anzahl der aufgetretenen Kommunikationsfehler (sowohl interne Fehler als auch Kommunikationsfehler)
SlaveState	IL_FBUS_SLAVE_STATE	Zeigt an, ob der Scanner eine aktive Verbindung zu allen konfigurierten Slaves besitzt siehe Kap. 5.8.3 "IL_FBUS_SLAVE_STATE" auf Seite 190
NumOfConfigSlaves	UDINT	Anzahl an konfigurierten Slaves
NumOfActiveSlaves	UDINT	Anzahl aktiver Slaves, d. h. Slaves, zu denen der Master eine Kommunikationsverbindung aufgebaut hat
NumOfDiagSlaves	UDINT	Anzahl an Slaves mit aktiver Diagnose

Tab. 7-42: Datenstruktur IL_ENIP_SCANNER_STATE

Im Ausgang "CommunicationCOS" ist der Status des Kommunikationsstacks binär kodiert.

Die darin enthaltene Information ist normalerweise nur in Problemfällen notwendig und wird dann von Bosch Rexroth interpretiert. Die Bits sind als Defines in der Library enthalten (ENIP_Defines).

Ausgang "CommunicationCOS"

Bit-Nr.	Bezeichnung	Beschreibung
0	COMM_COS_READY	Protokollstack ist gestartet und wartet auf Konfiguration
1	COMM_COS_RUN	Protokollstack ist konfiguriert und wartet auf Busverbindung
2	COMM_COS_BUS_ON	Protokollstack darf auf dem Bus kommunizieren
3	COMM_COS_CONFIG_LOCKED	Buskonfiguration ist gelockt und kann nicht verändert werden

Bit-Nr.	Bezeichnung	Beschreibung
4	COMM_COS_CONFIG_NEW	Neue Buskonfiguration ist vorhanden, aber noch nicht initialisiert
5	COMM_COS_RESTART_REQUIRED	Neustart der Kommunikation notwendig
6	COMM_COS_RESTART_REQUIRED_ENABLE	Neustart der Kommunikation möglich
....		übrige Werte: FALSE

7.5.5 IL_FBUS_COMMUNICATION_STATE

Kurzbeschreibung Der Aufzählungstyp **IL_FBUS_COMMUNICATION_STATE** dient dazu, den Status der Feldbus-Kommunikation darzustellen.

Element	Wert	Beschreibung
COMM_STATE_NOT_CONFIGURED	16#00000001	Kommunikation ist nicht konfiguriert
COMM_STATE_STOP	16#00000002	Kommunikation befindet sich im Stopp-Zustand
COMM_STATE_IDLE	16#00000003	Kommunikation befindet sich im Idle-Zustand
COMM_STATE_OPERATE	16#00000004	Kommunikation im zyklischen Betrieb

Tab. 7-43: Aufzählungstyp **IL_FBUS_COMMUNICATION_STATE**

7.5.6 IL_FBUS_SLAVE_STATE

Der Aufzählungstyp **IL_FBUS_SLAVE_STATE** dient dazu, den Status der Kommunikationsbeziehungen zu den Slaves darzustellen, d. h. der Status zeigt an, ob der Feldbus-Master eine aktive Verbindung zu allen konfigurierten Slaves besitzt.

Element	Wert	Beschreibung
SLAVE_STATE_OK	16#00000001	Kommunikation zu allen aktiven Slaves vorhanden
SLAVE_STATE_FAILED	16#00000002	Kommunikationsfehler mindestens zu einem Slave

Tab. 7-44: Aufzählungstyp **IL_FBUS_SLAVE_STATE**

7.5.7 Auszug aus der Fehlertabelle ETHERNET_IP_TABLE (ERROR_TABLE = 16#0151)

Die folgenden Error-Codes gelten für die Funktionsbausteine der Bibliothek RIL_EtherNetIP.

Die Error-Codes werden am Ausgang "ErrorIdent" als ERROR_STRUCT mit den Elementen "Additional1" und "Additional2" gemeldet.

ErrorID	Additional1	Additional2	
INPUT_INVALID_ERROR	16#00000001	16#00000000	Eingang "BusMaster" ist ungültig
		16#00000001	Eingang "CIPPath" ist ungültig
		16#00000002	Eingang "ServiceCode" darf nicht "0" sein
		16#00000003	Eingang "Class" darf nicht "0" sein
		16#00000004	Eingang "Attribute" darf nicht "0" sein
		16#00000005	Eingang "SizeOfValue" und "ValueAdr" dürfen nicht "0" sein
		16#00000006	Der Datenbereich am Eingang "ValueAdr" / "SizeOfValue" ist zu klein für die empfangenen Daten
		16#00000007	Es liegt ein ungültiger Eingangs-Parameter vor (interner Fehler)
SYSTEM_ERROR	16#00000002	(intern)	Interner Fehler (Interface Pointer)
COMMUNICATION_ERROR	16#00000003	(intern)	Überprüfen Sie die Eingangs-Parameter "ServiceCode", "Class" u. "Attribute", evtl. wird das adressierte CIP-Objekt vom Adapter auch nicht unterstützt. Interner Fehler (netX), siehe auch Tab. 7-46 "Spezielle Fehlercodes der EtherNet/IP_Table (ERROR_TABLE = 16#0151)" auf Seite 252
SYSTEM_ERROR	16#00000101	(intern)	Interner Fehler (Scanner nicht gefunden)
SYSTEM_ERROR	16#00000102	(intern)	Interner Fehler (Scanner Status)
SYSTEM_ERROR	16#00000103	(intern)	Interner Fehler (IO-Driver nicht bereit)
COMMUNICATION_ERROR	16#00000104	(intern)	Timeout: a) überprüfen Sie die Adresse am Eingang "CIPPath" (oder das adressierte Gerät ist nicht erreichbar) b)erhöhen Sie den Wert am Eingang "Timeout"
STATE_MACHINE_ERROR	16#00000105	(intern)	Interner Fehler (State Machine)
SYSTEM_ERROR	16#00000106	(intern)	Interner Fehler (Interface nicht aktiviert)
COMMUNICATION_ERROR	16#00000107	(intern)	Interner Fehler (Retry)
INPUT_RANGE_ERROR	16#00000108	(intern)	Der Adapter wurde nicht gefunden, überprüfen Sie den Eingang "CIPPath"
SYSTEM_ERROR	16#00000109	(intern)	Interner Fehler (Parameter nicht gefunden)
CALCULATION_ERROR	16#00000110	(intern)	Interner Fehler (Berechnungsfehler)
RESOURCE_ERROR	16#00000111	(intern)	Reservierter Puffer zu klein
INPUT_INVALID_ERROR	16#3002	16#0002	Ungültiger Sicherungsdateiname
ACCESS_ERROR	16#3003	16#0002	Ungültiges Format der Sicherungsdatei
ACCESS_ERROR	16#3003	16#0003	Sicherungsdatei kann nicht geöffnet werden
ACCESS_ERROR	16#3003	16#0004	Lesen oder Schreiben der Sicherungsdatei nicht möglich

Tab. 7-45: Error-Codes der RIL_EtherNetIP

Nachfolgende Tabelle enthält spezielle "Additional2" Error-Codes, die beim internen Fehler "Additional1" = 16#0003 auftreten können.

Additional2	
16#C01F0034	Ethernet/IP Objekt-Task während eines Resets
16#C00800C8	Task ist nicht initialisiert
16#C00800C9	Task ist beschäftigt (intern)
16#C00800D2	Konfigurationsdatenbasis nicht verfügbar
Fehlercodes Toolkit	
16#800A0012	Empfangsdatenpuffer zu klein
16#800C0017	Nachrichtenpuffer für Sendenachrichten ist voll
16#800C0019	Keine netX-Nachricht empfangen
16#800C001A	Nachrichtenpuffer zu klein für die eingehende Nachricht

Tab. 7-46: *Spezielle Fehlercodes der EtherNet/IP_Table (ERROR_TABLE = 16#0151)*

8 RIL_EtherNetIPAdapter.library

8.1 Einführung und Übersicht

Die Steuerungen können als Feldbuslave betrieben werden. Im Falle von EtherNet/IP nennt man diesen speziellen Feldbuslave EtherNet/IP adapter.

Die vorliegende Bibliothek beschreibt die Schnittstellen der EtherNet/IP adapter zum IEC-Anwenderprogramm. Über diese können die EtherNet/IP-Teilnehmer diagnostiziert werden.

Es werden die Methoden IL_EnableDoneBase und IL_ExecuteDoneBase verwendet und erweitert oder überschrieben

Zwei Schnittstellentypen werden unterstützt:

- EtherNet/IP adapter über die Engineering-Schnittstelle
- EtherNet/IP adapter über die Onboard- bzw. Funktionsmodul-Schnittstelle

Anwendungsgebiete und Funktionsbausteine von RIL_EtherNetIPAdapter.library

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine (POUs)	Beschreibung
Diagnose (Diagnosis), über die Onboard- bzw. Funktionsmodul-Schnittstelle	
IL_ENIPAdapterState, Seite 253	Diagnose: einfacher Status des Adapters
IL_ENIPAdapterStateDetails, Seite 255	Diagnose: Detailstatus des Ethernet/IP-Stacks des Adapters
Diagnose (Diagnosis/EngineeringPortAdapter), über die Engineering-Schnittstelle	
IL_Status auf Seite 256	Diagnose der zyklischen Kommunikation.
Mit Datentypen:	
IL_ENIP_ADAPTER_STATE auf Seite 259	Struktur, beinhaltet Diagnoseinformationen über den Zustand des Kommunikationsstacks, den Zustand der Buskommunikation, den Fehlerzähler, sowie einen Überblick über konfigurierte, aktive und eine Diagnose meldende Slaves.

Tab. 8-1: RIL_EtherNetIPAdapter-Bibliothek

Auswahl des Slaves Der Aufzählungstyp IL_BUSSLAVE aus der Bibliothek RIL_FeldbusTypes dient dazu, den jeweiligen EtherNet/IP adapter auszuwählen.
Siehe auch: [IL_BUSSLAVE, Seite 258](#)

8.2 Diagnose (Diagnosis)

8.2.1 IL_ENIPAdapterState

Kurzbeschreibung Dieser Funktionsbaustein gibt eine einfache Diagnose des EtherNet/IP Adapters zurück.
Diese Einfachdiagnose dient dazu, in einem Polling zyklisch abgefragt zu werden.

Zielsystem	Bibliothek /Ordner
MLC	RIL_EtherNetIPAdapter/ POU's/ Diagnosis
MTX	RIL_EtherNetIPAdapter/ POU's/ Diagnosis

Tab. 8-2: Referenztabelle

Schnittstellenbeschreibung

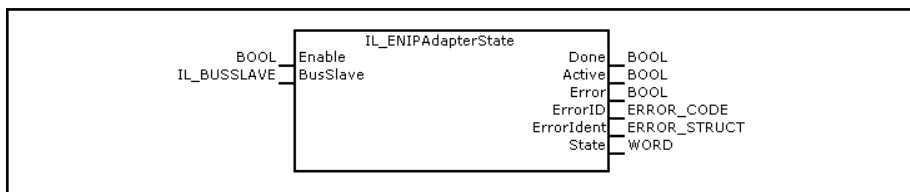


Abb. 8-1: Schnittstelle FB IL_ENIPAdapterState

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (permanent, pegelgesteuert)
	BusSlave	IL_BUSSLAVE	Instanz des Adapters in der Reihenfolge des Konfigurationsbaumes. Ist nur ein EtherNet/IP Adapter vorhanden, besitzt dieser immer den Wert IL_BUSSLAVE_0. siehe IL_BUSSLAVE auf Seite 258
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall (aus RIL_CommonTypes)
	ErrorIdent	ERROR_STRUCT	Detailldiagnose (ErrorTable ETHERNETIP_TABLE)
	State	WORD	Status des EtherNet/IP Adapters, siehe unten

Tab. 8-3: E/A-Schnittstelle FB IL_ENIPAdapterState

Im Ausgang "State" ist eine Grobklassifizierung von Diagnosen binär kodiert. Die Bits sind in der RIL_FeldbusTypes.library als Globale Konstanten "FieldbusStates" enthalten.

Ausgang "State"

Bit-Nr.	Bezeichnung	Beschreibung
0	IL_FBS_BUS_OFF	Keine Feldbus Kommunikation
2	IL_FBS_CONFIG_ERROR	Fehlerhafte Konfiguration durch den Feldbus-Master
4	IL_FBS_SLAVE_ERROR	Fehler des Gerätes

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
BusSlave	IL_BUSSLAVE	IL_BUSSLAVE_0	IL_BUSSLAVE_4	IL_BUSSLAVE_0	Steigende Flanke von Enable

Tab. 8-4: Min-/Max und Default-Werte der Eingänge IL_ENIPAdapterState

Funktionsbeschreibung Der Funktionsbaustein gibt die Statusinformationen des EtherNet/IP Adapters als Bitleiste in ‚State‘ zurück.

Die Einfachdiagnose wird zyklisch abgefragt. Wenn dieser Funktionsbaustein eine Diagnose meldet, sollte mit dem Funktionsbaustein [IL_ENIPAdapterStateDetails](#) auf Seite 255 eine Detaildiagnose geholt werden.

Fehlerbehandlung Der Funktionsbaustein nutzt die **Fehlertabelle ETHERNET_IP_TABLE** (ERROR_TABLE = 16#0151). Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	
SYSTEM_ERROR	16#0101	interne Verwendung	Adapter nicht in der Konfigurationstabelle gefunden

Tab. 8-5: Fehlercodes FB IL_ENIPAdapterState

8.2.2 IL_ENIPAdapterStateDetails

Kurzbeschreibung Der Funktionsbaustein ermittelt den detaillierten Status des EtherNet/IP adapter-Stacks. Er liefert Informationen über

- den Zustand des Kommunikationsstacks
- den Zustand der Buskommunikation
- die letzten Fehlerwerten

Zielsystem	Bibliothek /Ordner
MLC	RIL_EtherNetIPAdapter/ POU's/ Diagnosis
MTX	RIL_EtherNetIPAdapter/ POU's/ Diagnosis

Tab. 8-6: Referenztable

Schnittstellenbeschreibung

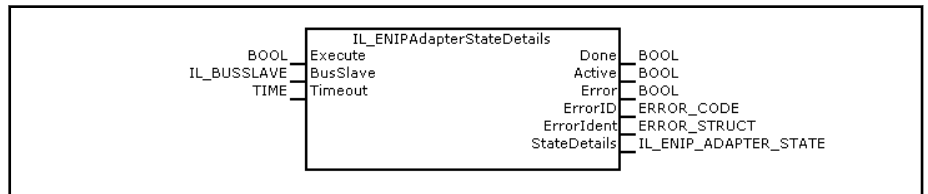


Abb. 8-2: IL_ENIPAdapterStateDetails FB-Schnittstelle

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusSlave	IL_BUSSLAVE	Instanz des Feldbus-Slaves in der Reihenfolge des Konfigurationsbaumes siehe IL_BUSSLAVE auf Seite 258
	Timeout	TIME	Timeout - Überwachung des FBs in ms. T#0ms = Timeout - Überwachung ist nicht aktiv
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers

E/A-Typ	Name	Datentyp	Beschreibung
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	AdapterState	IL_ENIP_ADAPTER_STATE	Detaillierte Statusinformationen des Adapter-Stacks siehe auch IL_ENIP_ADAPTER_STATE auf Seite 259

Tab. 8-7: E/A-Schnittstelle FB IL_ENIPAdapterStateDetails

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusSlave	IL_BUSSLAVE	IL_BUSSLAVE_0	IL_BUSSLAVE_4	IL_BUSSLAVE_0	Steigende Flanke von Execute
Timeout	TIME	T#0ms	T#3600s	T#2s	Steigende Flanke an Execute

Tab. 8-8: Min-/Max und Default-Werte der Eingänge IL_ENIPAdapterStateDetails

Funktionsbeschreibung Der Funktionsbaustein ermittelt bei Aktivierung einmalig den aktuellen Stand des Status des Adapters.

Fehlerbehandlung Der Funktionsbaustein nutzt die **Fehlertabelle ETHERNETIP_TABLE** (ERROR_TABLE = 16#0151).

Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	
SYSTEM_ERROR	16#0101	interne Verwendung	Scanner nicht in der Konfigurationstabelle
COMMUNICATION_ERROR	16#0104	0	Zeitüberschreitung
STATE_MACHINE_ERROR	16#0105	0	Fehler in der Statemaschine
COMMUNICATION_ERROR	16#0003	<i>EtherNet/IP adapter Diagnose</i> Allgemeine Fehlercodes, EtherNet/IP adapter , Seite 260, bzw. Spezielle Fehlercodes, EtherNet/IP adapter, Seite 261.	

Tab. 8-9: Fehlercodes FB IL_ENIPAdapterStateDetails

8.3 Diagnose (Diagnosis/EngineeringPortAdapter)

8.3.1 IL_Status

Kurzbeschreibung Dieser Funktionsbaustein gibt eine einfache Diagnose des EtherNet/IP adapters (Engineering port) zurück.

Diese Einfachdiagnose dient dazu, in einem Polling zyklisch abgefragt zu werden

Zielsystem	Bibliothek /Ordner
MLC	RIL_EtherNetIPAdapter/ POU's/ Diagnosis/EngineeringPortAdapter
MTX	RIL_EtherNetIPAdapter/ POU's/ Diagnosis/EngineeringPortAdapter

Tab. 8-10: Referenztable

Schnittstellenbeschreibung

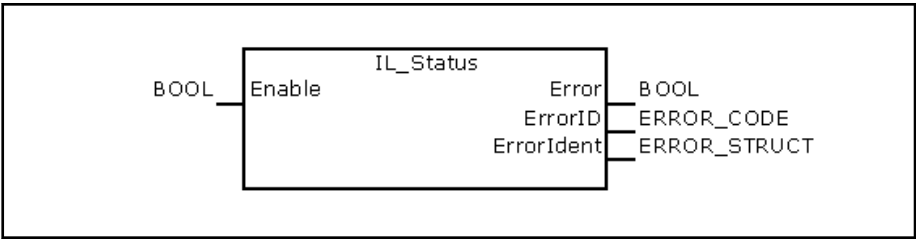


Abb. 8-3: IL_Status

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (permanent, pegelgesteuert)
VAR_OUTPUT	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall (aus RIL_CommonTypes)
	ErrorIdent	ERROR_STRUCT	Detaildiagnose (ErrorTable ETHERNETIP_TABLE)

Tab. 8-11: Schnittstelle von IL_Status

Min-/Max und Default-Werte der Eingänge

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich

Tab. 8-12: Min-/Max und Default-Werte der Eingänge IL_Status

Funktionsbeschreibung Bei TRUE an Enable, wird bei jedem Aufruf des FBs der Zustand der zyklischen EtherNet/IP-Kommunikation abgefragt.

Wenn ein Fehler der zyklischen Kommunikation festgestellt wird, geht der Zustand des Ausgangs „Error“ auf TRUE. Anhand der Fehlermeldung in Additional1 und Additional2 ist die Ursache des Kommunikationsfehlers ersichtlich.

Fehlerbehandlung Der Funktionsbaustein nutzt die **Fehlertabelle ETHERNETIP_TABLE** (ERROR_TABLE = 16#0151).

Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	
COMMUNICATION_ERROR	16#10000001	16#00000000	NoCyclicCom: Der Scanner hat mit IndraLogic-EtherNet/IP adapter noch keine zyklische Kommunikation aufgebaut.
COMMUNICATION_ERROR	16#10000002	16#00000000	Idle: EtherNet/IP scanner hat zyklische Kommunikation gestartet, aber das Idle-Flag im zyklischen Kanal gesetzt. Hiermit signalisiert der Scanner, dass sein Ausgangsabbild nicht gültig ist.

ErrorID	Additional1	Additional2	
COMMUNICATION_ERROR	16#10000003	16#00000000	Timeout: Zeitüberwachung der zyklischen Kommunikation meldet Fehler.
COMMUNICATION_ERROR	16#10000004	16#00000000	Closed: Zyklische Kommunikation wurde durch den EtherNet/IP scanner aktiv beendet.

Tab. 8-13: Fehlercodes FB IL_Status

Beispiel Im folgenden Beispiel wird gezeigt, wie der FB IL_Status anzuwenden ist.

Beispiel

```

(*****
** Variables for IL_Status
*****
PROGRAM PLC_PRG
VAR
    fbDiag:          IL_Status;
    diCtrValidInput: DINT;
    ...
END_VAR;
...
(* *****)
** Program
(* *****)
fbDiag(Enable:=TRUE);
IF(Diag.Error = TRUE) THEN
    (*Insert error handling here.*)
...
fbDiag(Enable:=FALSE ); (*Error (Enable) reset*)
ELSE
    (*Valid data: Insert input/output data processing here *)
    diCtrValidInput := diCtrValidInput +1;
END_IF
...

```

8.4 Datentypen (DataTypes)

8.4.1 IL_BUSSLAVE

Kurzbeschreibung

Der Aufzählungstyp IL_BUSSLAVE aus der RIL_FeldbusTypes.library dient dazu, den jeweiligen Feldbus-Slave auszuwählen. In einer Steuerung können neben der onboard Realtime-Ethernet-Schnittstelle über Funktionsmodule weitere EtherNet/IP Adapter vorhanden sein.

Die Geräte werden anhand ihrer aufsteigenden Reihenfolge in der Konfiguration unterschieden. Die Indizierung der Geräte geschieht dabei in der Reihenfolge

- Onboard
- Funktionsmodul 1
- Funktionsmodul 2
- Funktionsmodul 3
- Funktionsmodul 4

Element	Wert	Beschreibung
IL_BUSSLAVE_0	0	Erstes Gerät
IL_BUSSLAVE_1	1	Zweites Gerät
IL_BUSSLAVE_2	2	Drittes Gerät

Element	Wert	Beschreibung
IL_BUSSLAVE_3	3	Viertes Gerät
IL_BUSSLAVE_4	4	Fünftes Gerät

Tab. 8-14: Elemente des Aufzählungstyps IL_BUSSLAVE

8.4.2 IL_ENIP_ADAPTER_STATE

Kurzbeschreibung

Die Datenstruktur IL_ENIP_ADAPTER_STATE beinhaltet die Statusinformationen des EtherNet/IP Adapter Stacks.

Strukturelement	Typ	Beschreibung
LastGRC	UDINT	Letzter CIP General Error Code, Seite 262
LastERC	UDINT	Letzter Extended Error Code, Seite 263
CurrentConnection	UDINT	aufgebaute Verbindungen
CommunicationState	UDINT	Netzwerkstatus des Kommunikationskanals, siehe unten
CommunicationError	UDINT	Aktueller Fehlercode des Kommunikationskanals; Die Bedeutung des Fehlers ist der Fehlertabelle zu entnehmen, siehe unten.

Tab. 8-15: Datenstruktur IL_ENIP_ADAPTER_STATE

Das Element "CommunicationState" der Datenstruktur IL_ENIP_ADAPTER_STATE beschreibt den Zustand der Kommunikation des EtherNet/IP Adapters.

Strukturelement "CommunicationState"

Wert	Bezeichnung	Beschreibung
16#00000000	UNKNOWN	Kommunikationszustand ist nicht bekannt
16#00000001	NOT_CONFIGURED	Kommunikationskanal wurde nicht konfiguriert
16#00000002	STOP	Stop
16#00000003	IDLE	Idle
16#00000004	OPERATE	EtherNet/IP-Kommunikation

Das Element "CommunicationError" der Datenstruktur IL_ENIP_ADAPTER_STATE beschreibt das aktuelle Fehlercode des Kommunikationskanals.

Strukturelement "CommunicationError"

Wert	Bezeichnung	Beschreibung
16#00000000	SUCCESS	Kommunikationskanal läuft
16#C0000140	NETWORK FAULT	Netzwerkfehler
16#C0000141	CONNECTION CLOSED	Kommunikation wurde beendet
16#C0000142	CONNECTION TIMED OUT	Zeitüberwachung der Kommunikation meldet Fehler.
16#C0000143	LONELY NETWORK	einsames Netzwerk
16#C0000144	DUPLICATE NODE	doppelte Knoten
16#C0000145	CABLE DISCONNECT	Kabel wurde abgezogen

8.4.3 Allgemeine Fehlercodes, EtherNet/IP adapter

Die dargestellten Fehlercodes gelten für ENIP Adapter. Diese Fehlercodes sind allgemeine Fehlercodes, d.h. sie sind busunabhängig.

Die Fehlercodes werden als ERROR_STRUCT, Additional2 bei ERROR_STRUCT, **Additional1= 16#0003** gemeldet.

Additional2	
16#800A0001	Ungültiger Zeiger (NULL) an Treiber übergeben
16#800A0002	Keine Steckkarte mit dem angegebenen Namen/Index verfügbar
16#800A0003	Kein Kanal mit dem angegebenen Index verfügbar
16#800A0004	Ungültiges Handel an Treiber übergeben
16#800A0005	Ungültiger Parameter
16#800A0006	Ungültiger Befehl
16#800A0007	Ungültige Puffergröße
16#800A0008	Ungültige Zugriffsgröße
16#800A0009	Funktion fehlgeschlagen
16#800A000A	Datei konnte nicht geöffnet werden
16#800A000B	Dateigröße ist gleich Null
16#800A000C	Unzureichende Speicherkapazität um Datei zu laden
16#800A000D	Vergleich der Dateiprüfsummen fehlgeschlagen
16#800A000E	Fehler beim Lesen von Datei
16#800A000F	Ungültiger Dateityp
16#800A0010	Ungültiger Dateiname
16#800A0011	Treiberfunktion nicht vorhanden
16#800A0012	Angegebener Puffer ist zu klein
16#800A0013	Speicher kann nicht abgebildet werden
16#800A0014	Keine weiteren Einträge vorhanden
16#800B0001	Treiber ist nicht initialisiert
16#800B0002	Init-Statusfehler des Treibers
16#800B0003	Treiber Statuslesefehler
16#800B0004	Befehl ist aktiv auf Gerät
16#800B0005	Allgemeiner Fehler bei Download
16#800B0006	Falsche Treiberversion
16#800B0030	CIFx Treiber nicht in Betrieb
16#800B0031	Gerät konnte nicht initialisiert werden
16#800B0032	Kanal konnte nicht initialisiert werden (xOpenChannel wurde nicht aufgerufen)
16#800B0033	IOControl Aufruf fehlgeschlagen
16#800B0034	Treiber konnte nicht geöffnet werden
16#800C0010	Auf den Dual-Port Speicher konnte nicht zugegriffen werden (Speicherkarte nicht gefunden)

Additional2	
16#800C0011	Gerät nicht bereit (Ready-Flag fehlgeschlagen)
16#800C0012	Gerät nicht betriebsbereit (Running-Flag fehlgeschlagen)
16#800C0013	Watchdog-Test fehlgeschlagen
16#800C0015	Handshake-Flag Fehler
16#800C0016	Mailbox für gesendete Mails ist voll
16#800C0017	Timeout für Senden der Datenpakete
16#800C0018	Timeout für Empfangen der Datenpakete
16#800C0019	Kein Datenpaket verfügbar
16#800C001A	Zu wenig Speicherplatz in Mailbox
16#800C0020	Befehl-Timeout zurücksetzen
16#800C0021	COM-Flag nicht gesetzt
16#800C0022	EA-Datenaustausch fehlgeschlagen
16#800C0023	EA-Datenaustausch Timeout
16#800C0024	Unbekannter EA-Datenaustauschmodus
16#800C0025	Gerätfunktion fehlgeschlagen
16#800C0026	DPM-Größe size unterscheidet sich von Konfiguration
16#800C0027	Unbekannter Statusmodus
16#800C0028	Ausgangsport wird bereits verwendet
16#800C0029	Timeout Konfigurationsverriegelung
16#800C002A	Konfigurationsentriegelung Timeout
16#800C002B	HOST-Status Timeout setzen
16#800C002C	HOST-Status Timeout löschen
16#800C002D	Timeout während Kanalinitialisierung
16#800C002E	Bus EIN Timeout setzen
16#800C002F	Bus AUS Timeout setzen

Tab. 8-16: Allgemeine Fehlercodes

8.4.4 Spezielle Fehlercodes, EtherNet/IP adapter

In diesem Abschnitt werden die speziellen Fehlercodes des EtherNet/IP adapters dargestellt.

Die Fehlercodes werden als ERROR_STRUCT, Additional2 bei ERROR_STRUCT, **Additional1= 16#0003** gemeldet.

Additional2	
16#C0590001	Ungültiges Kommando
16#C0590004	Fehlerhafte TCP/IP Konfiguration
16#C0590009	Ungültiges Offset für E/A Daten

Tab. 8-17: EtherNet/IP adapter, spezielle Fehlercodes

8.4.5 CIP, Allgemeine Fehlercodes

In diesem Abschnitt werden die CIP, allgemeine Fehlercodes, dargestellt.

Code	
16#01	A connection-related service failed along the connection path
16#02	Resources needed for the object to perform the requested service were unavailable
16#03	See Status Code 16#20, which is the preferred value to use for this condition.
16#04	A path segment error has been encountered. Evaluation of the supplied path information failed.
16#05	The path references an unknown object class, instance or structure element causing the abort of path processing.
16#06	Only a part of the expected data could be transferred.
16#07	The messaging connection was lost.
16#08	The requested service has not been implemented or has not been defined for this object class or instance.
16#09	Detection of invalid attribute data
16#0A	An attribute in the Get_Attribute_List or Set_Attribute_List response has a status not equal to 0.
16#0B	The object is already in the mode or state which has been requested by the service
16#0C	The object is not able to perform the requested service in the current mode or state
16#0D	It has been tried to create an instance of an object which already exists.
16#0E	It has been tried to change an non-modifiable attribute.
16#0F	A check of permissions or privileges failed.
16#10	The current mode or state of the device prevents the execution of the requested service.
16#11	The data to be transmitted in the response buffer requires more space than the size of the allocated response buffer
16#12	The service specified an operation that is going to fragment a primitive data value, i.e. half a REAL data type.
16#13	The service did not supply all required data to perform the specified operation.
16#14	An unsupported attribute has been specified in the request
16#15	More data than was expected were supplied by the service.
16#16	The specified object does not exist in the device.
16#17	Fragmentation sequence for this service is not currently active for this data.
16#18	The attribute data of this object has not been saved prior to the requested service.
16#19	The attribute data of this object could not be saved due to a failure during the storage attempt.
16#1A	The service request packet was too large for transmission on a network in the path to the destination. The routing device was forced to abort the service.
16#1B	The service response packet was too large for transmission on a network in the path from the destination. The routing device was forced to abort the service.
16#1C	The service did not supply an attribute in a list of attributes that was needed by the service to perform the requested behavior.
16#1D	The service returns the list of attributes containing status information for invalid attrib
16#1E	An embedded service caused an error.

Code	
16#1F	A vendor specific error has occurred. This error should only occur when none of the other general error codes can correctly be applied.
16#20	A parameter which was associated with the request was invalid. The parameter does not meet the requirements of the CIP specification and/or the requirements defined in the specification of an application object.
16#21	An attempt was made to write to a write-once medium for the second time, or to modify a value that cannot be changed after being established once.
16#22	An invalid reply is received. Possible causes can for instance be among others a reply service code not matching the request service code or a reply message shorter than the expectable minimum size.
16#25	The key segment (i.e. the first segment in the path) does not match the destination module. More information about which part of the key check failed can be derived from the object specific status.
16#26	Path cannot be routed to an object due to lacking information or too much routing data have been included.
16#27	It has been attempted to set an attribute which may not be set in the current situation.
16#28	The Member ID specified in the request is not available within the specified class/ instance or attribute
16#29	A request to modify a member which cannot be modified has occurred

Tab. 8-18: CIP General Error Codes

8.4.6 Erweiterte Fehlercodes für Connection Manager

Nachstehend werden die erweiterten Fehlercodes für Connection Manager dargestellt.

Code	
16#0100	Connection already in use
16#0103	Transport type not supported
16#0106	More than one guy configuring
16#0107	Trying to close inactive connection
16#0108	Unsupported connection type
16#0109	Connection size mismatch
16#0110	Connection unconfigured
16#0111	Unsupportable RPI
16#0113	Connection Manager out of connections
16#0114	Vendor ID or Product Code mismatch
16#0115	Product Type mismatch
16#0116	Revision mismatch
16#0117	Nonexistant instance number
16#0118	Bad config instance number
16#0119	No controlling connection opened
16#011A	Application out of connections
16#0203	Using a timed out connection
16#0204	Unconnected Send timed out
16#0205	Unconnected Send parameter error

Code	
16#0301	No buffer memory available
16#0302	Insufficient bandwidth left
16#0303	Out of gen screeners
16#0304	Not configured to send RT data
16#0305	Signature does not match signature store in CCM
16#0306	CCM is not responding to request
16#0311	Nonexistant port
16#0312	Invalid link address in path
16#0315	Invalid segment in path
16#0316	Path & conn not equal in close
16#0317	Net segment not present or bad
16#0318	Link address to self invalid
16#0319	Resources in secondary unavailable
16#031D	Redundant connection mismatch
16#0320	Vendor specific: read write access fail
16#2105	Vendor specific: Access beyond end of the requested tag
16#2107	Vendor specific: Data type used in request does not match target tag's data type

Tab. 8-19: Erweiterten Fehlercodes für Connection Manager

9 RIL_Ethercat.library

9.1 Einführung und Übersicht

Die Bibliothek RIL_Ethercat.library ist die Schnittstelle von der SPS-Programmierumgebung zum EtherCAT Master.

Anwendungsgebiete und Funktionsbausteine der RIL_EtherCAT Bibliothek

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine	Beschreibung
Azyklische Kommunikation CoE (Acyclic Communication/CoE)	
IL_ECACoeReadSdo, Seite 266	Mit dem Funktionsbaustein IL_ECACoeReadSdo kann ein Service-Data-Object (SDO) von einem EtherCAT Slave Gerät azyklisch ausgelesen werden.
IL_ECACoeWriteSdo, Seite 269	Mit dem Funktionsbaustein IL_ECACoeWriteSdo kann ein Service-Data-Object (SDO) von einem EtherCAT Slave Gerät azyklisch geschrieben werden.
Azyklische Kommunikation SoE (Acyclic Communication/SoE)	
IL_ECASoeRead, Seite 271	Mit dem Funktionsbaustein IL_ECASoeRead kann ein Parameter von einem EtherCAT Slave Gerät azyklisch ausgelesen werden.
IL_ECASoeWrite, Seite 275	Mit dem Funktionsbaustein IL_ECASoeWrite kann ein Parameter von einem EtherCAT Slave Gerät azyklisch geschrieben werden.
IL_ECASoeCommand, Seite 277	Mit dem Funktionsbaustein IL_ECASoeCommand können SoE Kommandos eines EtherCAT Slave Gerätes azyklisch ausgeführt werden.
Diagnose (Diagnostics)	
IL_ECATGetNumConfigSlaves, Seite 280	Der Funktionsbaustein IL_ECATGetNumConfigSlaves gibt die Anzahl der konfigurierten Slaves im EtherCAT Master zurück.
IL_ECATGetNumOnlineSlaves, Seite 281	Der Funktionsbaustein IL_ECATGetNumOnlineSlaves gibt die Anzahl der aktuell verbundenen Slaves am EtherCAT Master zurück.
IL_ECATGetConfigEthercatAddrList, Seite 282	Mit dem Funktionsbaustein IL_ECATGetConfigEthercatAddrList können die EtherCAT Adressen der konfigurierten Slave Geräte eines EtherCAT Masters gelesen werden.
IL_ECATGetOnlineEthercatAddrList, Seite 284	Mit dem Funktionsbaustein IL_ECATGetOnlineEthercatAddrList können die EtherCAT Adressen der aktuell verbundenen Slave Geräte eines EtherCAT Masters gelesen werden.
IL_ECATGetSlaveConfigInfo, Seite 285	Mit dem Funktionsbaustein IL_ECATGetSlaveConfigInfo können Informationen zu einem konfigurierten Slave aus dem EtherCAT Master ausgelesen werden.
IL_ECATGetSlaveOnlineInfo, Seite 287	Mit dem Funktionsbaustein IL_ECATGetSlaveOnlineInfo können Informationen zu einem aktuell verbundenen Slave aus dem EtherCAT Master ausgelesen werden.
IL_ECATMasterState, Seite 289	Der Funktionsbaustein IL_ECATMasterState liefert eine Zusammenfassung von Diagnose-Informationen des EtherCAT Masters.
IL_ECATMasterBusStatistic, Seite 290	Der Funktionsbaustein IL_ECATMasterBusStatistic liefert statistische Diagnose-Informationen zur Bus-Kommunikation des EtherCAT Masters.

IL_ECATRemoteSlaveState, Seite 292	Der Funktionsbaustein IL_ECATRemoteSlaveState liefert Diagnose Status-Informationen eines EtherCAT Slave Gerätes.
IL_ECATRemoteSlaveSetErrorCounterPeriod, Seite 294	Der Funktionsbaustein IL_ECATRemoteSlaveSetErrorCounterPeriod aktiviert eine Master-Funktion zum zyklischen Einsammeln der Fehlerzähler von allen Slave Geräten.
IL_ECATRemoteSlaveGetErrorCounterPeriod, Seite 295	Der Funktionsbaustein IL_ECATRemoteSlaveGetErrorCounterPeriod liest das aktuelle Intervall des EtherCAT-Masters zum Einsammeln der Fehlerzähler der Slave Geräte.
IL_ECATRemoteSlaveGetErrorCounter, Seite 297	Der Funktionsbaustein IL_ECATRemoteSlaveGetErrorCounter liest die gesammelten Fehlerzähler eines Slave Gerätes aus dem Master aus.
IL_ECATRemoteSlaveClearErrorCounter, Seite 299	Der Funktionsbaustein IL_ECATRemoteSlaveClearErrorCounter löscht alle Fehlerzähler in allen Slave Geräten.
IL_ECATCoeReadDiagHistory, Seite 300	Mit dem Funktionsbaustein IL_ECATCoeReadDiagHistory kann das CoE "Diagnosis History" Objekt (Index 16#10F3) von einem EtherCAT Slave Gerät azyklisch ausgelesen werden.
Zustandsmaschine (State Machine)	
IL_ECATMasterGetBusState, Seite 304	Der Funktionsbaustein IL_ECATMasterGetBusState gibt den aktuellen Bus Zustand des EtherCAT Masters zurück.
IL_ECATMasterSetBusState, Seite 305	Mit dem Funktionsbaustein IL_ECATMasterSetBusState kann ein EtherCAT Bus Zustand beim EtherCAT Master angefordert werden.
IL_ECATRemoteSlaveGetDevState, Seite 307	Der Funktionsbaustein IL_ECATRemoteSlaveGetDevState gibt den aktuellen EtherCAT Zustand eines Slave Gerätes zurück.
IL_ECATRemoteSlaveSetDevState, Seite 308	Mit dem Funktionsbaustein IL_ECATRemoteSlaveSetDevState kann ein EtherCAT Zustand bei einem Slave Gerät angefordert werden.
Hilfsfunktionen (Utilities)	
IL_ECATSoeldnCoding, Seite 310	Die Funktion IL_ECATSoeldnCoding fasst die Einzelelemente Parametertyp, Parametersatz und Daten-Blocknummer zu einer Parameter-Identifikationsnummer zusammen.
IL_ECATSoeAttributeDecoding, Seite 311	Die Funktion IL_ECATSoeAttributeDecoding dekodiert das SoE Parameter "Attribut".
IL_ECATMasterStateDecoding, Seite 311	Die Funktion IL_ECATMasterStateDecoding dekodiert den Master "State" des Funktionsbausteins IL_ECATMasterState.
IL_ECATSlaveStateDecoding, Seite 312	Die Funktion IL_ECATSlaveStateDecoding dekodiert den Slave "State" des Funktionsbausteins IL_ECATRemoteSlaveState.

Tab. 9-1: RIL_Ethercat.library

EtherCAT® ist eine eingetragene Marke und patentierte Technologie, lizenziert durch die Beckhoff Automation GmbH, Deutschland.

9.2 Azyklische Kommunikation CoE (Acyclic Communication/CoE)

9.2.1 IL_ECATCoeReadSdo

Kurzbeschreibung Mit dem Funktionsbaustein IL_ECATCoeReadSdo kann ein Service-Data-Object (SDO) von einem EtherCAT-Slave-Gerät azyklisch ausgelesen werden.

Schnittstellenbeschreibung

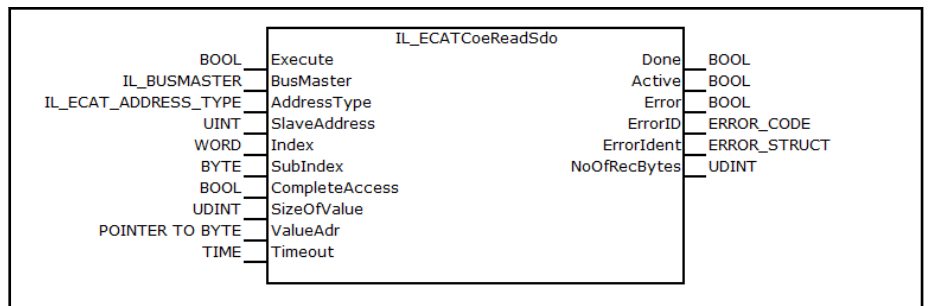


Abb. 9-1: Funktionsbaustein IL_ECACoeReadSdo

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des EtherCAT-Masters in der Reihenfolge des Konfigurationsbaumes
	AddressType	IL_ECAC_ADD-RESS_TYPE	Auswahl der Slave-Adressierung (EtherCAT-Adresse oder AutoInc-Adresse)
	SlaveAddress	UINT	EtherCAT-Adresse oder aktuelle AutoInc-Adresse am Bus (abhängig vom Eingang "AddressType")
	Index	WORD	Index des Objekts
	SubIndex	BYTE	SubIndex des Objekts
	CompleteAccess	BOOL	Mit TRUE kann der komplette Parameter gelesen werden (alle SubIndices)
	SizeOfValue	UDINT	Größe des Buffers bzw. der Variable in Bytes für den Empfang der Daten (SIZEOF(Value))
	ValueAdr	POINTER TO BYTE	Adresse des Buffers bzw. der Variable für den Empfang der Daten (ADR(Value))
	Timeout	TIME	Timeout-Überwachung
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	NoOfRecBytes	UDINT	Anzahl der Bytes, die in den "ValueAdr" Buffer kopiert wurden

Tab. 9-2: Schnittstelle IL_ECACoeReadSdo

Name	Datentyp	Min	Max	Default	Übernahme
Execute	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute

Name	Datentyp	Min	Max	Default	Übernahme
AddressType	IL_ECAC_ADD- RESS_TYPE	ECAT_AUTO- INC_ADDRESS	ECAT_ETHERC AT_ADDRESS	ECAT_ETHER CAT_ADD- RESS	Steigende Flanke an Execute
SlaveAddress	UINT	0	65535	0	Steigende Flanke an Execute
Index	WORD	0	65535	0	Steigende Flanke an Execute
SubIndex	BYTE	0	255 ¹⁾	0	Steigende Flanke an Execute
CompleteAccess	BOOL	-	-	FALSE	Steigende Flanke an Execute
SizeOfValue	UDINT	0	n.def.	0	Kontinuierlich
ValueAdr	POINTER TO BYTE	n.def.	n.def.	0	Kontinuierlich
Timeout	TIME	T#1ms	n.def	T#3s	Steigende Flanke an Execute

Tab. 9-3: Min-/Max- und Default-Werte der Eingänge IL_ECACoeReadSdo

Funktionsbeschreibung

Mit dem Funktionsbaustein IL_ECACoeReadSdo kann ein Service-Data-Object (SDO) von einem EtherCAT-Slave-Gerät azyklisch ausgelesen werden.

Voraussetzung ist, dass das Slave-Gerät das Mailboxprotokoll "CAN application protocol over EtherCAT" (CoE) unterstützt. Mailboxkommunikation ist nur in den EtherCAT-Zuständen PreOP, SafeOP und OP möglich.

Das Objekt, das ausgelesen werden soll, wird mit "Index" und "SubIndex" adressiert.

Mit der Einstellung "CompleteAccess" = TRUE können alle SubIndices eines Objekts mit einem Aufruf gelesen werden. Hierbei wird bei "SubIndex" dann der Start-SubIndex eingestellt:

- SubIndex = 0 → alle SubIndices inkl. SubIndex 0 (Start-SubIndex 0)
- SubIndex = 1 → alle SubIndices ohne SubIndex 0 (Start-SubIndex 1)



"CompleteAccess" wird nicht von allen Slaves unterstützt oder kann auch objektspezifisch sein. Wenn "CompleteAccess" nicht unterstützt wird, können folgende SDO-Fehlercodes auftreten:

- 16#F0ED0047: Nicht unterstützter Zugriff (CoE Abortcode 0x06010000)
- 16#F0EE0005: CompleteAccess nicht unterstützt (CoE Abortcode 0x06010004)

Implementierung in ST

Beispiel:

```

VAR
  fbECACoeReadSdo : IL_ECACoeReadSdo;
  boExecute : BOOL := FALSE;
  uiEthercatAddr : UINT := 1001;
  wIndex : WORD := 16#1018; // Identity Object
  bySubIndex : BYTE := 1; // Vendor ID
  udiVendorId : UDINT;
END_VAR

// Example to read the 'VendorId' from the Identity object
fbECACoeReadSdo(

```

¹⁾ Bei "CompleteAccess" = TRUE ist der Max-Wert = 1


```

Execute:= boExecute,
BusMaster:= ,
AddressType:= ,
SlaveAddress:= uiEthercatAddr,
Index:= wIndex,
SubIndex:= bySubIndex,
CompleteAccess:= ,
SizeOfValue:= SIZEOF(udiVendorId),
ValueAdr:= ADR(udiVendorId),
Timeout:= ,
Done=> ,
Active=> ,
Error=> ,
ErrorID=> ,
ErrorIdent=> ,
NoOfRecBytes=> );

```

Fehlerbeschreibung Spezifische Fehlercodes des Funktionsbausteins:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR	16#F0EC8000	16#00000001	Eingang "ValueAdr" ungültig: Null-Pointer
INPUT_INVALID_ERROR	16#F0EC8000	16#00000002	Eingang "SizeOfValue" ungültig: Wert "0" nicht erlaubt
INPUT_INVALID_ERROR	16#F0EC8000	16#00000003	Eingang "SubIndex" ungültig: nur "0" oder "1" erlaubt bei "CompleteAccess"=TRUE
INPUT_INVALID_ERROR	16#F0EC8000	16#00000004	Eingang "Timeout" ungültig: Wert "T#0ms" nicht erlaubt

Siehe auch allgemeine Fehlercodes "EtherCAT Fehlertabelle".

9.2.2 IL_ECATCoeWriteSdo

Kurzbeschreibung Mit dem Funktionsbaustein IL_ECATCoeWriteSdo kann ein Service-Data-Object (SDO) von einem EtherCAT-Slave-Gerät azyklisch geschrieben werden.

Schnittstellenbeschreibung

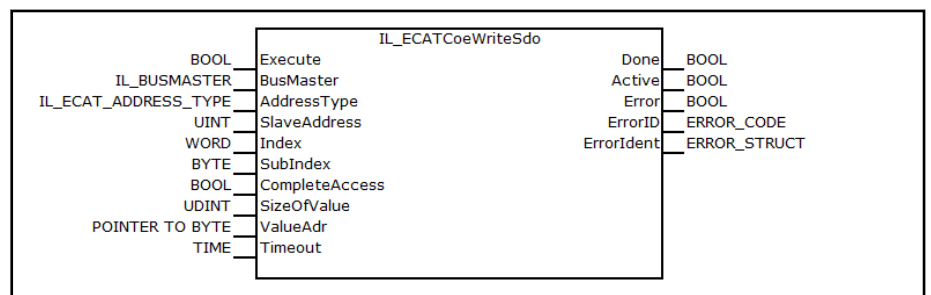


Abb. 9-2: Funktionsbeschreibung IL_ECATCoeWriteSdo

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des EtherCAT-Masters in der Reihenfolge des Konfigurationsbaumes
	AddressType	IL_ECAT_ADDRESS_TYPE	Auswahl der Slave-Adressierung (EtherCAT-Adresse oder AutoInc-Adresse)
	SlaveAddress	UINT	EtherCAT-Adresse oder aktuelle AutoInc-Adresse am Bus (abhängig vom Eingang "AddressType")
	Index	WORD	Index des Objekts
	SubIndex	BYTE	SubIndex des Objekts
	CompleteAccess	BOOL	Mit TRUE kann der komplette Parameter geschrieben werden (alle SubIndices)

	SizeOfValue	UDINT	Anzahl der Bytes an "ValueAdr" die geschrieben werden sollen (SIZEOF(Value))
	ValueAdr	POINTER TO BYTE	Adresse des Buffers bzw. der Variable mit den Daten, die geschrieben werden sollen (ADR(Value))
	Timeout	TIME	Timeout-Überwachung
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers

Tab. 9-4: Schnittstelle IL_ECACoeWriteSdo

Name	Datentyp	Min	Max	Default	Übernahme
Execute	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
AddressType	IL_ECAC_ADDRESSTYPE	ECAT_AUTOINC_ADDRESS	ECAT_ETHERCAT_ADDRESS	ECAT_ETHERCAT_ADDRESS	Steigende Flanke an Execute
SlaveAddress	UINT	0	65535	0	Steigende Flanke an Execute
Index	WORD	0	65535	0	Steigende Flanke an Execute
SubIndex	BYTE	0	255 ²⁾	0	Steigende Flanke an Execute
CompleteAccess	BOOL	-	-	FALSE	Steigende Flanke an Execute
SizeOfValue	UDINT	0	n.def.	0	Kontinuierlich
ValueAdr	POINTER TO BYTE	n.def.	n.def.	0	Kontinuierlich
Timeout	TIME	T#1ms	n.def	T#3s	Steigende Flanke an Execute

Tab. 9-5: Min-/Max- und Default-Werte der Eingänge IL_ECACoeWriteSdo

Funktionsbeschreibung

Mit dem Funktionsbaustein IL_ECACoeWriteSdo kann ein Service-Data-Object (SDO) von einem EtherCAT-Slave-Gerät azyklisch geschrieben werden.

Voraussetzung ist, dass das Slave-Gerät das Mailboxprotokoll "CAN application protocol over EtherCAT" (CoE) unterstützt. Mailboxkommunikation ist nur in den EtherCAT-Zuständen PreOP, SafeOP und OP möglich.

Das Objekt, das geschrieben werden soll, wird mit "Index" und "SubIndex" adressiert.

Mit der Einstellung "CompleteAccess" = TRUE können alle SubIndices eines Objekts mit einem Aufruf geschrieben werden. Hierbei wird bei "SubIndex" dann der Start-SubIndex eingestellt:

- SubIndex = 0 → alle SubIndices inkl. SubIndex 0 (Start-SubIndex 0)

²⁾ Bei "CompleteAccess" = TRUE ist der Max-Wert = 1

- SubIndex = 1 → alle SubIndices ohne SubIndex 0 (Start-SubIndex 1)



"CompleteAccess" wird nicht von allen Slaves unterstützt oder kann auch objektspezifisch sein. Wenn "CompleteAccess" nicht unterstützt wird, können folgende SDO-Fehlercodes auftreten:

- 16#F0ED0047: Nicht unterstützter Zugriff (CoE Abortcode 0x06010000)
- 16#F0EE0005: CompleteAccess nicht unterstützt (CoE Abortcode 0x06010004)

Implementierung in ST

Beispiel:

```
VAR
    fbECATCoeWriteSdo : IL_ECATCoeWriteSdo;
    boExecute : BOOL := FALSE;
    uiEthercatAddr : UINT := 1001;
    wIndex : WORD := 16#4000; // device specific Index/SubIndex
    bySubIndex : BYTE := 3; // ... which is writeable!
    byWriteValue : BYTE := 16#FF;
END_VAR

// Example to write a device specific byte value
fbECATCoeWriteSdo(
    Execute:= boExecute,
    BusMaster:= ,
    AddressType:= ,
    SlaveAddress:= uiEthercatAddr,
    Index:= wIndex,
    SubIndex:= bySubIndex,
    CompleteAccess:= ,
    SizeOfValue:= SIZEOF(byWriteValue),
    ValueAdr:= ADR(byWriteValue),
    Timeout:= ,
    Done=> ,
    Active=> ,
    Error=> ,
    ErrorID=> ,
    ErrorIdent=> );
```

Fehlerbeschreibung Spezifische Fehlercodes des Funktionsbausteins:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR	16#F0EC8000	16#00000001	Eingang "ValueAdr" ungültig: Null-Pointer
INPUT_INVALID_ERROR	16#F0EC8000	16#00000002	Eingang "SizeOfValue" ungültig: Wert "0" nicht erlaubt
INPUT_INVALID_ERROR	16#F0EC8000	16#00000003	Eingang "SubIndex" ungültig: nur "0" oder "1" erlaubt bei "CompleteAccess"=TRUE
INPUT_INVALID_ERROR	16#F0EC8000	16#00000004	Eingang "Timeout" ungültig: Wert "t#0ms" nicht erlaubt

Siehe auch allgemeine Fehlercodes "EtherCAT Fehlertabelle".

9.3 Azyklische Kommunikation SoE (Acyclic Communication/ SoE)

9.3.1 IL_ECATSoeRead

Kurzbeschreibung

Mit dem Funktionsbaustein IL_ECATSoeRead kann ein Parameter von einem EtherCAT-Slave-Gerät azyklisch ausgelesen werden.

Schnittstellenbeschreibung

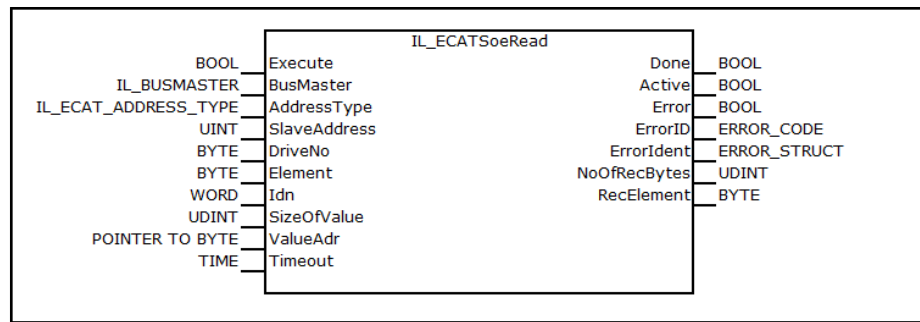


Abb. 9-3: Funktionsbaustein IL_ECATSoeRead

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des EtherCAT-Masters in der Reihenfolge des Konfigurationsbaumes
	AddressType	IL_ECAT_ADDRESS_TYPE	Auswahl der Slave-Adressierung (EtherCAT-Adresse oder AutoInc-Adresse)
	SlaveAddress	UINT	EtherCAT-Adresse oder aktuelle AutoInc-Adresse am Bus (abhängig vom Eingang "AddressType")
	DriveNo	BYTE	Nummer des Antriebs (bei Einzelachsgerät "0")
	Element	BYTE	Elementnummer des Parameters, siehe IL_ECAT_SOE_ELEMENT
	Idn	WORD	Identifikationsnummer des Parameters (siehe Kap. 9.6.1 "IL_ECATSoeldnCoding" auf Seite 310)
	SizeOfValue	UDINT	Größe des Buffers bzw. der Variable in Bytes für den Empfang der Daten (SIZEOF(Value))
	ValueAdr	POINTER TO BYTE	Adresse des Buffers bzw. der Variable für den Empfang der Daten (ADR(Value))
	Timeout	TIME	Timeout-Überwachung
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	NoOfRecBytes	UDINT	Anzahl der Bytes, die in den "ValueAdr"-Buffer kopiert wurden
	RecElement	BYTE	Empfangene Elemente, siehe IL_ECAT_SOE_ELEMENT (wenn mehr als ein Bit am Eingang "Element" gesetzt wurde)

Tab. 9-6: Schnittstelle IL_ECATSoeRead

Name	Datentyp	Min	Max	Default	Übernahme
Execute	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
AddressType	IL_ECAT_ADDRESS_TYPE	ECAT_AUTO_INCREMENT_ADDRESS	ECAT_ETHERCAT_ADDRESS	ECAT_ETHERCAT_ADDRESS	Steigende Flanke an Execute
SlaveAddress	UINT	0	65535	0	Steigende Flanke an Execute
DriveNo	BYTE	0	7	0	Steigende Flanke an Execute
Element	BYTE	16#01	16#FF	SOE_ELEMENT_OPDATA	Steigende Flanke an Execute
Idn	WORD	0	n.def.	0	Steigende Flanke an Execute
SizeOfValue	UDINT	0	n.def.	0	Kontinuierlich
ValueAdr	POINTER TO BYTE	n.def.	n.def.	0	Kontinuierlich
Timeout	TIME	T#1ms	n.def	T#3s	Steigende Flanke an Execute

Tab. 9-7: Min/Max und Default-Werte der Eingänge IL_ECATSoeRead

Funktionsbeschreibung

Mit dem Funktionsbaustein IL_ECATSoeRead kann ein Parameter von einem EtherCAT-Slave-Gerät azyklisch ausgelesen werden.

Voraussetzung ist, dass das Slave-Gerät das Mailboxprotokoll "Servo drive profile over EtherCAT" (SoE) unterstützt. Mailboxkommunikation ist nur in den EtherCAT-Zuständen PreOP, SafeOP und OP möglich.

Der Parameter, der ausgelesen werden soll, wird mit "Idn", "Element" und "DriveNo" adressiert. Für die Idn-Codierung wird die Funktion [Kap. 9.6.1 "IL_ECATSoeIdnCoding"](#) auf Seite 310 empfohlen.

Für den Eingang "Element" wird der Aufzählungstyp IL_ECAT_SOE_ELEMENT empfohlen. Es können auch mehrere Bits gesetzt werden, so dass mehrere Elemente mit einer azyklischen Anfrage ausgelesen werden sollen.

Zum Beispiel Sercos-Attribut und Daten:

```
fbECATSoeRead.Element := SOE_ELEMENT_ATTRIBUTE OR SOE_ELEMENT_OPDATA;
```

In den Antwortdaten werden die Werte der angefragten Elemente in der Reihenfolge der Element-Bits eingetragen. Am Ausgang "RecElement" kann geprüft werden, welche Elemente übertragen wurden.



Das Lesen von mehreren Elementen mit einer azyklischen Anfrage ist eine optionale Funktionalität in Slave Geräten. Wird dies vom Slave nicht unterstützt, antwortet der Slave nur mit den Daten vom ersten gesetzten Element-Bit (siehe Ausgang "RecElement"). Dies ist kein Fehler, d.h. auch der Funktionsbaustein meldet hier keinen Fehler. Die angeforderten Elemente müssen dann einzeln nacheinander ausgelesen werden.



Wird ein Listen-Parameter gelesen, steht bei den ausgelesenen Daten bei Byte 0 u. 1 die aktuelle Länge der Liste, bei Byte 2 u. 3 die maximale Länge der Liste und ab Byte 4 stehen die Daten des Listenparameters.

Implementierung in ST *Beispiel:*

```

=====
// Example A : read a UINT Parameter (S-0-0002)

VAR
    fbECATSoeRead : IL_ECATSoeRead;
    boExecute     : BOOL := FALSE;
    uiEthercatAddr : UINT := 1001;
    uiCycleTime   : UINT;
END_VAR

fbECATSoeRead(
    Execute:= boExecute,
    BusMaster:= ,
    AddressType:= ,
    SlaveAddress:= uiEthercatAddr,
    DriveNo:= ,
    Element:= ,
    Idn:= IL_ECATSoeIdnCoding(SOE_S_PARAM,0,2), // ^= S-0-0002
    SizeOfValue:= SIZEOF(uiCycleTime),
    ValueAdr:= ADR(uiCycleTime),
    Timeout:= ,
    Done=> ,
    Active=> ,
    Error=> ,
    ErrorID=> ,
    ErrorIdent=> ,
    NoOfRecBytes=> );

=====

// Example B : read a STRING Parameter (S-0-0095)

//-----
// Structure for SoE String param. (= List with var. length)
TYPE ST_SOE_STRING :
STRUCT
    ActLength : UINT; // actual length without 0-ending
    MaxLength : UINT; // max length without 0-ending
    Text      : STRING(255); // text value
END_STRUCT
END_TYPE
//-----

VAR
    fbECATSoeRead : IL_ECATSoeRead;
    boExecute     : BOOL := FALSE;
    uiEthercatAddr : UINT := 1001;
    rDiagnosis     : ST_SOE_STRING;
END_VAR

fbECATSoeRead(
    Execute:= boExecute,
    BusMaster:= ,
    AddressType:= ,
    SlaveAddress:= uiEthercatAddr,
    DriveNo:= ,
    Element:= ,
    Idn:= IL_ECATSoeIdnCoding(SOE_S_PARAM,0,95), // ^= S-0-0095
    SizeOfValue:= SIZEOF(rDiagnosis),
    ValueAdr:= ADR(rDiagnosis),
    Timeout:= ,
    Done=> ,
    Active=> ,
    Error=> ,
    ErrorID=> ,
    ErrorIdent=> ,
    NoOfRecBytes=> );

=====

// Example C : read an array (S-0-0017)

//-----
// Structure for a SoE List parameter (with WORD elements)
TYPE ST_SOE_WORD_LIST :
STRUCT
    ActLength : UINT; // actual length

```

```

        MaxLength      : UINT; // max length
        awIDN          : ARRAY[0..9999] OF WORD; // List elements
    END_STRUCT
END_TYPE
//-----

VAR
    fbECATSoeRead      : IL_ECATSoeRead;
    boExecute           : BOOL := FALSE;
    uiEthercatAddr      : UINT := 1001;
    rValue              : ST_SOE_WORD_LIST;
END_VAR

fbECATSoeRead(
    Execute:= boExecute,
    BusMaster:= ,
    AddressType:= ,
    SlaveAddress:= uiEthercatAddr,
    DriveNo:= ,
    Element:= ,
    Idn:= IL_ECATSoeIdnCoding(SOE_S_PARAM,0,17), // ^= S-0-0017
    SizeOfValue:= SIZEOF(rValue),
    ValueAdr:= ADR(rValue),
    Timeout:= ,
    Done=> ,
    Active=> ,
    Error=> ,
    ErrorID=> ,
    ErrorIdent=> ,
    NoOfRecBytes=> );

```

Fehlerbeschreibung Spezifische Fehlercodes des Funktionsbausteins:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR	16#F0EC8000	16#00000001	Eingang "ValueAdr" ungültig: Null-Pointer
INPUT_INVALID_ERROR	16#F0EC8000	16#00000002	Eingang "SizeOfValue" ungültig: Wert "0" nicht erlaubt
INPUT_INVALID_ERROR	16#F0EC8000	16#00000004	Eingang "Timeout" ungültig: Wert "T#0ms" nicht erlaubt

Siehe auch allgemeine Fehlercodes "EtherCAT Fehlertabelle".

9.3.2 IL_ECATSoeWrite

Kurzbeschreibung Mit dem Funktionsbaustein IL_ECATSoeWrite kann ein Parameter von einem EtherCAT-Slave-Gerät azyklisch geschrieben werden.

Schnittstellenbeschreibung

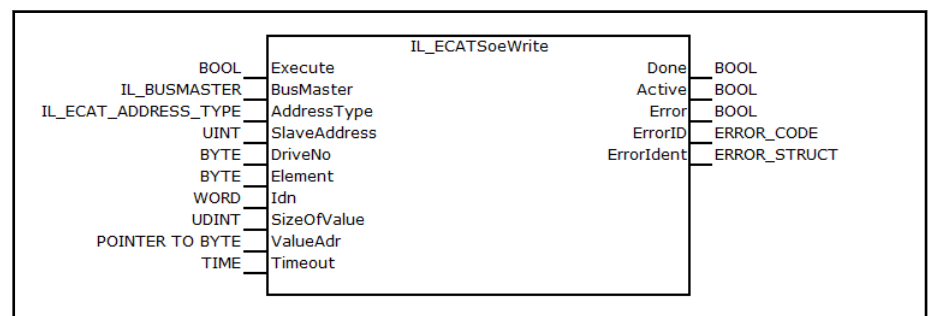


Abb. 9-4: Funktionsbaustein IL_ECATSoeWrite

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des EtherCAT-Masters in der Reihenfolge des Konfigurationsbaumes

E/A-Typ	Name	Datentyp	Beschreibung
	AddressType	IL_ECAC_ADDRESS_TYPE	Auswahl der Slave-Adressierung (EtherCAT-Adresse oder AutoInc-Adresse)
	SlaveAddress	UINT	EtherCAT-Adresse oder aktuelle AutoInc-Adresse am Bus (abhängig vom Eingang "AddressType")
	DriveNo	BYTE	Nummer des Antriebs (bei Einzelachsgerät "0")
	Element	BYTE	Elementnummer des Parameters, siehe IL_ECAC_SOE_ELEMENT
	Idn	WORD	Identifikationsnummer des Parameters (siehe Kap. 9.6.1 "IL_ECACSoeldnCoding" auf Seite 310)
	SizeOfValue	UDINT	Anzahl der Bytes an "ValueAdr", die geschrieben werden sollen (SIZEOF(Value))
	ValueAdr	POINTER TO BYTE	Adresse des Buffers bzw. der Variable mit den Daten, die geschrieben werden sollen (ADR(Value))
	Timeout	TIME	Timeout-Überwachung
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers

Tab. 9-8: Schnittstelle IL_ECACSoeWrite

Name	Datentyp	Min	Max	Default	Übernahme
Execute	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
AddressType	IL_ECAC_ADDRESS_TYPE	ECAC_AUTOINC_ADDRESS	ECAC_ETHERCAT_ADDRESS	ECAC_ETHERCAT_ADDRESS	Steigende Flanke an Execute
SlaveAddress	UINT	0	65535	0	Steigende Flanke an Execute
DriveNo	BYTE	0	7	0	Steigende Flanke an Execute
Element	BYTE	16#01	16#FF	SOE_ELEMENT_OPDATA	Steigende Flanke an Execute
Idn	WORD	0	n.def.	0	Steigende Flanke an Execute
SizeOfValue	UDINT	0	n.def.	0	Kontinuierlich
ValueAdr	POINTER TO BYTE	n.def.	n.def.	0	Kontinuierlich
Timeout	TIME	T#1ms	n.def	T#3s	Steigende Flanke an Execute

Tab. 9-9: Min-/Max- und Default-Werte der Eingänge IL_ECACSoeWrite

Funktionsbeschreibung Mit dem Funktionsbaustein IL_ECATSoeWrite kann ein Parameter von einem EtherCAT-Slave-Gerät azyklisch geschrieben werden.

Voraussetzung ist, dass das Slave-Gerät das Mailboxprotokoll "Servo drive profile over EtherCAT" (SoE) unterstützt. Mailboxkommunikation ist nur in den EtherCAT-Zuständen PreOP, SafeOP und OP möglich.

Der Parameter, der geschrieben werden soll, wird mit "Idn", "Element" und "DriveNo" adressiert. Für die Idn-Codierung wird die Funktion [Kap. 9.6.1 "IL_ECATSoeIdnCoding" auf Seite 310](#) empfohlen.



Soll ein Listenparameter geschrieben werden, müssen 4 Bytes am Anfang der zu schreibenden Daten ergänzt werden: Byte 0 u. 1 muss die aktuelle Länge der Daten enthalten und Byte 2 u. 3 ist reserviert für die maximale Länge der Daten (diese maximale Länge braucht nicht gesetzt werden).

Implementierung in ST

Beispiel:

```
VAR
    fbECATSoeWrite : IL_ECATSoeWrite;
    boExecute : BOOL := FALSE;
    uiEthercatAddr : UINT := 1001;
    uiWriteValue : UINT := 16#1234;
END_VAR

// Example to write a '2 Byte Parameter' (here: S-0-0000)
fbECATSoeWrite(
    Execute:= boExecute,
    BusMaster:= ,
    AddressType:= ,
    SlaveAddress:= uiEthercatAddr,
    DriveNo:= ,
    Element:= ,
    Idn:= IL_ECATSoeIdnCoding(SOE_S_PARAM,0,0), // ^= S-0-0000
    SizeOfValue:= SIZEOF(uiWriteValue),
    ValueAdr:= ADR(uiWriteValue),
    Timeout:= ,
    Done=> ,
    Active=> ,
    Error=> ,
    ErrorID=> ,
    ErrorIdent=> );
```

Fehlerbeschreibung

Spezifische Fehlercodes des Funktionsbausteins:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR	16#F0EC8000	16#00000001	Eingang "ValueAdr" ungültig: Null-Pointer
INPUT_INVALID_ERROR	16#F0EC8000	16#00000002	Eingang "SizeOfValue" ungültig: Wert "0" nicht erlaubt
INPUT_INVALID_ERROR	16#F0EC8000	16#00000004	Eingang "Timeout" ungültig: Wert "T#0ms" nicht erlaubt

Siehe auch allgemeine Fehlercodes "EtherCAT Fehlertabelle".

9.3.3 IL_ECATSoeCommand

Kurzbeschreibung

Mit dem Funktionsbaustein IL_ECATSoeCommand können SoE-Kommandos eines EtherCAT-Slave-Gerätes azyklisch ausgeführt werden.

Schnittstellenbeschreibung

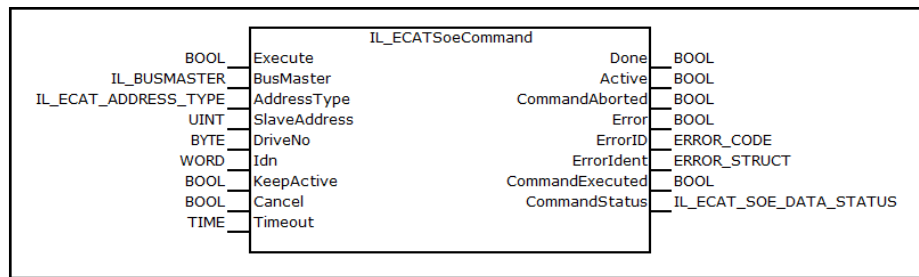


Abb. 9-5: Funktionsbaustein IL_ECATSoeCommand

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des EtherCAT-Masters in der Reihenfolge des Konfigurationsbaumes
	AddressType	IL_ECAT_ADDRESS_TYPE	Auswahl der Slave-Adressierung (EtherCAT-Adresse oder AutoInc-Adresse)
	SlaveAddress	UINT	EtherCAT-Adresse oder aktuelle AutoInc-Adresse am Bus (abhängig vom Eingang "AddressType")
	DriveNo	BYTE	Nummer des Antriebs (bei Einzelachsgerät "0")
	Idn	WORD	Identifikationsnummer des Kommandoparameters (siehe Kap. 9.6.1 "IL_ECATSoeIdnCoding" auf Seite 310)
	KeepActive	BOOL	Hält das Kommando nach Abarbeitung aktiv (zustandsgesteuert)
	Cancel	BOOL	Abbruch des laufenden Kommandos (bei steigender Flanke)
	Timeout	TIME	Timeout-Überwachung von intern verwendeten azyklischen Diensten
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	CommandAborted	BOOL	Kommando erfolgreich abgebrochen und zurückgesetzt
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	Command-Executed	BOOL	Kommando erfolgreich abgeschlossen
	CommandStatus	IL_ECAT_SOE_DATA_STATUS	Status des Kommando-Parameters

Tab. 9-10: Schnittstelle IL_ECATSoeCommand

Name	Datentyp	Min	Max	Default	Übernahme
Execute	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
AddressType	IL_ECATAADDRESS_TYPE	ECAT_AUTOINC_ADDRESS	ECAT_ETHERCAT_ADDRESS	ECAT_ETHERCAT_ADDRESS	Steigende Flanke an Execute
SlaveAddress	UINT	0	65535	0	Steigende Flanke an Execute
DriveNo	BYTE	0	7	0	Steigende Flanke an Execute
Idn	WORD	0	n.def.	0	Steigende Flanke an Execute
KeepActive	BOOL	-	-	FALSE	Kontinuierlich
Cancel	BOOL	-	-	FALSE	Kontinuierlich
Timeout	TIME	T#1ms	n.def	T#3s	Steigende Flanke an Execute

Tab. 9-11: Min-/Max- und Default-Werte der Eingänge IL_ECATSocCommand

Funktionsbeschreibung

Mit dem Funktionsbaustein IL_ECATSocCommand können SoE-Kommandos eines EtherCAT-Slave-Gerätes azyklisch ausgeführt werden.

Voraussetzung ist, dass das Slave-Gerät das Mailboxprotokoll "Servo drive profile over EtherCAT" (SoE) unterstützt. Mailboxkommunikation ist nur in den EtherCAT-Zuständen PreOP, SafeOP und OP möglich.

Der Kommandoparameter, der ausgeführt werden soll, wird mit "Idn" und "DriveNo" adressiert. Für die Idn-Codierung wird die Funktion [Kap. 9.6.1 "IL_ECATSocIdnCoding" auf Seite 310](#) empfohlen.

Das Kommando wird nach Abschluss der Bearbeitung automatisch zurückgesetzt. Die Kommandoausführung kann, abhängig vom Kommando selbst, jederzeit über den Eingang "Cancel" abgebrochen werden.

Mit dem Eingang "KeepActive" kann das Kommando nach erfolgreicher Ausführung weiter aktiv gehalten werden. Wird im Folgenden "KeepActive" deaktiviert, wird das Kommando ebenfalls automatisch zurückgesetzt.

Implementierung in ST

Beispiel:

```

VAR
    fbECATSocCommand : IL_ECATSocCommand;
    boExecute : BOOL := FALSE;
    uiEthercatAddr : UINT := 1001;
END_VAR

// Example to execute the Reset command (S-0-0099)
fbECATSocCommand(
    Execute:= boExecute,
    BusMaster:= ,
    AddressType:= ,
    SlaveAddress:= uiEthercatAddr,
    DriveNo:= ,
    Idn:= IL_ECATSocIdnCoding(SOE_S_PARAM,0,99), // ^= S-0-0099
    KeepActive:= ,
    Cancel:= ,
    Timeout:= ,
    Done=> ,
    Active=> ,
    CommandAborted=> ,
    Error=> ,
    ErrorID=> ,
    ErrorIdent=> ,
    CommandExecuted=> ,
    CommandStatus=> );

```

Fehlerbeschreibung

Spezifische Fehlercodes des Funktionsbausteins:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR	16#F0EC8000	internal	Eingang "Idn" ist kein Kommando-Parameter
DEVICE_ERROR	16#F0EC8010	internal	Ungültiger Kommando Daten-Status, siehe Ausgang "CommandStatus".

Siehe auch allgemeine Fehlercodes "EtherCAT Fehlertabelle".

9.4 Diagnose (Diagnostics)

9.4.1 IL_ECAGetNumConfigSlaves

Kurzbeschreibung Der Funktionsbaustein IL_ECAGetNumConfigSlaves gibt die Anzahl der konfigurierten Slaves im EtherCAT-Master zurück.

Schnittstellenbeschreibung

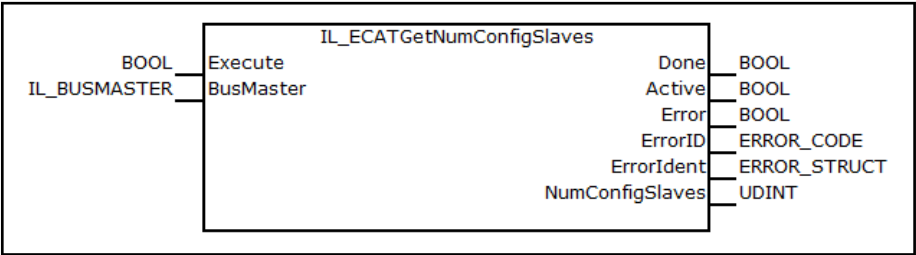


Abb. 9-6: Funktionsbaustein IL_ECAGetNumConfigSlaves

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des EtherCAT-Masters in der Reihenfolge des Konfigurationsbaumes
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	NumConfigSlaves	UDINT	Anzahl der konfigurierten Slaves im EtherCAT-Master

Tab. 9-12: Schnittstelle IL_ECAGetNumConfigSlaves

Name	Datentyp	Min	Max	Default	Übernahme
Execute	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute

Tab. 9-13: Min-/Max- und Default-Werte der Eingänge IL_ECAGetNumConfigSlaves

Funktionsbeschreibung Der Funktionsbaustein IL_ECAGetNumConfigSlaves gibt die Anzahl der konfigurierten Slaves im EtherCAT-Master zurück.

Implementierung in ST Beispiel:

```

VAR
    fbECATGetNumConfigSlaves: IL_ECATGetNumConfigSlaves;
    boExecute : BOOL := FALSE;
    udiNumConfigSlaves : UDINT;
END_VAR

fbECATGetNumConfigSlaves (
    Execute:= boExecute,
    BusMaster:= ,
    Done=> ,
    Active=> ,
    Error=> ,
    ErrorID=> ,
    ErrorIdent=> ,
    NumConfigSlaves=> udiNumConfigSlaves );

```

Fehlerbeschreibung Keine spezifischen Fehlercodes, siehe allgemeine Fehlercodes "EtherCAT Fehlertabelle".

9.4.2 IL_ECATGetNumOnlineSlaves

Kurzbeschreibung Der Funktionsbaustein IL_ECATGetNumOnlineSlaves gibt die Anzahl der aktuell verbundenen Slaves am EtherCAT-Master zurück.

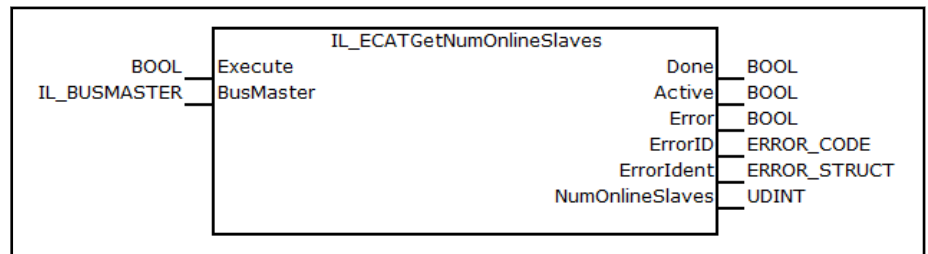
Schnittstellenbeschreibung

Abb. 9-7: Funktionsbaustein IL_ECATGetNumOnlineSlaves

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des EtherCAT-Masters in der Reihenfolge des Konfigurationsbaumes
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	NumOnlineSlaves	UDINT	Anzahl der aktuell verbundenen Slaves am EtherCAT Master

Tab. 9-14: Schnittstelle IL_ECATGetNumOnlineSlaves

Name	Datentyp	Min	Max	Default	Übernahme
Execute	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute

Tab. 9-15: Min-/Max- und Default-Werte der Eingänge IL_ECATGetNumOnlineSlaves

Funktionsbeschreibung Der Funktionsbaustein IL_ECATGetNumOnlineSlaves gibt die Anzahl der aktuell verbundenen Slaves am EtherCAT-Master zurück.

Implementierung in ST *Beispiel:*

```

VAR
    fbECATGetNumOnlineSlaves: IL_ECATGetNumOnlineSlaves;
    boExecute : BOOL := FALSE;
    udiNumOnlineSlaves : UDINT;
END_VAR

fbECATGetNumOnlineSlaves(
    Execute:= boExecute,
    BusMaster:= ,
    Done=> ,
    Active=> ,
    Error=> ,
    ErrorID=> ,
    ErrorIdent=> ,
    NumOnlineSlaves=> udiNumOnlineSlaves );

```

Fehlerbeschreibung Keine spezifischen Fehlercodes, siehe allgemeine Fehlercodes "EtherCAT Fehlertabelle".

9.4.3 IL_ECATGetConfigEthercatAddrList

Kurzbeschreibung Mit dem Funktionsbaustein IL_ECATGetConfigEthercatAddrList können die EtherCAT-Adressen der konfigurierten Slave-Geräte eines EtherCAT-Masters gelesen werden.

Schnittstellenbeschreibung

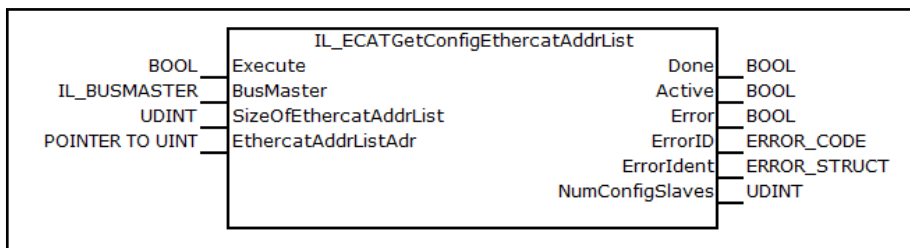


Abb. 9-8: Funktionsbaustein IL_ECATGetConfigEthercatAddrList

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des EtherCAT-Masters in der Reihenfolge des Konfigurationsbaumes
	SizeOfEthercatAddrList	UDINT	Größe des Datenbereichs (in Bytes), in dem die EtherCAT-Adressen geschrieben werden (SIZE-OF(EthercatAddrList))
	EthercatAddrListAdr	POINTER TO UINT	Zeiger auf einen Datenbereich, in dem die EtherCAT-Adressen geschrieben werden (ADR(EthercatAddrList))
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig

E/A-Typ	Name	Datentyp	Beschreibung
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	NumConfigSlaves	UDINT	Anzahl der konfigurierten Slaves (= gültige Einträge in "EthercatAddrList")

Tab. 9-16: Schnittstelle IL_ECAGetConfigEthercatAddrList

Name	Datentyp	Min	Max	Default	Übernahme
Execute	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
SizeOfEthercatAddrList	UDINT	0	n.def.	0	Steigende Flanke an Execute
EthercatAddrListAdr	POINTER TO UINT	n.def.	n.def.	0	Kontinuierlich

Tab. 9-17: Min-/Max- und Default-Werte der Eingänge IL_ECAGetConfigEthercatAddrList

Funktionsbeschreibung Mit dem Funktionsbaustein IL_ECAGetConfigEthercatAddrList können die EtherCAT-Adressen der konfigurierten Slave-Geräte eines EtherCAT-Masters gelesen werden.

Implementierung in ST *Beispiel:*

```

VAR
    fbECAGetConfigEthercatAddrList : IL_ECAGetConfigEthercatAddrList;
    boExecute : BOOL := FALSE;
    // array with max number of configured slaves, here: '128'
    arEthercatAddrList : ARRAY [0..127] OF UINT;
    udiNumConfigSlaves : UDINT;
END_VAR

// Example to read the configured EtherCAT Addresses
fbECAGetConfigEthercatAddrList(
    Execute:= boExecute,
    BusMaster:= ,
    SizeOfEthercatAddrList:= SIZEOF(arEthercatAddrList),
    EthercatAddrListAdr:= ADR(arEthercatAddrList),
    Done=> ,
    Active=> ,
    Error=> ,
    ErrorID=> ,
    ErrorIdent=> ,
    NumConfigSlaves=> udiNumConfigSlaves);

```

Fehlerbeschreibung Spezifische Fehlercodes des Funktionsbausteins:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR	16#F0EC8000	internal	Eingang "EthercatAddrListAdr" ungültig: Null-Pointer
INPUT_INVALID_ERROR	16#F0EC8001	internal	Eingang "SizeOfEthercatAddrListAdr" ungültig: Datenbereich ist zu klein

Siehe auch allgemeine Fehlercodes "EtherCAT Fehlertabelle".

9.4.4 IL_ECATCHetOnlineEthercatAddrList

Kurzbeschreibung Mit dem Funktionsbaustein IL_ECATCHetOnlineEthercatAddrList können die EtherCAT-Adressen der aktuell verbundenen Slave-Geräte eines EtherCAT-Masters gelesen werden.

Schnittstellenbeschreibung

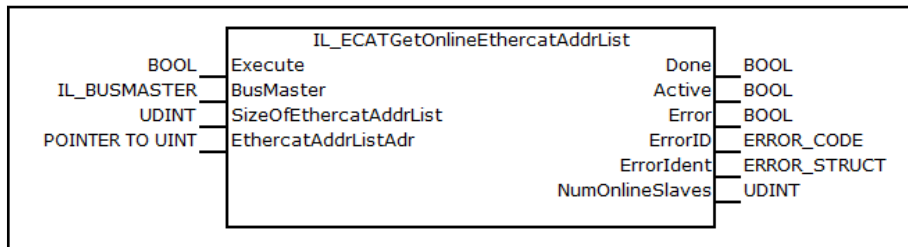


Abb. 9-9: Funktionsbaustein IL_ECATCHetOnlineEthercatAddrList

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des EtherCAT-Masters in der Reihenfolge des Konfigurationsbaumes
	SizeOfEthercatAddrList	UDINT	Größe des Datenbereichs (in Bytes), in dem die EtherCAT-Adressen geschrieben werden (SIZEOF(EthercatAddrList))
	EthercatAddrListAdr	POINTER TO UINT	Zeiger auf einen Datenbereich, in dem die EtherCAT-Adressen geschrieben werden (ADR(EthercatAddrList))
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	NumOnlineSlaves	UDINT	Anzahl der aktuell verbundenen Slaves (= gültige Einträge in "EthercatAddrList")

Tab. 9-18: Schnittstelle IL_ECATCHetOnlineEthercatAddrList

Name	Datentyp	Min	Max	Default	Übernahme
Execute	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
SizeOfEthercatAddrList	UDINT	0	n.def.	0	Steigende Flanke an Execute
EthercatAddrListAdr	POINTER TO UINT	n.def.	n.def.	0	Kontinuierlich

Tab. 9-19: Min-/Max- und Default-Werte der Eingänge IL_ECATCHetOnlineEthercatAddrList

Funktionsbeschreibung Mit dem Funktionsbaustein IL_ECATCHetOnlineEthercatAddrList können die EtherCAT-Adressen der aktuell verbundenen Slave-Geräte eines EtherCAT-Masters gelesen werden.

Implementierung in ST *Beispiel:*

```

VAR
  fbECATGetOnlineEthercatAddrList : IL_ECATGetOnlineEthercatAddrList;
  boExecute: BOOL := FALSE;
  // array with max number of connected slaves, here '128'
  arEthercatAddrList : ARRAY [0..127] OF UINT;
  udiNumOnlineSlaves : UDINT;
END_VAR

// Example to read the currently connected EtherCAT Addresses
fbECATGetOnlineEthercatAddrList(
  Execute:= boExecute,
  BusMaster:= ,
  SizeOfEthercatAddrList:= SIZEOF(arEthercatAddrList),
  EthercatAddrListAdr:= ADR(arEthercatAddrList),
  Done=> ,
  Active=> ,
  Error=> ,
  ErrorID=> ,
  ErrorIdent=> ,
  NumOnlineSlaves=> udiNumOnlineSlaves );

```

Fehlerbeschreibung Spezifische Fehlercodes des Funktionsbausteins:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR	16#F0EC8000	internal	Eingang "EthercatAddrListAdr" ungültig: Null-Pointer
INPUT_INVALID_ERROR	16#F0EC8001	internal	Eingang "SizeOfEthercatAddrListAdr" ungültig: Datenbereich ist zu klein

Siehe auch allgemeine Fehlercodes "EtherCAT Fehlertabelle".

9.4.5 IL_ECATGetSlaveConfigInfo

Kurzbeschreibung

Mit dem Funktionsbaustein IL_ECATGetSlaveConfigInfo können Informationen zu einem konfigurierten Slave aus dem EtherCAT-Master ausgelesen werden.

Schnittstellenbeschreibung

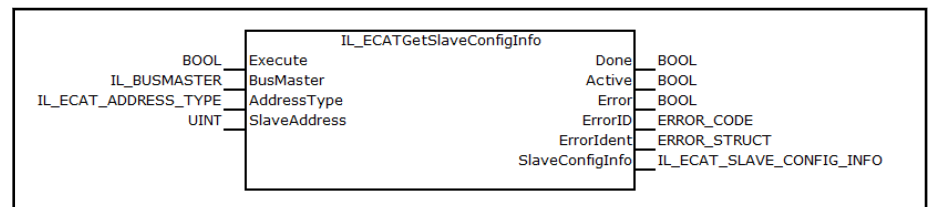


Abb. 9-10: Funktionsbaustein IL_ECATGetSlaveConfigInfo

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des EtherCAT-Masters in der Reihenfolge des Konfigurationsbaumes
	AddressType	IL_ECAT_ADDRESS_TYPE	Auswahl der Slave-Adressierung (EtherCAT-Adresse oder AutoInc-Adresse)
	SlaveAddress	UINT	EtherCAT-Adresse oder konfigurierte AutoInc-Adresse (abhängig vom Eingang "AddressType")
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig

E/A-Typ	Name	Datentyp	Beschreibung
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	SlaveConfigInfo	IL_ECATCHSLAVE_CONFIG_INFO	Informationen zu einem konfigurierten Slave

Tab. 9-20: Schnittstelle IL_ECATCHSlaveConfigInfo

Name	Datentyp	Min	Max	Default	Übernahme
Execute	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
AddressType	IL_ECATCHADDRESS_TYPE	ECATCH_AUTO_INCREMENT_ADDRESS	ECATCH_ETHERCAT_ADDRESS	ECATCH_ETHERCAT_ADDRESS	Steigende Flanke an Execute
SlaveAddress	UINT	0	65535	0	Steigende Flanke an Execute

Tab. 9-21: Min-/Max- und Default-Werte der Eingänge IL_ECATCHSlaveConfigInfo

Funktionsbeschreibung

Mit dem Funktionsbaustein IL_ECATCHSlaveConfigInfo können Informationen zu einem konfigurierten Slave aus dem EtherCAT-Master ausgelesen werden.



Bei der EtherCAT-Konfiguration der Slaves muss die AutoInc-Adresse fortlaufend ohne Lücken vergeben sein [0, 0xFFFF (-1), 0xFFFE (-2), ...]. Auf diese Weise können die Info-Daten von allen konfigurierten Slaves mit einer einfachen Schleife ausgelesen werden; siehe nachfolgendes Programmier-Beispiel.

Implementierung in ST**Beispiel:**

```

VAR
    boExecute : BOOL := FALSE;
    boDone    : BOOL := FALSE;
    boError   : BOOL := FALSE;
    // arrays with max number of configured slaves, here: '128'
    arEthercatAddrList : ARRAY[0..127] OF UINT;
    arSlaveNameList : ARRAY[0..127] OF STRING(ECATCH_MAX_SLAVE_NAME_LEN);
    udiNumConfigSlaves : UDINT;
    uiSlaveIdx : UINT := 0;
    fbECATCHGetNumConfigSlaves: IL_ECATCHGetNumConfigSlaves;
    fbECATCHGetSlaveConfigInfo: IL_ECATCHGetSlaveConfigInfo;
END_VAR

// Example to read the EthercatAddr and the SlaveName
// from all configured Slaves

IF (boExecute = TRUE AND boDone = FALSE) THEN

    // get number of configured slaves
    fbECATCHGetNumConfigSlaves(
        Execute:= TRUE,
        NumConfigSlaves=> udiNumConfigSlaves );
    IF (TRUE = fbECATCHGetNumConfigSlaves.Error) THEN
        boError := TRUE;
    ELSIF (TRUE = fbECATCHGetNumConfigSlaves.Done) THEN

        // read SlaveConfigData from one slave after the other
        IF (uiSlaveIdx < udiNumConfigSlaves) THEN

```

```

fbECATGetSlaveConfigInfo(
    Execute:= TRUE,
    AddressType:= ECAT_AUTOINC_ADDRESS,
    SlaveAddress:= (0 - uiSlaveIdx) );
IF (TRUE = fbECATGetSlaveConfigInfo.Error) THEN
    boError := TRUE;
ELSIF (TRUE = fbECATGetSlaveConfigInfo.Done) THEN
    arEthercatAddrList[uiSlaveIdx] :=
        fbECATGetSlaveConfigInfo.SlaveConfigInfo.EthercatAddr;
    arSlaveNameList[uiSlaveIdx] :=
        fbECATGetSlaveConfigInfo.SlaveConfigInfo.SlaveName;
    // next slave
    fbECATGetSlaveConfigInfo(Execute:=FALSE);
    uiSlaveIdx := uiSlaveIdx + 1;
END_IF

ELSE // uiSlaveIdx
    boDone := TRUE; // Finished
END_IF
END_IF // fbECATGetNumConfigSlaves

ELSIF (boExecute = FALSE) THEN
    boDone := FALSE;
    boError:= FALSE;
    uiSlaveIdx := 0;
    fbECATGetNumConfigSlaves(Execute:=FALSE);
    fbECATGetSlaveConfigInfo(Execute:=FALSE);
END_IF

```

Fehlerbeschreibung Keine spezifischen Fehlercodes, siehe allgemeine Fehlercodes "EtherCAT Fehlertabelle".

9.4.6 IL_ECATGetSlaveOnlineInfo

Kurzbeschreibung Mit dem Funktionsbaustein IL_ECATGetSlaveOnlineInfo können Informationen zu einem aktuell verbundenen Slave aus dem EtherCAT-Master ausgelesen werden.

Schnittstellenbeschreibung

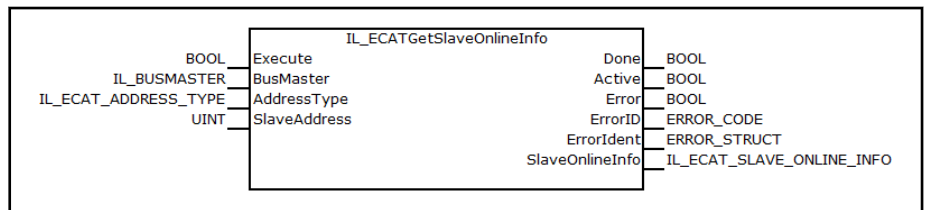


Abb. 9-11: Funktionsbaustein IL_ECATGetSlaveOnlineInfo

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des EtherCAT-Masters in der Reihenfolge des Konfigurationsbaumes
	AddressType	IL_ECAT_ADDRESS_TYPE	Auswahl der Slave-Adressierung (EtherCAT-Adresse oder AutoInc-Adresse)
	SlaveAddress	UINT	EtherCAT-Adresse oder aktuelle AutoInc-Adresse am Bus (abhängig vom Eingang "AddressType")
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers

E/A-Typ	Name	Datentyp	Beschreibung
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	SlaveOnlineInfo	IL_ECATCHSLAVE_ONLINE_INFO	Informationen zu einem aktuell verbundenen Slave

Tab. 9-22: Schnittstelle IL_ECATCHSlaveOnlineInfo

Name	Datentyp	Min	Max	Default	Übernahme
Execute	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
AddressType	IL_ECATCHADDRESS_TYPE	ECATCH_AUTO_INCREMENT_ADDRESS	ECATCH_ETHERCAT_ADDRESS	ECATCH_ETHERCAT_ADDRESS	Steigende Flanke an Execute
SlaveAddress	UINT	0	65535	0	Steigende Flanke an Execute

Tab. 9-23: Min-/Max- und Default-Werte der Eingänge IL_ECATCHSlaveOnlineInfo

Funktionsbeschreibung Mit dem Funktionsbaustein IL_ECATCHSlaveOnlineInfo können Informationen zu einem aktuell verbundenen Slave aus dem EtherCAT-Master ausgelesen werden.



Der EtherCAT-Master führt automatisch einen Bus-Scan durch, wenn eine Topologie-Änderung erkannt wird ("DL Status Event"). Die Funktion IL_ECATCHSlaveOnlineInfo liest die gesammelten Daten aus dem EtherCAT-Master aus.

Implementierung in ST *Beispiel:*

```

VAR
    boExecute : BOOL := FALSE;
    boDone    : BOOL := FALSE;
    boError   : BOOL := FALSE;
    // array with max number of connected slaves, here: '128'
    arSlaveOnlineInfo : ARRAY[0..127] OF IL_ECATCHSLAVE_ONLINE_INFO;
    udiNumOnlineSlaves: UDINT;
    uiSlaveIdx : UINT := 0;
    fbECATCHGetNumOnlineSlaves : IL_ECATCHGetNumOnlineSlaves;
    fbECATCHSlaveOnlineInfo : IL_ECATCHSlaveOnlineInfo;
END_VAR

// Example to read all SlaveOnlineInfo data

IF (boExecute = TRUE AND boDone = FALSE) THEN

    // get number of currently connected slaves
    fbECATCHGetNumOnlineSlaves(
        Execute:= TRUE,
        NumOnlineSlaves=> udiNumOnlineSlaves);
    IF (TRUE = fbECATCHGetNumOnlineSlaves.Error) THEN
        boError := TRUE;
    ELSIF (TRUE = fbECATCHGetNumOnlineSlaves.Done) THEN

        // read OnlineSlaveInfo from one slave after the other
        IF (uiSlaveIdx < udiNumOnlineSlaves) THEN

            fbECATCHSlaveOnlineInfo(
                Execute:= TRUE,
                AddressType:= ECATCH_AUTO_INCREMENT_ADDRESS,
                SlaveAddress:= (0 - uiSlaveIdx),
                SlaveOnlineInfo => arSlaveOnlineInfo[uiSlaveIdx] );
            IF (TRUE = fbECATCHSlaveOnlineInfo.Error) THEN
                boError := TRUE;
            // Note: maybe the number of OnlineSlaves has just changed
            ELSIF (TRUE = fbECATCHSlaveOnlineInfo.Done) THEN

```

```

// next slave
fbECATGetSlaveOnlineInfo(Execute:=FALSE);
uiSlaveIdx := uiSlaveIdx + 1;
END_IF

ELSE // uiSlaveIdx
  boDone := TRUE; // Finished
END_IF
END_IF // fbECATGetNumOnlineSlaves

ELSIF (boExecute = FALSE) THEN
  boDone := FALSE;
  boError:= FALSE;
  uiSlaveIdx := 0;
  fbECATGetNumOnlineSlaves(Execute:=FALSE);
  fbECATGetSlaveOnlineInfo(Execute:=FALSE);
END_IF

```

Fehlerbeschreibung Keine spezifischen Fehlercodes, siehe allgemeine Fehlercodes "EtherCAT Fehlertabelle".

9.4.7 IL_ECATMasterState

Kurzbeschreibung Der Funktionsbaustein IL_ECATMasterState liefert eine Zusammenfassung von Diagnoseinformationen des EtherCAT-Masters.

Schnittstellenbeschreibung

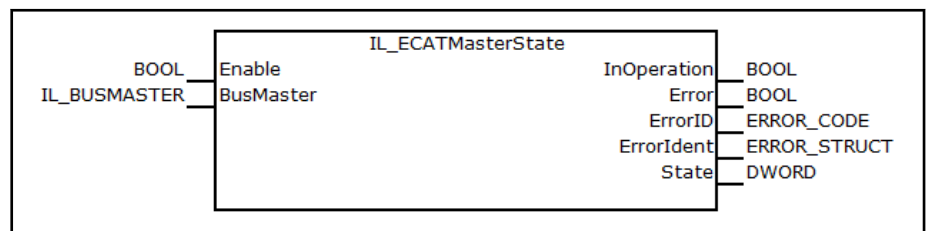


Abb. 9-12: Funktionsbaustein IL_ECATMasterState

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (permanent, pegelgesteuert)
	BusMaster	IL_BUSMASTER	Instanz des EtherCAT-Masters in der Reihenfolge des Konfigurationsbaumes
VAR_OUTPUT	InOperation	BOOL	Zyklischer Bearbeitungszustand
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	State	DWORD	Diagnose-Statusinformationen des EtherCAT-Masters. Details siehe IL_ECAT_MASTER_STATE

Tab. 9-24: Schnittstelle IL_ECATMasterState

Name	Datentyp	Min	Max	Default	Übernahme
Enable	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Enable

Tab. 9-25: Min-/Max- und Default-Werte der Eingänge IL_ECATMasterState

Funktionsbeschreibung Der Funktionsbaustein IL_ECATMasterState liefert eine Zusammenfassung von Diagnoseinformationen des EtherCAT-Masters.

Ist der Ausgang "State" = 0, dann liegt keine Diagnose vor und der Master ist OK. Für eine Beschreibung der Diagnosebits des "State"-Ausgang siehe Struktur IL_ECATCHMASTER_STATE, sowie Hilfsfunktion IL_ECATCHMASTERStateDecoding.

Implementierung in ST Beispiel:

```
VAR
  fbECATCHMASTERState : IL_ECATCHMASTERState;
  boEnable      : BOOL := FALSE;
  boMasterOK    : BOOL; // FALSE = Master diagnosis available
  rMasterState  : IL_ECATCHMASTER_STATE;
  eActBusState  : IL_ECATCHMASTER_STATE;
END_VAR

// Example to read Master State diagnosis (cyclic call)
fbECATCHMASTERState(
  Enable:= boEnable,
  BusMaster:= ,
  InOperation=> ,
  Error=> ,
  ErrorID=> ,
  ErrorIdent=> ,
  State=> );

IF (TRUE = fbECATCHMASTERState.InOperation) THEN
  // Master OK
  boMasterOK := NOT( DWORD_TO_BOOL(fbECATCHMASTERState.State) );
  // Master diagnosis bits
  rMasterState := IL_ECATCHMASTERStateDecoding( fbECATCHMASTERState.State );
  // Actual Bus State
  IF (FALSE = rMasterState.NotInStateOp) THEN
    eActBusState := ECATCHMASTER_STATE_OP;
  ELSE
    eActBusState := DWORD_TO_DINT( SHR((fbECATCHMASTERState.State AND 16#0070), 4) );
  END_IF
ELSE
  boMasterOK := FALSE;
  eActBusState := ECATCHMASTER_STATE_UNKNOWN;
  ; // ToDo error handling
END_IF
```

Fehlerbeschreibung Keine spezifischen Fehlercodes, siehe allgemeine Fehlercodes "EtherCAT Fehlertabelle".

9.4.8 IL_ECATCHMASTERBusStatistic

Kurzbeschreibung Der Funktionsbaustein IL_ECATCHMASTERBusStatistic liefert statistische Diagnoseinformationen zur Buskommunikation des EtherCAT-Masters.

Schnittstellenbeschreibung

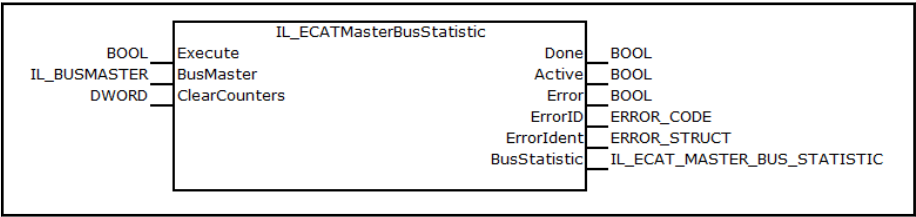


Abb. 9-13: Funktionsbaustein IL_ECATCHMASTERBusStatistic

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des EtherCAT-Masters in der Reihenfolge des Konfigurationsbaumes
	ClearCounters	DWORD	Reset der Frame und Datagramm Zähler nach dem Lesen (1 = löscht alle Zähler)

E/A-Typ	Name	Datentyp	Beschreibung
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	BusStatistic	IL_ECATCHMASTERBUS_STATISTIC	Statistische Diagnoseinformationen zur Buskommunikation des EtherCAT-Masters

Tab. 9-26: Schnittstelle IL_ECATCHMASTERBUSStatistic

Name	Datentyp	Min	Max	Default	Übernahme
Execute	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
ClearCounters	DWORD	0	n.def.	0	Steigende Flanke an Execute

Tab. 9-27: Min-/Max- und Default-Werte der Eingänge IL_ECATCHMASTERBUSStatistic

Funktionsbeschreibung

Der Funktionsbaustein IL_ECATCHMASTERBUSStatistic liefert statistische Diagnoseinformationen zur Buskommunikation des EtherCAT-Masters.

Wenn der Eingang "ClearCounters" > 0 gesetzt wird, werden folgende Zähler nach dem Lesen der Daten zurückgesetzt (Bit x vom Eingang "ClearCounters"):

- Bit 0: löscht alle Zähler (ClearCounters = 1)
- Bit 1: löscht (SI7) Zähler "TxFrames"
- Bit 2: löscht (SI8) Zähler "RxFrames"
- Bit 3: löscht (SI9) Zähler "LostFrames"
- Bit 4: löscht (SI10) Zähler "CyclicFrames"
- Bit 5: löscht (SI11) Zähler "CyclicDatagrams"
- Bit 6: löscht (SI12) Zähler "AcyclicFrames"
- Bit 7: löscht (SI13) Zähler "AcyclicDatagrams"

Implementierung in ST

Beispiel:

```

VAR
    fbECATCHMASTERBUSStatistic : IL_ECATCHMASTERBUSStatistic;
    boExecute : BOOL := FALSE;
    // 'ClearCounters': set value '1' to clear all counters
    dwClearCounters : DWORD := 0;
    rBusStatistic : IL_ECATCHMASTERBUS_STATISTIC;
END_VAR

// Example to read the Master Bus Statistic
fbECATCHMASTERBUSStatistic(
    Execute:= boExecute,
    BusMaster:= ,
    ClearCounters:= dwClearCounters,
    Done=> ,
    Active=> ,
    Error=> ,
    ErrorID=> ,
    ErrorIdent=> ,
    BusStatistic=> rBusStatistic);

```

Fehlerbeschreibung

Keine spezifischen Fehlercodes, siehe allgemeine Fehlercodes "EtherCAT Fehlertabelle".

9.4.9 IL_ECATRemoteSlaveState

Kurzbeschreibung

Der Funktionsbaustein IL_ECATRemoteSlaveState liefert Diagnose-Statusinformationen eines EtherCAT-Slave-Gerätes.

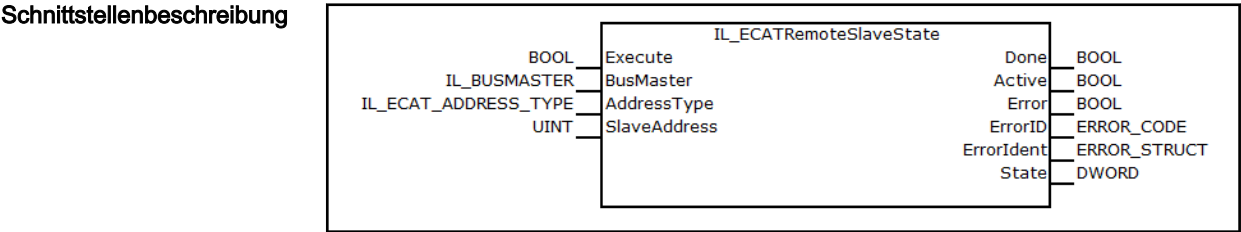


Abb. 9-14: Funktionsbaustein IL_ECATRemoteSlaveState

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des EtherCAT-Masters in der Reihenfolge des Konfigurationsbaumes
	AddressType	IL_ECAT_ADD-RESS_TYPE	Auswahl der Slave-Adressierung (EtherCAT-Adresse oder AutoInc-Adresse)
	SlaveAddress	UINT	EtherCAT-Adresse oder aktuelle AutoInc-Adresse am Bus (abhängig vom Eingang "AddressType")
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	State	DWORD	Diagnose Status-Informationen des Slave Gerätes, Details siehe IL_ECAT_SLAVE_STATE

Tab. 9-28: Schnittstelle IL_ECATRemoteSlaveState

Name	Datentyp	Min	Max	Default	Übernahme
Execute	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMAS-TER	IL_BUSMAS-TER_0	IL_BUSMAS-TER_0	IL_BUSMAS-TER_0	Steigende Flanke an Execute
AddressType	IL_ECAT_ADD-RESS_TYPE	ECAT_AUTO-INC_ADDRESS	ECAT_ETHERCAT_ADDRESS	ECAT_ETHERCAT_ADDRESS	Steigende Flanke an Execute
SlaveAddress	UINT	0	65535	0	Steigende Flanke an Execute

Tab. 9-29: Min-/Max- und Default-Werte der Eingänge IL_ECATRemoteSlaveState

Funktionsbeschreibung

Der Funktionsbaustein IL_ECATRemoteSlaveState liefert Diagnose-Statusinformationen eines EtherCAT-Slave-Gerätes.

Ist der Ausgang "State" = 0, dann liegt keine Diagnose vor und der Slave ist OK. Für eine Beschreibung der Diagnose-Bits des "State"-Ausgang siehe Struktur IL_ECAT_SLAVE_STATE, sowie Hilfsfunktion IL_ECATSlaveStateDecoding.

Implementierung in ST *Beispiel:*

```

VAR CONSTANT
    // max number of configured slaves, here: '128'
    uiMAX_CONFIG_SLAVES : UINT := 128;
END_VAR
VAR
    boEnable           : BOOL := FALSE;
    boInOperation      : BOOL := FALSE;
    boError            : BOOL := FALSE;
    arEthercatAddrList : ARRAY [0..(uiMAX_CONFIG_SLAVES-1)] OF UINT;
    arSlaveOK          : ARRAY [0..(uiMAX_CONFIG_SLAVES-1)] OF BOOL;
    arSlaveState       : ARRAY [0..(uiMAX_CONFIG_SLAVES-1)] OF IL_ECAT_SLAVE_STATE;
    arSlaveDevState    : ARRAY [0..(uiMAX_CONFIG_SLAVES-1)] OF IL_ECAT_STATE;
    udiNumConfigSlaves : UDINT := 0;
    uiSlaveIdx         : UINT := 0;
    pDword             : POINTER TO DWORD;
    fbECATGetConfigEthercatAddrList : IL_ECATGetConfigEthercatAddrList;
    fbECATRemoteSlaveState          : IL_ECATRemoteSlaveState;
END_VAR

// Example to read the RemoteSlaveState from all configured slaves

IF (boEnable = TRUE) THEN

    IF (boInOperation = FALSE) THEN
        (* Init *)

        // get all configured EtherCAT addresses
        fbECATGetConfigEthercatAddrList(
            Execute:= TRUE,
            SizeOfEthercatAddrList:= SIZEOF(arEthercatAddrList),
            EthercatAddrListAdr:= ADR(arEthercatAddrList),
            NumConfigSlaves=> udiNumConfigSlaves );
        IF (fbECATGetConfigEthercatAddrList.Error) THEN
            boError := TRUE;
        ELSIF (fbECATGetConfigEthercatAddrList.Done = TRUE) THEN
            boInOperation := TRUE;
        END_IF
    ELSE //
        (* InOperation *)

        // Hint: you may read the RemoteSlaveStates only if MasterState <> 0

        // read the RemoteSlaveState from one slave after the other
        IF (uiSlaveIdx < udiNumConfigSlaves) THEN

            fbECATRemoteSlaveState(
                Execute:= TRUE,
                AddressType:= ECAT_ETHERCAT_ADDRESS,
                SlaveAddress:= arEthercatAddrList[uiSlaveIdx],
                State=> );

            IF (TRUE = fbECATRemoteSlaveState.Error) THEN
                boError := TRUE;
            ELSIF (TRUE = fbECATRemoteSlaveState.Done) THEN
                // Slave diagnosis
                arSlaveOK[uiSlaveIdx] :=
                    NOT( DWORD_TO_BOOL(fbECATRemoteSlaveState.State) );
                arSlaveState[uiSlaveIdx] :=
                    IL_ECATSlaveStateDecoding( fbECATRemoteSlaveState.State );

                // Slave device state
                IF (TRUE = arSlaveState[uiSlaveIdx].NotOnline) THEN
                    arSlaveDevState[uiSlaveIdx] := ECAT_STATE_UNKNOWN;
                ELSIF (TRUE = NOT(arSlaveState[uiSlaveIdx].NotInStateOp)) THEN
                    arSlaveDevState[uiSlaveIdx] := ECAT_STATE_OP;
                ELSE
                    pDword := ADR(arSlaveState[uiSlaveIdx]);
                    arSlaveDevState[uiSlaveIdx] :=
                        DWORD_TO_DINT( SHR((pDword^ AND 16#0070), 4) );
                END_IF
            END_IF
        END_IF
    END_IF

```

```
// go to the next slave
fbECATRemoteSlaveState(Execute:=FALSE);
uiSlaveIdx := uiSlaveIdx + 1;
END_IF

ELSE // uiSlaveIdx
  // start again with first slave
  uiSlaveIdx := 0;
END_IF
END_IF // InOperation

ELSE // boEnable
  boError      := FALSE;
  boInOperation := FALSE;
  uiSlaveIdx   := 0;
  fbECATGetConfigEthercatAddrList(Execute:=FALSE);
  fbECATRemoteSlaveState(Execute:=FALSE);
END_IF
```

Fehlerbeschreibung Keine spezifischen Fehlercodes, siehe allgemeine Fehlercodes "EtherCAT Fehlertabelle".

9.4.10 IL_ECATRemoteSlaveSetErrorCounterPeriod

Kurzbeschreibung Der Funktionsbaustein IL_ECATRemoteSlaveSetErrorCounterPeriod aktiviert eine Master-Funktion zum zyklischen Einsammeln der Fehlerzähler von allen Slave-Geräten.

Schnittstellenbeschreibung

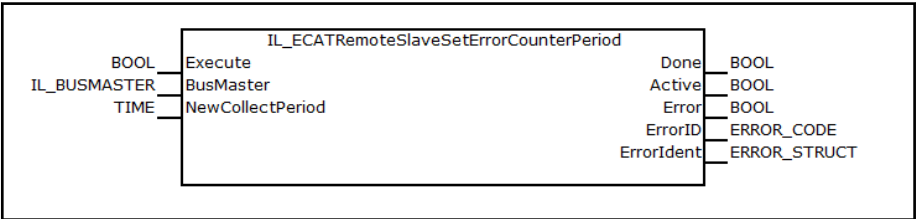


Abb. 9-15: Funktionsbaustein IL_ECATRemoteSlaveSetErrorCounterPeriod

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des EtherCAT-Masters in der Reihenfolge des Konfigurationsbaumes
	NewCollectPeriod	TIME	Intervall zum Einsammeln der Fehlerzähler (deaktiviert mit T#0S), Empfehlung: Wert > "100 ms * NumConfigSlaves".
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers

Tab. 9-30: Schnittstelle IL_ECATRemoteSlaveSetErrorCounterPeriod

Name	Datentyp	Min	Max	Default	Übernahme
Execute	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
NewCollectPeriod	TIME	T#0ms	n. def.	T#10s	Steigende Flanke an Execute

Tab. 9-31: Min-/Max- und Default-Werte der Eingänge IL_ECATRemoteSlaveSetErrorCounterPeriod

Funktionsbeschreibung

Der Funktionsbaustein IL_ECATRemoteSlaveSetErrorCounterPeriod aktiviert eine Master-Funktion zum zyklischen Einsammeln der Fehlerzähler von allen Slave-Geräten.

Am Eingang "NewCollectPeriod" wird das Aktualisierungsintervall gesetzt, in dem der Master die Fehlerzähler von einem Slave nach dem anderen einsammelt. Es wird empfohlen, eine "NewCollectPeriod" > "100 ms * NumConfigSlaves" einzustellen, um die Kommunikationslast der azyklischen Buszugriffe zu begrenzen.

Der Wert "NewCollectPeriod" = T#0S deaktiviert diese Master-Funktion.



Diese Master-Funktion zum Einsammeln der Fehlerzähler kann auch durch einen anderen Client (z.B. MLPI) de-/aktiviert werden.

Mit dem Funktionsbaustein IL_ECATRemoteSlaveGetErrorCounter können die eingesammelten Fehlerzähler der Slaves aus dem EtherCAT-Master ausgelesen werden.

Mit dem Funktionsbaustein IL_ECATRemoteSlaveClearErrorCounter können die Fehlerzähler in allen Slaves gelöscht werden.

Implementierung in ST

Beispiel:

```

VAR
    fbECATRemoteSlaveSetErrorCounterPeriod : IL_ECATRemoteSlaveSetErrorCounterPeriod;
    boExecute : BOOL := FALSE;
    tNewCollectPeriod : TIME := T#10S;
END_VAR

fbECATRemoteSlaveSetErrorCounterPeriod(
    Execute:= boExecute,
    BusMaster:= ,
    NewCollectPeriod:= tNewCollectPeriod,
    Done=> ,
    Active=> ,
    Error=> ,
    ErrorID=> ,
    ErrorIdent=> );

```

Fehlerbehandlung

Keine spezifischen Fehlercodes, siehe allgemeine Fehlercodes "EtherCAT Fehlertabelle".

9.4.11 IL_ECATRemoteSlaveGetErrorCounterPeriod

Kurzbeschreibung

Der Funktionsbaustein IL_ECATRemoteSlaveGetErrorCounterPeriod liest das aktuelle Intervall des EtherCAT-Masters zum Einsammeln der Fehlerzähler der Slave Geräte.

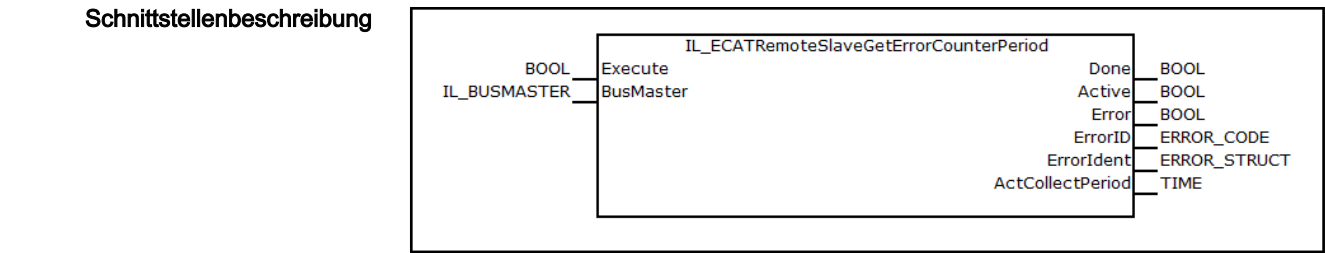


Abb. 9-16: Funktionsbaustein IL_ECATERemoteSlaveGetErrorCounterPeriod

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des EtherCAT-Masters in der Reihenfolge des Konfigurationsbaumes
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	ActCollectPeriod	TIME	Das aktuelle Intervall des EtherCAT-Masters zum Einsammeln der Fehlerzähler der Slave Geräte (deaktiviert mit T#0S)

Tab. 9-32: Schnittstelle IL_ECATERemoteSlaveGetErrorCounterPeriod

Name	Datentyp	Min	Max	Default	Übernahme
Execute	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute

Tab. 9-33: Min-/Max- und Default-Werte der Eingänge IL_ECATERemoteSlaveGetErrorCounterPeriod

Funktionsbeschreibung

Der Funktionsbaustein IL_ECATERemoteSlaveGetErrorCounterPeriod liest das aktuelle Intervall des EtherCAT-Masters zum Einsammeln der Fehlerzähler der Slave-Geräte.

Eine "ActCollectPeriod" = T#0S zeigt an, dass die Sammelfunktion im Master deaktiviert ist.

Mit dem Funktionsbaustein IL_ECATERemoteSlaveSetErrorCounterPeriod kann ein neues Intervall gesetzt werden.



Da verschiedene Clients (z.B. MLPI) dieses Sammelintervall ändern können, z. B. auch Deaktivieren, kann mit dem Funktionsbaustein IL_ECATERemoteSlaveGetErrorCounterPeriod der aktuelle Wert überprüft werden.

Implementierung in ST Beispiel:

VAR

```

fbECATRemoteSlaveGetErrorCounterPeriod : IL_ECATRemoteSlaveGetErrorCounterPeriod;
boExecute : BOOL := FALSE;
tActCollectPeriod : TIME;
END_VAR

fbECATRemoteSlaveGetErrorCounterPeriod(
    Execute:= boExecute,
    BusMaster:= ,
    Done=> ,
    Active=> ,
    Error=> ,
    ErrorID=> ,
    ErrorIdent=> ,
    ActCollectPeriod => tActCollectPeriod );

```

Fehlerbehandlung Keine spezifischen Fehlercodes, siehe allgemeine Fehlercodes "EtherCAT Fehlertabelle".

9.4.12 IL_ECATRemoteSlaveGetErrorCounter

Kurzbeschreibung Der Funktionsbaustein IL_ECATRemoteSlaveGetErrorCounter liest die gesammelten Fehlerzähler eines Slave-Gerätes aus dem Master aus.

Schnittstellenbeschreibung

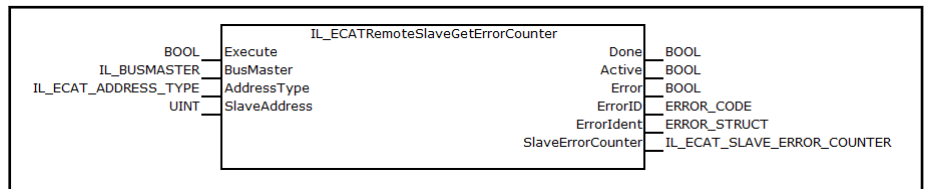


Abb. 9-17: Funktionsbaustein IL_ECATRemoteSlaveGetErrorCounter

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des EtherCAT-Masters in der Reihenfolge des Konfigurationsbaumes
	AddressType	IL_ECAT_ADDRESS_TYPE	Auswahl der Slave-Adressierung (EtherCAT-Adresse oder AutoInc-Adresse)
	SlaveAddress	UINT	EtherCAT-Adresse oder aktuelle AutoInc-Adresse am Bus (abhängig vom Eingang "AddressType")
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	SlaveErrorCounter	IL_ECAT_SLAVE_ERROR_COUNTER	Fehlerzähler des Slave-Gerätes

Tab. 9-34: Schnittstelle IL_ECATRemoteSlaveGetErrorCounter

Name	Datentyp	Min	Max	Default	Übernahme
Execute	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute

Name	Datentyp	Min	Max	Default	Übernahme
AddressType	IL_ECATA_ADD- RESS_TYPE	ECAT_AUTO- INC_ADDRESS	ECAT_ETHERC AT_ADDRESS	ECAT_ETHER CAT_ADD- RESS	Steigende Flanke an Execute
SlaveAddress	UINT	0	65535	0	Steigende Flanke an Execute

Tab. 9-35: Min-/Max- und Default-Werte der Eingänge IL_ECATRemoteSlaveGetErrorCounter

Funktionsbeschreibung

Der Funktionsbaustein IL_ECATRemoteSlaveGetErrorCounter liest die gesammelten Fehlerzähler eines Slave-Gerätes aus dem Master aus.

Die Master-Funktion zum Einsammeln der Slave-Fehlerzähler muss vorher mit dem Funktionsbaustein IL_ECATRemoteSlaveSetErrorCounterPeriod aktiviert werden.



Der Funktionsbaustein IL_ECATRemoteSlaveGetErrorCounter meldet keinen Fehler, wenn die Master-Funktion zum Einsammeln nicht aktiviert ist.

Implementierung in ST

Beispiel:

```

VAR CONSTANT
    // max number of configured slaves, here: '128'
    uiMAX_CONFIG_SLAVES : UINT := 128;
END_VAR
VAR
    boExecute : BOOL := FALSE;
    boDone    : BOOL := FALSE;
    boError    : BOOL := FALSE;
    arEthercatAddrList : ARRAY[0..(uiMAX_CONFIG_SLAVES-1)] OF UINT;
    arSlaveErrorCounter : ARRAY[0..(uiMAX_CONFIG_SLAVES-1)] OF IL_ECAT_SLAVE_ERROR_COUNTER;
    uiSlaveIdx : UINT := 0;
    udiNumConfigSlaves : UDINT;
    fbECATGetConfigEthercatAddrList : IL_ECATGetConfigEthercatAddrList;
    fbECATRemoteSlaveGetErrorCounter : IL_ECATRemoteSlaveGetErrorCounter;
END_VAR

// Example to read the collected slave's error counter from the Master
// (activate with IL_ECATRemoteSlaveSetErrorCounterPeriod)

IF (boExecute = TRUE AND boDone = FALSE) THEN
    // get all configured EtherCAT addresses
    fbECATGetConfigEthercatAddrList(
        Execute:= TRUE,
        SizeOfEthercatAddrList:= SIZEOF(arEthercatAddrList),
        EthercatAddrListAdr:= ADR(arEthercatAddrList),
        NumConfigSlaves=> udiNumConfigSlaves );
    IF (fbECATGetConfigEthercatAddrList.Error) THEN
        boError := TRUE;
    ELSIF (fbECATGetConfigEthercatAddrList.Done = TRUE) THEN
        // read Slave ErrorCounter from one slave after the other
        IF (uiSlaveIdx < udiNumConfigSlaves) THEN
            fbECATRemoteSlaveGetErrorCounter(
                Execute:= TRUE,
                AddressType:= ECAT_ETHERCAT_ADDRESS,
                SlaveAddress:= arEthercatAddrList[uiSlaveIdx],
                SlaveErrorCounter=> arSlaveErrorCounter[uiSlaveIdx] );

            IF (TRUE = fbECATRemoteSlaveGetErrorCounter.Error) THEN
                boError := TRUE;
            ELSIF (TRUE = fbECATRemoteSlaveGetErrorCounter.Done) THEN
                // next slave
                fbECATRemoteSlaveGetErrorCounter(Execute:=FALSE);
                uiSlaveIdx := uiSlaveIdx + 1;
            END_IF
        ELSE // uiSlaveIdx
            boDone := TRUE; // Finished
        END_IF
    END_IF

```

```

        END_IF
    END_IF

ELSIF (boExecute = FALSE) THEN
    boDone := FALSE;
    boError:= FALSE;
    uiSlaveIdx := 0;
    fbECATGetConfigEthercatAddrList (Execute:=FALSE);
    fbECATRemoteSlaveGetErrorCounter (Execute:=FALSE);
END_IF

```

Fehlerbeschreibung Keine spezifischen Fehlercodes, siehe allgemeine Fehlercodes "EtherCAT Fehlertabelle".

9.4.13 IL_ECATRemoteSlaveClearErrorCounter

Kurzbeschreibung Der Funktionsbaustein IL_ECATRemoteSlaveClearErrorCounter löscht alle Fehlerzähler in allen Slave-Geräten.

Schnittstellenbeschreibung

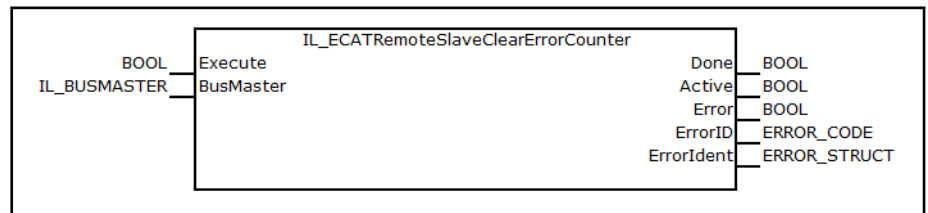


Abb. 9-18: Funktionsbaustein IL_ECATRemoteSlaveClearErrorCounter

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des EtherCAT-Masters in der Reihenfolge des Konfigurationsbaumes
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers

Tab. 9-36: Schnittstelle IL_ECATRemoteSlaveClearErrorCounter

Name	Datentyp	Min	Max	Default	Übernahme
Enable	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute

Tab. 9-37: Min-/Max- und Default-Werte der Eingänge IL_ECATRemoteSlaveClearErrorCounter

Funktionsbeschreibung Der Funktionsbaustein IL_ECATRemoteSlaveClearErrorCounter löscht alle Fehlerzähler in allen Slave-Geräten.

Implementierung in ST *Beispiel:*

```

VAR
    fbECATRemoteSlaveClearErrorCounter: IL_ECATRemoteSlaveClearErrorCounter;
    boExecute : BOOL := FALSE;
END_VAR

```

```
fbECATRemoteSlaveClearErrorCounter(
  Execute:= boExecute,
  BusMaster:= ,
  Done=> ,
  Active=> ,
  Error=> ,
  ErrorID=> ,
  ErrorIdent=> );
```

Fehlerbeschreibung Keine spezifischen Fehlercodes, siehe allgemeine Fehlercodes "EtherCAT Fehlertabelle".

9.4.14 IL_ECATCoeReadDiagHistory

Kurzbeschreibung Mit dem Funktionsbaustein IL_ECATCoeReadDiagHistory kann das CoE "Diagnosis History" Objekt (Index 16#10F3) von einem EtherCAT-Slave-Gerät azyklisch ausgelesen werden.

Schnittstellenbeschreibung

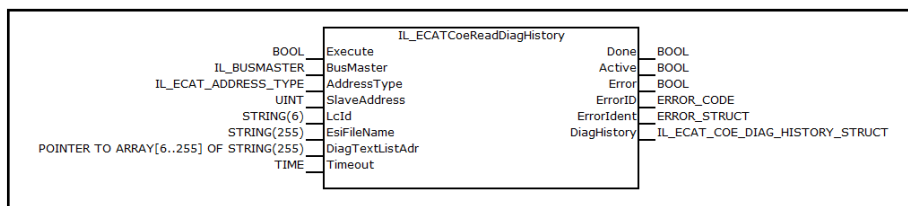


Abb. 9-19: Funktionsbaustein IL_ECATCoeReadDiagHistory

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des EtherCAT-Masters in der Reihenfolge des Konfigurationsbaumes
	AddressType	IL_ECAT_ADDRESS_TYPE	Auswahl der Slave-Adressierung (EtherCAT-Adresse oder AutoInc-Adresse)
	SlaveAddress	UINT	EtherCAT-Adresse oder aktuelle AutoInc-Adresse am Bus (abhängig vom Eingang "AddressType")
	LcId	STRING(6)	[Optional] "Local Id" (Language Code Identifier): Sprach-ID, siehe ESI Datei, welche Sprachen bei den Diagnosetexten unterstützt werden (z.B. "1033" = EN-US, "1031" = DE-DE)
	EsiFileName	STRING(255)	[Optional] Dateiname der ESI-Datei inkl. Pfad (auf Flashkarte der Steuerung)
	DiagTextListAdr	POINTER TO ARRAY[6..255] OF STRING(255)	[Optional] Diagnose Text-Liste über die Diagnose Nachrichten. Achtung: der Datentyp muss genau übereinstimmen, da der Array[Index] mit dem Index der "Diag-Messages" zusammenhängt.
	Timeout	TIME	Timeout-Überwachung für die intern verwendeten azyklischen Dienste.
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig

E/A-Typ	Name	Datentyp	Beschreibung
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	DiagHistory	IL_ECACOE_DIAG_HISTORY_STRUCT	"Diagnosis History" Objekt (Index 16#10F3) des Slave Gerätes

Tab. 9-38: Schnittstelle IL_ECACoeReadDiagHistory

Name	Datentyp	Min	Max	Default	Übernahme
Execute	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
AddressType	IL_ECACOE_ADDRESSTYPE	ECAT_AUTOINC_ADDRESS	ECAT_ETHERCAT_ADDRESS	ECAT_ETHERCAT_ADDRESSTYPE	Steigende Flanke an Execute
SlaveAddress	UINT	0	65535	0	Steigende Flanke an Execute
LcId	STRING(6)	-	-	'1033'	Steigende Flanke an Execute
EsiFileName	STRING(255)	-	-	'	Steigende Flanke an Execute
DiagTextListAdr	POINTER TO ARRAY[6..255] OF STRING(255)	n.def	n.def	0	Kontinuierlich
Timeout	TIME	T#1ms	n.def	T#3s	Steigende Flanke an Execute

Tab. 9-39: Min-/Max- und Default-Werte der Eingänge IL_ECACoeReadDiagHistory

Funktionsbeschreibung

Mit dem Funktionsbaustein IL_ECACoeReadDiagHistory kann das CoE "Diagnosis History" Objekt (Index 16#10F3) von einem EtherCAT-Slave-Gerät azyklisch ausgelesen werden.

Voraussetzung ist, dass das Slave-Gerät das Mailboxprotokoll "CAN application protocol over EtherCAT" (CoE) unterstützt. Mailboxkommunikation ist nur in den EtherCAT-Zuständen PreOP, SafeOP und OP möglich.



Das "Diagnosis History" Objekt (Index 16#10F3) ist optional und wird nicht von jedem CoE-Slave unterstützt. Zudem bietet das Objekt optionale Funktionen an. Welche Funktionen von einem Slave unterstützt werden, sind bei Bedarf in der Produktdokumentation oder Gerätebeschreibung (ESI) des Slave-Gerätes nachzuschlagen.

Allgemein:

- Es gibt verschiedene Typen von Diagnosenachrichten: Info, Warnung, Fehler
- Es können bis zu 250 Diagnosenachrichten im Slave-Gerät gespeichert werden (Ringpuffer)
- Eine Diagnose-Nachricht enthält eine "TextId" als Referenz für eine ausführliche Fehlerbeschreibung in der Gerätebeschreibung (ESI)
- Der FB stellt die Diagnosen-Nachrichten (SI6-255) nicht als Rohdaten dar, sondern in einer erweiterten Datenstruktur um eine einfache Auswertung zu ermöglichen

- Man unterscheidet zwei Modi:

- "Overwrite Mode"
- "Acknowledge Mode"

Welcher Modus aktiv ist, wird bei "Flags" (SI5, Bit 0.4) angezeigt. Eine Umschaltung des Modus (Schreibzugriff) wird nicht von allen Slaves unterstützt

- Normalerweise ist das "NewDiagMessages" Bit (SI4) Teil der zyklischen Kommunikation (PDO Konfiguration) des Slaves um anzuzeigen, wenn eine neue Diagnosenachricht vorhanden ist

"Overwrite Mode":

- Wenn der Nachrichtenpuffer voll ist, überschreibt eine neue Nachricht die älteste Nachricht im Puffer. Auch unbestätigte Nachrichten werden überschrieben
- Bestätigen von Nachrichten:
Durch Schreiben der "NewestAckMessage" (SI3) werden alle älteren Nachrichten bis zu dem Wert (SubIndex) bestätigt, der geschrieben wird; unabhängig davon, ob diese vorher gelesen wurden
- Löschen von Nachrichten:
Durch Schreiben der "NewestAckMessage" (SI3) mit dem Wert "0" werden alle Nachrichten gelöscht, auch wenn diese unbestätigt oder ungelesen waren (optionale Funktionalität im Slave)
- Wird ein SubIndex-Wert geschrieben, der keine Nachricht enthält, können folgende CoE Abortcodes auftreten:
 - 0x06090030
 - 0x06090031
 - 0x06090032

"Acknowledge Mode":

- Unbestätigte Nachrichten werden nicht überschrieben. D.h. wenn der Nachrichtenpuffer voll ist, werden neue Nachrichten nicht gespeichert und gehen verloren
- Bestätigen von Nachrichten:
Durch Schreiben der "NewestAckMessage" (SI3) werden alle älteren Nachrichten bis zu dem Wert (SubIndex) bestätigt, der geschrieben wird; unabhängig davon, ob diese vorher gelesen wurden
- Löschen von Nachrichten:
Durch Schreiben der "NewestAckMessage" (SI3) mit dem Wert "0" werden alle Nachrichten gelöscht, die vorher bestätigt wurden (optionale Funktionalität im Slave)
- Wird ein SubIndex-Wert geschrieben, der keine Nachricht enthält, können folgende CoE Abortcodes auftreten:
 - 0x06090030
 - 0x06090031
 - 0x06090032

Zusätzliche Hinweise:

- Zum Schreiben eines SDO-SubIndex kann der Funktionsbaustein IL_ECACoeWriteSdo mit folgenden Parametern verwendet werden:
 - Index = 16#10F3 ("Diagnosis History" Objekt)

- SubIndex = 3 (SI3)
- Beim Schreiben der "Flags" (SI5) zur Einstellung der Betriebsmodi des Diagnoseobjekts ist zu beachten, dass einige Steuerbits der "Flags" vom Slave-Gerät ggf. nicht geändert werden können. Bei diesen Steuerbits muss der Schreibwert dann dem Istwert entsprechen, andernfalls können folgende CoE-Abortcodes auftreten:
 - 0x06090030 u.a.
- **ESI-Datei:**
 - Der Eingang "EsiFileName" kann optional verwendet werden. Die ESI-Datei muss hierfür manuell auf der Flashkarte der Steuerung abgelegt werden. Der Dateiname muss dann inklusive Pfad angegeben werden.
Zum Beispiel auf der USER-Partition in einem Unterordner ESI:
strEsiFileName : STRING := '/USER/ESI/S20-EC-BK.xml'
 - Der Funktionsbaustein überprüft nicht, ob die angegebene ESI mit dem adressierten Slave-Gerät übereinstimmt. Es wird der erste "DiagMessages" Eintrag in der ESI verwendet
 - Die ESI muss im Format UTF-8 vorliegen; bei einem anderen Format ist dieses Format ggf. zu konvertieren
 - Wird die angegebene Sprach-ID (Eingang "Lcld") in der ESI nicht gefunden, wird alternativ der Lcld Wert '1033' (EN-US) versucht; wird dieser auch nicht gefunden, entsteht ein Fehler
 - Der Funktionsbaustein hat folgende Einschränkungen bei der generischen Parameterliste einer Diagnosenachricht, die im "MessageText" aus der ESI referenziert werden:
 - (a) Es werden maximal bis zu 12 Parameter unterstützt
 - (b) Die Parameter müssen in aufsteigender Reihenfolge im "MessageText" referenziert werden

SI:	Diagnosis Message
6	ERROR: Ack=Yes, SlaveTime: 2000-01-01T00:01:41, DiagCode: 0x3412E800, (TextId: 0x3412) Module 2: I/O supply failure
7	INFO: Ack=Yes, SlaveTime: 2000-01-01T00:01:42, DiagCode: 0x3412E800, (TextId: 0x3412) Module 2: I/O supply failure
8	ERROR: Ack=No, SlaveTime: 2000-01-01T00:01:44, DiagCode: 0x5160E800, (TextId: 0x5160) Module 4: Faulty supply voltage
9	INFO: Ack=No, SlaveTime: 2000-01-01T00:01:45, DiagCode: 0x5160E800, (TextId: 0x5160) Module 4: Faulty supply voltage
10	

Abb. 9-20: Exemplarischer Screenshot für die Darstellung der Diagnosenachricht (Eingang "DiagTextListAdr")

Implementierung in ST Beispiel:

```

VAR
  fbECATCoeReadDiagHistory: IL_ECATCoeReadDiagHistory;
  boExecute : BOOL := FALSE;
  uiEthercatAddr : UINT := 1001;
  // ESI example: you have to store manually the corresponding ESI file
  // on the compact flash of the control,
  // here: USER-partition with subfolder 'ESI'
  strEsiFileName : STRING := '/USER/ESI/S20-EC-BK.xml';
  // see ESI which LcId is supported, e.g. '1033' = EN-US; '1031' = DE-DE
  strLcId : STRING(6) := '1033';
  arDiagTextList : ARRAY[6..255] OF STRING(255);
END_VAR

// Example to read CoE 'Diagnosis History' (16#10F3)
// Note: Slave must support CoE and the 'Diagnosis History' (16#10F3) object
fbECATCoeReadDiagHistory(
  Execute:= boExecute,
  BusMaster:= ,
  AddressType:= ,
  SlaveAddress:= uiEthercatAddr,
  LcId:= strLcId,

```

```
// assign only if ESI corresponds to your device
EsiFileName:= , // strEsiFileName,
DiagTextListAdr := ADR(arDiagTextList),
Timeout:= ,
Done=> ,
Active=> ,
Error=> ,
ErrorID=> ,
ErrorIdent=> ,
DiagHistory=> );
```

Fehlerbeschreibung Spezifische Fehlercodes des Funktionsbausteins:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR	16#F0EC8000	16#00000004	Eingang "Timeout" ungültig: Wert "T#0ms" nicht erlaubt
ACCESS_ERROR	16#F0EC8020	internal	Fehler beim Datei-Öffnen, überprüfe Eingang "EsiFileName" (Dateiname inklusive Pfad)
ACCESS_ERROR	16#F0EC8021	internal	Fehler beim Datei-Lesen (intern)
ACCESS_ERROR	16#F0EC8022	internal	Fehler beim Datei Position-Setzen (intern)
ACCESS_ERROR	16#F0EC8023	internal	a.) Datei Tag <DiagMessages> nicht gefunden, überprüfe ESI-Datei ob <DiagMessages> unterstützt werden. b.) Es wird nur UTF-8 Kodierung unterstützt; falls ESI-Datei eine andere Kodierung hat, dann versuchen diese Datei in UTF-8 zu konvertieren.

Siehe auch allgemeine Fehlercodes "EtherCAT Fehlertabelle".

9.5 Zustandsmaschine (State machine)

9.5.1 IL_ECATMasterGetBusState

Kurzbeschreibung Der Funktionsbaustein IL_ECATMasterGetBusState gibt den aktuellen Buszustand des EtherCAT-Masters zurück.

Schnittstellenbeschreibung

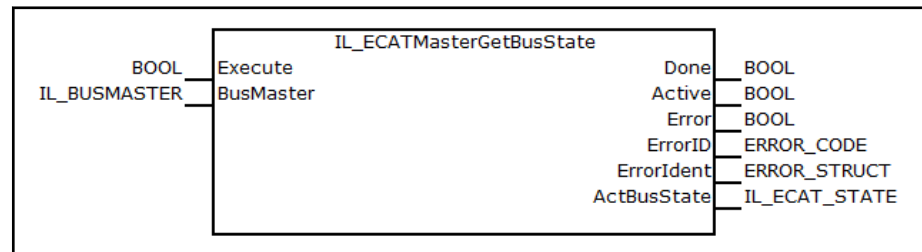


Abb. 9-21: Funktionsbaustein IL_ECATMasterGetBusState

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des EtherCAT-Masters in der Reihenfolge des Konfigurationsbaumes
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers

E/A-Typ	Name	Datentyp	Beschreibung
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	ActBusState	IL_ECATA_STATE	Aktueller Buszustand des EtherCAT-Masters

Tab. 9-40: Schnittstelle IL_ECATA_MasterGetBusState

Name	Datentyp	Min	Max	Default	Übernahme
Execute	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute

Tab. 9-41: Min-/Max- und Default-Werte der Eingänge IL_ECATA_MasterGetBusState

Funktionsbeschreibung Der Funktionsbaustein IL_ECATA_MasterGetBusState gibt den aktuellen Buszustand des EtherCAT-Masters zurück.

Implementierung in ST *Beispiel:*

```

VAR
    fbECATA_MasterGetBusState: IL_ECATA_MasterGetBusState;
    boExecute : BOOL := FALSE;
    eActBusState : IL_ECATA_STATE;
END_VAR

fbECATA_MasterGetBusState (
    Execute:= boExecute,
    BusMaster:= ,
    Done=> ,
    Active=> ,
    Error=> ,
    ErrorID=> ,
    ErrorIdent=> ,
    ActBusState=> eActBusState );

```

Fehlerbeschreibung Keine spezifischen Fehlercodes, siehe allgemeine Fehlercodes "EtherCAT Fehlertabelle".

9.5.2 IL_ECATA_MasterSetBusState

Kurzbeschreibung Mit dem Funktionsbaustein IL_ECATA_MasterSetBusState kann ein EtherCAT-Buszustand beim EtherCAT-Master angefordert werden.

Schnittstellenbeschreibung

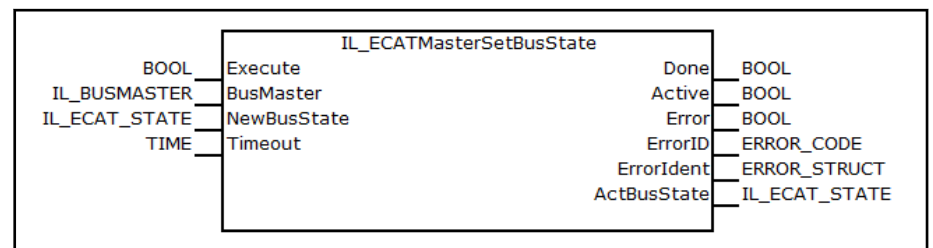


Abb. 9-22: Funktionsbaustein IL_ECATA_MasterSetBusState

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des EtherCAT-Masters in der Reihenfolge des Konfigurationsbaumes
	NewBusState	IL_ECATA_STATE	EtherCAT-Buszustand, der angefordert wird
	Timeout	TIME	Timeout-Überwachung (mit T#0s deaktiviert)

E/A-Typ	Name	Datentyp	Beschreibung
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	ActBusState	IL_ECATA_STATE	Aktueller Buszustand des EtherCAT-Masters

Tab. 9-42: Schnittstelle IL_ECATMasterSetBusState

Name	Datentyp	Min	Max	Default	Übernahme
Execute	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
NewBusState	IL_ECATA_STATE	ECATA_STATE_INIT	ECATA_STATE_OP	ECATA_STATE_UNKNOWN	Steigende Flanke an Execute
Timeout	TIME	T#0s	n.def	T#0s	Steigende Flanke an Execute

Tab. 9-43: Min-/Max- und Default-Werte der Eingänge IL_ECATMasterSetBusState

Funktionsbeschreibung

Mit dem Funktionsbaustein IL_ECATMasterSetBusState kann ein EtherCAT-Buszustand beim EtherCAT-Master angefordert werden.

Der EtherCAT Master versucht dann den Buszustand und den Zustand von allen Slaves in den angeforderten Zustand zu schalten.

Folgende EtherCAT Zustände können angefordert werden:

- Init
- PreOP
- SafeOP
- OP

Wird der angeforderte Buszustand innerhalb der Timeout-Überwachung ("Timeout" > T#0s) nicht erreicht, dann meldet der Funktionsbaustein einen Fehler, die interne Kommando-Ausführung kann aber nicht gestoppt werden!



Ist der Master in einem bestimmten Zustand (z. B. OP), aber einige Slaves sind nicht in diesem Zustand, dann kann IL_ECATMasterSetBusState auch mit diesem Zielzustand aufgerufen werden. Der Master versucht dann alle Slaves in den angeforderten Zustand zu schalten.

Hierbei ist zu beachten, dass der Ausgang "ActBusState" nur den Buszustand anzeigt. Um zu ermitteln, ob alle Slaves im Master-Zustand sind, kann der Diagnose-Funktionsbaustein IL_ECATMasterState verwendet werden.

Implementierung in ST

Beispiel:

```
VAR
  fbECATMasterSetBusState: IL_ECATMasterSetBusState;
  boExecute: BOOL := FALSE;
```

```

eNewBusState: IL_ECATCH_STATE := ECATCH_STATE_OP;
tTimeout: TIME := T#30S;
END_VAR

// Example to request a new bus state
fbECATCHMasterSetBusState(
  Execute:= boExecute,
  BusMaster:= ,
  NewBusState:= eNewBusState,
  Timeout:= tTimeout,
  Done=> ,
  Active=> ,
  Error=> ,
  ErrorID=> ,
  ErrorIdent=> ,
  ActBusState=> );

```

Fehlerbeschreibung Spezifische Fehlercodes des Funktionsbausteins:

ErrorID	Additional1	Additional2	Beschreibung
ACCESS_ERROR	16#F0ED000B	internal	Eingang "NewBusState" ungültig: ungültiger Parameter-Wert

Siehe auch allgemeine Fehlercodes "EtherCAT Fehlertabelle".

9.5.3 IL_ECATCHRemoteSlaveGetDevState

Kurzbeschreibung Der Funktionsbaustein IL_ECATCHRemoteSlaveGetDevState gibt den aktuellen EtherCAT-Zustand eines Slave-Gerätes zurück.

Schnittstellenbeschreibung

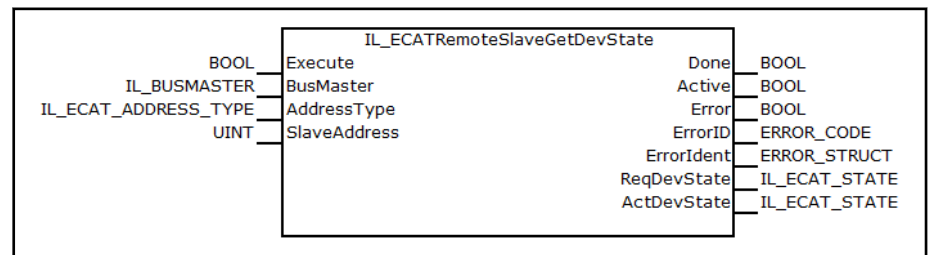


Abb. 9-23: Funktionsbaustein IL_ECATCHRemoteSlaveGetDevState

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des EtherCAT-Masters in der Reihenfolge des Konfigurationsbaumes
	AddressType	IL_ECATCH_ADDRESS_TYPE	Auswahl der Slave-Adressierung (EtherCAT-Adresse oder AutoInc-Adresse)
	SlaveAddress	UINT	EtherCAT-Adresse oder aktuelle AutoInc-Adresse am Bus (abhängig vom Eingang "AddressType")
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers

E/A-Typ	Name	Datentyp	Beschreibung
	ReqDevState	IL_ECATA_STATE	Angeforderter EtherCAT-Zustand des Slave-Gerätes
	ActDevState	IL_ECATA_STATE	Aktueller EtherCAT-Zustand des Slave-Gerätes

Tab. 9-44: Schnittstelle IL_ECATRemoteSlaveGetDevState

Name	Datentyp	Min	Max	Default	Übernahme
Execute	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
AddressType	IL_ECATA_ADDRESS_TYPE	ECAT_AUTO_INCREMENT_ADDRESS	ECAT_ETHERCAT_ADDRESS	ECAT_ETHERCAT_ADDRESS	Steigende Flanke an Execute
SlaveAddress	UINT	0	65535	0	Steigende Flanke an Execute

Tab. 9-45: Min-/Max- und Default-Werte der Eingänge IL_ECATRemoteSlaveGetDevState

Funktionsbeschreibung Der Funktionsbaustein IL_ECATRemoteSlaveGetDevState gibt den aktuellen EtherCAT-Zustand eines Slave-Gerätes zurück.

Implementierung in ST *Beispiel:*

```

VAR
    fbECATRemoteSlaveGetDevState: IL_ECATRemoteSlaveGetDevState;
    boExecute : BOOL := FALSE;
    uiEthercatAddr : UINT := 1001;
    eActDevState : IL_ECATA_STATE;
    eReqDevState : IL_ECATA_STATE;
END_VAR

fbECATRemoteSlaveGetDevState(
    Execute:= boExecute,
    BusMaster:= ,
    AddressType:= ,
    SlaveAddress:= uiEthercatAddr,
    Done=> ,
    Active=> ,
    Error=> ,
    ErrorID=> ,
    ErrorIdent=> ,
    ReqDevState=> eReqDevState,
    ActDevState=> eActDevState );

```

Fehlerbeschreibung Keine spezifischen Fehlercodes, siehe allgemeine Fehlercodes "EtherCAT Fehlertabelle".

9.5.4 IL_ECATRemoteSlaveSetDevState

Kurzbeschreibung Mit dem Funktionsbaustein IL_ECATRemoteSlaveSetDevState kann ein EtherCAT-Zustand bei einem Slave-Gerät angefordert werden.

Schnittstellenbeschreibung

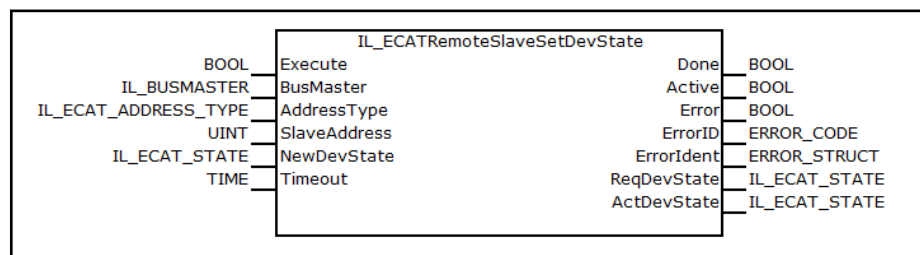


Abb. 9-24: Funktionsbaustein IL_ECATRemoteSlaveSetDevState

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Instanz des EtherCAT-Masters in der Reihenfolge des Konfigurationsbaumes
	AddressType	IL_ECATOR_ADD- RESS_TYPE	Auswahl der Slave-Adressierung (EtherCAT-Adresse oder AutoInc-Adresse)
	SlaveAddress	UINT	EtherCAT-Adresse oder aktuelle AutoInc-Adresse am Bus (abhängig vom Eingang "AddressType")
	NewDevState	IL_ECATOR_STATE	EtherCAT-Zustand, der beim Slave-Gerät angefordert wird
	Timeout	TIME	Timeout-Überwachung (mit T#0s deaktiviert)
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen, Ausgangsdaten sind ungültig
	ErrorID	ERROR_CODE	Standardisierte Grobklassifizierung des Fehlers
	ErrorIdent	ERROR_STRUCT	Bausteinspezifische Detailinformation des Fehlers
	ReqDevState	IL_ECATOR_STATE	Angeforderter EtherCAT-Zustand des Slave-Gerätes
	ActDevState	IL_ECATOR_STATE	Aktueller EtherCAT-Zustand des Slave-Gerätes

Tab. 9-46: Schnittstelle IL_ECATORRemoteSlaveSetDevState

Name	Datentyp	Min	Max	Default	Übernahme
Execute	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
AddressType	IL_ECATOR_ADD- RESS_TYPE	ECATOR_AUTO- INC_ADDRESS	ECATOR_ETHERCAT_ ADDRESS	ECATOR_ETHERCAT_ ADDRESS	Steigende Flanke an Execute
SlaveAddress	UINT	0	65535	0	Steigende Flanke an Execute
NewDevState	IL_ECATOR_STATE	ECATOR_STATE_IN IT	ECATOR_STATE_OP	ECATOR_STATE_UNKNOWN	Steigende Flanke an Execute
Timeout	TIME	T#0s	n.def	T#0s	Steigende Flanke an Execute

Tab. 9-47: Min-/Max- und Default-Werte der Eingänge IL_ECATORRemoteSlaveSetDevState

Funktionsbeschreibung

Mit dem Funktionsbaustein IL_ECATORRemoteSlaveSetDevState kann ein EtherCAT-Zustand bei einem Slave-Gerät angefordert werden.

Folgende EtherCAT-Zustände können angefordert werden:

- Init
- PreOP
- Bootstrap
- SafeOP
- OP

Hierbei gibt es folgende Randbedingungen:

- Der angeforderte Zustand in einem Slave-Gerät darf nicht höher sein als der EtherCAT-Master-Buszustand (z.B. ist der Master in PreOP, kann ein Slave-Gerät nicht in SafeOP oder OP geschaltet werden)
- Der EtherCAT-Slave-Zustand "Bootstrap" ist optional in Slave-Geräten. Der Zustand "Bootstrap" kann nur vom EtherCAT-Slave-Zustand "Init" aus erreicht werden
- Wird der angeforderte Bus-Zustand innerhalb der Timeout-Überwachung ("Timeout" > T#0s) nicht erreicht, dann meldet der Funktionsbaustein einen Fehler, die interne Kommando-Ausführung kann aber nicht gestoppt werden!

Implementierung in ST *Beispiel:*

```
VAR
    fbECATRemoteSlaveSetDevState : IL_ECATRemoteSlaveSetDevState;
    boExecute : BOOL := FALSE;
    uiEthercatAddr : UINT := 1001;
    eNewDevState : IL_ECAT_STATE := ECAT_STATE_PREOP;
    tTimeout: TIME := T#20S;
END_VAR

// Example to request a new EtherCAT state at a slave device
fbECATRemoteSlaveSetDevState(
    Execute:= boExecute,
    BusMaster:= ,
    AddressType:= ,
    SlaveAddress:= uiEthercatAddr,
    NewDevState:= eNewDevState,
    Timeout:= tTimeout,
    Done=> ,
    Active=> ,
    Error=> ,
    ErrorID=> ,
    ErrorIdent=> ,
    ReqDevState=> ,
    ActDevState=> );
```

Fehlerbeschreibung Spezifische Fehlercodes des Funktionsbausteins:

ErrorID	Additional1	Additional2	Beschreibung
ACCESS_ERROR	16#F0ED000B	Internal	Eingang "NewDevState" ungültig: ungültiger Parameterwert
ACCESS_ERROR	16#F0ED000E	Internal	Eingang "NewDevState" ungültig: ungültiger Zustandswechsel

Siehe auch allgemeine Fehlercodes "EtherCAT Fehlertabelle".

9.6 Hilfsfunktionen (Utilities)

9.6.1 IL_ECATSoeldnCoding

Kurzbeschreibung Die Funktion IL_ECATSoeldnCoding fasst die Einzelelemente Parametertyp, Parametersatz und Daten-Blocknummer zu einer Parameter-Identifikationsnummer zusammen.

Schnittstellenbeschreibung

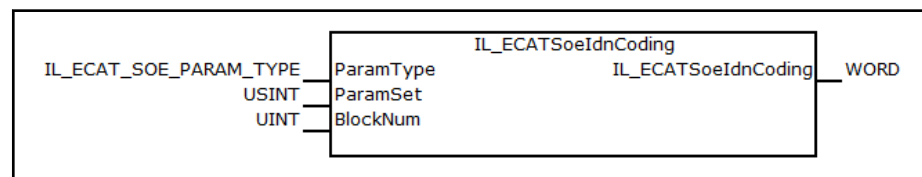


Abb. 9-25: Funktion IL_ECATSoeldnCoding

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	ParamType	IL_ECATT_SOE_PARAM_TYPE	Parametertyp (S- oder P-Parameter)
	ParamSet	USINT	Parametersatz (0..7)
	BlockNum	UINT	Daten-Blocknummer
RETURN VALUE	IL_ECATTSoeIdnCoding	WORD	Parameter-Identifikationsnummer

Funktionsbeschreibung Die Funktion IL_ECATTSoeIdnCoding fasst die Einzelelemente Parametertyp, Parametersatz und Daten-Blocknummer zu einer Parameter-Identifikationsnummer zusammen.

Implementierung in ST *Beispiel:*

```
fbSoeRead.Idn := IL_ECATTSoeIdnCoding(SOE_S_PARAM, 0, 1504); // ^= S-0-1504
```

Fehlerbeschreibung Die Funktion IL_ECATTSoeAttributeDecoding hat keine Fehlerbehandlung.

9.6.2 IL_ECATTSoeAttributeDecoding

Kurzbeschreibung Die Funktion IL_ECATTSoeAttributeDecoding dekodiert das SoE Parameter-"Attribut".

Schnittstellenbeschreibung

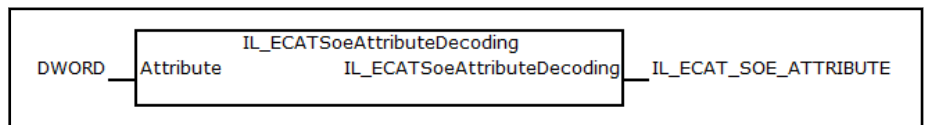


Abb. 9-26: Funktion IL_ECATTSoeAttributeDecoding

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Attribute	DWORD	SoE Parameter "Attribut" Wert
RETURN VALUE	IL_ECATTSoeAttributeDecoding	IL_ECATT_SOE_ATTRIBUTE	Parameter-Attribut als Strukturtyp

Funktionsbeschreibung Die Funktion IL_ECATTSoeAttributeDecoding dekodiert das SoE Parameter-"Attribut".

Implementierung in ST *Beispiel:*

```
VAR
    dwAttribute : DWORD; // read value
    rAttribute   : IL_ECATT_SOE_ATTRIBUTE;
END_VAR

rAttribute := IL_ECATTSoeAttributeDecoding( dwAttribute );
```

Fehlerbeschreibung Die Funktion IL_ECATTSoeAttributeDecoding hat keine Fehlerbehandlung.

9.6.3 IL_ECATTMasterStateDecoding

Kurzbeschreibung Die Funktion IL_ECATTMasterStateDecoding dekodiert den Master "State" des Funktionsbausteins IL_ECATTMasterState.

Schnittstellenbeschreibung

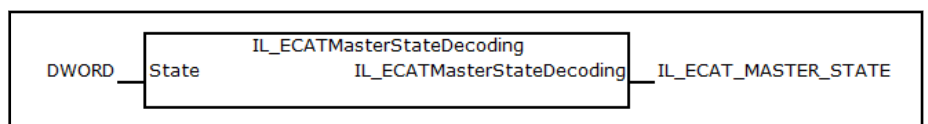


Abb. 9-27: Funktion IL_ECATTMasterStateDecoding

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	State	DWORD	Master "State" des Funktionsbausteins IL_ECATMasterState
RETURN VALUE	IL_ECATMasterStateDecoding	IL_ECAT_MASTER_STATE	Bit-Struktur für komfortablen Zugriff auf den Master "State"

Funktionsbeschreibung Die Funktion IL_ECATMasterStateDecoding dekodiert den Master "State" des Funktionsbausteins IL_ECATMasterState.

Implementierung in ST *Beispiel:*

```
VAR
    rMasterState : IL_ECAT_MASTER_STATE;
END_VAR

rMasterState := IL_ECATMasterStateDecoding( fbECATMasterState.State );
```

Fehlerbeschreibung Die Funktion IL_ECATMasterStateDecoding hat keine Fehlerbehandlung.

9.6.4 IL_ECATSlaveStateDecoding

Kurzbeschreibung Die Funktion IL_ECATSlaveStateDecoding dekodiert den Slave "State" des Funktionsbausteins IL_ECATRemoteSlaveState.

Schnittstellenbeschreibung

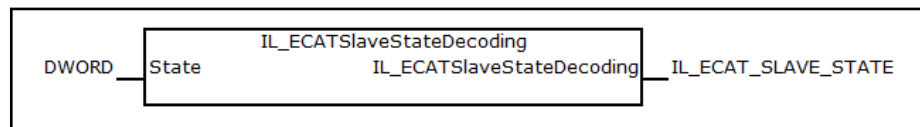


Abb. 9-28: Funktion IL_ECATSlaveStateDecoding

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	State	DWORD	Slave "State" des Funktionsbausteins IL_ECATRemoteSlaveState
RETURN VALUE	IL_ECATSlaveStateDecoding	IL_ECAT_SLAVE_STATE	Bit-Struktur für komfortablen Zugriff auf den Slave "State"

Funktionsbeschreibung Die Funktion IL_ECATSlaveStateDecoding dekodiert den Slave "State" des Funktionsbausteins IL_ECATRemoteSlaveState.

Implementierung in ST *Beispiel:*

```
VAR
    rSlaveState : IL_ECAT_SLAVE_STATE;
END_VAR

rSlaveState := IL_ECATSlaveStateDecoding( fbECATRemoteSlaveState.State );
```

Fehlerbeschreibung Die Funktion IL_ECATSlaveStateDecoding hat keine Fehlerbehandlung.

9.7 Datentypen (DataTypes)

9.7.1 IL_BUSMASTER

Kurzbeschreibung Der Aufzählungstyp IL_BUSMASTER aus der RIL_FieldbusTypes.library dient dazu, den jeweiligen EtherCAT-Master auszuwählen.

Momentan wird nur eine EtherCAT-Master-Instanz unterstützt.

Element	Wert	Beschreibung
IL_BUSMASTER_0	0	Erster EtherCAT-Master
IL_BUSMASTER_1	1	Nicht unterstützt
IL_BUSMASTER_2	2	Nicht unterstützt
IL_BUSMASTER_3	3	Nicht unterstützt
IL_BUSMASTER_4	4	Nicht unterstützt

Tab. 9-48: Elemente des Aufzählungstyps IL_BUSMASTER

9.7.2 IL_ECAT_ADDRESS_TYPE

Kurzbeschreibung Der Aufzählungstyp IL_ECAT_ADDRESS_TYPE beschreibt den EtherCAT-Adressierungstyp.

Name	Wert	Beschreibung
ECAT_AUTOINC_ADDRESS	0	Auto Inkrement Adresse
ECAT_ETHERCAT_ADDRESS	1	EtherCAT-Adresse ("Fixed Physical Address")

Tab. 9-49: Elemente des Aufzählungstyps IL_ECAT_ADDRESS_TYPE

9.7.3 IL_ECAT_BASE_DATA_TYPE

Kurzbeschreibung Der Aufzählungstyp IL_ECAT_BASE_DATA_TYPE beschreibt die Datentypen entsprechend ETG-Kodierung.

Name	Wert	Beschreibung
ECAT_DATATYPE_BOOLEAN	16#0001	BOOL
ECAT_DATATYPE_INTEGER8	16#0002	Short Integer
ECAT_DATATYPE_INTEGER16	16#0003	Integer
ECAT_DATATYPE_INTEGER32	16#0004	Double Integer
ECAT_DATATYPE_UNSIGNED8	16#0005	Unsigned Short Integer
ECAT_DATATYPE_UNSIGNED16	16#0006	Unsigned Integer
ECAT_DATATYPE_UNSIGNED32	16#0007	Unsigned Double Integer
ECAT_DATATYPE_REAL32	16#0008	Floating Point
ECAT_DATATYPE_VISIBLE_STRING	16#0009	STRING(n), 1 Byte je Zeichen
ECAT_DATATYPE_OCTET_STRING	16#000A	ARRAY(0..n) OF BYTE, 1 Byte je Zeichen (+1)
ECAT_DATATYPE_UNICODE_STRING	16#000B	ARRAY(0..n) OF WORD, 2 Byte je Zeichen (+1)
ECAT_DATATYPE_TIME_OF_DAY	16#000C	-
ECAT_DATATYPE_TIME_DIFFERENCE	16#000D	-
ECAT_DATATYPE_DOMAIN	16#000F	-
ECAT_DATATYPE_INTEGER24	16#0010	-
ECAT_DATATYPE_REAL64	16#0011	Long Float
ECAT_DATATYPE_INTEGER40	16#0012	-
ECAT_DATATYPE_INTEGER48	16#0013	-
ECAT_DATATYPE_INTEGER56	16#0014	-

Name	Wert	Beschreibung
ECAT_DATATYPE_INTEGER64	16#0015	Long Integer
ECAT_DATATYPE_UNSIGNED24	16#0016	-
ECAT_DATATYPE_UNSIGNED40	16#0018	-
ECAT_DATATYPE_UNSIGNED48	16#0019	-
ECAT_DATATYPE_UNSIGNED56	16#001A	-
ECAT_DATATYPE_UNSIGNED64	16#001B	Unsigned Long Integer
ECAT_DATATYPE_GUID	16#001D	Globally unique Identifier, 128 Bit
ECAT_DATATYPE_BYTE	16#001E	1 Octet
ECAT_DATATYPE_WORD	16#001F	2 Octet
ECAT_DATATYPE_DWORD	16#0020	4 Octet
ECAT_DATATYPE_PDO_MAPPING	16#0021	-
ECAT_DATATYPE_IDENTITY	16#0023	-
ECAT_DATATYPE_COMMAND_PAR	16#0025	-
ECAT_DATATYPE_PDO_PARAMETER	16#0027	-
ECAT_DATATYPE_ENUM	16#0028	-
ECAT_DATATYPE_SYNC_PAR	16#0029	-
ECAT_DATATYPE_RECORD	16#002A	-
ECAT_DATATYPE_BACKUP_PARAMETER	16#002B	-
ECAT_DATATYPE_MOD_DEV_PROFILE	16#002C	Modular Device Profile
ECAT_DATATYPE_BITARR8	16#002D	8 individual Bits
ECAT_DATATYPE_BITARR16	16#002E	16 individual Bits
ECAT_DATATYPE_BITARR32	16#002F	32 individual Bits
ECAT_DATATYPE_BIT1	16#0030	-
ECAT_DATATYPE_BIT2	16#0031	-
ECAT_DATATYPE_BIT3	16#0032	-
ECAT_DATATYPE_BIT4	16#0033	-
ECAT_DATATYPE_BIT5	16#0034	-
ECAT_DATATYPE_BIT6	16#0035	-
ECAT_DATATYPE_BIT7	16#0036	-
ECAT_DATATYPE_BIT8	16#0037	-

Tab. 9-50: Elemente des Aufzählungstyps IL_ECAT_BASE_DATA_TYPE

9.7.4 IL_ECAT_COE_DIAG_HISTORY_FLAGS

Kurzbeschreibung Die Struktur IL_ECAT_COE_DIAG_HISTORY_FLAGS beschreibt Einstellungen zur Funktionsweise des CoE "Diagnosis History" Objekts.

Strukturelement	Datentyp	Beschreibung
EnableEmergencySending	BIT	Bit 0.0: Senden von Emergency Nachrichten entsprechend ETG. 1000-6 [R/(W ¹)]
DisableInfoMessages	BIT	Bit 0.1: Deaktivierung von "Info" Nachrichten [R/W]
DisableWarningMessages	BIT	Bit 0.2: Deaktivierung von "Warnung" Nachrichten [R/W]
DisableErrorMessages	BIT	Bit 0.3: Deaktivierung von "Fehler" Nachrichten [R/(W ³)]
ModeSelection	BIT	Bit 0.4: Modus: 0 = "Overwrite Mode", 1 = "Acknowledge Mode" [R/(W ¹)]
OverwriteDiscardInfo	BIT	Bit 0.5: Buffer-Überlauf, Nachrichten wurden überschrieben ("Overwrite Mode") oder verworfen ("Acknowledge Mode") [R]
Res06	BIT	Bit 0.6: Reserviert
...	...	...
Res17	BIT	Bit 1.7: Reserviert

Tab. 9-51: Elemente der Struktur IL_ECAT_COE_DIAG_HISTORY_FLAGS

9.7.5 IL_ECAT_COE_DIAG_HISTORY_STRUCT

Kurzbeschreibung Die Struktur IL_ECAT_COE_DIAG_HISTORY_STRUCT beschreibt die Datenstruktur des CoE "Diagnosis History" Objekts (Index 16#10F3). Die Diagnosenachrichten (SI6: "DiagMessages") besitzen eine erweiterte Datenstruktur und entsprechen nicht den Rohdaten des Objekts (siehe [Kap. 9.4.14 "IL_ECATCoeReadDiagHistory"](#) auf Seite 300).

Strukturelement	Datentyp	Beschreibung
NumOfEntries	BYTE	SI0: Anzahl der Einträge (max. SubIndex)
MaxDiagMessages	BYTE	SI1: Max. Anzahl der Diagnosenachrichten (ab SI6)
NewestMessage	BYTE	SI2: SubIndex der neuesten Diagnosenachricht [6..255] (Default = 0)
NewestAckMessage	BYTE	SI3: [R/W] Bedeutung abhängig vom Modus ("Overwrite Mode" oder "Acknowledge Mode"), siehe unten
NewDiagMessages	BYTE	SI4: Wert 1 = neue Diagnose Nachrichten verfügbar: "Overwrite Mode": Nachrichten noch nicht gelesen "Acknowledge Mode": Nachrichten noch nicht bestätigt
Flags	IL_ECAT_COE_DIAG_HISTORY_FLAGS	SI5: [R/(W)] Einstellungen zur Funktionsweise des Objekts
DiagMessages	ARRAY[6..255] OF IL_ECAT_COE_DIAG_MSG_EXT	SI6 (SI7..255 optional): Diagnosenachrichten, max. 250 Einträge

Tab. 9-52: Elemente der Struktur IL_ECAT_COE_DIAG_HISTORY_STRUCT

NewestAckMessage (SI3) Bedeutung abhängig vom Modus:

3) Schreibzugriff ist optional / geräte-spezifisch.

Read/Write SI3	Beschreibung Overwrite Mode (SI5:Bit4 = 0)
Read = 0	Wenn der Nachrichten-Buffer überschrieben wird, setzt der Slave SI3 = 0
Write = 0	(Optional) der Slave löscht alle Nachrichten (SI2, SI3, SI4 u. SI5:Bit5 werden zurückgesetzt)
Write = 1..5	Der Slave meldet einen Fehler (z.B. SDO Abortcode 0x06090030 oder 0x06090031)
Write = 6..[SI0]	Es werden alle Nachrichten bis zu dem geschriebenen Wert quittiert
Write > [SI0]..255	Der Slave meldet einen Fehler (z.B. SDO Abortcode 0x06090030 oder 0x06090031)

Tab. 9-53: Overwrite Mode

Read/Write SI3	Beschreibung Acknowledge Mode (SI5:Bit4 = 1)
Read = 0	Es wurden bisher keine Nachrichten quittiert
Read <> 0	SubIndex der zuletzt quittierten Nachricht (6..255)
Write = 0	(Optional) alle quittierten Nachrichten werden gelöscht
Write = 1..5	Der Slave meldet einen Fehler (z.B. SDO Abortcode 0x06090030 oder 0x06090031)
Write = 6..[SI0]	Es werden alle Nachrichten bis zu dem geschriebenen Wert quittiert
Write > [SI0]..255	Der Slave meldet einen Fehler (z.B. SDO Abortcode 0x06090030 oder 0x06090031)

Tab. 9-54: Acknowledge Mode

9.7.6 IL_ECACOE_DIAG_MSG_EXT

Kurzbeschreibung Die Struktur IL_ECACOE_DIAG_MSG_EXT beschreibt eine erweiterte Datenstruktur für eine Diagnosenachricht des CoE "Diagnosis History" Objekts (siehe [Kap. 9.4.14 "IL_ECACoeReadDiagHistory" auf Seite 300](#)).

Strukturelement	Datentyp	Beschreibung
DiagCode_LWord	WORD	LWord (Bit 0-15) HWord (Bit 16-31)
DiagCode_HWord	WORD	0x0000 - 0xDFFF → nicht verwendet 0xE000 - 0xE7FF → herstellerspezifisch 0xE800 → Emergency ErrorCode nach DS301 oder DS4xxx 0xE801 - 0xEDFF → reserviert 0xEE00 - 0xEFFF → Profil-spezifisch 0xF000 - 0xFFFF → nicht verwendet
DiagType	IL_ECACOE_DIAG_TYPE	Diagnose-Typ ("Flags" Bit 0-3)
Acknowledged	BOOL	TRUE = Nachricht ist bestätigt
TimeStampNs	ULINT	Zeitstempel in ns ; DC oder lokaler Zeitstempel, wenn DC 64 Bit nicht unterstützt wird (0 = kein Zeitstempel)
IsLocalTimeStamp	BOOL	Zeitstempel ist lokal vom Slave-Gerät ("Flags" Bit 4)

Strukturelement	Datentyp	Beschreibung
TextId	WORD	Text ID als Referenz zum Diagnosetext in der ESI-Datei (0 = keine ID)
NumOfParams	BYTE	Anzahl von Parametern in dieser Diagnosenachricht ("Flags" Bit 8-15)
Data	ARRAY[0..127] OF BYTE	[Optional] Daten von Parameter 1 sowie ggf. weitere Parameter
MessageText	STRING(127)	[Optional] "MessageText" der ESI (wenn eine ESI-Datei angegeben ist)

Tab. 9-55: Elemente der Struktur IL_ECAT_COE_DIAG_MSG_EXT

9.7.7 IL_ECAT_COE_DIAG_MSG_RAW

Kurzbeschreibung Die Struktur IL_ECAT_COE_DIAG_MSG_RAW beschreibt die Rohdaten einer Diagnosenachricht des CoE "Diagnosis History" Objekts.

Strukturelement	Datentyp	Beschreibung
DiagCode_LWord	WORD	LWord (Bit 0-15) HWord (Bit 16-31) 0x0000 - 0xDFFF → nicht verwendet 0xE000 - 0xE7FF → herstellerspezifisch 0xE800 → Emergency ErrorCode nach DS301 oder DS4xxx 0xE801 - 0xEDFF → reserviert 0xEE00 - 0xEFFF → Profil-spezifisch 0xF000 - 0xFFFF → nicht verwendet
DiagCode_HWord	WORD	
Flags_LByte	BYTE	
Flags_HByte	BYTE	
TextId	WORD	
TimeStamp	ULINT	Zeitstempel in ns: DC oder lokaler Zeitstempel, wenn DC 64 Bit nicht unterstützt wird (0 = kein Zeitstempel)
Data	ARRAY[0..127] OF BYTE	(Optional) Daten von Parameter 1 sowie ggf. weitere Parameter

Tab. 9-56: Elemente der Struktur IL_ECAT_COE_DIAG_MSG_RAW

9.7.8 IL_ECAT_COE_DIAG_TYPE

Kurzbeschreibung Der Aufzählungstyp IL_ECAT_COE_DIAG_TYPE beschreibt den Diagnosentyp einer CoE-Diagnosenachricht.

Name	Wert	Beschreibung
COE_DIAG_INFO	0	Diagnose "Info" Nachricht
COE_DIAG_WARNING	1	Diagnose "Warnung" Nachricht
COE_DIAG_ERROR	2	Diagnose "Fehler" Nachricht

Tab. 9-57: Elemente des Aufzählungstyps IL_ECAT_COE_DIAG_TYPE

9.7.9 IL_ECAT_DL_STATUS

Kurzbeschreibung Die Struktur IL_ECAT_DL_STATUS beschreibt den Data Link Status eines EtherCAT-Slaves (ESC-Register 0x0110:0x0111).

Strukturelement	Datentyp	Beschreibung
PdiOperationalEepromLoaded	BIT	Bit 0.0: PDI funktionsfähig, EEPROM richtig geladen
PdiWatchdogStatus	BIT	Bit 0.1: Watchdog geladen/läuft
EnhancedLinkDetection	BIT	Bit 0.2: Erweiterte Link-Erkennung für mindestens einen Port
Reserved03	BIT	Bit 0.3: Reserviert
LinkPort0	BIT	Bit 0.4: Link vorhanden an Port 0
LinkPort1	BIT	Bit 0.5: Link vorhanden an Port 1
LinkPort2	BIT	Bit 0.6: Link vorhanden an Port 2
LinkPort3	BIT	Bit 0.7: Link vorhanden an Port 3
LoopClosedPort0	BIT	Bit 1.0: Port 0 geschlossen
CommunicationPort0	BIT	Bit 1.1: Kommunikation an Port 0
LoopClosedPort1	BIT	Bit 1.2: Port 1 geschlossen
CommunicationPort1	BIT	Bit 1.3: Kommunikation an Port 1
LoopClosedPort2	BIT	Bit 1.4: Port 2 geschlossen
CommunicationPort2	BIT	Bit 1.5: Kommunikation an Port 2
LoopClosedPort3	BIT	Bit 1.6: Port 3 geschlossen
CommunicationPort3	BIT	Bit 1.7: Kommunikation an Port 3

Tab. 9-58: Elemente der Struktur IL_ECAT_DL_STATUS

9.7.10 IL_ECAT_ESC_FEATURES

Kurzbeschreibung Die Struktur IL_ECAT_ESC_FEATURES beschreibt die Eigenschaften eines EtherCAT-Slave-Controllers (ESC-Register 0x0008:0x0009).

Strukturelement	Datentyp	Beschreibung
FmmuOperationByte	BIT	Bit 0.0: Fieldbus Memory Management Unit (FALSE = Bit orientiert, TRUE = Byte orientiert)
Reserved01	BIT	Bit 0.1: Reserviert
DcSupport	BIT	Bit 0.2: Distributed Clocks wird unterstützt
Dc64Support	BIT	Bit 0.3: Distributed Clocks 64 Bit wird unterstützt
LowJitterEbus	BIT	Bit 0.4: Jitter minimiert
EnhancedLinkDetectionEbus	BIT	Bit 0.5: Erweiterte Link-Erkennung EBUS
EnhancedLinkDetectionMii	BIT	Bit 0.6: Erweiterte Link-Erkennung MII
SeparateHandlingFcsErrors	BIT	Bit 0.7: FCS-Fehler werden einzeln ausgewertet
EnhancedDcSyncActivation	BIT	Bit 1.0: Erweiterte DC Sync Aktivierung
LrwCommandNotSupported	BIT	Bit 1.1: Kommando LRW wird nicht unterstützt
RwCommandNotSupported	BIT	Bit 1.2: Kommandos BRW/APRW/FPRW werden nicht unterstützt

Strukturelement	Datentyp	Beschreibung
FixedFmmuSyncManConfig	BIT	Bit 1.3: Feste FMMU/SyncManager Konfiguration
Reserved14	BIT	Bit 1.4: Reserviert
Reserved15	BIT	Bit 1.5: Reserviert
Reserved16	BIT	Bit 1.6: Reserviert
Reserved17	BIT	Bit 1.7: Reserviert

Tab. 9-59: Elemente der Struktur IL_ECATCHES_FEATURES

9.7.11 IL_ECATCHES_MASTER_BUS_STATISTIC

Kurzbeschreibung Die Struktur IL_ECATCHES_MASTER_BUS_STATISTIC beschreibt die statistischen Diagnoseinformationen zur Buskommunikation des EtherCAT-Masters.

Strukturelement	Datentyp	Beschreibung
Res1	WORD	SI1: Reserviert
ConfigChecksum	DWORD	SI2: Konfigurationschecksumme (CRC32)
NumOnlineSlaves	UDINT	SI3: Anzahl verbundener Slaves
NumOnlineDcSlaves	UDINT	SI4: Anzahl verbundener Slaves, die DC unterstützen
NumConfigSlaves	UDINT	SI5: Anzahl konfigurierter Slaves
NumConfigMbxSlaves	UDINT	SI6: Anzahl konfigurierter Slaves, die Mailbox unterstützen
CntTxFrames	UDINT	SI7: Zähler gesendete Frames
CntRxFrames	UDINT	SI8: Zähler empfangene Frames
CntLostFrames	UDINT	SI9: Zähler verlorene Frames
CntCyclicFrames	UDINT	SI10: Zähler zyklische Frames
CntCyclicDatagrams	UDINT	SI11: Zähler zyklische Datagramme
CntAcyclicFrames	UDINT	SI12: Zähler azyklische Frames
CntAcyclicDatagrams	UDINT	SI13: Zähler azyklische Datagramme
ClearCounters	DWORD	SI14: [R/W] Zurücksetzen der Frame-Zähler und Datagramm-Zähler (1 = löscht alle Zähler)

Tab. 9-60: Elemente der Struktur IL_ECATCHES_MASTER_BUS_STATISTIC

9.7.12 IL_ECATCHES_MASTER_STATE

Kurzbeschreibung Die Struktur IL_ECATCHES_MASTER_STATE beschreibt eine Zusammenfassung von Diagnoseinformationen des EtherCAT-Masters.

Strukturelement	Datentyp	Beschreibung
NoEthernetLink	BIT	Bit 0.0: Kein Master Ethernet-Link
NotInStateOp	BIT	Bit 0.1: Master ist nicht im Zustand "Operational"
NotInReqState	BIT	Bit 0.2: Master ist nicht im angeforderten Zustand
Res03	BIT	Bit 0.3: Reserviert
StateInit	BIT	Bit 0.4: Master ist im Zustand "Init"

Strukturelement	Datentyp	Beschreibung
StatePreOp	BIT	Bit 0.5: Master ist im Zustand "Pre-Operational"
StateSafeOp	BIT	Bit 0.6: Master ist im Zustand "Safe-Operational"
Res07	BIT	Bit 0.7: Reserviert
TopologyNotOk	BIT	Bit 1.0: Topologie ist nicht OK: Die konfigurierten Slaves stimmen nicht mit den Online-Slaves überein, z. B. es fehlen Slaves oder die Konfiguration der Topologie stimmt nicht mit der realen Topologie am Bus überein
SlavesNotInMasterState	BIT	Bit 1.1: Ein oder mehrere Slaves sind nicht im Master-Zustand
SlavesError	BIT	Bit 1.2: Ein oder mehrere Slaves melden einen Fehler
Res13	BIT	Bit 1.3: Reserviert
DcNotInSync	BIT	Bit 1.4: DC Synchronisation ist nicht innerhalb der konfigurierten Grenzwerte (nur wenn DC aktiviert ist)
Res15	BIT	Bit 1.5: Reserviert
Res16	BIT	Bit 1.6: Reserviert
Res17	BIT	Bit 1.7: Reserviert
Res20	BIT	Bit 2.0: Reserviert
...		...
Res37		Bit 3.7: Reserviert

Tab. 9-61: Elemente der Struktur IL_ECAT_MASTER_STATE

9.7.13 IL_ECAT_MBX_CONFIG_PROTOCOLS

Kurzbeschreibung Die Struktur IL_ECAT_MBX_CONFIG_PROTOCOLS beschreibt die unterstützten Mailboxprotokolle eines konfigurierten Slave-Gerätes.

Strukturelement	Datentyp	Beschreibung
MbxProtocolAoE	BIT	Bit 0.0: ADS over EtherCAT (Automation Device Specification)
MbxProtocolEoE	BIT	Bit 0.1: Ethernet over EtherCAT
MbxProtocolCoE	BIT	Bit 0.2: CAN application protocol over EtherCAT
MbxProtocolFoE	BIT	Bit 0.3: File access over EtherCAT
MbxProtocolSoE	BIT	Bit 0.4: Servo drive profile over EtherCAT
MbxProtocolVoE	BIT	Bit 0.5: Vendor specific profile over EtherCAT
Reserved06	BIT	Bit 0.6: Reserviert
...	...	...
Reserved37	BIT	Bit 3.7: Reserviert

Tab. 9-62: Elemente der Struktur IL_ECAT_MBX_CONFIG_PROTOCOLS

9.7.14 IL_ECAT_MBX_ONLINE_PROTOCOLS

Kurzbeschreibung Die Struktur IL_ECAT_MBX_ONLINE_PROTOCOLS beschreibt die unterstützten Mailboxprotokolle eines am Bus verbundenen Slave-Gerätes.

Strukturelement	Datentyp	Beschreibung
MbxProtocolAoE	BIT	Bit 0.0: ADS over EtherCAT (Automation Device Specification)
MbxProtocolEoE	BIT	Bit 0.1: Ethernet over EtherCAT
MbxProtocolCoE	BIT	Bit 0.2: CAN application protocol over EtherCAT
MbxProtocolFoE	BIT	Bit 0.3: File access over EtherCAT
MbxProtocolSoE	BIT	Bit 0.4: Servo drive profile over EtherCAT
MbxProtocolVoE	BIT	Bit 0.5: Vendor specific profile over EtherCAT
Reserved06	BIT	Bit 0.6: Reserviert
...	...	...
Reserved17	BIT	Bit 1.7: Reserviert

Tab. 9-63: Elemente der Struktur IL_ECATCHER_MBX_ONLINE_PROTOCOLS

9.7.15 IL_ECATCHER_MBX_SIZE_INFO

Kurzbeschreibung Die Struktur IL_ECATCHER_MBX_SIZE_INFO beschreibt die Mailboxgröße eines Slave-Gerätes.

Strukturelement	Datentyp	Beschreibung
SizeIn	UDINT	Mailbox Eingangsgröße (Slave Empfangen)
SizeOut	UDINT	Mailbox Ausgangsgröße (Slave Senden)

Tab. 9-64: Elemente der Struktur IL_ECATCHER_MBX_SIZE_INFO

9.7.16 IL_ECATCHER_MEMORY_INFO

Kurzbeschreibung Die Struktur IL_ECATCHER_MEMORY_INFO beschreibt einen Speicherbereich.

Strukturelement	Datentyp	Beschreibung
Offset	UDINT	Speicheroffset
Size	UDINT	Speichergröße

Tab. 9-65: Elemente der Struktur IL_ECATCHER_MEMORY_INFO

9.7.17 IL_ECATCHER_PORT_ERROR_COUNTER

Kurzbeschreibung Die Struktur IL_ECATCHER_PORT_ERROR_COUNTER beschreibt die Fehlerzähler eines Ports.

Strukturelement	Datentyp	Beschreibung
InvalidFrameCounter	BYTE	Fehlerzähler ungültiger Frame empfangen (Register 0x0300/0x0302/0x0304/0x0306)
RxErrorCounter	BYTE	Fehlerzähler ungültiger Frame empfangen (MII/EBUS) (Register 0x0301/0x0303/0x0305/0x0307)
FwdRxErrorCounter	BYTE	Fehlerzähler ungültiger Frame weitergeleitet (Register 0x0308:0x030B)
LostLinkCounter	BYTE	Fehlerzähler Linkverlust (Register 0x0310:0x0313), nur wenn Loop-Erkennung "Auto" und der Port "offen" ist.

Tab. 9-66: Elemente der Struktur IL_ECATCHER_PORT_ERROR_COUNTER

9.7.18 IL_ECATCHPORT_STATE

Kurzbeschreibung Die Struktur IL_ECATCHPORT_STATE beschreibt den EtherCAT-Port-Status eines Slave-Gerätes.

Im Vergleich zum DL-Status wird hier nur der Status der physikalischen Ports dargestellt.

Strukturelement	Datentyp	Beschreibung
ConnectionPort0	BIT	Bit 0.0: Verbindung vorhanden an Port 0
ConnectionPort1	BIT	Bit 0.1: Verbindung vorhanden an Port 1
ConnectionPort2	BIT	Bit 0.2: Verbindung vorhanden an Port 2
ConnectionPort3	BIT	Bit 0.3: Verbindung vorhanden an Port 3
LinkPort0	BIT	Bit 0.4: Link vorhanden an Port 0
LinkPort1	BIT	Bit 0.5: Link vorhanden an Port 1
LinkPort2	BIT	Bit 0.6: Link vorhanden an Port 2
LinkPort3	BIT	Bit 0.7: Link vorhanden an Port 3
LoopClosedPort0	BIT	Bit 1.0: Port 0 geschlossen (nur physikalische Ports)
LoopClosedPort1	BIT	Bit 1.1: Port 1 geschlossen (nur physikalische Ports)
LoopClosedPort2	BIT	Bit 1.2: Port 2 geschlossen (nur physikalische Ports)
LoopClosedPort3	BIT	Bit 1.3: Port 3 geschlossen (nur physikalische Ports)
CommunicationPort0	BIT	Bit 1.4: Kommunikation an Port 0
CommunicationPort1	BIT	Bit 1.5: Kommunikation an Port 1
CommunicationPort2	BIT	Bit 1.6: Kommunikation an Port 2
CommunicationPort3	BIT	Bit 1.7: Kommunikation an Port 3

Tab. 9-67: Elemente der Struktur IL_ECATCHPORT_STATE

Das Diagnosebit "Connection(PortX)" ist eine logische Verknüpfung der anderen Statusbits:

Communication	LoopClosed	LinkStatus	=> Connection	Hinweis
0	0	0	0	z.B. Port nicht konfiguriert
0	0	1	1	-
0	1	0	0	Standard "Loop closed"
0	1	1	0	evtl. "alte" Geräte
1	0	0	1	evtl. "alte" Geräte
1	0	1	1	Standard "Loop Open"
1	1	0	0	-
1	1	1	0	z. B. EK1122 "closed Port"

9.7.19 IL_ECATCHSLAVE_CONFIG_INFO

Kurzbeschreibung Die Struktur IL_ECATCHSLAVE_CONFIG_INFO beschreibt die Informationen eines im EtherCAT-Master konfigurierten Slave-Gerätes.

Strukturelement	Datentyp	Beschreibung
AutoIncAddr	UINT	Autoinkrementadresse
EthercatAddr	UINT	EtherCAT-Adresse ("Fixed Physical Address")
IdentifyAdo	UINT	Adress-Offset (ADO) für Identifizierungskommando
IdentifyValue	UINT	Identifizierungswert
SlaveHandle	DWORD	Interne SlaveID (interne Verwendung)
HcGroupIdx	DWORD	Index der Hot Connect Group (0 = mandatory)
PreviousEthercatAddr	UINT	EtherCAT-Adresse des vorherigen Slaves (in der Topologie)
PreviousPort	UINT	Port des vorherigen Slaves (in der Topologie)
SlaveIdentity	IL_ECOT_SLAVE_IDENTITY_INFO	Slave Identitätsinformation
SlaveName	STRING(127)	Konfigurierter Geräteiname
MbxProtocols	IL_ECOT_MBX_CONFIG_PROTOCOLS	Unterstützte Mailboxprotokolle
MbxStandard	IL_ECOT_MBX_SIZE_INFO	Mailbox Standardgröße
MbxBootstrap	IL_ECOT_MBX_SIZE_INFO	Mailbox Bootstrapgröße
ProcessDataIn	ARRAY[0..3] OF IL_ECOT_MEMORY_INFO	Prozessdatenbereich Eingänge
ProcessDataOut	ARRAY[0..3] OF IL_ECOT_MEMORY_INFO	Prozessdatenbereich Ausgänge
NumProcessVarsIn	UINT	Anzahl Eingangsprozessdaten Variablen
NumProcessVarsOut	UINT	Anzahl Ausgangsprozessdaten Variablen
PortDescriptor	BYTE	Port-Deskriptor (0b10 = EBUS, 0b11 = MII/...) (Port0^Bit0:1, Port1^Bit2:3, usw.)
Reserved01	ARRAY[0..2] OF BYTE	Reserviert
WkcStateDiagOffsIn	ARRAY[0..3] OF UINT	Intern (interne Verwendung)
WkcStateDiagOffsOut	ARRAY[0..3] OF UINT	Intern (interne Verwendung)
Reserved02	ARRAY[0..2] OF UDINT	Reserviert
IsPresent	BOOL	Slave ist aktuell am Bus verbunden (online)
IsHcGroupPresent	BOOL	Hot Connect Group des Slaves ist aktuell am Bus verbunden
DcSupport	BOOL	Slave unterstützt DC (Distributed Clocks)

Tab. 9-68: Elemente der Struktur IL_ECOT_SLAVE_CONFIG_INFO

9.7.20 IL_ECOT_SLAVE_ERROR_COUNTER

Kurzbeschreibung Die Struktur IL_ECOT_SLAVE_ERROR_COUNTER beschreibt die Fehlerzähler eines Slave-Gerätes.

Strukturelement	Datentyp	Beschreibung
ProcUnitErrorCounter	BYTE	Fehlerzähler "Processing Unit" (z.B. FCS) (Register 0x030C)
PdiErrorCounter	BYTE	Fehlerzähler "Process Data Interface" (PDI) (Register 0x030D)
PortErrorCounter	ARRAY[0..3] OF IL_ECAT_PORT_ERROR_COUNTER	Fehlerzähler je Port

Tab. 9-69: Elemente der Struktur IL_ECAT_SLAVE_ERROR_COUNTER

9.7.21 IL_ECAT_SLAVE_IDENTITY_INFO

Kurzbeschreibung Die Struktur IL_ECAT_SLAVE_IDENTITY_INFO beschreibt die Identitätsdaten eines EtherCAT-Slave-Gerätes.

Strukturelement	Datentyp	Beschreibung
VendorId	UDINT	Herstellerkennung (z.B. Rexroth = 16#24)
ProductCode	UDINT	Produktcode (herstellerspezifisch)
RevisionNumber	UDINT	Revisionsnummer
SerialNumber	UDINT	Seriennummer

Tab. 9-70: Elemente der Struktur IL_ECAT_SLAVE_IDENTITY_INFO

9.7.22 IL_ECAT_SLAVE_ONLINE_INFO

Kurzbeschreibung Die Struktur IL_ECAT_SLAVE_ONLINE_INFO beschreibt die "Online"-Informationen eines Slave-Gerätes, das am EtherCAT-Master angeschlossen ist.

Strukturelement	Datentyp	Beschreibung
AutoIncAddr	UINT	Autoinkrementadresse
EthercatAddr	UINT	EtherCAT-Adresse ("Fixed Physical Address") (ESC Register 0x0010)
StationAlias	UINT	Station Alias ("Second Slave Address") (ESC Register 0x0012)
IdentifyValue	UINT	Identifizierungswert (nur wenn ein ADO konfiguriert ist, siehe IL_ECAT_SLAVE_CONFIG_INFO → IdentifyAdo)
SlaveHandle	DWORD	Interne SlaveID (interne Verwendung)
PortSlaveHandles	ARRAY[0..3] OF DWORD	Port interne SlaveIDs (interne Verwendung)
SlaveIdentity	IL_ECAT_SLAVE_IDENTITY_INFO	Slave Identitätsinformation (EEPROM Offset 0x0008)
EscType	BYTE	EtherCAT Slave Controller Typ (ESC Register 0x0000)
EscRevision	BYTE	EtherCAT Slave Controller Revisionsnummer (ESC Register 0x0001)
EscBuild	WORD	EtherCAT Slave Controller Buildnummer (ESC Register 0x0002)

Strukturelement	Datentyp	Beschreibung
EscFeatures	IL_ECAT_ESC_FEATURES	EtherCAT Slave Controller unterstützte Features (ESC Register 0x0008)
PortDescriptor	BYTE	Port-Deskriptor (ESC Register 0x0007) (0b10 = EBUS, 0b11 = MII/...) (Port0^Bit0:1, Port1^Bit2:3, usw.)
Reserved01	BYTE	Reserviert
AlStatus	WORD	Application Layer Status (ESC Register 0x0130)
AlStatusCode	WORD	Application Layer Status Code (ESC Register 0x0134) (wird bei einer Zustandsänderung des Slaves im Master zurückgesetzt)
MbxProtocols	IL_ECAT_MBX_ONLINE_PROTOCOLS	Unterstützte Mailboxprotokolle (EEPROM Offset 0x001C); dieser Wert hier ist "0", wenn der Slave im Master konfiguriert ist, siehe IL_ECAT_SLAVE_CONFIG_INFO.
DlStatus	IL_ECAT_DL_STATUS	Data Link Status (ESC Register 0x0110)
PortState	IL_ECAT_PORT_STATE	Port-Status
PreviousPort	UINT	Port des vorherigen Slaves (in der Topologie)
SystemTimeDifference	UDINT	Systemzeit-Differenz in ns (ESC Register 0x092C) (dieser Wert ist nur gültig, wenn DC konfiguriert ist)
SlaveDelay	UDINT	Zeitverzögerung "hinter" dem Slave in ns (dieser Wert ist nur gültig, wenn DC konfiguriert ist)
PropagationDelay	UDINT	Übertragungsverzögerung in ns (ESC Register 0x0928) (dieser Wert ist nur gültig, wenn DC konfiguriert ist)
Reserved02	ARRAY[0..3] OF DWORD	Reserviert
DcSupport	BOOL	Slave unterstützt DC (Distributed Clocks)
Dc64Support	BOOL	Slave unterstützt 64 Bit DC (Distributed Clocks)
IsRefClock	BOOL	Slave ist Referenzuhr
LineCrossed	BOOL	EtherCAT-Verkabelung des Slaves ist fehlerhaft (IN-OUT Port vertauscht)

Tab. 9-71: Elemente der Struktur IL_ECAT_SLAVE_ONLINE_INFO

9.7.23 IL_ECAT_SLAVE_STATE

Kurzbeschreibung Die Struktur IL_ECAT_SLAVE_STATE beschreibt Diagnose-Statusinformationen eines Slave-Gerätes.

Strukturelement	Datentyp	Beschreibung
NotOnline	BIT	Bit 0.0: Slave ist nicht mit dem Bus verbunden
NotInStateOp	BIT	Bit 0.1: Slave ist nicht im Zustand "Operational"
Error	BIT	Bit 0.2: Slave meldet Fehler, siehe "AL Status Code"
Res03	BIT	Bit 0.3: Reserviert
StateInit	BIT	Bit 0.4: Slave ist im Zustand "Init"

Strukturelement	Datentyp	Beschreibung
StatePreOp	BIT	Bit 0.5: Slave ist im Zustand "Pre-Operational"
StateSafeOp	BIT	Bit 0.6: Slave ist im Zustand "Safe-Operational"
Res07	BIT	Bit 0.7: Reserviert
Res10	BIT	Bit 1.0: Reserviert
...	...	...
Res37	BIT	Bit 3.7: Reserviert

Tab. 9-72: Elemente der Struktur IL_ECAT_SLAVE_STATE

9.7.24 IL_ECAT_SOE_ATTRIBUTE

Kurzbeschreibung Die Struktur IL_ECAT_SOE_ATTRIBUTE beschreibt das SoE-Parameterattribut (siehe [Kap. 9.6.2 "IL_ECATSoeAttributeDecoding" auf Seite 311](#)).

Strukturelement	Datentyp	Beschreibung
Res31	BOOL	Bit 31: Reserviert
WriteProtectedCp4	BOOL	Bit 30: Parameter ist schreibgeschützt im Zustand OP (CP4)
WriteProtectedCp3	BOOL	Bit 29: Parameter ist schreibgeschützt im Zustand SafeOP (CP3)
WriteProtectedCp2	BOOL	Bit 28: Parameter ist schreibgeschützt im Zustand PreOP (CP2)
DecPoints	USINT	Bit 24-27: Anzahl Nachkommastellen
Res23	BOOL	Bit 23: Reserviert
DataType	IL_ECAT_SOE_DATA_TYPE	Bit 20-22: Datentyp und Anzeigeformat
IsCommand	BOOL	Bit 19: Parameter ist ein Kommando
DataLength	IL_ECAT_SOE_DATA_LENGTH	Bit 16-18: Datenlänge
ConversionFactor	UINT	Bit 0-15: Faktor für Darstellung

Tab. 9-73: Elemente der Struktur IL_ECAT_SOE_ATTRIBUTE

9.7.25 IL_ECAT_SOE_DATA_LENGTH

Kurzbeschreibung Der Aufzählungstyp IL_ECAT_SOE_DATA_LENGTH unterstützt die Auswertung des SoE-Parameterattributs bzgl. Datenlänge und Unterscheidung Listenparameter (VAR_x_BYTE) oder Einzelparameter (FIX_x_BYTE) (siehe [Kap. 9.6.2 "IL_ECATSoeAttributeDecoding" auf Seite 311](#)).

Name	Wert	Beschreibung
SOE_DATALENGTH_RES0	0	Reserviert
SOE_DATALENGTH_FIX_2_BYTE	1	Datenlänge ist 2 Byte
SOE_DATALENGTH_FIX_4_BYTE	2	Datenlänge ist 4 Byte
SOE_DATALENGTH_FIX_8_BYTE	3	Datenlänge ist 8 Byte
SOE_DATALENGTH_VAR_1_BYTE	4	Datenlänge ist variabel mit 1 Byte Daten
SOE_DATALENGTH_VAR_2_BYTE	5	Datenlänge ist variabel mit 2 Byte Daten

Name	Wert	Beschreibung
SOE_DATALENGTH_VAR_4_BYTE	6	Datenlänge ist variabel mit 4 Byte Daten
SOE_DATALENGTH_VAR_8_BYTE	7	Datenlänge ist variabel mit 8 Byte Daten

Tab. 9-74: Elemente des Aufzählungstyps IL_ECAT_SOE_DATA_LENGTH

9.7.26 IL_ECAT_SOE_DATA_STATUS

Kurzbeschreibung Der Aufzählungstyp IL_ECAT_SOE_DATA_STATUS beschreibt den Datenstatus eines SoE-Parameters (z.B. Bit 0-3 werden für den Status einer Kommandoausführung verwendet).

Name	Wert	Beschreibung
SOE_DATA_CMD_NOT_SET	0	Kommando ist nicht aktiviert oder wurde abgebrochen
SOE_DATA_CMD_SET	1	Kommando wurde aktiviert
SOE_DATA_CMD_END_OK	3	Kommandoausführung wurde erfolgreich abgeschlossen
SOE_DATA_CMD_INTERRUPTED	5	Kommandoausführung wurde unterbrochen
SOE_DATA_CMD_RUNNING	7	Kommando ist aktiviert, aber noch nicht abgeschlossen
SOE_DATA_CMD_ERROR	15	Fehler: Kommandoausführung nicht möglich
SOE_DATA_DATUM_INVALID	256	(Bit 8) Daten nicht gültig

Tab. 9-75: Elemente des Aufzählungstyps IL_ECAT_SOE_DATA_STATUS

9.7.27 IL_ECAT_SOE_DATA_TYPE

Kurzbeschreibung Der Aufzählungstyp IL_ECAT_SOE_DATA_TYPE unterstützt die Auswertung des SoE-Parameterattributs bzgl. Datentyp und Anzeigeformat (siehe [Kap. 9.6.2 "IL_ECATSoeAttributeDecoding" auf Seite 311](#)).

Name	Wert	Beschreibung
SOE_DATATYPE_BINARY	0	Binärzahl (Darstellung binär)
SOE_DATATYPE_UDEC	1	Vorzeichenlose Ganzzahl (Darstellung Dezimal)
SOE_DATATYPE_DEC	2	Ganzzahl mit Vorzeichen (Darstellung Dezimal)
SOE_DATATYPE_HEX	3	Vorzeichenlose Ganzzahl (Darstellung Hexadezimal)
SOE_DATATYPE_TEXT	4	Erweiterter Zeichensatz (Darstellung Text)
SOE_DATATYPE_IDN	5	Vorzeichenlose Ganzzahl (Darstellung IDN, z.B. "S-0-0000")
SOE_DATATYPE_FLOAT	6	Gleitkommazahl (Darstellung Dezimal mit Exponent)
SOE_DATATYPE_RES7	7	Reserviert

Tab. 9-76: Elemente des Aufzählungstyps IL_ECAT_SOE_DATA_TYPE

9.7.28 IL_ECAT_SOE_ELEMENT

Kurzbeschreibung Der Aufzählungstyp IL_ECAT_SOE_ELEMENT beschreibt die Elemente eines SoE-Parameters.

Name	Wert	Beschreibung
SOE_ELEMENT_STATUS	16#01	Servicekanal Datenstatus
SOE_ELEMENT_NAME	16#02	Name (max. 64 Bytes) (optional)

Name	Wert	Beschreibung
SOE_ELEMENT_ATTRIBUTE	16#04	Attribut
SOE_ELEMENT_UNIT	16#08	Einheit (optional)
SOE_ELEMENT_MIN	16#10	Minimalwert (optional)
SOE_ELEMENT_MAX	16#20	Maximalwert (optional)
SOE_ELEMENT_OPDATA	16#40	Betriebsdaten
SOE_ELEMENT_DEFAULT	16#80	Defaultwert (optional)

Tab. 9-77: Elemente des Aufzählungstyps IL_ECAT_SOE_ELEMENT

9.7.29 IL_ECAT_SOE_PARAM_TYPE

Kurzbeschreibung Der Aufzählungstyp IL_ECAT_SOE_PARAM_TYPE unterscheidet zwischen SoE-Standardparametern und produktspezifischen Parametern.

Name	Wert	Beschreibung
SOE_S_PARAM	0	Standard IDN ("S-Parameter")
SOE_P_PARAM	1	Produktspezifische IDN ("P-Parameter")

Tab. 9-78: Elemente des Aufzählungstyps IL_ECAT_SOE_PARAM_TYPE

9.7.30 IL_ECAT_STATE

Kurzbeschreibung Der Aufzählungstyp IL_ECAT_STATE beschreibt den EtherCAT-Zustand.

Name	Wert	Beschreibung
ECAT_STATE_UNKNOWN	0	EtherCAT Zustand "unbekannt"
ECAT_STATE_INIT	1	EtherCAT Zustand "Init"
ECAT_STATE_PREOP	2	EtherCAT Zustand "Pre-Operational"
ECAT_STATE_BOOTSTRAP	3	EtherCAT Zustand "Bootstrap" (nur bei Slaves) ⁴⁾
ECAT_STATE_SAFEOP	4	EtherCAT Zustand "Safe-Operational"
ECAT_STATE_OP	8	EtherCAT Zustand "Operational"

Tab. 9-79: Elemente des Aufzählungstyps IL_ECAT_STATE

9.8 Fehlercodes

9.8.1 EtherCAT Fehlertabelle

Alle Fehlercodes des EtherCAT-Masters sind auf dem Steuerungssystem MLC Bestandteil des MLC-Diagnosesystems:

- Als Kategorie wird in der Regel "F0" verwendet (= nicht fataler Fehler)
- Der EtherCAT-Master nutzt derzeit die Diagnosegruppen 0xEC, 0xED, 0xEE u. 0xEF
- Es folgt eine vierstellige Fehlernummer

Siehe Fehlerbeschreibungen unter MLC-Diagnosen.

⁴⁾ Optionaler Zustand bei Slave Geräten hauptsächlich für FW-Update; kann nur vom Slave-Zustand "Init" aus erreicht werden.



Der Fehlercode kann über die "Index"-Suche in der IndraWorks Online-Hilfe aufgerufen werden, z.B. "F0EC0001".

Bei den Funktionsbausteinen wird der Fehlercode am Ausgang "ErrorIdent.Additional1" angezeigt.

Spezifische Fehler der Funktionsbausteine sind in der jeweiligen Funktionsbeschreibung nachzuschlagen.

Eine Auflistung der EtherCAT-Fehlercodes ist in der RIL_Ethercat-Bibliothek auch als GVL "ErrorCodes" enthalten.

10 RIL_MappingList.library

10.1 Einführung und Überblick

Für den azyklischen Zugriff auf Feldbuslaves (DP-Slave, PNIO-Device und ENIP-Adapter) wird eine Mapping-Tabelle auf den Feldbuslaves realisiert.

Darin wird für die busspezifischen azyklischen Feldbuszugriffe eine Abbildung auf Datenobjekte der Steuerung hinterlegt.

Bei einem azyklischen Feldbuszugriff wird diese Abbildung (=Adressierungs)-Vorschrift ausgeführt.

Es ist geplant, die Mapping-Tabelle über ein Konfigurationstool auf der Oberfläche zu erstellen und auf die Steuerung zu laden.

Solange dieser Konfigurator nicht verfügbar ist, kann die Mapping-Tabelle mit der RIL_MappingList.library aus dem IEC-Programm gefüllt und verändert werden.

In der Mapping-Tabelle werden zu feldbusspezifischen (Objekt-) Zugriffen SPS-Adressen hinterlegt, über die der Nutzdatenzugriff erfolgt.

Diese Adressen können sein:

- symbolische SPS-Operanden
- absolute Speicheradressen

Die Mapping-Tabelle wird aus den Inhalten der IO-Konfiguration (Endausbau, abhängig vom Mapping-Konfigurator in der Oberfläche) oder über die hier beschriebenen Bausteinaufrufe aus dem IEC-Programm gefüllt.

Die Mapping-Tabelle existiert einmal für jeden Feldbuslave.

Die Feldbusadressen können wahlfrei innerhalb der, vom jeweiligen Feldbus zugelassenen Werte, verwendet werden.

Die Anzahl der möglichen Einträge in die Mapping-Tabelle ist derzeit auf 256 Einträge beschränkt.

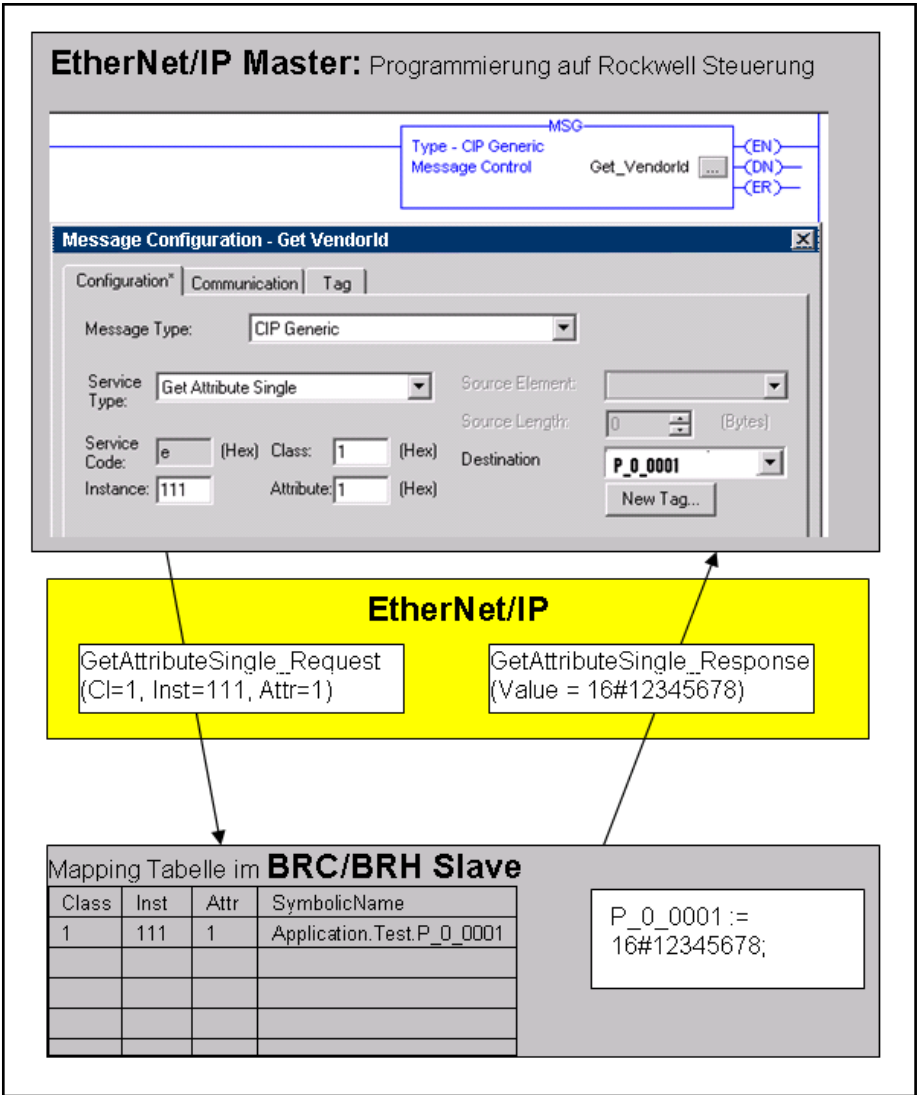


Abb. 10-1: Beispielhafter Zugriff über Ethernet/IP

Anwendungsgebiete und Funktionsbausteine von RIL_MappingList.library

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine	Beschreibung
POUs	
IL_SlaveMapListInit, Seite 333	Initialisieren und Löschen der Mapping-Tabelle
IL_SlaveMapListAddEntry, Seite 334	Hinzufügen eines Eintrages in die Mapping-Tabelle
Datentypen (DataTypes):	
IL_FIELDBUSTYPE auf Seite 338	dient dazu, die Art des jeweiligen Feldbusses auszuwählen
IL_FIELDBUSOBJECT auf Seite 337	beinhaltet die Adressierungen der einzelnen Feldbus-Objekte
IL_ADDRESSTYPE auf Seite 337	dient dazu, die Art der zu mappenden Variablen auszuwählen

Tab. 10-1: RIL_MappingList-Bibliothek

Auswahl des Slaves Der Aufzählungstyp IL_BUSSLAVE aus der RIL_FieldbusTypes.library dient dazu, den jeweiligen Feldbus-Slave auszuwählen.
 Siehe auch: [IL_BUSSLAVE auf Seite 339](#)

10.2 POU's

10.2.1 IL_SlaveMapListInit

Kurzbeschreibung Dieser Funktionsbaustein initialisiert und löscht die gesamte Mapping-Tabelle.

Damit ist sichergestellt, dass keine historischen Einträge vorhanden sind.

Bei einer Nutzung der RIL_MappingList Library sollte vor dem Hinzufügen von Zugriffseinträgen die Liste initialisiert werden, um Wechselwirkungen mit anderen Methoden zum Anlegen von Zugriffseinträgen zu vermeiden.

Ebenso kann mit diesem FB der Zugriff von außen über den Feldbus wieder unterbunden werden.

Zielsystem	Bibliothek /Ordner
MLC	RIL_MappingList/ POU's
MTX	RIL_MappingList/ POU's

Tab. 10-2: Referenztable

Schnittstellenbeschreibung

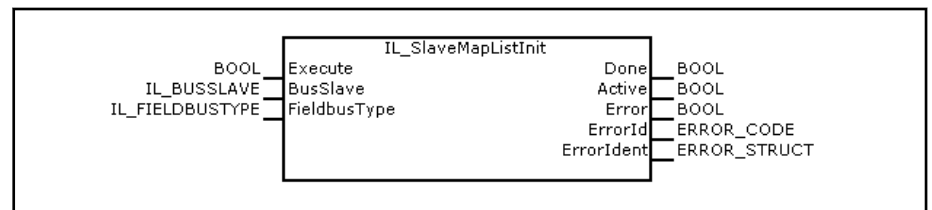


Abb. 10-2: Schnittstelle FB IL_SlaveMapListInit

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusSlave	IL_BUSSLAVE	Instanz des Feldbus-slaves in der Reihenfolge des Konfigurationsbaumes. Ist nur ein Slave/Device/Adapter vorhanden, besitzt dieser immer den Wert IL_BUSSLAVE_0. IL_BUSSLAVE auf Seite 339
	FieldType	IL_FIELDBUSTYPE	Auswahl des Feldbus-typs (Profinet IO; Profibus DP, EtherNet/IP) siehe auch IL_FIELDBUSTYPE auf Seite 338
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall (aus RIL_CommonTypes)
	ErrorIdent	ERROR_STRUCT	Detalldiagnose (ErrorTable NO_TABLE_USED)

Tab. 10-3: IL_SlaveMapListInit E/A-Schnittstelle

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	kontinuierlich
BusSlave	IL_BUSSLAVE	IL_BUSSLAVE_0	IL_BUSSLAVE_4	IL_BUSSLAVE_0	Steigende Flanke von Execute
FieldbusType	IL_FIELDBUS-TYPE	IL_DP_SLAVE	IL_ENIP_ADAP-TER	IL_DP_SLAVE	Steigende Flanke von Execute

Tab. 10-4: Min-/Max und Default-Werte der Eingänge IL_SlaveMapListInit

Funktionsbeschreibung

Der Funktionsbaustein IL_SlaveMapListInit dient dazu, eine vorhandene Mapping-Liste zu löschen.

Eine Mapping-Tabelle kann durch folgende Aktionen vorhanden sein:

- die Mapping-Tabelle wurde durch die Oberfläche generiert,
- die Mapping-Tabelle wurde früher durch das SPS-Programm generiert.

Der FB muss in den folgenden Fällen aufgerufen werden

- wenn die Mapping-Tabelle nicht über die Oberfläche generiert wird,
- wenn Einträge der Mapping-Tabelle entfernt werden sollen (d.h. komplettes Löschen der Tabelle).

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle **NO_TABLE_USED 16#0000**. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	
SYSTEM_ERROR	16#0101	16#0000 16#0001	BusSlave nicht in der Konfigurationstabelle
STATE_MACHINE_ERROR	16#0105	0	Fehler in der Statemaschine
INPUT_RANGE_ERROR	16#0108	0	Fieldbustyp nicht unterstützt

Tab. 10-5: Fehlercodes FB IL_SlaveMapListInit

10.2.2 IL_SlaveMapListAddEntry

Kurzbeschreibung

Der Funktionsbaustein fügt der Mapping-Tabelle einen Eintrag hinzu. Der Eintrag ist fest mit der Feldbusobjektadresse verbunden. Eine Feldbusobjektadresse kann nur einmal belegt werden.

Zielsystem	Bibliothek /Ordner
MLC	RIL_MappingList/ POU's
MTX	RIL_MappingList/ POU's

Tab. 10-6: Referenztable

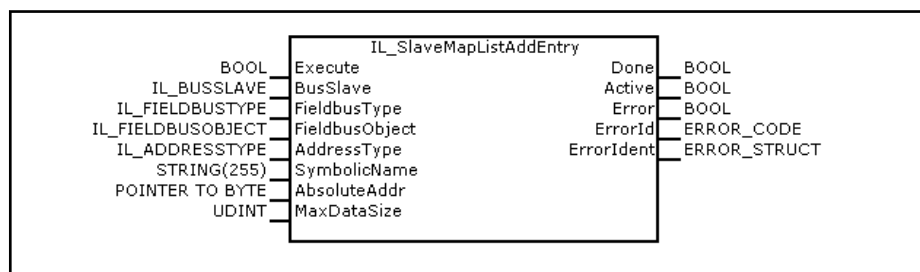
Schnittstellenbeschreibung

Abb. 10-3: Schnittstelle FB IL_SlaveMapListAddEntry

	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flanken-gesteuert)
	BusSlave	IL_BUSSLAVE	Instanz des Feldbusslaves in der Reihenfolge des Konfiguration-baumes. Ist nur ein Slave/Device/Adapter vorhanden, besitzt dieser immer den Wert IL_BUSSLAVE_0. IL_BUSSLAVE auf Seite 339
	FieldbusType	IL_FIELDBUSTYPE	Auswahl des Feldbustyps (ProfinetIO; ProfibusDP, EtherNet/IP) siehe auch IL_FIELDBUSTYPE auf Seite 338
	FieldbusObject	IL_FIELDBUSOB- JECT	Feldbusspezifische Objekt-Adresse siehe auch IL_FIELDBUSOBJECT auf Seite 337
	AddressType	IL_ADRESSTYPE	Art der Adressierung siehe auch IL_ADRESSTYPE auf Seite 337
	SymbolicName	STRING(255)	Symbolischer Name einer Variablen aus dem IEC-Programm Das Symbol muss den einschlägigen IEC –Programmierichtlinien entsprechen. Um die Eindeutigkeit sicherzustellen kann es not-wendig sein auch Task und POU angeben zu müssen.
	AbsoluteAddr	POINTER TO BYTE	Zeiger auf einen absolut adressierten Datenbereich
	MaxDataSize	UDINT	Maximal zulässige Bytelänge für den Objektzugriff. Mit dieser Angabe kann der Lese/Schreibe Zugriff auf die ange-gebene Byteanzahl beschränkt werden. Bei Input Adresstype = FBT_SYMBOL_IEC (_RO) kann diese Grenze auch unterhalb der Größe des über SymbolicName ad-ressierten Objektes liegen.
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall (aus RIL_CommonTy-pes)
	ErrorIdent	ERROR_STRUCT	Detailldiagnose (ErrorTable NO_TABLE_USED)

Tab. 10-7: E/A-Schnittstelle IL_SlaveMapListAddEntry

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusSlave	IL_BUSSLAVE	IL_BUSSLAVE_0	IL_BUSSLAVE_4	IL_BUSSLAVE_0	Steigende Flanke von Execute
FieldbusType	IL_FIELDBUS- TYPE	IL_DP_SLAVE	IL_ENIP_ADAPTER	IL_DP_SLAVE	Steigende Flanke von Execute
FieldbusObject	IL_FIELDBU- SBJECT	feldbusabhängig	feldbusabhängig		Steigende Flanke von Execute
AddressType	IL_ADRESS- TYP	FBT_SYMBOL_IEC	FBT_ADR_IEC_RO	FBT_SYM- BOL_IEC	Steigende Flanke von Execute

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
SymbolicName	STRING(255)	n.a.	n.a.	"	Steigende Flanke von Execute
AbsoluteAddr	POINTER TO BYTE	n.a.	n.a.	0	Steigende Flanke von Execute
MaxDataSize	UDINT	0	feldbusabhängig	0	Steigende Flanke von Execute

Tab. 10-8: Min-/Max und Default-Werte der Eingänge IL_SlaveMapListAddEntry

Funktionsbeschreibung

Beispiel: Eintragen des symbolischen lesenden Zugriffs auf die IEC-Variable ‚wrvar‘ über Feldbusindex 16#1001 auf Slot und Subslot 1 für ein PN-Device. Der schreibende Zugriff ist gesperrt Das Symbol ‚wrvar‘ muss in der Symboltabelle eingetragen sein:

Programmauszug

```
fbMapAdd: IL_SlaveMapListAddEntry;
wrvar:    DINT;

fbMapAdd.FielfbusType:=      IL_FIELDBUSTYPE.IL_PNIO_DEVICE;
fbMapAdd.BusSlave:=          IL_BUSSLAVE_0;
fbMapAdd.FielfbusObject.Index:= 16#1001;
fbMapAdd.FielfbusObject.Slot:= 1;
fbMapAdd.FielfbusObject.SubSlot:= 1;
fbMapAdd.MaxDataSize:=      SIZEOF(wrvar);
fbMapAdd.AddressType:=      IL_ADDRESSSTYPE.IL_SYMBOL_IEC_RO; //read only
fbMapAdd.SymbolicName:=      'Application.MappingTest.wrvar';

fbMapAdd (Execute:= TRUE);
```

Beispiel: Eintragen des absolute adressierten lesenden und schreibenden Zugriffs auf die IEC-Variable ‚wrvar‘ über Feldbusindex 16#1002 auf Slot und Subslot 1 für ein PN-Device.

Programmauszug

```
fbMapAdd: IL_SlaveMapListAddEntry;
wrvar:    DINT;

fbMapAdd.Execute:=          TRUE;
fbMapAdd.FielfbusType:=      IL_FIELDBUSTYPE.IL_PNIO_DEVICE;
fbMapAdd.FielfbusObject.Index:= 16#1002;
fbMapAdd.FielfbusObject.Slot:= 1;
fbMapAdd.FielfbusObject.SubSlot:= 1;
fbMapAdd.MaxDataSize:=      SIZEOF(wrvar);
fbMapAdd.AddressType:=      IL_ADDRESSSTYPE.IL_ADDR_IEC;
fbMapAdd.AbsolutAddr:=      ADDR(wrvar);

fbMapAdd (Execute:= TRUE);
```

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle **NO_TABLE_USED 16#0000**. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	
SYSTEM_ERROR	16#0101	16#0000 16#0001	BusSlave nicht in der Konfigurationstabelle
STATE_MACHINE_ERROR	16#0105	0	Fehler in der Statemaschine
INPUT_RANGE_ERROR	16#0108	0	Feldbustyp nicht unterstützt
INPUT_RANGE_ERROR	16#0109	0	Adresstype nicht unterstützt
SYSTEM_ERROR	16#010A	0	Kein Eintrag mehr verfügbar.
SYSTEM_ERROR	16#010B	0	Eintrag ist schon verwendet
INPUT_RANGE_ERROR	16#010C	0	Symbol in SymbolicName nicht gefunden

ErrorID	Additional1	Additional2	
INPUT_RANGE_ERROR	16#010D	0	FieldbusObject ungültig
INPUT_RANGE_ERROR	16#010E	0	FieldbusObject ist reserviert
INPUT_RANGE_ERROR	16#010F	0	AbsoluteAddr ungültig
INPUT_RANGE_ERROR	16#0110	0	MaxDataSize ist Null
INPUT_RANGE_ERROR	16#0111	0	MaxDataSize ist zu groß
INPUT_RANGE_ERROR	16#0112	0	Index bereits reserviert
INPUT_RANGE_ERROR	16#0113	0	Slot bereits reserviert
INPUT_RANGE_ERROR	16#0114	0	Subslot bereits reserviert
INPUT_RANGE_ERROR	16#0115	0	Class bereits reserviert
INPUT_RANGE_ERROR	16#0116	0	Instanz bereits reserviert
INPUT_RANGE_ERROR	16#0117	0	Attribute bereits reserviert

Tab. 10-9: Fehlercodes FB IL_SlaveMapListAddEntry

10.3 Datentypen (DataTypes)

10.3.1 IL_ADDRESSTYPE

Kurzbeschreibung Der Aufzählungstyp IL_ADDRESSTYPE dient dazu, die Art der zu mappenden Variablen auszuwählen.

Element	Wert	Beschreibung
IL_SYMBOL_IEC	0	Symbolischer IEC-Variablenname; der Parameter SymbolicName wird ausgewertet; Lesender und schreibender Zugriff erlaubt
IL_ADDR_IEC	1	Absolute IEC-Adresse; der Parameter AbsoluteAddr wird ausgewertet. Lesender und schreibender Zugriff erlaubt
IL_SYMBOL_IEC_RO	2	Symbolischer IEC-Variablenname; der Parameter SymbolicName wird ausgewertet; Nur lesender Zugriff erlaubt
IL_ADDR_IEC_RO	3	Absolute IEC-Adresse; der Parameter AbsoluteAddr wird ausgewertet. Nur lesender Zugriff erlaubt

Tab. 10-10: Elemente des Aufzählungstyps IL_ADDRESSTYPE

10.3.2 IL_FIELDBUSOBJECT

Kurzbeschreibung Die Datenstruktur IL_FIELDBUSOBJECT beinhaltet die Adressierungen der einzelnen Feldbus-Objekte.

	Typ	Notwenig bei ProfibusDP IL_DP_SLAVE	Notwenig bei ProfinetIO IL_PNIO_DEVICE	Notwenig bei EthernetIP IL_ENIP_ADAPTER	Beschreibung
Slot	WORD	ja	ja		Slotnummer
Index	WORD	ja	ja		Index
SubSlot	WORD		ja		Subslotnummer
Class	WORD			ja	Klasse

	Typ	Notwendig bei ProfibusDP IL_DP_SLAVE	Notwendig bei ProfinetIO IL_PNIO_DEVICE	Notwendig bei EthernetIP IL_ENIP_ADAPTER	Beschreibung
Instance	WORD			ja	Instanz
Attribute	WORD			ja	Attribut

Tab. 10-11: Schnittstellenbeschreibung Datenstruktur IL_FIELDBUSOBJECT

Adressierung der Feldbusobjekte bei ProfibusDP:

	Typ	Min.-Wert	Max.-Wert	Default-Wert	Beschreibung
Slot	WORD	0	254		Slotnummer
Index	WORD	0	254		Index

Tab. 10-12: IL_FIELDBUSOBJECT für ProfibusDP

Adressierung der Feldbusobjekte bei Profinet IO:

	Typ	Min.-Wert	Max.-Wert	Default-Wert	Beschreibung
Slot	WORD	0	16#7FFF		Slotnummer
Index	WORD	0	16#7FFF		Index (16#8000 bis 16#ffff ist reserviert.)
SubSlot	WORD	1	16#8FFF		Subslotnummer

Tab. 10-13: IL_FIELDBUSOBJECT für ProfinetIO

Adressierung der Feldbusobjekte bei EtherNet/IP:

	Typ	Min.-Wert	Max.-Wert	Default-Wert	Beschreibung
Class	WORD	0	16#FFFF		
Instance	WORD	1	16#FFFF		
Attribute	WORD	1	16#FFFF		

Tab. 10-14: IL_FIELDBUSOBJECT für EtherNet/IP

10.3.3 IL_FIELDBUSTYPE

Kurzbeschreibung Der Aufzählungstyp IL_FIELDBUSTYPE dient dazu, die Art des jeweiligen Feldbusses auszuwählen.

Derzeit gibt es Feldbusslaves für

- Profibus DP,
- Profinet IO,
- Ethernet/IP.

Element	Wert	Beschreibung
IL_DP_SLAVE	0	Profibus DP
IL_PNIO_DEVICE	1	Profinet IO
IL_ENIP_ADAPTER	2	EtherNet/IP

Tab. 10-15: Elemente des Aufzählungstyps IL_FIELDBUSTYPE

10.3.4 IL_BUSSLAVE

Kurzbeschreibung

Der Aufzählungstyp IL_BUSSLAVE aus der RIL_FieldbusTypes.library dient dazu, den jeweiligen Feldbus-Slave auszuwählen.

In einer Steuerung können neben der onboard Realtime-Ethernet-Schnittstelle über Funktionsmodule weitere Controller vorhanden sein.

Die Geräte werden anhand ihrer aufsteigenden Reihenfolge in der Konfiguration unterschieden. Die Indizierung der Geräte geschieht dabei in der Reihenfolge

- Onboard,
- Funktionsmodul 1,
- Funktionsmodul 2,
- Funktionsmodul 3,
- Funktionsmodul 4,

d.h. ist beispielsweise die onboard Realtime-Ethernet-Schnittstelle ein PROFINET IO Controller und ist auf dem FM1 ein PROFINET IO Device konfiguriert, so erhält das Device auf dem FM1 den Wert IL_BUSSLAVE_0.

Element	Wert	Beschreibung
IL_BUSSLAVE_0	0	Erstes Gerät
IL_BUSSLAVE_1	1	Zweites Gerät
IL_BUSSLAVE_2	2	Drittes Gerät
IL_BUSSLAVE_3	3	Viertes Gerät
IL_BUSSLAVE_4	4	Fünftes Gerät

Tab. 10-16: Elemente des Aufzählungstyps IL_BUSSLAVE

11 RIL_SercosIII.library

11.1 Einführung und Übersicht

Die Bibliothek RIL_SercosIII.library ist die Schnittstelle von der SPS-Programmierungsumgebung zu den Sercos III IO-Teilnehmern. Über diese Schnittstelle können zum Beispiel die Sercos III IO-Teilnehmer diagnostiziert werden oder azyklisch Sercos-Parameter ausgetauscht werden.

Zielsysteme Die Bibliothek kann auf folgenden Systemen eingesetzt werden:

Zielsystem	Bibliothek
MLC/MTX	RMB_SercosIII_Util
MLD mit IndraLogic 2G ab 20VRS	RMB_SercosIII_Util

Anwendungsgebiete und Funktionsbausteine von RIL_SercosIII

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine	Beschreibung
Azyklische Dienste (AcyclicCommunication)	
IL_SIIISvcRead, Seite 343	Lesen eines Parameters über den Sercos III Service-Kanal
IL_SIIISvcWrite, Seite 348	Schreiben eines Parameters über den Sercos III Service-Kanal
Azyklische Dienste/Parameter (AcyclicCommunication/Parameter)	
IL_SIIICommand, Seite 351	Ausführen der Kommandos eines Sercos III-Slaves über den Sercos III-Service-Kanal
IL_SIIIReadParameter, Seite 358	Kann über den Sercos III-Service-Kanal 2- oder 4 Byte-Parameter von einem Slave-Teilnehmer lesen (DINT)
IL_SIIIReadRealParameter, Seite 363	Kann über den Sercos III-Service-Kanal 2- oder 4 Byte-Parameter von einem Slave-Teilnehmer lesen (REAL)
IL_SIIIReadStringParameter, Seite 365	Kann über den Sercos III-Service-Kanal Textlisten von einem Slave-Teilnehmer lesen (STRING)
IL_SIIISvcWriteParameter, Seite 367	Kann über den Sercos III-Service-Kanal 2- oder 4 Byte-Parameter eines Slave-Teilnehmers schreiben (DINT)
IL_SIIISvcWriteRealParameter, Seite 370	Kann über den Sercos III-Service-Kanal 2- oder 4 Byte-Parameter eines Slave-Teilnehmers schreiben (REAL)
Mit Datentyp:	
IL_SERCOS_DATA_STATUS, Seite 372	Status eines vom durch den Funktionsbaustein IL_SIIICommand aus der RIL_SercosIII.library gestarteten Kommandos wird als Klartext lesbar angezeigt
Sichern und Wiederherstellen (Backup_Restore)	
IL_SIIIParallelBackup, Seite 373	Sichert die Parameter eines Sercos-Gerätes in einer textbasierten Datei auf einem Speichermedium des Sercos-Masters.
IL_SIIIParallelRestore, Seite 379	Importiert die S- und P-Parameter eines Sercos-Gerätes von einer textbasierten Datei
Mit Datentypen:	
IL_BACKUP_OPTIONS, Seite 386	Mögliche Auswahl der Parameter zum Backup

IL_BACKUP_ERRORS, Seite 386	Fehler bei Backup/ Restore
Diagnose (Diagnostics)	
IL_SIIIGetConfigSlaveAdrList, Seite 387	Erzeugt Liste der Sercos-Adressen aller konfigurierten SPS-Sercos-Teilnehmer
IL_SIIIGetDiagSlaveAdrList, Seite 388	Erzeugt Liste der Sercos-Adressen aller SPS-Sercos-Teilnehmer für die Diagnoseinformationen vorliegen
IL_SIIIGetLiveSlaveAdrList, Seite 390	Erzeugt Liste der Sercos-Adressen aller Sercos Teilnehmer, die am Bus gefunden wurden
IL_SIIIMasterState, Seite 391	Beschreibt den Status des Sercos-Masters
IL_SIIIMasterStateDetails, Seite 393	Beschreibt den Detail-Status des Sercos-Masters mit Fehlerzählern und Topologie-Informationen
IL_SIIIRemoteSlaveState, Seite 394	Beschreibt den Status eines am Sercos-Master betriebenen Slaves
IL_SIIIRemoteSlaveStateDetails, Seite 396	Beschreibt den Detail-Status eines am Sercos-Master betriebenen Slaves
Mit Datentypen:	
IL_SIII_REMOTE_SLAVE_STATE_DETAILS, Seite 399 IL_SIII_REMOTE_SLAVE_STATE_EX, Seite 401 IL_SIII_REMOTE_SLAVE_STATE_EX_BITS, Seite 401 IL_SIII_DEVICE_STATUS, Seite 402 IL_SIII_DEVICE_STATUS_UNION, Seite 402 IL_SIII_DEVICE_STATUS_UNION_BITS, Seite 403 IL_SIII_TOPOLOGY_PORT_STATUS, Seite 403 IL_SIII_DIAGNOSTIC_NUMBER, Seite 404 IL_SIII_DIAGNOSTIC_NUMBER_R, Seite 405 IL_SIII_DIAGNOSTIC_NUMBER_SOURCE_GROUP, Seite 405 IL_SIII_DIAGNOSTIC_NUMBER_SOURCE_TYPE, Seite 405 IL_SIII_DIAGNOSTIC_NUMBER_CLASS, Seite 406 IL_SIII_DEVICE_TYPE, Seite 406 IL_SIII_DRIVE_DIAG, Seite 406 IL_SIII_DRV_ERROR, Seite 407 IL_SIII_DRV_ERROR_BITS, Seite 407 IL_SIII_DRV_WARNING, Seite 408 IL_SIII_DRV_WARNING_BITS, Seite 408 IL_SIII_DRV_MESSAGE, Seite 409 IL_SIII_DRV_MESSAGE_BITS, Seite 409 IL_SIII_IO_DIAG, Seite 410 IL_SIII_IO_DIAG_STATUS_CODE_TYPE, Seite 411 IL_SIII_IO_DIAG_STATUS_CODE, Seite 411	

Hilfsfunktionen (Utilities)	
IL_SIIAttribute , Seite 418	Aufschlüsselung der Elemente des Sercos-Attributes
IL_SIIIElementsToldn , Seite 421	Fasst die Einzelelemente einer IDN in einen MB_IDN-Wert zusammen
IL_SIIIGetSercosPhase , Seite 422	Liest die Sercos-Phase eines Sercos-Busmasters
IL_SIIISetSercosPhase , Seite 424	Schreibt die Sercos-Phase eines Sercos-Busmasters
IL_SIIISStringToldn , Seite 426	Wandelt einen String der Form "S-0-0001.0.0" oder in Kurzformen "S-0-0001" oder "S1" in einen MB_IDN-Wert
Mit Datentypen:	
IL_SIII_DATA_LENGTH , Seite 427	Unterstützt die Auswertung des Sercos Attributes bezüglich Datlänge und Unterscheidung Listen- oder Einzelparameter
IL_SIII_DATA_TYPE , Seite 427	Unterstützt die Auswertung des Sercos Attributes bezüglich Datentyp und Anzeigeformat
IL_SIII_ELEMENT , Seite 427	Beschreibt die Sercos-Elemente eines Parameters
IL_SIII_PARAM_TYPE , Seite 428	Unterscheidet zwischen Standard-IDN und produktspezifischer IDN
IL_SIII_SERCOS_PHASE , Seite 428	Enthält die möglichen aktiven Sercos-Phasen

Tab. 11-1: RIL_SercosIII-Bibliothek

Auswahl des Masters

Der Aufzählungstyp **IL_BUSMASTER** aus der RIL_FieldbusTypes.library dient dazu, den jeweiligen Sercos-III-Master auszuwählen.

Momentan wird nur der Onboard-Master unterstützt.

Siehe auch [IL_BUSMASTER](#), Seite 429.

11.2 Azyklische Dienste (AcyclicCommunication)

11.2.1 IL_SIIISvcRead

Kurzbeschreibung

Mit dem Funktionsbaustein IL_SIIISvcRead können über den Sercos III-Service-Kanal Parameter von einem Slave-Teilnehmer gelesen werden.

Zielsystem	Bibliothek/ Ordner
MLC	RIL_SercosIII/ AcyclicCommunication
MTX	RIL_SercosIII/ AcyclicCommunication
MLD (IL 2G)	RIL_SercosIII/ AcyclicCommunication

Tab. 11-2: Referenztable

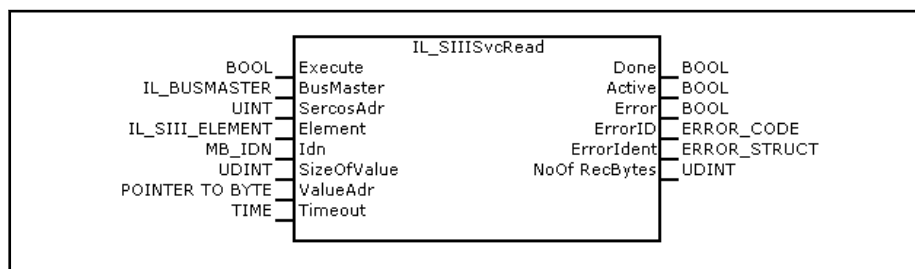
Schnittstellenbeschreibung

Abb. 11-1: Funktionsbaustein IL_SIIISvcRead

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable BusMaster unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 429
	SercosAdr	UINT	Sercos III-Adresse des Slave-Teilnehmers.
	Element	IL_SIII_ELEMENT	Sercos-Daten-Block-Element. Mit diesem Eingang wird entschieden, ob der Name, das Attribute, die Einheit, der Minimal Wert, Maximal Wert oder das Datum eines Parameters gelesen wird. Mit Sercos-Daten-Block-Element 1 (Element := IL_STATUS;) wird der Service-Kanal Data Status gelesen. siehe auch IL_SIII_ELEMENT auf Seite 427
	Idn	MB_IDN	IDN des Parameters. Siehe Funktion IL_SIIIElementsToldn auf Seite 421 .
	SizeOfValue	UDINT	Größe des Buffers bzw. der Variablen in Bytes, der für den Empfang der Daten zur Verfügung gestellt wird (SIZEOF(Value))
	ValueAdr	POINTER TO BYTE	Adresse des Buffers bzw. der Variablen, der für den Empfang der Daten zur Verfügung gestellt wird.
	Timeout	TIME	Timeout-Überwachung des FBs in ms. T#0ms = Timeout-Überwachung ist nicht aktiv.
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig.
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig.
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCT	Detaildiagnose
	NoOfRecBytes	UDINT	Anzahl der in den ValueAdr Buffer kopierten Bytes.

Tab. 11-3: Schnittstelle IL_SIIISvcRead

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	USINT	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
SercosAdr	UINT	0 (MLD: lokale Achse) 1 (andere Systeme)	511	0	Steigende Flanke an Execute
Element	IL_SIII_ELEMENT	1	7	7	Steigende Flanke an Execute
Idn	MB_IDN	0	16#FFFFFFFF	0	Steigende Flanke an Execute

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
SizeOfValue	UDINT	0	16#FFFF	0	Steigende Flanke an Execute
ValueAdr	POINTER TO BYTE	n.def.	n.def.	0	Steigende Flanke an Execute
Timeout	TIME	T#0s	T#60 min	T#2s	Steigende Flanke an Execute

Tab. 11-4: Min-/Max und Default-Werte der Eingänge IL_SIII_SvcRead

Fehlerbehandlung

Der Funktionsbaustein nutzt zur Darstellung von Fehlern des Sercos-Service-Kanals die Fehlertabelle SERCOS_TABLE (ERROR_TABLE = 16#0010).

Zusätzlich sind folgende Fehler des Bausteines in der Fehlertabelle **F_RELATED_TABLE** (ERROR_TABLE = 16#0170) bzw. **SERCOS_TABLE** (ERROR_TABLE = 16#0010) zu finden.

ErrorID	Additional1	Additional2	
Fehler des Bausteines in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170)			
INPUT_RANGE_ERROR	16#00000002	16#00000001	Eingangswert des Parameters "BusMaster" ist ungleich IL_BUSMASTER_0 und außerhalb des Eingangswertebereichs
INPUT_RANGE_ERROR	16#00000002	16#00000002	Eingangswert des Parameters "SercosAdr" ist 0 oder > 511 MLD: Eingangswert des Parameters "SercosAdr" > 511
INPUT_RANGE_ERROR	16#00000002	16#00000003	Eingangswert des Parameters "Element" ist <> 7
INPUT_RANGE_ERROR	16#00000002	16#00000004	Eingangswert des Parameters "Idn" ist > 16#FFFFFFFF
INPUT_RANGE_ERROR	16#00000002	16#00000006	Eingangswert des Parameters "ValueAdr" ist 0
INPUT_RANGE_ERROR	16#00000002	16#00000007	Eingangswert des Parameters "Timeout" > T#60 min
STATE_MACHINE_ERROR	16#00000006	16#00000001	Fehler in der Zustandsmaschine
OTHER_ERROR	16#00000015	16#00000001	Zeitüberwachungsfehler wurde festgestellt
Fehler des Bausteines in der Fehlertabelle SERCOS_TABLE (ERROR_TABLE = 16#0010)			
COMMUNICATION_ERROR	16#00008002	16#00000000	Eingangswert des Parameters "SizeOfValue" ist zu klein. Die empfangenen Daten passen nicht in den zur Verfügung gestellten Speicher

Tab. 11-5: Fehlercodes Funktionsbaustein IL_SIII_SvcRead in F_RELATED_TABLE := 16#0170 bzw. SERCOS_TABLE (ERROR_TABLE = 16#0010).

Beispiele**Lesen von Einzelparameter und Datenbausteinelementen**

Beim Lesen von Einzelparameter und Datenbausteinelementen **sollten** der Datentyp der Variablen am Eingang "ValueAdr" zur Datengröße des zu lesenden Elements passen.

Im folgenden Beispiel wird der Sercos-Parameter S-0-1002.0.0, Sercos-Zykluszeit gelesen.

Es werden alle Sercos Datenblockelemente des Parameters gelesen.

Hilfsstrukturen

```

TYPE LIST_STRING:
STRUCT
  uiCurLength: UINT;
  uiMaxLength: UINT;

```

```

    strValue:    STRING;
END_STRUCT
END_TYPE

```

Variablendefinition

```

VAR
    fbSIIISvcRead:          IL_SIIISvcRead;

    uiServiceChannelStatus_E1:    UINT;
    tSercosCycleTimeName_E2:    LIST_STRING;
    dwSercosCycleTime_Attribute_E3:  DWORD;
    tSercosCycleTime_Unit_E4:    LIST_STRING;
    udiSercosCycleTime_Min_E5:    UDINT;
    udiSercosCycleTime_Max_E6:    UDINT;
    udiSercosCycleTime_Value_E7:  UDINT;
END_VAR

```

Service Kanal Datenstatus lesen

```

fbSIIISvcRead.Execute:=    TRUE;
fbSIIISvcRead.SercosAdr:=  65;
fbSIIISvcRead.Element:=    IL_STATUS;
fbSIIISvcRead.Idn:=        IL_SIIIElementsToIdn(IL_S_PARAM,0,1002,0,0);
fbSIIISvcRead.SizeOfValue:=  SIZEOF(uiServiceChannelStatus_E1);
fbSIIISvcRead.ValueAdr:=    ADR(uiServiceChannelStatus_E1);

fbSIIISvcRead();

```

Parameter-Name lesen

```

fbSIIISvcRead.Execute:=    TRUE;
fbSIIISvcRead.SercosAdr:=  65;
fbSIIISvcRead.Element:=    IL_NAME;
fbSIIISvcRead.Idn:=        IL_SIIIElementsToIdn(IL_S_PARAM,0,1002,0,0);
fbSIIISvcRead.SizeOfValue:=  SIZEOF(tSercosCycleTimeName_E2);
fbSIIISvcRead.ValueAdr:=    ADR(tSercosCycleTimeName_E2);

fbSIIISvcRead();

IF fbSIIISvcRead.Done = TRUE THEN
    tSercosCycleTimeName_E2.strValue[tSercosCycleTimeName_E2.uiCurLength]:=0;
END_IF

```

Parameter-Attribut lesen

```

fbSIIISvcRead.Execute:=    TRUE;
fbSIIISvcRead.SercosAdr:=  65;
fbSIIISvcRead.Element:=    IL_ATTRIBUTE;
fbSIIISvcRead.Idn:=        IL_SIIIElementsToIdn(IL_S_PARAM,0,1002,0,0);
fbSIIISvcRead.SizeOfValue:=  SIZEOF(dwSercosCycleTime_Attribute_E3);
fbSIIISvcRead.ValueAdr:=    ADR(dwSercosCycleTime_Attribute_E3);

fbSIIISvcRead();

```

Parameter Einheit lesen

```

fbSIIISvcRead.Execute:=    TRUE;
fbSIIISvcRead.SercosAdr:=  65;
fbSIIISvcRead.Element:=    IL_UNIT;
fbSIIISvcRead.Idn:=        IL_SIIIElementsToIdn(IL_S_PARAM,0,1002,0,0);
fbSIIISvcRead.SizeOfValue:=  SIZEOF(tSercosCycleTime_Unit_E4);
fbSIIISvcRead.ValueAdr:=    ADR(tSercosCycleTime_Unit_E4);

fbSIIISvcRead();

IF fbSIIISvcRead.Done = TRUE THEN
    tSercosCycleTimeUnit_E4.strValue[tSercosCycleTimeUnit_E4.uiCurLength]:=0;
END_IF

```

Parameter-Minimalwert lesen

```

fbSIIISvcRead.Execute:=    TRUE;
fbSIIISvcRead.SercosAdr:=  65;
fbSIIISvcRead.Element:=    IL_MINVALUE;

```

```

fbSIIISvcRead.Idn:=      IL_SIIIElementsToIdn(IL_S_PARAM,0,1002,0,0);
fbSIIISvcRead.SizeOfValue:= SIZEOF(udiSercosCycleTime_Min_E5);
fbSIIISvcRead.ValueAdr:=  ADR(udiSercosCycleTime_Min_E5);

fbSIIISvcRead();

```

Parameter-Maximalwert lesen

```

fbSIIISvcRead.Execute:=  TRUE;
fbSIIISvcRead.SercosAdr:= 65;
fbSIIISvcRead.Element:=  IL_MAXVALUE;
fbSIIISvcRead.Idn:=      IL_SIIIElementsToIdn(IL_S_PARAM,0,1002,0,0);
fbSIIISvcRead.SizeOfValue:= SIZEOF(udiSercosCycleTime_Max_E6);
fbSIIISvcRead.ValueAdr:=  ADR(udiSercosCycleTime_Max_E6);

fbSIIISvcRead();

```

Parameter-Datum lesen

```

fbSIIISvcRead.Execute:=  TRUE;
fbSIIISvcRead.SercosAdr:= 65;
fbSIIISvcRead.Element:=  IL_OPDATA;
fbSIIISvcRead.Idn:=      IL_SIIIElementsToIdn(IL_S_PARAM,0,1002,0,0);
fbSIIISvcRead.SizeOfValue:= SIZEOF(udiSercosCycleTime_Value_E7);
fbSIIISvcRead.ValueAdr:=  ADR(udiSercosCycleTime_Value_E7);

fbSIIISvcRead();

```

Nach dem erfolgreichen Lesen steht in ValueAdr zum Beispiel der Wert 100000.

Parameter S-0-1002 hat 3 Nachkommastellen. Siehe Attribut.

Somit ist die Sercos-Zykluszeit 1000 µs.



Wenn ein Parameter mit Anzeigeformat FLOAT verwendet wird, sollte der Eingangsparameter von ValueAdr den Datentyp REAL haben.

Bei einem Parameter mit nachkommabehaftetem Dezimalwert ist der Datentyp Integer zu verwenden und die Nachkommastellen zu berücksichtigen.

Lesen von Listen

Im folgenden Beispiel wird der Sercos-Parameter S-0-0017.0.0, IDN-Liste aller Betriebsdaten, gelesen.

Beim Lesen von Listen ist zu berücksichtigen, dass bei einer Liste die Istlänge und Maxlänge Teil der Liste sind.

2 Byte	2 Byte	<Istlänge> Bytes
Istlänge	Maxlänge	Nutzdaten

Hilfsstrukturen

```

TYPE LIST_DINT:
STRUCT
    uiCurLength:  UINT;
    uiMaxLength:  UINT;
    adiValue:     ARRAY[0..10000] OF DINT;
END_STRUCT
END_TYPE

```

Variablendefinition

```

VAR
    fbSIIISvcRead:      IL_SIIISvcRead;

```

```
tListOfIdn_Value_E7: LIST_DINT;  
END_VAR
```

Parameter-Datum lesen von Liste S-0-0017.0.0

```
fbSIIISvcRead.Execute:= TRUE;  
fbSIIISvcRead.SercosAdr:= 65;  
fbSIIISvcRead.Element:= IL_OPDATA;  
fbSIIISvcRead.Idn:= IL_SIIIElementsToIdn(IL_S_PARAM,0,17,0,0);  
fbSIIISvcRead.SizeOfValue:= SIZEOF(tListOfIdn_Value_E7);  
fbSIIISvcRead.ValueAdr:= ADR(tListOfIdn_Value_E7);  
  
fbSIIISvcRead();
```

11.2.2 IL_SIIISvcWrite

Kurzbeschreibung Mit dem Funktionsbaustein IL_SIIISvcWrite können über den Sercos III-Service-Kanal Parameter eines Slave-Teilnehmers geschrieben werden.

Zielsystem	Bibliothek/ Ordner
MLC	RIL_SercosIII/ AcyclicCommunication
MTX	RIL_SercosIII/ AcyclicCommunication
MLD (IL 2G)	RIL_SercosIII/ AcyclicCommunication

Tab. 11-6: Referenztabelle

Schnittstellenbeschreibung

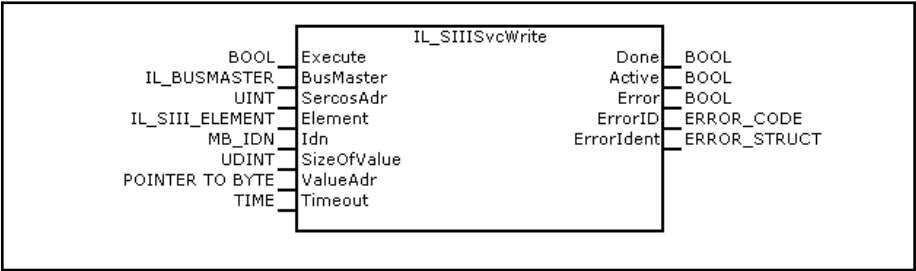


Abb. 11-2: Funktionsbaustein IL_SIIISvcWrite

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flanken-gesteuert)
	BusMaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable BusMaster unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 429
	SercosAdr	UINT	Sercos III-Adresse des Slave-Teilnehmers.
	Element	IL_SIII_ELEMENT	Sercos-Daten-Block-Element. Momentan wird nur IL_OPDATA unterstützt. siehe auch IL_SIII_ELEMENT auf Seite 427
	Idn	MB_IDN	IDN des Parameters siehe Funktion IL_SIIIElementsToIdn auf Seite 421 .
	SizeOfValue	UDINT	Anzahl der Bytes, die über den Service-Kanal geschrieben werden sollen.

E/A-Typ	Name	Datentyp	Beschreibung
	ValueAdr	POINTER TO BYTE	Adresse des Buffers bzw. der Variablen, der für das Senden der Daten zur Verfügung gestellt wird.
	Timeout	TIME	Timeout-Überwachung des FBs in ms. T#0ms = Timeout-Überwachung ist nicht aktiv.
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCT	Detaildiagnose

Tab. 11-7: Schnittstelle IL_SIII_SvcWrite

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	BYTE	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
SercosAdr	UINT	0 (MLD: lokale Achse) 1 (andere Systeme)	511	0	Steigende Flanke an Execute
Element	IL_SIII_ELEMENT	7	7	7	Steigende Flanke an Execute
Idn	MB_IDN	0	16#FFFFFFFF	0	Steigende Flanke an Execute
SizeOfValue	UDINT	0	16#FFFF	0	Steigende Flanke an Execute
ValueAdr	POINTER TO BYTE	n.def.	n.def.	0	Steigende Flanke an Execute
Timeout	TIME	T#0s	T#60 min	T#2s	Steigende Flanke an Execute

Tab. 11-8: Min-/Max und Default-Werte der Eingänge IL_SIII_SvcWrite

Fehlerbehandlung

Der Funktionsbaustein nutzt zur Darstellung von Fehlern des Sercos-Service-Kanals die Fehlertabelle SERCOS_TABLE (ERROR_TABLE = 16#0010).

Zusätzlich sind folgende Fehler des Bausteines in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additional2	
INPUT_RANGE_ERROR	16#00000002	16#00000001	Eingangswert des Parameters "BusMaster" ist ungleich IL_BUSMASTER_0 und außerhalb des Eingangswertebereiches
INPUT_RANGE_ERROR	16#00000002	16#00000002	Eingangswert des Parameters "SercosAdr" ist 0 oder > 511
INPUT_RANGE_ERROR	16#00000002	16#00000003	Eingangswert des Parameters "Element" ist <> 7
INPUT_RANGE_ERROR	16#00000002	16#00000004	Eingangswert des Parameters "Idn" ist > 16#FFFFFFFF

ErrorID	Additional1	Additional2	
INPUT_RANGE_ERROR	16#00000002	16#00000005	Eingangswert des Parameters "SizeOfValue" ist > 16#FFFF
INPUT_RANGE_ERROR	16#00000002	16#00000006	Eingangswert des Parameters "ValueAdr" ist 0
INPUT_RANGE_ERROR	16#00000002	16#00000007	Eingangswert des Parameters "Timeout" > T#60 min
RESOURCE_ERROR	16#00000005	16#00000000	Der FB ist in der vorhandenen Firmware nicht implementiert, z. B. in der MTX-Emulation
STATE_MACHINE_ERROR	16#00000006	16#00000001	Fehler in der Zustandsmaschine
OTHER_ERROR	16#00000015	16#00000001	Zeitüberwachungsfehler wurde festgestellt

Tab. 11-9: Fehlercodes Funktionsbaustein IL_SIIISvcWrite in F_RELATED_TABLE := 16#0170

Beispiele Schreiben von Einzelparameter

Im folgenden Beispiel wird der Sercos Parameter S-0-0099.0.0, Reset-Kommando, geschrieben.

Variablendefinition

```
VAR
  uiDummy:   UINT := 3;
  fbSIIISvcWrite: IL_SIIISvcWrite;
END_VAR
```

Parameter S-0-0099.0.0 schreiben

```
fbSIIISvcWrite.Execute:= TRUE;
fbSIIISvcWrite.SercosAdr:= 0;
fbSIIISvcWrite.Element:= IL_OPDATA;
fbSIIISvcWrite.Idn:= IL_SIIIElementsToIdn(IL_S_PARAM,0,99,0,0);
fbSIIISvcWrite.SizeOfValue:= SIZEOF(uiDummy);
fbSIIISvcWrite.ValueAdr:= ADR(uiDummy);

fbSIIISvcWrite ();
```

Schreiben von Listen

Im folgenden Beispiel wird der Sercos-Parameter S-0-1020.0.0, IP Adresse, geschrieben.

Beim Schreiben von Listen ist zu berücksichtigen, dass bei einer Liste die Istlänge und Maxlänge Teil der Liste sind. Deshalb müssen zur Istlänge der Liste 4 Byte hinzuaddiert werden (siehe Beispiel "fbSIIISvcWrite.SizeOfValue").

2 Byte	2 Byte	<Istlänge> Bytes
Istlänge	Maxlänge	Nutzdaten

Hilfsstrukturen

```
TYPE LIST_USINT:
  STRUCT
    uiCurLength: UINT;
    uiMaxLength: UINT;
    ausiValue:   ARRAY[0..3] OF USINT;
  END_STRUCT
END_TYPE
```

Variablendefinition

```
VAR
  fbSIIISvcWrite: IL_SIIISvcWrite;
  // IndraLogic 2G Syntax
  tIpAddress: LIST_USINT:= (uiCurLength:=4, ausiValue:=[192,186,73,1]);
```

```
(* IndraLogic lx Syntax
tIpAddress: LIST_USINT:= (uiCurLength:=4, ausiValue:=192,186,73,1);*)
END_VAR
```

Parameter S-0-1020.0.0 schreiben

```
fbSIIISvcWrite.Execute:= TRUE;
fbSIIISvcWrite.SercosAdr:= 65;
fbSIIISvcWrite.Idn:= IL_SIIIElementsToIdn(IL_S_PARAM,0,1020,0,0);
fbSIIISvcWrite.SizeOfValue:= tIpAddress.uiCurLength + 4;
fbSIIISvcWrite.ValueAdr:= ADR(tIpAddress);

fbSIIISvcWrite ();
```

11.3 Azyklische Dienste/Parameter (AcyclicCommunication/Parameter)

11.3.1 IL_SIICommand

Kurzbeschreibung Mit dem Funktionsbaustein IL_SIICommand können über den Sercos III-Service-Kanal Kommandos eines Slave-Teilnehmers ausgeführt werden.

Zielsystem	Bibliothek / Ordner
MLC	RIL_SercosIII/ AcyclicCommunication\ Parameter
MTX	RIL_SercosIII/ AcyclicCommunication\ Parameter
MLD (IL 2G)	RIL_SercosIII/ AcyclicCommunication\ Parameter

Tab. 11-10: Referenztabelle Funktionsbaustein IL_SIICommand

Schnittstellenbeschreibung

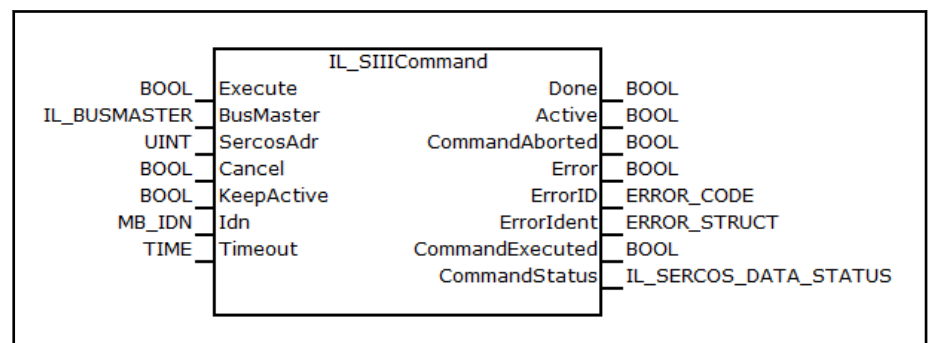


Abb. 11-3: Funktionsbaustein IL_SIICommand

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flanken-gesteuert)
	BusMaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable Bus-Master unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0. IL_BUSMASTER, Seite 429
	SercosAdr	UINT	Sercos III-Adresse des Slave-Teilnehmers.
	Cancel	BOOL	bricht das laufende Kommando bei steigender Flanke ab
	KeepActive	BOOL	hält das Kommando nach Abarbeitung aktiv

E/A-Typ	Name	Datentyp	Beschreibung
	Idn	MB_IDN	IDN des Parameters siehe Funktionen IL_SIIIElementsToldn , Seite 421, bzw. IL_SIIIStringToldn , Seite 426.
	Timeout	TIME	Timeout-Überwachung des FBs in ms. T#0ms = Timeout-Überwachung ist nicht aktiv.
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig.
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig.
	Command-Aborted	BOOL	Kommando erfolgreich abgebrochen und zurückgesetzt
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen.
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall.
	ErrorIdent	ERROR_STRUCT	Detaildiagnose
	Command-Executed	BOOL	Kommando erfolgreich abgeschlossen
	CommandStatus	IL_SERCOS_DATA_STATUS	Status des gewählten Parameters IL_SERCOS_DATA_STATUS , Seite 372

Tab. 11-11: Schnittstelle IL_SIIICommand

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	USINT	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
SercosAdr	UINT	0 (MLD: lokale Achse) 1 (andere Systeme)	511	0	Steigende Flanke an Execute
Cancel	BOOL			FALSE	Kontinuierlich
KeepActive	BOOL			FALSE	Kontinuierlich
Idn	MB_IDN	0	16#FFFFFFFF	0	Steigende Flanke an Execute
Timeout	TIME	T#0s	T#60 min	T#2s	Steigende Flanke an Execute

Tab. 11-12: Min-/Max und Default-Werte der Eingänge IL_SIIICommand

Signal-Zeit-Diagramm

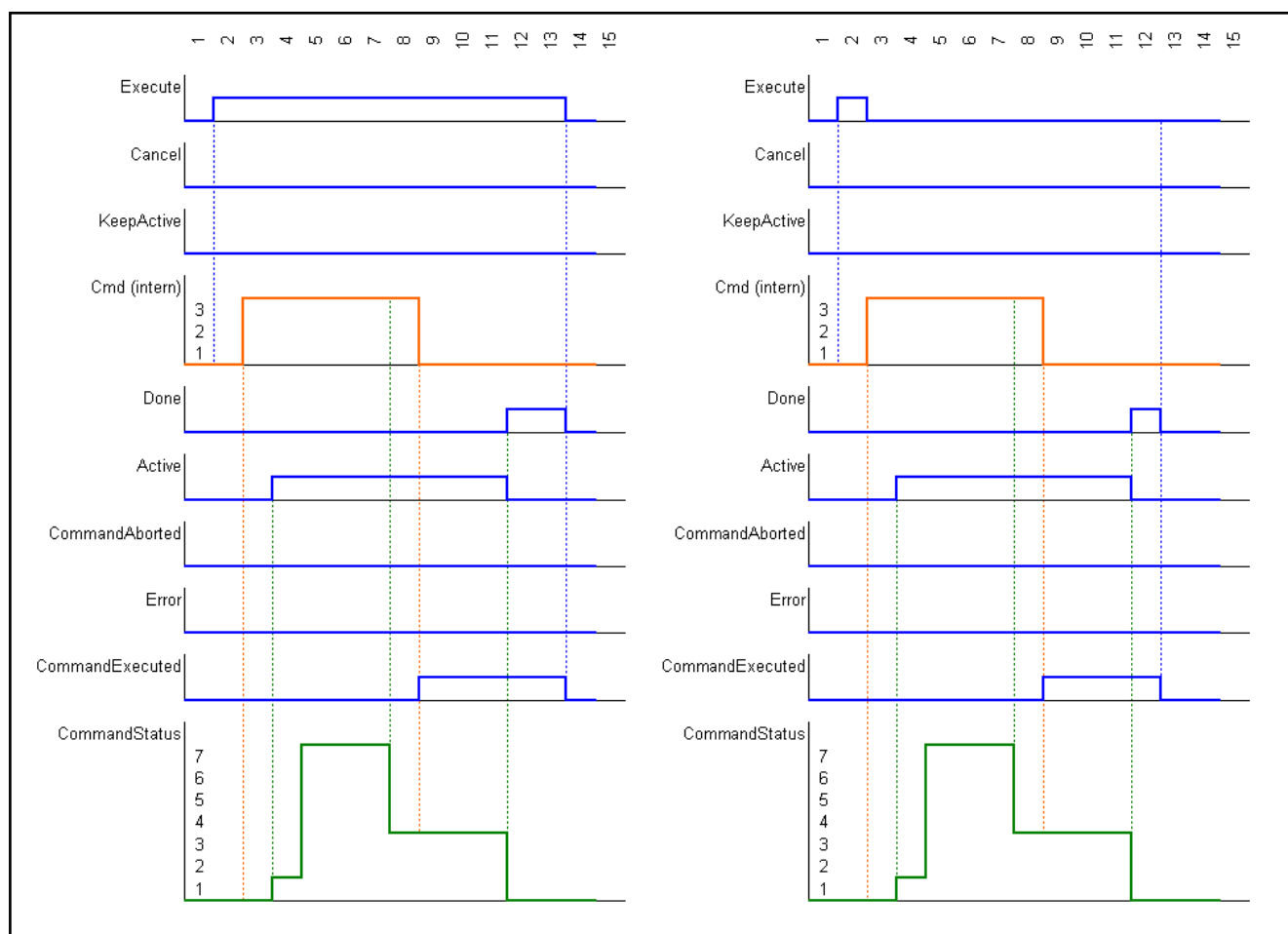


Abb. 11-4: Signal-Zeit-Diagramm für Signalverlauf "Execute-Done"

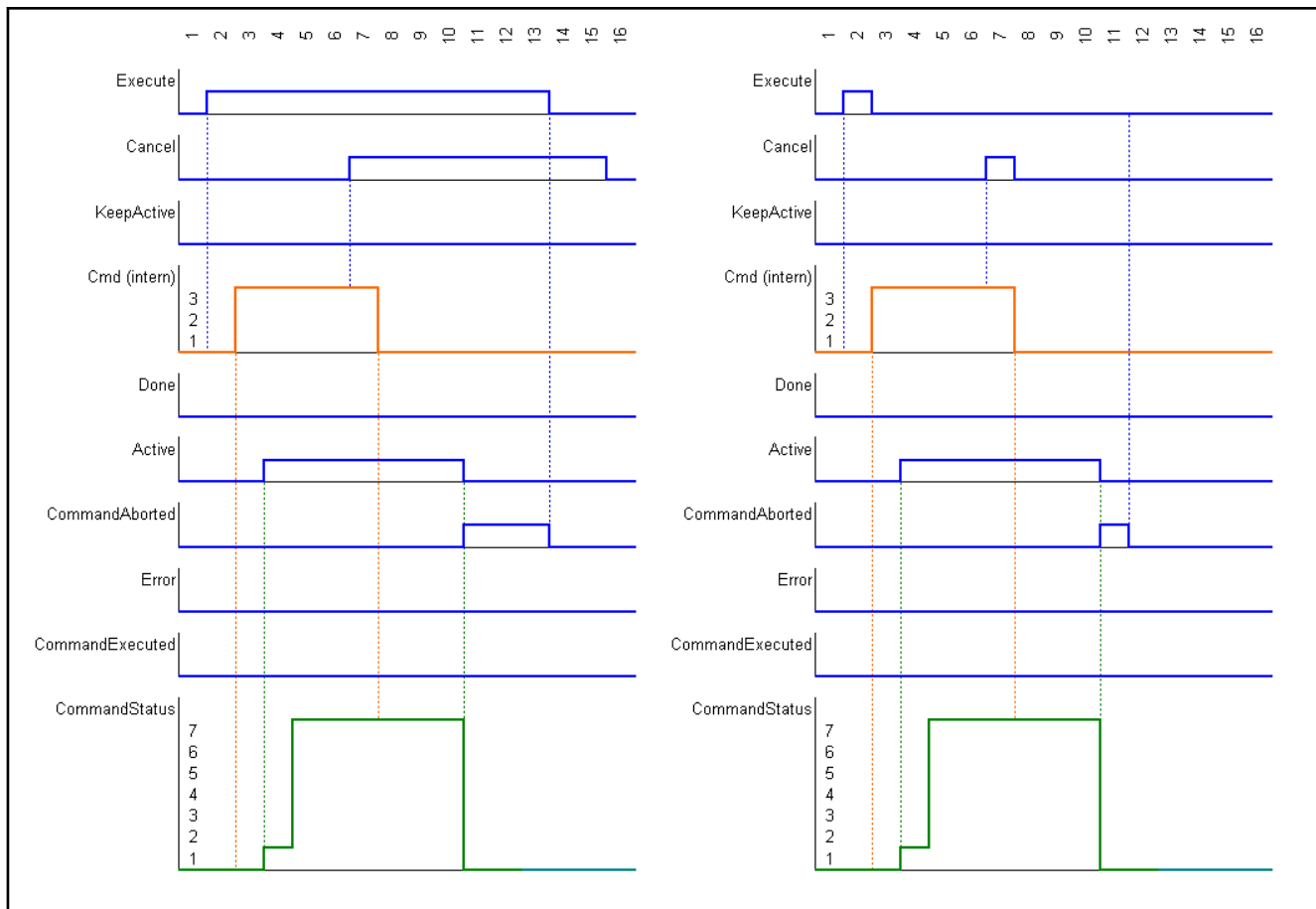


Abb. 11-5: Signal-Zeit-Diagramm für Signalverlauf "Execute-Cancel-CommandAborted"

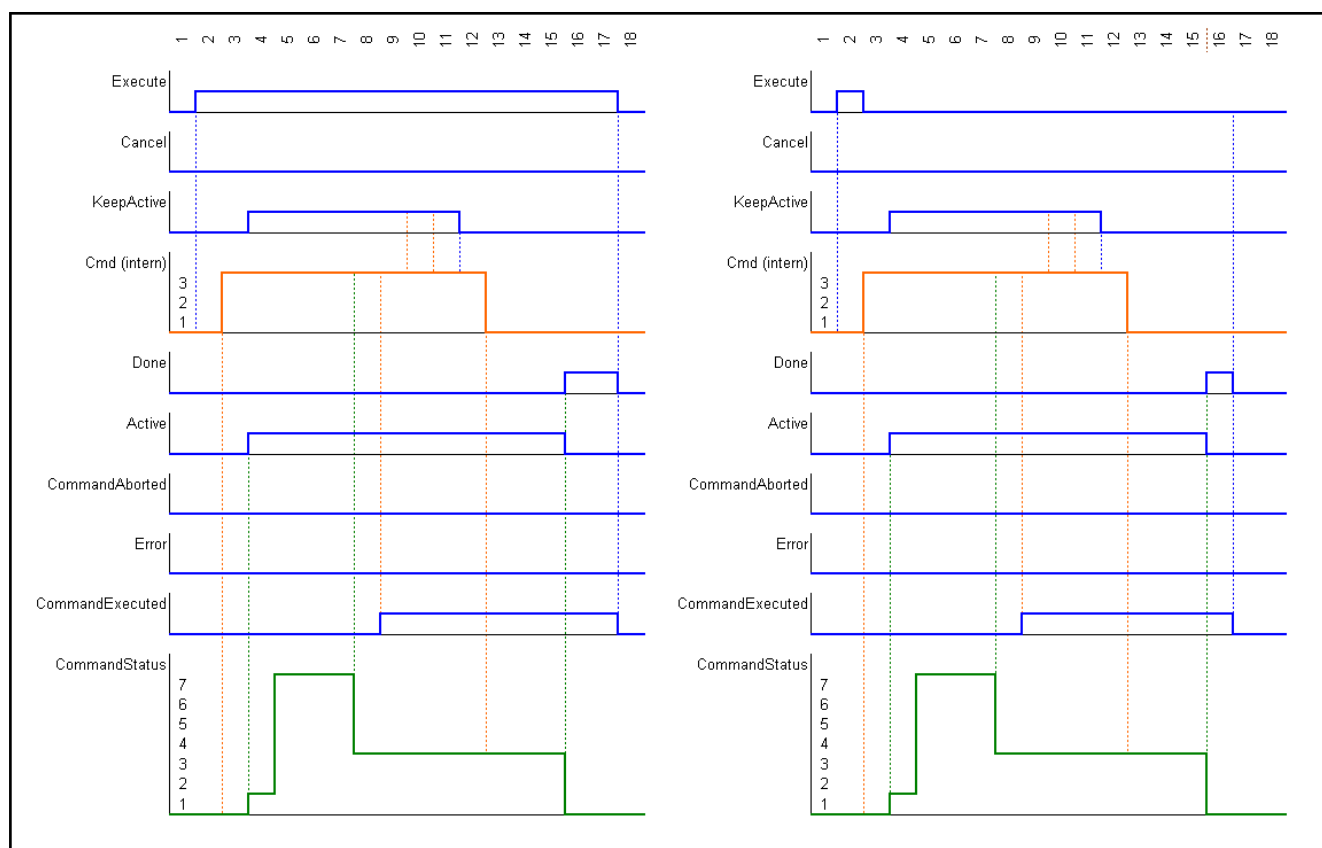


Abb. 11-6: Signal-Zeit-Diagramm für Signalverlauf "Execute-KeepActive-Done "

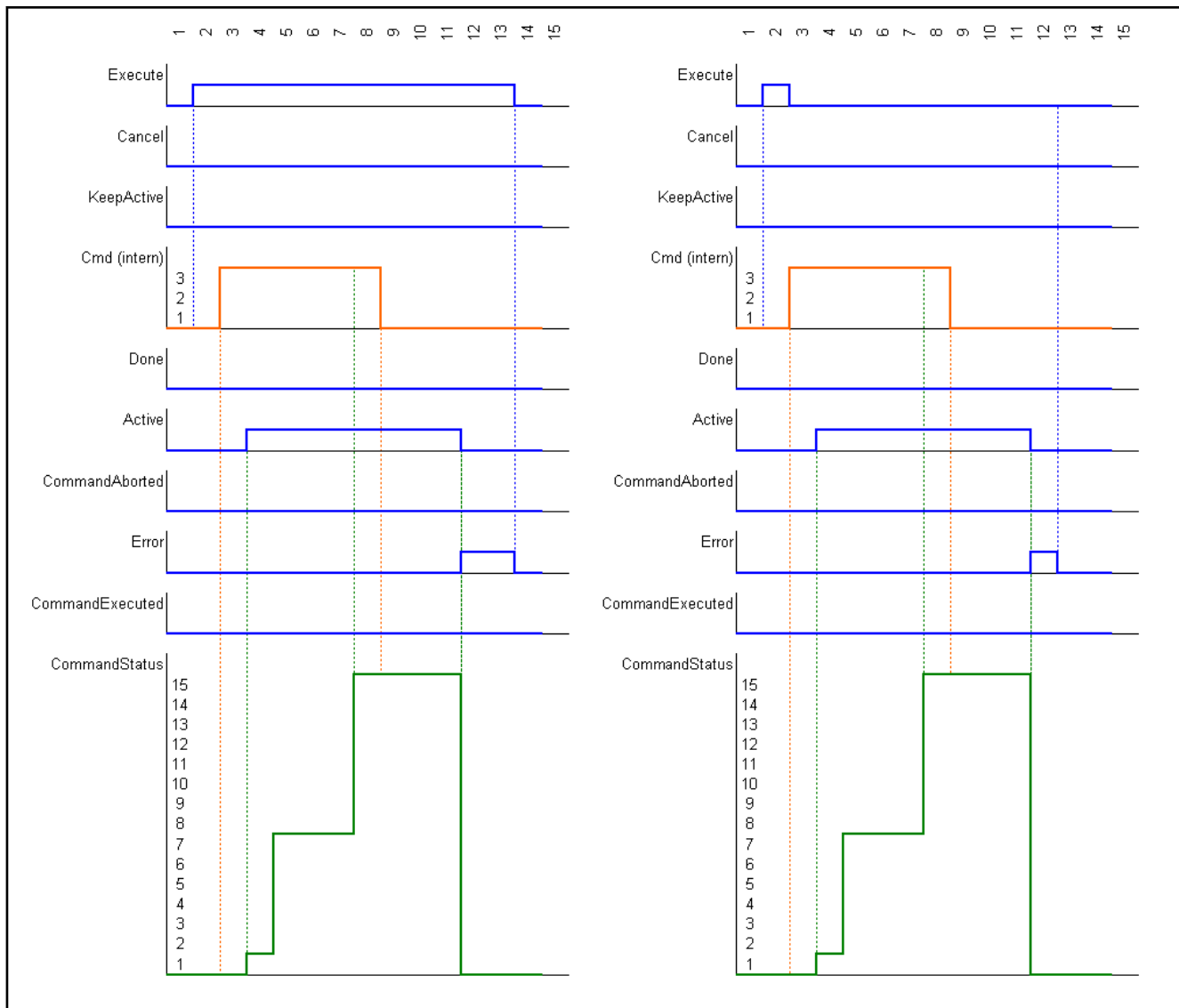


Abb. 11-7: Signal-Zeit-Diagramm für Signalverlauf "Execute-Error"

Beschreibung

Der Funktionsbaustein "IL_SIIICommand" startet nach der Bearbeitungsfreigabe mittels "Execute" die Kommandoausführung des über den Eingang "Idn" festgelegten Kommandos. Das Kommando wird im allgemeinen nach Abschluss der Bearbeitung automatisch zurückgesetzt.

Die Kommandoausführung kann, abhängig vom Kommando selbst, jederzeit über den Eingang "Cancel" abgebrochen werden.

Der Eingang "KeepActive" ermöglicht es, das Kommando nach erfolgreicher Ausführung weiter aktiv zu halten. Wird im Folgenden der Eingang "KeepActive" deaktiviert, wird das Kommando ebenfalls automatisch zurückgesetzt.

Der erfolgreiche Abschluss bzw. erfolgreiche Abbruch eines Kommandos wird über die Ausgänge "Done" bzw. "CommandAborted" für die Zeitdauer von mindestens einem Aufruf signalisiert.

Die aktive Bearbeitung des Kommandos wird über den Ausgang "Active" und das entsprechende Datum am Ausgang "CommandStatus" dargestellt.

Der aktuelle Status der Kommandoausführung wird permanent am Ausgang "CommandStatus" signalisiert, der Abschluss der Kommandoausführung wird durch den Ausgang "CommandExecuted" quittiert.

Tritt während der Bearbeitung des Kommandos ein Fehler auf, wird dies über den Ausgang "Error", sowie über "ErrorID", "ErrorIdnt" für die Zeitdauer von mindestens einem Aufruf signalisiert.

CommandStatus:

- 0 - Kommando nicht gesetzt und nicht freigegeben
- 1 - Kommando gesetzt und freigegeben
- 3 - Kommando korrekt abgearbeitet
- 5 - Kommandoabarbeitung unterbrochen
- 7 - Kommandoabarbeitung läuft
- 15 - Fehler, Kommandoabarbeitung unmöglich
- 256 - Wert des Parameters ausserhalb seines Min.- / Max.-Bereiches

Fehlerbehandlung

Der Funktionsbaustein nutzt zur Darstellung von Fehlern des Sercos-Service-Kanals die Fehlertabelle SERCOS_TABLE (ERROR_TABLE = 16#0010).

Zusätzlich sind Fehler der intern genutzten Funktionsbausteine IL_SIIISvcRead [Fehlerbehandlung, Seite 345](#), und IL_SIIISvcWrite [Fehlerbehandlung, Seite 349](#), und folgende Fehler des Bausteines in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additional2	
STATE_MACHINE_ERROR	16#00000006	State	Fehler in der Zustandsmaschine
INPUT_RANGE_ERROR	16#00002905	Wert des Attributes	Parameter ist kein Kommando

Tab. 11-13: Fehlercodes Funktionsbaustein IL_SIIICommand in F_RELATED_TABLE := 16#0170

Beispiel mit automatischem Rücksetzen

Ausführen Kommando S-0-0099

Im folgenden Beispiel wird das Sercos-Kommando S-0-0099.0.0, C0500 Reset Zustandsklasse 1, ausgeführt.

Variablendefinition

```
VAR
    fbSIIICommand:    IL_SIIICommand;
END_VAR
```

Programmcode S-0-0099 ausführen

```
fbSIIICommand.Execute:= TRUE;
fbSIIICommand.SercosAdr:= 1;
fbSIIICommand.Idn:= IL_SIIIElementsToIdn(IL_S_PARAM,0,0099,0,0);

fbSIIICommand();

IF fbSIIICommand.Done THEN
    ; (* command executed successfull *)
END_IF
```

Nach dem erfolgreichen Ausführen ist ein Fehler gelöscht (wenn die Fehlerursache vorher beseitigt wurde).

Beispiel mit Cancel

Ausführen Kommando S-0-0139

Im folgenden Beispiel wird das Sercos-Kommando S-0-0139.0.0, C1600 Kommando Parkende Achse, für 5 Sekunden ausgeführt und dann mit dem Eingang "Cancel" beendet.

Das Kommando kann nur in Sercos-Phase 2 oder in Parametriermodus PM ausgeführt werden.

Variablendefinition

```

VAR
    fbSIIICommand:    IL_SIIICommand;
    fbTON:            TON;
END_VAR

```

Programmcode S-0-0139 ausführen

```

fbSIIICommand.Execute:=    TRUE;
fbSIIICommand.SercosAdr:=  1;
fbSIIICommand.Idn:=        IL_SIIIElementsToIdn(IL_S_PARAM,0,0139,0,0);

IF fbSIIICommand.CommandStatus = IL_SERCOS_DATA_STATUS.IL_SERCOS_DATA_IN_PROCESS THEN
    (* command started successfull, start timer *)
    fbTON(IN:= TRUE, PT:= T#5S );
    (* Cancel the command after 5 seconds *)
    fbSIIICommand.Cancel := fbTON.Q;
END_IF

fbSIIICommand();

IF fbSIIICommand.Done THEN
    ; (* finished *)
END_IF

```

Nach dem erfolgreichen Ausführen ist das Kommando "Parkende Achse" wieder beendet.

Beispiel mit KeepActive**Ausführen Kommando S-0-0149**

Im folgenden Beispiel wird das Sercos-Kommando S-0-0149.0.0, C1300 Kommando Fahren auf Festanschlag, ausgeführt und dann durch Rücksetzen des Einganges "KeepActive" beendet.

Das Kommando kann nur mit Antriebsfreigabe ausgeführt werden.

Variablendefinition

```

VAR
    fbSIIICommand:    IL_SIIICommand;
    fbTON:            TON;
END_VAR

```

Programmcode S-0-0149 ausführen

```

fbSIIICommand.Execute:=    TRUE;
fbSIIICommand.SercosAdr:=  1;
fbSIIICommand.Idn:=        IL_SIIIElementsToIdn(IL_S_PARAM,0,0149,0,0);

IF fbSIIICommand.CommandStatus = IL_SERCOS_DATA_STATUS.IL_SERCOS_DATA_END_OK THEN
    (* command started successfull, start timer *)
    fbTON(IN:= TRUE, PT:= T#5S );
END_IF
(* Cancel the command after 5 seconds *)
fbSIIICommand.KeepActive := NOT fbTON.Q;

fbSIIICommand();

IF fbSIIICommand.Done THEN
    ; (* finished *)
END_IF

```

Nach dem erfolgreichen Ausführen ist das Kommando "Fahren auf Festanschlag" wieder beendet.

11.3.2 IL_SIIReadParameter

Kurzbeschreibung

Mit dem Funktionsbaustein IL_SIIReadParameter können über den Sercos III-Service-Kanal 2- oder 4Byte-Parameter von einem Slave-Teilnehmer gelesen werden. Der Parameterwert wird unter Beachtung des Attributes des Parameters in einen DINT-Wert gewandelt.



Der Ausgang Value (DINT) muss je nach Datentyp des Parameters in den richtigen SPS Datentyp konvertiert werden. Die Datengröße 2 oder 4 Byte muss nicht berücksichtigt werden, wenn auf der SPS immer 4Byte-Datentypen verwendet werden.

- Datentyp DEC (dezimal mit Vorzeichen): Value kann in eine DINT Variable kopiert werden
- Datentyp UDEC (dezimal ohne Vorzeichen): Value mit DINT_TO_UDINT in eine UDINT Variable konvertieren
- Datentypen BINARY, HEX und IDN: Value mit DINT_TO_DWORD in eine DWORD Variable konvertieren

Beispiele zu den jeweiligen Datentypen siehe unten.

Siehe auch Funktion [IL_SIIIAttribute](#), Seite 418, und Datentyp [IL_SIII_DATA_TYPE](#), Seite 427.

Am Ausgang "Attribute" liefert der Funktionsbaustein IL_SIIIReadParameter die notwendigen Informationen zur korrekten Konvertierung (Auswertung mit der Funktion IL_SIIIAttribute).

Zielsystem	Bibliothek/ Ordner
MLC	RIL_SercosIII/ AcyclicCommunication\ Parameter
MTX	RIL_SercosIII/ AcyclicCommunication\ Parameter
MLD (IL 2G)	RIL_SercosIII/ AcyclicCommunication\ Parameter

Tab. 11-14: Referenztablelle

Schnittstellenbeschreibung

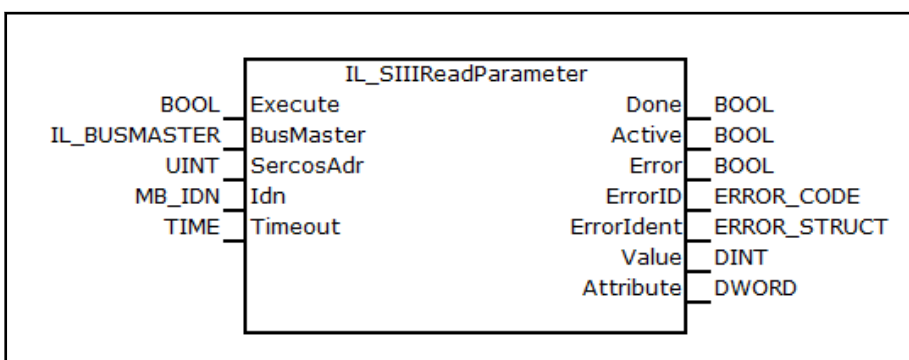


Abb. 11-8: Funktionsbaustein IL_SIIIReadParameter

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable BusMaster unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0. IL_BUSMASTER , Seite 429
	SercosAdr	UINT	Sercos III-Adresse des Slave-Teilnehmers.
	Idn	MB_IDN	IDN des Parameters. siehe Funktionen IL_SIIIElementsToldn , Seite 421, bzw. IL_SII-StringToldn , Seite 426.

E/A-Typ	Name	Datentyp	Beschreibung
	Timeout	TIME	Timeout-Überwachung des FBs in ms. T#0ms = Timeout-Überwachung ist nicht aktiv.
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig.
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig.
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen.
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall.
	ErrorIdent	ERROR_STRUCT	Detaildiagnose
	Value	DINT	Wert des Parameters
	Attribute	DWORD	Attribut des Parameters

Tab. 11-15: Schnittstelle IL_SIIIReadParameter

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	USINT	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
SercosAdr	UINT	0 (MLD: lokale Achse) 1 (andere Systeme)	511	0	Steigende Flanke an Execute
Idn	MB_IDN	0	16#FFFFFFFF	0	Steigende Flanke an Execute
Timeout	TIME	T#0s	T#60 min	T#2s	Steigende Flanke an Execute

Tab. 11-16: Min-/Max und Default-Werte der Eingänge IL_SIIIReadParameter

Fehlerbehandlung

Der Funktionsbaustein nutzt zur Darstellung von Fehlern des Sercos-Service-Kanals die Fehlertabelle SERCOS_TABLE (ERROR_TABLE = 16#0010).

Zusätzlich sind Fehler des intern genutzten Funktionsbausteines IL_SIIISvc-Read, [Fehlerbehandlung, Seite 345](#), und folgende Fehler des Bausteines in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additional2	
STATE_MACHINE_ERROR	16#00000006	State	Fehler in der Zustandsmaschine
INPUT_RANGE_ERROR	16#00002900	Wert des Attributes	Datenlänge 8 Byte Parameter oder Liste kann nicht gewandelt werden
INPUT_RANGE_ERROR	16#00002904	Wert des Attributes	Nur Parametertypen UDEC, DEC, BINARY, HEX oder IDN können gewandelt werden.

Tab. 11-17: Fehlercodes Funktionsbaustein IL_SIIIReadParameter in F_RELATED_TABLE := 16#0170

Beispiel für Datentyp DEC (dezimal mit Vorzeichen)**Lesen S-0-0051**

Im folgenden Beispiel wird der Sercos-Parameter S-0-0051.0.0, Lage-Istwert Geber 1 gelesen.

Variablendefinition

```

VAR
    fbSIIIReadParameter:    IL_SIIIReadParameter;
    diValue: DINT;
END_VAR

```

Programmcode S-0-0051 lesen

```

fbSIIIReadParameter.Execute:= TRUE;
fbSIIIReadParameter.SercosAdr:= 1;
fbSIIIReadParameter.Idn:=IL_SIIIElementsToIdn(IL_S_PARAM,0,0051,0,0);

fbSIIIReadParameter();

IF fbSIIIReadParameter.Done = TRUE THEN
    (* S-0-0051.0.0 is signed decimal -> copy direct *)
    diValue := fbSIIIReadParameter.Value;
END_IF

```

Nach dem erfolgreichen Lesen steht in diValue zum Beispiel der Wert "1234567".

Beispiel für Datentyp UDEC (dezimal ohne Vorzeichen)

Lesen S-0-0091

Im folgenden Beispiel wird der Sercos-Parameter S-0-0091.0.0, Geschwindigkeits-Grenzwert bipolar gelesen.

Variablendefinition

```

VAR
    fbSIIIReadParameter:    IL_SIIIReadParameter;
    udiValue: UDINT;
END_VAR

```

Programmcode S-0-0091 lesen

```

fbSIIIReadParameter.Execute:= TRUE;
fbSIIIReadParameter.SercosAdr:= 1;
fbSIIIReadParameter.Idn:=IL_SIIIElementsToIdn(IL_S_PARAM,0,0091,0,0);

fbSIIIReadParameter();

IF fbSIIIReadParameter.Done = TRUE THEN
    (* S-0-0091.0.0 is unsigned decimal -> convert *)
    udiValue := DINT_TO_UDINT(fbSIIIReadParameter.Value);
END_IF

```

Nach dem erfolgreichen Lesen steht in udiValue zum Beispiel der Wert "10000000".

Beispiel für Datentyp BINARY (gilt auch für HEX und IDN)

Lesen S-0-0135

Im folgenden Beispiel wird der Sercos-Parameter S-0-0135.0.0, Antriebs-Status gelesen.

Variablendefinition

```

VAR
    fbSIIIReadParameter:    IL_SIIIReadParameter;
    wValue: WORD;
END_VAR

```

Programmcode S-0-0135 lesen

```

fbSIIIReadParameter.Execute:= TRUE;
fbSIIIReadParameter.SercosAdr:= 1;
fbSIIIReadParameter.Idn:= IL_SIIIElementsToIdn(IL_S_PARAM,0,0135,0,0);

fbSIIIReadParameter();

IF fbSIIIReadParameter.Done = TRUE THEN
    (* S-0-0135.0.0 is binary -> convert *)

```

**Beispiel für Lesen mit Auswertung
des Datentypes**

```
wValue := DINT_TO_WORD(fbSIIIReadParameter.Value);
END_IF
```

Nach dem erfolgreichen Lesen steht in wValue zum Beispiel der Wert "16#4000".

Lesen S-0-0390 mit Auswertung des Attributes

Im folgenden Beispiel wird der Sercos-Parameter S-0-0390.0.0, Diagnose-Nummer gelesen. Die Auswertung des Attributes ist hier eigentlich nicht notwendig, da der Datentyp des S-0-0390 bekannt ist. Das Beispiel soll zeigen, wie eine Auswertung beim Lesen von Parametern mit verschiedenen Attributen aussehen könnte.

Variablendefinition

```
VAR
    fbSIIIReadParameter:    IL_SIIIReadParameter;
    diValue: DINT;
    iValue: INT;
    udiValue: UDINT;
    uiValue: UINT;
    dwValue: DWORD;
    wValue: WORD;
    enDataType:             IL_SIII_DATA_TYPE;
    enDataLength:           IL_SIII_DATA_LENGTH;
END_VAR
```

Programmcode S-0-0390 lesen mit Auswertung des Attributes

```
fbSIIIReadParameter.Execute:= TRUE;
fbSIIIReadParameter.SercosAdr:= 1;
fbSIIIReadParameter.Idn:= IL_SIIIElementsToIdn(IL_S_PARAM,0,0390,0,0);

fbSIIIReadParameter();

IF fbSIIIReadParameter.Done = TRUE THEN
    IL_SIIIAttribute(Attribute:= fbSIIIReadParameter.Attribute ,
                    DecPoints=> ,
                    DataType=>enDataType ,
                    DataLength=>enDataLength );
    CASE enDataType OF
        IL_SIII_DATA_TYPE.DEC:
            (* signed decimal *)
            IF enDataLength = IL_SIII_DATA_LENGTH.FIX_4_BYTE THEN
                diValue := fbSIIIReadParameter.Value;
            ELSE (* 2 byte *)
                iValue:= DINT_TO_INT(fbSIIIReadParameter.Value);
            END_IF

        IL_SIII_DATA_TYPE.UDEC:
            (* unsigned decimal *)
            IF enDataLength = IL_SIII_DATA_LENGTH.FIX_4_BYTE THEN
                udiValue:= DINT_TO_UDINT(fbSIIIReadParameter.Value);
            ELSE (* 2 byte *)
                uiValue:= DINT_TO_UINT(fbSIIIReadParameter.Value);
            END_IF

        IL_SIII_DATA_TYPE.BINARY,
        IL_SIII_DATA_TYPE.HEX,
        IL_SIII_DATA_TYPE.IDN :
            (* all in (D)WORD *)
            IF enDataLength = IL_SIII_DATA_LENGTH.FIX_4_BYTE THEN
                dwValue := DINT_TO_DWORD(fbSIIIReadParameter.Value);
            ELSE (* 2 byte *)
                wValue := DINT_TO_WORD(fbSIIIReadParameter.Value);
            END_IF

        ELSE
            ; (* not possible (would be fbSIIIReadParameter.Error) *)
        END_CASE
    END_IF
```

Nach dem erfolgreichen Lesen steht in dwValue zum Beispiel der Wert "16#000A0013".

11.3.3 IL_SIIIReadRealParameter

Kurzbeschreibung Mit dem Funktionsbaustein IL_SIIIReadRealParameter können über den Sercos III-Service-Kanal 2- oder 4Byte-Parameter von einem Slave-Teilnehmer gelesen werden. Der Parameterwert wird unter Beachtung des Attributes in einen REAL-Wert gewandelt.

Zielsystem	Bibliothek/ Ordner
MLC	RIL_SercosIII/ AcyclicCommunication\ Parameter
MTX	RIL_SercosIII/ AcyclicCommunication\ Parameter
MLD (IL 2G)	RIL_SercosIII/ AcyclicCommunication\ Parameter

Tab. 11-18: Referenztablelle

Schnittstellenbeschreibung

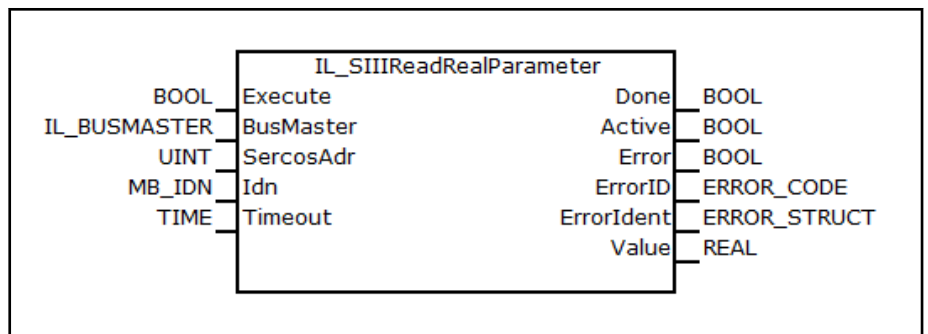


Abb. 11-9: Funktionsbaustein IL_SIIIReadRealParameter

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable BusMaster unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0 IL_BUSMASTER, Seite 429
	SercosAdr	UINT	Sercos III-Adresse des Slave-Teilnehmers
	Idn	MB_IDN	IDN des Parameters. siehe Funktionen IL_SIIIElementsToldn, Seite 421 , bzw. IL_SIIStringToldn, Seite 426 .
	Timeout	TIME	Timeout-Überwachung des FBs in ms. T#0ms = Timeout-Überwachung ist nicht aktiv
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCT	Detaildiagnose
	Value	REAL	Wert des Parameters

Tab. 11-19: Schnittstelle IL_SIIIReadRealParameter

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	USINT	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
SercosAdr	UINT	0 (MLD: lokale Achse) 1 (andere Systeme)	511	0	Steigende Flanke an Execute
Idn	MB_IDN	0	16#FFFFFFFF	0	Steigende Flanke an Execute
Timeout	TIME	T#0s	T#60 min	T#2s	Steigende Flanke an Execute

Tab. 11-20: Min-/Max und Default-Werte der Eingänge IL_SIIReadRealParameter

Fehlerbehandlung

Der Funktionsbaustein nutzt zur Darstellung von Fehlern des Sercos-Service-Kanals die Fehlertabelle SERCOS_TABLE (ERROR_TABLE = 16#0010).

Zusätzlich sind Fehler des intern genutzten Funktionsbausteines IL_SIIISvcRead, [Fehlerbehandlung, Seite 345](#), und folgende Fehler des Bausteines in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additional2	
STATE_MACHINE_ERROR	16#00000006	State	Fehler in der Zustandsmaschine
INPUT_RANGE_ERROR	16#00002900	Wert des Attributes	Datenlänge 8 Byte Parameter oder Liste kann nicht gewandelt werden
INPUT_RANGE_ERROR	16#00002901	Wert des Attributes	Nur Parametertypen UDEC, DEC oder FLOAT können gewandelt werden.

Tab. 11-21: Fehlercodes Funktionsbaustein IL_SIIReadRealParameter in F_RELATED_TABLE := 16#0170

Beispiel Lesen S-0-0051

Im folgenden Beispiel wird der Sercos-Parameter S-0-0051.0.0, Lage-Istwert Geber 1 gelesen.

Variablendefinition

```
VAR
    fbSIIReadRealParameter:    IL_SIIReadRealParameter;
    rValue:    REAL;
END_VAR
```

Programmcode S-0-0051 lesen

```
fbSIIReadRealParameter.Execute:=    TRUE;
fbSIIReadRealParameter.SercosAdr:=    1;
fbSIIReadRealParameter.Idn:=    IL_SIIElementsToIdn(IL_S_PARAM,0,0051,0,0);

fbSIIReadRealParameter();

IF fbSIIReadRealParameter.Done = TRUE THEN
    rValue := fbSIIReadRealParameter.Value;
END_IF
```

Nach dem erfolgreichen Lesen steht in rValue zum Beispiel der Wert "123.4567".

11.3.4 IL_SIIIReadStringParameter

Kurzbeschreibung Mit dem Funktionsbaustein IL_SIIIReadStringParameter können über den Sercos III-Service-Kanal Textlisten von einem Slave-Teilnehmer gelesen werden.

Zielsystem	Bibliothek/ Ordner
MLC	RIL_SercosIII/ AcyclicCommunication\ Parameter
MTX	RIL_SercosIII/ AcyclicCommunication\ Parameter
MLD (IL 2G)	RIL_SercosIII/ AcyclicCommunication\ Parameter

Tab. 11-22: Referenztablelle

Schnittstellenbeschreibung

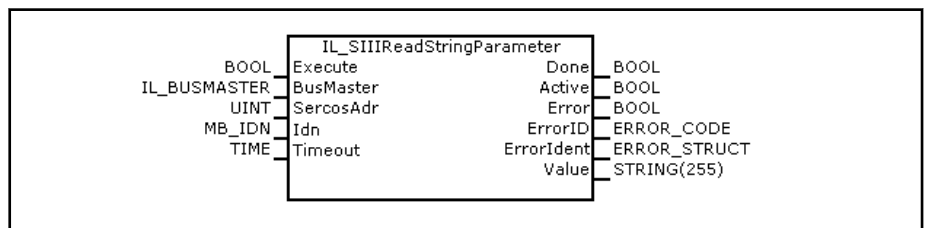


Abb. 11-10: Funktionsbaustein IL_SIIIReadStringParameter

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable BusMaster unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0. IL_BUSMASTER, Seite 429
	SercosAdr	UINT	Sercos III-Adresse des Slave-Teilnehmers.
	Idn	MB_IDN	IDN des Parameters siehe Funktionen IL_SIIIElementsToldn, Seite 421 , bzw. IL_SII-StringToldn, Seite 426 .
	Timeout	TIME	Timeout-Überwachung des FBs in ms. T#0ms = Timeout-Überwachung ist nicht aktiv.
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig.
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig.
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen.
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall.
	ErrorIdent	ERROR_STRUCT	Detaildiagnose
	Value	STRING(255)	Zeichenkette mit maximal 255 Zeichen

Tab. 11-23: Schnittstelle IL_SIIIReadStringParameter

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	USINT	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
SercosAdr	UINT	0 (MLD: lokale Achse) 1 (andere Systeme)	511	0	Steigende Flanke an Execute
Idn	MB_IDN	0	16#FFFFFFFF	0	Steigende Flanke an Execute
Timeout	TIME	T#0s	T#60 min	T#2s	Steigende Flanke an Execute

Tab. 11-24: Min-/Max und Default-Werte der Eingänge IL_SIIIReadStringParameter

Fehlerbehandlung

Der Funktionsbaustein nutzt zur Darstellung von Fehlern des Sercos-Service-Kanals die Fehlertabelle SERCOS_TABLE (ERROR_TABLE = 16#0010).

Zusätzlich sind Fehler des intern genutzten Funktionsbausteines IL_SIIISvcRead, [Fehlerbehandlung, Seite 345](#), und folgende Fehler des Bausteines in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additional2	
STATE_MACHINE_ERROR	16#00000006	State	Fehler in der Zustandsmaschine
INPUT_RANGE_ERROR	16#00002902	Wert des Attributes	Zeichenkette im Parameter ist zu lang (max. 254 byte + \0)
INPUT_RANGE_ERROR	16#00002903	Wert des Attributes	Nur Parameter mit Datentyp Text und Länge ein Byte variable Liste können gelesen werden

Tab. 11-25: Fehlercodes Funktionsbaustein IL_SIIIReadStringParameter in F_RELATED_TABLE := 16#0170

Beispiel Lesen S-0-0095

Im folgenden Beispiel wird der Sercos-Parameter S-0-0095.0.0, Diagnose, gelesen.

Variablendefinition

```
VAR
    fbSIIIReadStringParameter: IL_SIIIReadStringParameter;
    strValue: STRING;
END_VAR
```

Programmcode S-0-0095 lesen

```
fbSIIIReadStringParameter.Execute:= TRUE;
fbSIIIReadStringParameter.SercosAdr:= 1;
fbSIIIReadStringParameter.Idn:= IL_SIIIElementsToIdn(IL_S_PARAM,0,0095,0,0);

fbSIIIReadStringParameter();

IF fbSIIIReadStringParameter.Done = TRUE THEN
    strValue := fbSIIIReadStringParameter.Value;
END_IF
```

Nach dem erfolgreichen Lesen steht in strValue zum Beispiel der Wert "A0013 Bereit zur Leistungszuschaltung".

11.3.5 IL_SIIWriteParameter

Kurzbeschreibung Mit dem Funktionsbaustein IL_SIIWriteParameter können über den Sercos III-Service-Kanal 2- oder 4Byte-Parameter eines Slave-Teilnehmers geschrieben werden.



Der Eingang Value (DINT) muss je nach Datentyp des Parameters konvertiert werden.

Die Datenlänge 2 oder 4 Byte muss nicht berücksichtigt werden, wenn auf der SPS immer 4Byte-Datentypen verwendet werden.

- Datentyp DEC (dezimal mit Vorzeichen): DINT Variable kann in Value kopiert werden
- Datentyp UDEC (dezimal ohne Vorzeichen): UDINT Variable mit UDINT_TO_DINT konvertieren und in Value kopieren
- Datentypen BINARY, HEX und IDN: DWORD Variable mit DWORD_TO_DINT konvertieren und in Value kopieren

Beispiele zu den jeweiligen Datentypen siehe unten.

Siehe auch Funktion [IL_SIIAttribute](#), Seite 418, und Datentyp [IL_SII_DATA_TYPE](#), Seite 427.

Zielsystem	Bibliothek/ Ordner
MLC	RIL_SercosIII/ AcyclicCommunication\ Parameter
MTX	RIL_SercosIII/ AcyclicCommunication\ Parameter
MLD (IL 2G)	RIL_SercosIII/ AcyclicCommunication\ Parameter

Tab. 11-26: Referenztablelle

Schnittstellenbeschreibung

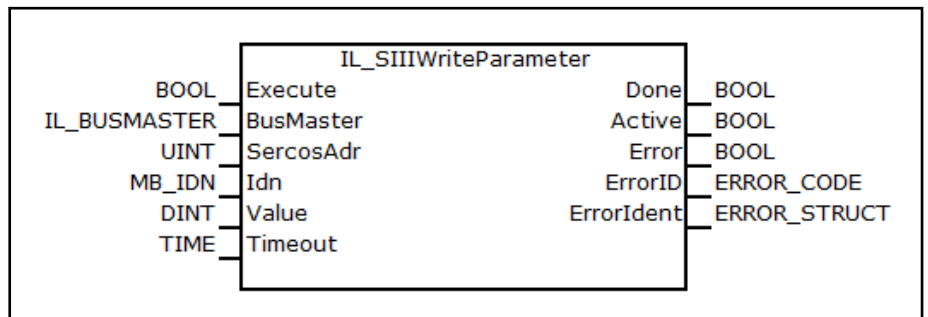


Abb. 11-11: Funktionsbaustein IL_SIIWriteParameter

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable BusMaster unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0. IL_BUSMASTER , Seite 429
	SercosAdr	UINT	Sercos III-Adresse des Slave-Teilnehmers.

E/A-Typ	Name	Datentyp	Beschreibung
	Idn	MB_IDN	IDN des Parameters siehe Funktionen IL_SIIIElementsToldn , Seite 421, bzw. IL_SII-StringToldn , Seite 426.
	Value	DINT	Wert, der in den Parameter geschrieben werden soll
	Timeout	TIME	Timeout-Überwachung des FBs in ms. T#0ms = Timeout-Überwachung ist nicht aktiv.
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig.
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig.
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen.
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall.
	ErrorIdent	ERROR_STRUCT	Detailldiagnose

Tab. 11-27: Schnittstelle IL_SIIWriteParameter

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	USINT	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
SercosAdr	UINT	0 (MLD: lokale Achse) 1 (andere Systeme)	511	0	Steigende Flanke an Execute
Idn	MB_IDN	0	16#FFFFFFFF	0	Steigende Flanke an Execute
Value	DINT	Abhängig vom Datentyp des Parameters	Abhängig vom Datentyp des Parameters	0	Steigende Flanke an Execute
Timeout	TIME	T#0s	T#60 min	T#2s	Steigende Flanke an Execute

Tab. 11-28: Min-/Max und Default-Werte der Eingänge IL_SIIWriteParameter

Fehlerbehandlung

Der Funktionsbaustein nutzt zur Darstellung von Fehlern des Sercos-Service-Kanals die Fehlertabelle SERCOS_TABLE (ERROR_TABLE = 16#0010).

Zusätzlich sind Fehler des intern genutzten Funktionsbausteines IL_SIIISvc-Read, [Fehlerbehandlung](#), Seite 345, und folgende Fehler des Bausteines in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additional2	
STATE_MACHINE_ERROR	16#00000006	State	Fehler in der Zustandsmaschine
INPUT_RANGE_ERROR	16#00002900	Wert des Attributes	Datenlänge 8 Byte Parameter oder Liste kann nicht gewandelt werden
INPUT_RANGE_ERROR	16#00002904	Wert des Attributes	Nur Parametertypen UDEC, DEC, HEX, BINARY oder IDN können gewandelt werden.

Tab. 11-29: Fehlercodes Funktionsbaustein IL_SIIWriteParameter in F_RELATED_TABLE := 16#0170

Beispiel für Datentyp DEC (dezimal mit Vorzeichen)

Schreiben S-0-0258

Im folgenden Beispiel wird der Sercos-Parameter S-0-0258.0.0, Zielposition, mit dem Wert -42,1234 beschrieben.

Variablendefinition

```
VAR
    fbSIIWriteParameter:    IL_SIIWriteParameter;
    diValue: DINT;
END_VAR
```

Programmcode S-0-0258 schreiben

```
fbSIIWriteParameter.Execute:= TRUE;
fbSIIWriteParameter.SercosAdr:= 1;
fbSIIWriteParameter.Idn:= IL_SIIElementsToIdn(IL_S_PARAM,0,0258,0,0);
diValue := REAL_TO_DINT(-42.1234*10000); (* *10000 because of 4 decimal places
                                           (rotatoric, preferred scaling)
                                           S-0-0051.0.0 is signed decimal -> copy direct *)
fbSIIWriteParameter.Value := diValue;

fbSIIWriteParameter();

IF fbSIIWriteParameter.Done = TRUE THEN
    ; (* .... *)
END_IF
```

Nach dem erfolgreichen Schreiben steht im Parameter S-0-0258 zum Beispiel der Wert "-42.1234".

Beispiel für Datentyp UDEC (dezimal ohne Vorzeichen)

Schreiben S-0-0091

Im folgenden Beispiel wird der Sercos-Parameter S-0-0091.0.0, Geschwindigkeits-Grenzwert, mit dem Wert 1042,5678 beschrieben.

Variablendefinition

```
VAR
    fbSIIWriteParameter:    IL_SIIWriteParameter;
    udiValue: UDINT;
END_VAR
```

Programmcode S-0-0091 schreiben

```
fbSIIWriteParameter.Execute:= TRUE;
fbSIIWriteParameter.SercosAdr:= 1;
fbSIIWriteParameter.Idn:= IL_SIIElementsToIdn(IL_S_PARAM,0,0091,0,0);
udiValue := REAL_TO_UDINT(1042.5678*10000); (* 10000 because of 4 decimal places
                                           (rotatoric, preferred scaling)
                                           S-0-0091.0.0 is unsigned decimal -> convert from UDINT to DINT *)
fbSIIWriteParameter.Value := UDINT_TO_DINT(udiValue);

fbSIIWriteParameter();

IF fbSIIWriteParameter.Done = TRUE THEN
    ; (* .... *)
END_IF
```

Nach dem erfolgreichen Schreiben steht im Parameter S-0-0091 zum Beispiel der Wert "1042.5678".

Beispiel für Datentyp BINARY (gilt auch für HEX und IDN)

Schreiben S-0-0169

Im folgenden Beispiel wird der Sercos-Parameter S-0-0169.0.0, Messtaster-Steuerparameter, mit dem Wert 0b0000.0001.0110.0011 beschrieben.

Variablendefinition

```
VAR
    fbSIIWriteParameter:    IL_SIIWriteParameter;
    wValue: WORD;
END_VAR
```

Programmcode S-0-0169 lesen

```
fbSIIWriteParameter.Execute:=    TRUE;
fbSIIWriteParameter.SercosAdr:=  1;
fbSIIWriteParameter.Idn:=        IL_SIIElementsToIdn(IL_S_PARAM,0,0169,0,0);
wValue := 2#0000000101100011;
(* S-0-0169.0.0 is 2 byte BINARY -> convert from WORD to DINT *)
fbSIIWriteParameter.Value := WORD_TO_DINT(wValue);

fbSIIWriteParameter();

IF fbSIIWriteParameter.Done = TRUE THEN
    ; (* .... *)
END_IF
```

Nach dem erfolgreichen Schreiben steht im Parameter S-0-0169 zum Beispiel der Wert "0b0000.0001.0110.0011".

11.3.6 IL_SIIWriteRealParameter

Kurzbeschreibung Mit dem Funktionsbaustein IL_SIIWriteRealParameter können über den Sercos III-Service-Kanal 2- oder 4Byte-Parameter eines Slave-Teilnehmers geschrieben werden. Der Wert wird unter Beachtung des Attributes in einen REAL-Wert gewandelt und in den Parameter geschrieben.

Zielsystem	Bibliothek/ Ordner
MLC	RIL_SercosIII/ AcyclicCommunication\ Parameter
MTX	RIL_SercosIII/ AcyclicCommunication\ Parameter
MLD (IL 2G)	RIL_SercosIII/ AcyclicCommunication\ Parameter

Tab. 11-30: Referenztablelle

Schnittstellenbeschreibung

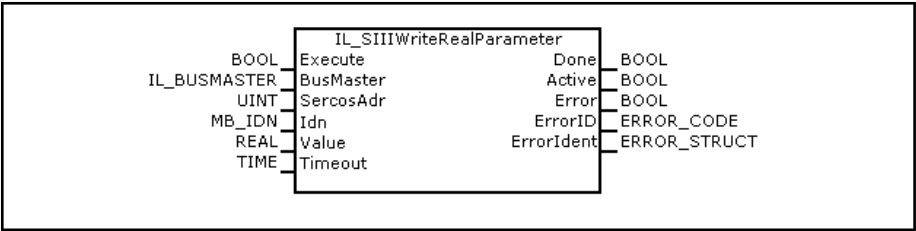


Abb. 11-12: Funktionsbaustein IL_SIIWriteRealParameter

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable BusMaster unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0. IL_BUSMASTER, Seite 429

E/A-Typ	Name	Datentyp	Beschreibung
	SercosAdr	UINT	Sercos III-Adresse des Slave-Teilnehmers.
	Idn	MB_IDN	IDN des Parameters. Siehe Funktionen IL_SIIIElementsToldn , Seite 421, bzw. IL_SII-StringToldn , Seite 426.
	Value	REAL	Wert, der in den Parameter geschrieben werden soll
	Timeout	TIME	Timeout-Überwachung des FBs in ms. T#0ms = Timeout-Überwachung ist nicht aktiv.
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig.
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig.
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen.
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall.
	ErrorIdent	ERROR_STRUCT	Detaildiagnose

Tab. 11-31: Schnittstelle IL_SIIWriteRealParameter

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	USINT	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
SercosAdr	UINT	0 (MLD: lokale Achse) 1 (andere Systeme)	511	0	Steigende Flanke an Execute
Idn	MB_IDN	0	16#FFFFFFF	0	Steigende Flanke an Execute
Value	REAL	Abhängig von Datentyp und Dezimalstellen des Parameters	Abhängig von Datentyp und Dezimalstellen des Parameters	0	Steigende Flanke an Execute
Timeout	TIME	T#0s	T#60 min	T#2s	Steigende Flanke an Execute

Tab. 11-32: Min-/Max und Default-Werte der Eingänge IL_SIIWriteRealParameter

Fehlerbehandlung

Der Funktionsbaustein nutzt zur Darstellung von Fehlern des Sercos-Service-Kanals die Fehlertabelle SERCOS_TABLE (ERROR_TABLE = 16#0010).

Zusätzlich sind Fehler der intern genutzten Funktionsbausteine IL_SIIISvcRead [Fehlerbehandlung](#), Seite 345, und IL_SIIISvcWrite [Fehlerbehandlung](#), Seite 349, und folgende Fehler des Bausteines in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additional2	
STATE_MACHINE_ERROR	16#00000006	State	Fehler in der Zustandsmaschine
INPUT_RANGE_ERROR	16#00002900	Wert des Attributes	Datenlänge 8 Byte Parameter oder Liste kann nicht gewandelt werden
INPUT_RANGE_ERROR	16#00002901	Wert des Attributes	Nur Parametertypen UDEC, DEC oder FLOAT können gewandelt werden.

Tab. 11-33: Fehlercodes Funktionsbaustein IL_SIIWriteRealParameter in F_REALATED_TABLE := 16#0170

Beispiel Schreiben S-0-0038

Im folgenden Beispiel wird der Sercos-Parameter S-0-0038.0.0, Geschwindigkeits-Grenzwert positiv, mit dem Wert 123,456 beschrieben.

Variablendefinition

```
VAR
    fbSIIWriteRealParameter:    IL_SIIWriteRealParameter;
END_VAR
```

Programmcode S-0-0038 schreiben

```
fbSIIWriteRealParameter.Execute:=    TRUE;
fbSIIWriteRealParameter.SercosAdr:=  1;
fbSIIWriteRealParameter.Idn:=        IL_SIIElementsToIdn(IL_S_PARAM,0,0038,0,0);
fbSIIWriteRealParameter.Value:=      123.456;

fbSIIWriteRealParameter();

IF fbSIIWriteRealParameter.Done THEN
    ; // finished write successfull
END_IF
```

Nach dem erfolgreichen Schreiben steht im Parameter S-0-0038 zum Beispiel der Wert "123.4567".

11.3.7 IL_SERCOS_DATA_STATUS

Kurzbeschreibung Der Status eines durch den Funktionsbaustein IL_SIICommand, RIL_SercosIII.library, gestarteten Kommandos wird mittels ENUM-Typ "IL_SERCOS_DATA_STATUS" als Klartext lesbar angezeigt.

Zielsystem	Bibliothek / Ordner
MLC	RIL_SercosIII / AcyclicCommunication / Parameter
MTX	RIL_SercosIII / AcyclicCommunication / Parameter
MLC (IL 2G)	RIL_SercosIII / AcyclicCommunication / Parameter

Tab. 11-34: Bibliothekszuordnung

Programm:

```
TYPE IL_SERCOS_DATA_STATUS:
(* Status of the parameter *)
(
    IL_SERCOS_DATA_NOT_SET      := 0,    (* Not set and not enabled *)
    IL_SERCOS_DATA_SETTING     := 1,    (* Set and enabled *)
    IL_SERCOS_DATA_END_OK      := 3,    (* Command correctly executed *)
    IL_SERCOS_DATA_INTERRUPTED := 5,    (* Command execution interrupted *)
    IL_SERCOS_DATA_IN_PROCESS  := 7,    (* In process *)
    IL_SERCOS_DATA_BREAK_ERROR := 15,   (* Error, command execution impossible *)
    IL_SERCOS_DATA_DATUM_FAULT := 256   (* Data invalid *)
);
END_TYPE
```




Der Datenstatus "IL_SERCOS_DATA_DATUM_FAULT" besagt, dass der Wert des Parameters außerhalb seiner Gültigkeitsgrenzen liegt, d.h. kleiner als sein Minimalwert oder größer als sein zulässiger Maximalwert ist.

11.4 Sichern und Wiederherstellen (BackupRestore)

11.4.1 IL_SIIIPParameterBackup

Kurzbeschreibung Der Funktionsbaustein **IL_SIIIPParameterBackup** sichert die Parameter eines Sercos-Gerätes in einer textbasierten Datei auf einem Speichermedium des Sercos-Masters.

Der Baustein besitzt eine ähnliche Schnittstelle wie die Bausteine ML_AxisBackup, bzw. ML_AxisBackup_Type02, basiert aber nur auf den Kommunikationsmechanismen des Sercos-Feldbus. Dadurch können nur S- und P-Parameter gesichert werden, dafür aber für beliebige Busteilnehmer, wie z.B. IO-Geräte.

Siehe auch

- ML_AxisBackup
- ML_AxisBackup_Type02

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek/ Ordner
MLC	RIL_SercosIII / BackupRestore
MTX	RIL_SercosIII / BackupRestore
MLD (IL 2G)	RIL_SercosIII / BackupRestore

Tab. 11-35: Referenztablelle IL_SIIIPParameterBackup

Schnittstellenbeschreibung

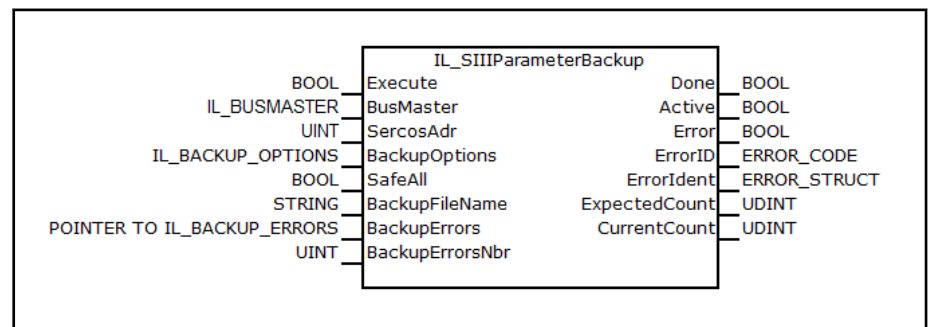


Abb. 11-13: Funktionsbaustein IL_SIIIPParameterBackup

	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (flanken-gesteuert)
	BusMaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable BusMaster unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 429

	Name	Datentyp	Beschreibung
	SercosAdr	UINT	Sercos Adresse des Slave-Teilnehmers. Bei einer MLD muss für die lokale Achse als Sercos Adresse der Wert 0 verwendet werden.
	BackupOptions	IL_BACKUP_OPTIONS	Umfang der Parametersicherung IL_BACKUP_OPTIONS, Seite 386
	SafeAll	BOOL	TRUE: es werden alle Sercos Elemente (IDN, Attribute, Name, Einheit, Min.- und Max.-Werte und Daten) in der Parameterdatei gesichert. FALSE: Es werden nur die zum Sichern notwendigen Sercos Elemente gesichert (IDN, Attribute und Daten)
	BackupFileName	STRING	Dateiname unter dem die Parameterdatei gespeichert werden soll. Bei Dateisystemen mit unterschiedlichen Pfaden sollte der gesamte Pfad angegeben werden.
	BackupErrors	POINTER TO IL_BACKUP_ERRORS	Pointer auf ein Array von IL_BACKUP_ERRORS Strukturen Kap. 11.4.3 "IL_BACKUP_ERRORS" auf Seite 386
	BackupErrorsNbr	UINT	Anzahl der Felder im Array BackupErrors
VAR_OUT-PUT	Done	BOOL	Bearbeitung abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten gültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCT	Fehlerdiagnose (siehe Fehlercodes, Seite 378)
	Expected-Count	UDINT	Anzahl an erwarteten Parametern für den Export
	CurrentCount	UDINT	Anzahl bereits exportierter Parameter

Tab. 11-36: Schnittstellenvariablen Funktionsbaustein IL_SIIIParameterBackup

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	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	USINT	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
SercosAdr	UINT	0 (MLD: lokale Achse) 1 (andere Systeme)	511 (MLD) 253 (andere Systeme)	0	Steigende Flanke an Execute
BackupOptions	IL_BACKUP_OPTIONS	ALL_PARAM	TO_SAVE_PARAM	TO_SAVE_PARAM	Steigende Flanke an Execute
SafeAll	BOOL	FALSE	TRUE	FALSE	Steigende Flanke an Execute

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
BackupFile-Name	STRING	mindestens ein Buchstabe	max. 80 Buchstaben, incl. Pfadangabe	-	Steigende Flanke an Execute
BackupErrors	POINTER TO IL_BACKUP_ERRORS	> 0			Kontinuierlich
BackupErrorsNbr	UINT	0	65535	0	Kontinuierlich

Tab. 11-37: Min-/Max- und Defaultwerte Funktionsbaustein IL_SIIParameterBackup

Funktionsbeschreibung

Der Funktionsbaustein IL_SIIParameterBackup sichert die S- und P-Parameter eines Sercos Gerätes in einer textbasierten Datei. Ist die Datei bereits vorhanden, so wird sie überschrieben. Die Datei kann vom Funktionsbaustein IL_SIIParameterRestore wieder gelesen werden.

BusMaster

Dies ist die Sercos Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable BusMaster unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0.

SercosAdr

Hier ist die Sercos Adresse des Teilnehmers anzugeben, dessen Parameter gesichert werden sollen. Bei einer MLD muss für die lokale Achse die Adresse 0 angegeben werden. Ansonsten ist die kleinste mögliche Sercos Adresse MB_MIN_SERCOS_ADDRESS und die größte mögliche Adresse MB_MAX_SERCOS_ADDRESS. Je nach Zielsystem haben diese Definitionen unterschiedliche Werte.

BackupOptions

Über den Eingang "BackupOptions" werden die zu sichernden Parameter ausgewählt. Die folgenden Auswahlmöglichkeiten sind vorhanden:

- ALL_PARAM: Es werden alle Parameter gesichert (Inhalt der Liste S-0-0017). Diese Option wird vor allem zum Überprüfen der Konfiguration benötigt
- MODIFIED_PARAM: Es werden die von den Default-Werten abweichenden Parameter gesichert. Diese Variante ist noch nicht umgesetzt, weshalb auch hier der S-0-0192 verwendet wird.
- TO_SAVE_PARAM: Es werden die für eine Wiederherstellung notwendigen Parameter gesichert (S-0-0192). Dies ist die Voreinstellung

Ist in einem Sercos Gerät der entsprechende Sercos Parameter nicht vorhanden, so wird kein Fehler erzeugt, sondern lediglich eine Parameterdatei erzeugt, die außer den Kopfzeilen keine Parameter enthält.

SafeAll

Hat dieser Eingang den Wert FALSE, so werden nur die notwendigen Daten (IDN, Attribute, Daten) in die Parameterdatei gesichert. Hat der Eingang den Wert TRUE, so werden alle Daten (zusätzlich noch Name, Einheit, Min.- und Max.-Werte) in der Parameterdatei angegeben. Für die Sicherung der Daten ist es nicht notwendig die zusätzlichen Daten zu sichern. Lediglich für die bessere Lesbarkeit, für Debug Zwecke, oder zum Ermitteln von gültigen Eingabewerten ist es sinnvoll die zusätzlichen Daten zu sichern.

Das Sichern der zusätzlichen Daten nimmt außerdem mehr Zeit in Anspruch, da wesentlich mehr Daten gelesen werden müssen.

BackupFileName

Über "BackupFileName" muss der Name der Datei auf dem Speichermedium angegeben werden. Groß-/Kleinschreibung ist dabei zu beachten. Es sollte immer der gesamte Pfad angegeben werden. Der Pfad wird dabei nicht automatisch erstellt, d.h. wird ein Pfad mit Unterverzeichnissen angegeben, so müssen diese Unterverzeichnisse bereits existieren. Die Datei selbst muss nicht existieren, sie wird angelgt, bzw. bei bereits existierender Datei, überschrieben.

BackupErrors

Dieser Eingang muss mit einem Zeiger auf ein Feld von IL_BACKUP_ERRORS belegt werden. Die Struktur besteht wiederum aus einer Parameternummer (MB_IDN) und einem Fehlercode (DWORD). In diesem Feld werden alle aufgetretenen Fehler während dem Importieren der Parameterdatei aufgelistet. Dabei muss zwischen Fehlern unterschieden werden, die im Baustein selbst entstehen (z.B. falsche Eingangswerte angelegt) und Fehlern, die während dem Schreiben von Parametern entstehen. Nur letztere werden in diese Struktur eingetragen. Die dabei eingetragenen Fehlernummern stammen von den Bausteinen IL_SIIISvcWrite und IL_SIIISvcRead und sind aus der SERCOS_TABLE.

Am Fehlerausgang des Bausteins wird bei einem Eintrag in der Struktur ein Sammelfehler eingetragen (Additional1 = 16#1E03). Der Grund dafür ist, dass bei einem Fehler während dem Lesen eines Parameters nicht einfach abgebrochen wird, sondern mit dem Lesen des nächsten Parametrs fortgefahren wird.

BackupErrorsNbr

Hier wird angegeben, wieviele Strukturen im Feld von "BackupErrors" enthalten sind. Hier arbeitet man am besten mit dem "sizeof" Operator.

ExpectedCount

Dieser Ausgang gibt an, wieviele Parameter insgesamt zu Schreiben sind. Der Ausgang wird in der Initialisierungsphase des Bausteins auf 0 gesetzt und erst im Laufe der Abarbeitung auf seinen endgültigen Wert gesetzt. D.h. solange der Ausgang noch den Wert 0 besitzt, wurde noch kein Wert ermittelt.

CurrentCount

Dieser Ausgang gibt die Anzahl der bereits gelesenen Parameter an. Der Ausgang wird in der Initialisierungsphase des Bausteins auf 0 gesetzt und erst im Laufe der Abarbeitung des Bausteins aktualisiert. Der Wert kann dazu verwendet werden, um eine Fortschrittsanzeige zu implementieren.

Zeitverhalten

Der Funktionsbaustein IL_SIIIParallelBackup arbeitet asynchron. Nach dem Start durch eine steigende Flanke am Eingang "Execute" liest er die Eingänge "BusMaster", "SercosAdr", "BackupOptions", "SafeAll" und "BackupFileName" einmalig ein, die Eingänge "BackupErrors" und "BackupErrorsNbr" werden kontinuierlich gelesen. Solange der Funktionsbaustein arbeitet, ist der Ausgang "Active" gesetzt. Nach erfolgreichem Abschluss der Operation ist der Ausgang "Done" gesetzt. Tritt ein Fehler auf, wird der Ausgang "Error" gesetzt.



Der Baustein sollte nicht in der Motion-Task verwendet werden, da er zeitweise längere Ausführungszeiten benötigt.

Kompatibilität mit anderen Parameter Dateien

Der Baustein kann selbstverständlich die von Baustein IL_SIIIPParameterRestore benötigten Dateien erzeugen. Dabei sind in der Parameter Datei neben den Parametern auch noch Zusatzinformationen enthalten, die je nach Sicherungsmethode unterschiedlich sein können. Hauptunterschied ist dabei die Adressierung, die bei diesem Baustein die Sercos Adresse umfasst, bei einer Sicherung mit IndraWorks dagegen die logische Adresse.

Die folgenden Tabelle gibt einen Überblick über die unterschiedlichen Sicherungsvarianten und Besonderheiten, die dabei berücksichtigt werden müssen.

Kompatibilität der einzelnen Backup/Restore Bausteine

Backup mit	Restore mit	Bemerkungen
IL_SercosIIIPParameterBackup IndraWorksDS	IL_SercosIIIPParameterRestore	✓ Parameterdatei kann direkt verwendet werden. Beide Bausteine verwenden die Sercos-Adresse als Adressierungsart.
	Nur MLC : ML_AxisRestore	(✓) Funktioniert nur wenn die Sercos-Adresse gleich der Logischen Adresse ist.
	Nur MLC : ML_AxisRestore_Type02	Import funktioniert nicht. Es wird ein Fehler generiert (Additional1 = 0x00001E16, Additional2 = 0x000A0008)
IndraWorks	IL_SercosIIIPParameterRestore	(✓) Bei Import muss für den Eingang "SercosAdrSource" die logische Adresse angegeben werden, für "SercosAdrDestination" die Sercos-Adresse.
	Nur MLC : ML_AxisRestore	✓ Parameterdatei kann direkt verwendet werden. Beide Bausteine verwenden die Logische Adresse als Adressierungsart.
	Nur MLC : ML_AxisRestore_Type02	✓ Parameterdatei kann direkt verwendet werden. Beide Bausteine verwenden die Logische Adresse als Adressierungsart.

Backup mit	Restore mit	Bemerkungen
Nur MLC : ML_AxisBackup	IL_SercosIIIPParameterRestore	Import funktioniert nicht. Es wird ein Fehler generiert (Additional1 = 0x00001E03, Additional2 = 0x00000000). In der Fehlerstruktur sind Fehler mit unterschiedlichen Nummern eingetragen.
	Nur MLC : ML_AxisRestore	✓ Parameterdatei kann direkt verwendet werden. Beide Bausteine verwenden die Logische Adresse als Adressierungsart.
	Nur MLC : ML_AxisRestore_Type02	✓ Parameterdatei kann direkt verwendet werden. Beide Bausteine verwenden die Logische Adresse als Adressierungsart.
Nur MLC : ML_AxisRestore_Type02	IL_SercosIIIPParameterRestore	(✓) Bei Import muss für den Eingang "SercosAdrSource" die logische Adresse angegeben werden, für "SercosAdrDestination" die Sercos-Adresse.
	Nur MLC : ML_AxisRestore	✓ Parameterdatei kann direkt verwendet werden. Beide Bausteine verwenden die Logische Adresse als Adressierungsart.
	Nur MLC : ML_AxisRestore_Type02	✓ Parameterdatei kann direkt verwendet werden. Beide Bausteine verwenden die Logische Adresse als Adressierungsart.

Tab. 11-38: Kompatibilität der einzelnen Backup- und Restore-Möglichkeiten

Fehlercodes Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE**.

Er kann in Additional1/Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Table	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	F_RELATED_TABL	16#0006	16#xxxx	Unerlaubter Zustand des Funktionsbausteins. Additional2 gibt eine interne Fehlernummer an.
INPUT_RANGE_ERROR	SERCOS_TABLE	16#0002	16#0001	Ungültiger Eingang "BusMaster"
INPUT_RANGE_ERROR	SERCOS_TABLE	16#0002	16#0002	Ungültiger Eingang "SercosAdr"
INPUT_RANGE_ERROR	F_RELATED_TABLE	16#1E00	16#0000	Der Eingang "BackupOptions" hat einen ungültigen Wert.
RESOURCE_ERROR	F_RELATED_TABLE	16#1E03	16#0000	Fehler beim Lesen eines Parameters. Weitere Informationen sind in der Fehlerstruktur "BackupErrors" enthalten.

ErrorID	Table	Additional1	Additional2	Beschreibung
OTHER_ERROR	F_RELATED_TABLE	16#1E04	16#0000	Fehler beim Lesen einer Parameterliste. Die Liste ist zu groß um im Baustein gespeichert werden zu können.
RESOURCE_ERROR	F_RELATED_TABLE	16#1E05	16#0000	"BackupErrors" ist 0 oder "BackupErrorsNbr" ist 0.
ACCESS_ERROR	F_RELATED_TABLE	16#1E10	16#0000	Fehler beim Erstellen der Parameterdatei.
ACCESS_ERROR	F_RELATED_TABLE	16#1E12	16#0000	Fehler beim Schreiben der Parameterdatei.
ACCESS_ERROR	F_RELATED_TABLE	16#1E14	16#0000	Fehler beim Schließen der Parameterdatei.

Tab. 11-39: Fehlercodes FB IL_SIIIParallelBackup

Beispiel Code

Im Folgenden soll ein Beispiel Code die prinzipielle Arbeitsweise erläutern. Das Beispiel ist nicht vollständig, insbesondere die Fehlerbehandlung fehlt komplett. Es soll vielmehr dazu dienen direkt kopiert zu werden und dann eigene Änderungen durchzuführen.

Deklaration

```

VAR CONSTANT
  // Number of maximum errors to display
  MAX_ERRORS: UINT := 50;
END_VAR
VAR
  // Backup function block
  fbParBackup : IL_SIIIParallelBackup;
  // Error strukture for function block
  rBackupErrors: ARRAY [1..MAX_ERRORS] OF IL_BACKUP_ERRORS;
  // ID number of SERCOS master 0
  rBusMaster: IL_BUSMASTER := IL_BUSMASTER_0;
  // File name, use complete path if available on your target
  strFile: STRING := 'backup.par';
  // Defines which parameters to backup
  rOptions: IL_BACKUP_OPTIONS := TO_SAVE_PARAM;
  // Progress of backup in %
  rBackupProgress: REAL;
END_VAR

```

Implementation

```

// Function block call
fbParBackup(Execute:= TRUE, Done=> , Active=> , Error=> , ErrorID=>,
  ErrorIdent=>, BusMaster:= rBusMaster, SercosAdr:= 1,
  BackupOptions:= rOptions, SafeAll:= FALSE, BackupFileName:= strFile,
  BackupErrors:= ADR(rBackupErrors), BackupErrorsNbr:= MAX_ERRORS,
  ExpectedCount=> , CurrentCount=> );

// Get progress in %.
// Note, that ExpectedCount will be 0 during initialisation
IF fbParBackup.ExpectedCount <> 0 THEN
  rBackupProgress := (fbParBackup.CurrentCount * 100)
    / fbParBackup.ExpectedCount;
ELSE
  rBackupProgress := 0;
END_IF

```

11.4.2 IL_SIIIParallelRestore

Kurzbeschreibung

Der Funktionsbaustein IL_SIIIParallelRestore importiert die S- und P-Parameter eines Sercos-Gerätes von einer textbasierten Datei. Die Datei kann mit

dem Funktionsbaustein IL_SIIIPParameterBackup erzeugt werden (Kap. 11.4.1 "IL_SIIIPParameterBackup" auf Seite 373).

Der Baustein besitzt eine ähnliche Schnittstelle wie die Bausteine ML_AxisRestore, bzw. ML_AxisRestore_Type02, basiert aber nur auf den Kommunikationsmechanismen des Sercos-Feldbus. Dadurch können nur S-und P-Parameter gesichert werden, dafür aber für beliebige Busteilnehmer, wie z.B. IO-Geräte.

Siehe auch

- ML_AxisRestore
- ML_AxisRestore_Type02

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek/ Ordner
MLC	RIL_SercosIII / BackupRestore
MTX	RIL_SercosIII / BackupRestore
MLD (IL 2G)	RIL_SercosIII / BackupRestore

Tab. 11-40: Referenztablelle IL_SIIIPParameterRestore

Schnittstellenbeschreibung

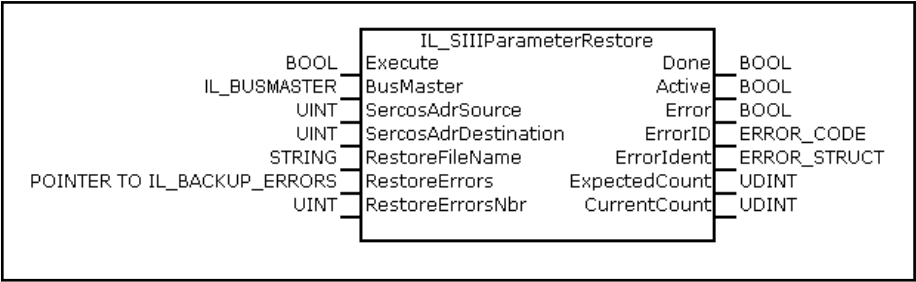


Abb. 11-14: Funktionsbaustein IL_SIIIPParameterRestore

	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (flankengesteuert)
	BusMaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable BusMaster unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 429
	SercosAdrSource	UINT	Sercos-Adresse des Slave-Teilnehmers von dem die Daten importiert werden sollen. Bei einer MLD muss für die lokale Achse als Sercos-Adresse der Wert 0 verwendet werden. Hintergrund: In einer Parameterdatei können Sicherungen von mehreren Geräten enthalten sein. Hier wird die Quelle innerhalb einer Parameterdatei angegeben.
	SercosAdrDestination	UINT	Sercos-Adresse des Slave-Teilnehmers auf den die Daten importiert werden sollen. Bei einer MLD muss für die lokale Achse als Sercos-Adresse der Wert 0 verwendet werden.
	RestoreFileName	STRING	Dateiname der Parameterdatei, die importiert werden soll. Bei Dateisystemen mit unterschiedlichen Pfaden sollte der gesamte Pfad angegeben werden.
	RestoreErrors	POINTER TO IL_BACKUP_ERRORS	Pointer auf ein Array von IL_BACKUP_ERRORS-Strukturen Kap. 11.4.3 "IL_BACKUP_ERRORS" auf Seite 386
	RestoreErrorsNbr	UINT	Anzahl der Felder im Array 'RestoreErrors'
VAR_OUTPUT	Done	BOOL	Bearbeitung abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten gültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCT	Fehlerdiagnose (siehe Fehlercodes, Seite 385)
	ExpectedCount	UDINT	Anzahl an erwarteten Parametern für den Import
	CurrentCount	UDINT	Anzahl bereits importierter Parameter

Tab. 11-41: Schnittstellenvariablen Funktionsbaustein IL_SIIIParameterRestore

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	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	USINT	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
SercosAdr-Source	UINT	1	511 (MLD) 253 (andere Systeme)	0	Steigende Flanke an Execute

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
SercosAdr-Destination	UINT	0 (MLD: lokale Achse) 1 (andere Systeme)	511 (MLD) 253 (andere Systeme)	0	Steigende Flanke an Execute
RestoreFile-Name	STRING	mindestens ein Buchstabe	max. 80 Buchstaben, incl. Pfadangabe	-	Steigende Flanke an Execute
RestoreErrors	POINTER TO IL_BACKUP_ERRORS	> 0			Kontinuierlich
RestoreErrorsNbr	UINT	0	65535	0	Kontinuierlich

Tab. 11-42: Min-/Max- und Defaultwerte Funktionsbaustein IL_SIIIPParameterRestore

Funktionsbeschreibung

Der Funktionsbaustein IL_SIIIPParameterRestore importiert die S- und P-Parameter eines Sercos-Gerätes von einer textbasierten Datei. Die Datei kann mit dem Funktionsbaustein IL_SIIIPParameterBackup erzeugt werden.

BusMaster

Dies ist die Sercos-Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable BusMaster unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0.

SercosAdrSource

Hier ist die Sercos-Adresse des Teilnehmers anzugeben, dessen Parameter importiert werden sollen. Die kleinste mögliche Sercos-Adresse ist MB_MIN_SERCOS_ADDRESS und die größte mögliche Adresse MB_MAX_SERCOS_ADDRESS. Je nach Zielsystem haben diese Definitionen unterschiedliche Werte.

SercosAdrDestination

Hier ist die Sercos-Adresse des Teilnehmers anzugeben, auf den die Parameter importiert werden sollen. Bei einer MLD muss für die lokale Achse die Adresse 0 angegeben werden. Ansonsten ist die kleinste mögliche Sercos-Adresse MB_MIN_SERCOS_ADDRESS und die größte mögliche Adresse MB_MAX_SERCOS_ADDRESS. Je nach Zielsystem haben diese Definitionen unterschiedliche Werte.

RestoreFileName

Über "RestoreFileName" muss der Name der Datei auf dem Speichermedium angegeben werden. Es sollte immer der gesamte Pfad angegeben werden. Groß-/Kleinschreibung ist dabei zu beachten.

RestoreErrors

Dieser Eingang muss mit einem Zeiger auf ein Feld von IL_BACKUP_ERRORS belegt werden. Die Struktur besteht wiederum aus einer Parameternummer (MB_IDN) und einem Fehlercode (DWORD).

In diesem Feld werden alle aufgetretenen Fehler während dem Importieren der Parameterdatei aufgelistet. Dabei muss zwischen Fehlern unterschieden werden, die im Baustein selbst entstehen (z.B. falsche Eingangswerte angelegt) und Fehlern, die während dem Schreiben von Parametern entstehen. Nur letztere werden in diese Struktur eingetragen. Die dabei eingetragenen

Fehlernummern stammen von den Bausteinen IL_SIIISvcWrite und IL_SIIISvcRead und sind aus der SERCOS_TABLE.

Am Fehlerausgang des Bausteins wird bei einem Eintrag in der Struktur ein Sammelfehler eingetragen (Additional1 = 16#1E03). Der Grund dafür ist, dass bei einem Fehler während dem Schreiben eines Parameters nicht einfach abgebrochene wird, sondern mit dem Schreiben des nächsten Parameters fortgefahren wird.

RestoreErrorsNbr

Hier wird angegeben, wieviele Strukturen im Feld von "RestoreErrors" enthalten sind. Hier arbeitet man am besten mit dem "sizeof" Operator.

ExpectedCount

Dieser Ausgang gibt an, wieviele Parameter insgesamt zu Schreiben sind. Der Ausgang wird in der Initialisierungsphase des Bausteins auf 0 gesetzt und erst im Laufe der Abarbeitung auf seinen endgültigen Wert gesetzt. D.h. solange der Ausgang noch den Wert 0 besitzt, wurde noch kein Wert ermittelt.

CurrentCount

Dieser Ausgang gibt die Anzahl der bereits gelesenen Parameter an. Der Ausgang wird in der Initialisierungsphase des Bausteins auf 0 gesetzt und erst im Laufe der Abarbeitung des Bausteins aktualisiert. Der Wert kann dazu verwendet werden, um eine Fortschrittsanzeige zu implementieren.

Zeitverhalten

Der Funktionsbaustein IL_SIIIPParameterRestore arbeitet asynchron. Nach dem Start durch eine steigende Flanke am Eingang "Execute" liest er die Eingänge "BusMaster", "SercosAdrSource", "SercosAdrDestination" und "RestoreFileName" einmalig ein, die Eingänge "RestoreErrors" und "RestoreErrorsNbr" werden kontinuierlich gelesen. Solange der Funktionsbaustein arbeitet, ist der Ausgang "Active" gesetzt. Nach erfolgreichem Abschluss der Operation ist der Ausgang "Done" gesetzt. Tritt ein Fehler auf, wird der Ausgang "Error" gesetzt.



Der Baustein sollte nicht in der Motion-Task verwendet werden, da er zeitweise längere Ausführungszeiten benötigt.

Kompatibilität mit anderen Parameter Dateien

Der Baustein kann selbstverständlich die von Baustein IL_SIIIPParameterBackup erzeugten Dateien verarbeiten. Dabei sind in der Parameter Datei neben den Parametern auch noch Zusatzinformationen enthalten, die je nach Sicherungsmethode unterschiedlich sein können. Hauptunterschied ist dabei die Adressierung, die bei diesem Baustein die Sercos-Adresse umfasst, bei einer Sicherung mit IndraWorks dagegen die logische Adresse.

Die folgenden Tabelle gibt einen Überblick über die unterschiedlichen Sicherungsvarianten und Besonderheiten, die dabei berücksichtigt werden müssen.

Kompatibilität der einzelnen Backup/Restore Bausteine

Backup mit	Restore mit	Bemerkungen
IL_SercosIIIParameterBackup IndraWorksDS	IL_SercosIIIParameterRestore	✓ Parameterdatei kann direkt verwendet werden. Beide Bausteine verwenden die Sercos-Adresse als Adressierungsart.
	Nur MLC : ML_AxisRestore	(✓) Funktioniert nur wenn die Sercos-Adresse gleich der Logischen Adresse ist.
	Nur MLC : ML_AxisRestore_Type02	Import funktioniert nicht. Es wird ein Fehler generiert (Additional1 = 0x00001E16, Additional2 = 0x000A0008)
IndraWorks	IL_SercosIIIParameterRestore	(✓) Bei Import muss für den Eingang "SercosAdrSource" die logische Adresse angegeben werden, für "SercosAdrDestination" die Sercos-Adresse.
	Nur MLC : ML_AxisRestore	✓ Parameterdatei kann direkt verwendet werden. Beide Bausteine verwenden die Logische Adresse als Adressierungsart.
	Nur MLC : ML_AxisRestore_Type02	✓ Parameterdatei kann direkt verwendet werden. Beide Bausteine verwenden die Logische Adresse als Adressierungsart.
Nur MLC : ML_AxisBackup	IL_SercosIIIParameterRestore	Import funktioniert nicht. Es wird ein Fehler generiert (Additional1 = 0x00001E03, Additional2 = 0x00000000). In der Fehlerstruktur sind Fehler mit unterschiedlichen Nummern eingetragen.
	Nur MLC : ML_AxisRestore	✓ Parameterdatei kann direkt verwendet werden. Beide Bausteine verwenden die Logische Adresse als Adressierungsart.
	Nur MLC : ML_AxisRestore_Type02	✓ Parameterdatei kann direkt verwendet werden. Beide Bausteine verwenden die Logische Adresse als Adressierungsart.
Nur MLC : ML_AxisRestore_Type02	IL_SercosIIIParameterRestore	(✓) Bei Import muss für den Eingang "SercosAdrSource" die logische Adresse angegeben werden, für "SercosAdrDestination" die Sercos-Adresse.

Backup mit	Restore mit	Bemerkungen
	Nur MLC : ML_AxisRestore	✓ Parameterdatei kann direkt verwendet werden. Beide Bausteine verwenden die Logische Adresse als Adressierungsart.
	Nur MLC : ML_AxisRestore_Type02	✓ Parameterdatei kann direkt verwendet werden. Beide Bausteine verwenden die Logische Adresse als Adressierungsart.

Tab. 11-43: Kompatibilität der einzelnen Backup- und Restore-Möglichkeiten

Fehlercodes

Der Funktionsbaustein nutzt die Fehlertabelle **F_RELATED_TABLE** und **SERCOS_TABLE**.

Er kann in Additional1/Additional2 nachfolgende Fehlermeldungen erzeugen. Zusätzliche Fehler können durch unterlagerte Baustein entstehen. Deren Fehlernummern können in der

ErrorID	Table	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	F_RELATED_TABLE	16#0006	16#xxxx	Unerlaubter Zustand des Funktionsbausteins. Additional2 gibt eine interne Fehlernummer an.
INPUT_RANGE_ERROR	SERCOS_TABLE	16#0002	16#0001	Ungültiger Eingang "BusMaster"
INPUT_RANGE_ERROR	SERCOS_TABLE	16#0002	16#0002	Ungültiger Eingang "SercosAdrSource"
INPUT_RANGE_ERROR	SERCOS_TABLE	16#0002	16#0003	Ungültiger Eingang "SercosAdrDestination"
RESOURCE_ERROR	F_RELATED_TABLE	16#1E03	16#0000	Fehler beim Schreiben eines Parameters. Weitere Informationen sind in der Fehlerstruktur "RestoreErrors" enthalten.
RESOURCE_ERROR	F_RELATED_TABLE	16#1E05	16#0000	"RestoreErrors" ist 0 oder "RestoreErrorsNbr" ist 0
ACCESS_ERROR	F_RELATED_TABLE	16#1E11	16#0000	Fehler beim Öffnen der Parameterdatei
ACCESS_ERROR	F_RELATED_TABLE	16#1E13	16#0000	Fehler beim Lesen der Parameterdatei
ACCESS_ERROR	F_RELATED_TABLE	16#1E14	16#0000	Fehler beim Schließen der Parameterdatei
OTHER_ERROR	F_RELATED_TABLE	16#1E15	16#xxxxxxxx	Fehler beim Interpretieren der Parameterdatei. In Additional2 wird die Zeilennummer innerhalb der Parameterdatei angegeben.

Tab. 11-44: Fehlercodes FB_IL_SIIIParameterRestore

Beispiel Code

Im Folgenden soll ein Beispiel Code die prinzipielle Arbeitsweise erläutern. Das Beispiel ist nicht vollständig, insbesondere die Fehlerbehandlung fehlt komplett. Es soll vielmehr dazu dienen direkt kopiert zu werden und dann eigene Änderungen durchzuführen.

Deklaration

```

VAR CONSTANT
    // Number of maximum errors to display
    MAX_ERRORS: UINT := 50;
END_VAR
VAR
    // Restore function block
    fbParRestore : IL_SIIIPParameterRestore;
    // Error structure for function block
    // (will contain parameter numbers and error codes)
    rRestoreErrors: ARRAY [1..MAX_ERRORS] OF IL_BACKUP_ERRORS;
    // ID number of Sercos master 0
    rBusMaster: IL_BUSMASTER := IL_BUSMASTER_0;
    // File name, use complete path if available on your target
    strFile: STRING := 'backup.par';
    // Progress of backup in %
    rRestoreProgress: REAL;
END_VAR

```

Implementation

```

// Function block call
fbParRestore(Execute:= TRUE, Done=> , Active=> , Error=> , ErrorID=>,
    ErrorIdent=> , BusMaster:= rBusMaster, SercosAdrSource:= 1,
    SercosAdrDestination:= 1,
    RestoreFileName:= strFile, RestoreErrors:= ADR(rRestoreErrors),
    RestoreErrorsNbr:= MAX_ERRORS, ExpectedCount=> , CurrentCount=> );

// Get progress in %.
// Note, that ExpectedCount will be 0 during initialisation
IF fbParRestore.ExpectedCount <> 0 THEN
    rRestoreProgress := (fbParRestore.CurrentCount * 100)
                        / fbParRestore.ExpectedCount;
ELSE
    rRestoreProgress := 0;
END_IF

```

11.4.3 IL_BACKUP_ERRORS

Typdefinitionen Die folgende Definition beschreibt den Datentyp IL_BACKUP_ERRORS, der in nachstehenden Funktionsbausteinen verwendet wird.

- [IL_SIIIPParameterBackup, Seite 373](#)
- [IL_SIIIPParameterRestore, Seite 379](#)

Die Typ-Definitionen sind in der Bibliothek enthalten.

Der Datentyp "IL_BACKUP_ERRORS" ist wie folgt definiert:

Strukturelement	Datentyp	Beschreibung
ParameterNumber	MB_IDN	Parameternummer im IDN Format
ErrorCode	DWORD	Fehlercode

Tab. 11-45: Definition des Datentyps IL_BACKUP_ERRORS

11.4.4 IL_BACKUP_OPTIONS

Typdefinitionen Die folgende Definition beschreibt den Aufzählungstyp IL_BACKUP_OPTIONS, der in nachstehendem Funktionsbaustein verwendet wird.

- [IL_SIIIPParameterBackup, Seite 373](#)

Die Typ-Definitionen sind in der Bibliothek enthalten.

Der Datentyp "IL_BACKUP_OPTIONS" ist wie folgt definiert:

Aufzählungselement	Wert	Beschreibung
ALL_PARAM	1	Alle Parameter
MODIFIED_PARAM	2	Vom Default-Wert abweichende Parameter
TO_SAVE_PARAM	3	Zur Wiederherstellung notwendige Parameter

Tab. 11-46: Definition der Aufzählungselemente des Datentyps IL_BACKUP_OPTIONS

11.5 Diagnose (Diagnostics)

11.5.1 IL_SIIIGetConfigSlaveAdrList

Kurzbeschreibung Mit dem Funktionsbaustein IL_SIIIGetConfigSlaveAdrList erhält man eine Liste der Sercos-Adressen aller konfigurierten SPS-Sercos-Teilnehmer.

Zielsystem	Bibliothek/ Ordner
MLC	RIL_SercosIII/ Diagnostics
MTX	RIL_SercosIII/ Diagnostics
MLD (IL 2G)	RIL_SercosIII/ Diagnostics

Tab. 11-47: Referenztable

Schnittstellenbeschreibung

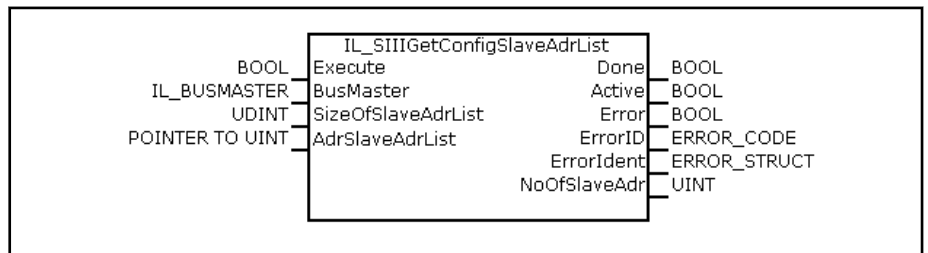


Abb. 11-15: Funktionsbaustein IL_SIIIGetConfigSlaveAdrList

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable "BusMaster" unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 429
	SizeOfSlaveAdrList	UDINT	Größe des Puffers an der Adresse "ValueAdr" in Bytes, der für den Empfang der Daten zur Verfügung gestellt wird (SIZEOF(SlaveAdrList))
	AdrSlaveAdrList	POINTER TO UINT	Adresse des Puffers, der für die Liste der Slave-Adressen zur Verfügung gestellt wird (ADR(SlaveAdrList)).
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig.
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig.
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall

E/A-Typ	Name	Datentyp	Beschreibung
	ErrorIdent	ERROR_STRUCT	Detaildiagnose
	NoOfSlaveAdr	UINT	Anzahl der in den "ValueAdr"-Puffer kopierten Bytes.

Tab. 11-48: Schnittstelle IL_SIIIGetConfigSlaveAdrList

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	BYTE	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
SizeOfSlaveAdrList	UDINT	0	16#FFFFFFFF	0	Steigende Flanke an Execute
AdrSlaveAdrList	POINTER TO UINT	n.def.	n.def.	0	Steigende Flanke an Execute

Tab. 11-49: Min-/Max und Default-Werte der Eingänge IL_SIIIGetConfigSlaveAdrList

Fehlerbehandlung Der Funktionsbaustein nutzt zur Darstellung von Fehlern des Sercos-Service-Kanals die Fehlertabelle SERCOS_TABLE (ERROR_TABLE = 16#0010).
Zusätzlich sind folgende Fehler des Bausteines in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additional2	
INPUT_RANGE_ERROR	16#0002	16#0001	Eingangswert des Parameters "BusMaster" ist ungleich IL_BUSMASTER_0 und außerhalb des Eingangswertebereiches.
INPUT_RANGE_ERROR	16#0002	16#0006	Eingangswert des Parameters "AdrSlaveAdrList" ist 0
STATE_MACHINE_ERROR	16#0006	16#0001	Fehler in der Zustandsmaschine
INPUT_INVALID_ERROR	16#2907	benötigter Wert für "SizeOfSlaveAdrList"	Eingangswert des Parameters "SizeOfSlaveAdrList" ist kleiner als benötigt

Tab. 11-50: Fehlercodes FB IL_SIIIGetConfigSlaveAdrList in F_RELATED_TABLE := 16#0170

11.5.2 IL_SIIIGetDiagSlaveAdrList

Kurzbeschreibung Mit dem Funktionsbaustein IL_SIIIGetDiagSlaveAdrList erhält man eine Liste der Sercos-Adressen aller SPS-Sercos-Teilnehmer für die Diagnoseinformationen vorliegen.

Zielsystem	Bibliothek/ Ordner
MLC	RIL_SercosIII/ Diagnostics
MTX	RIL_SercosIII/ Diagnostics
MLD (IL 2G)	RIL_SercosIII/ Diagnostics

Tab. 11-51: Referenztable

Schnittstellenbeschreibung

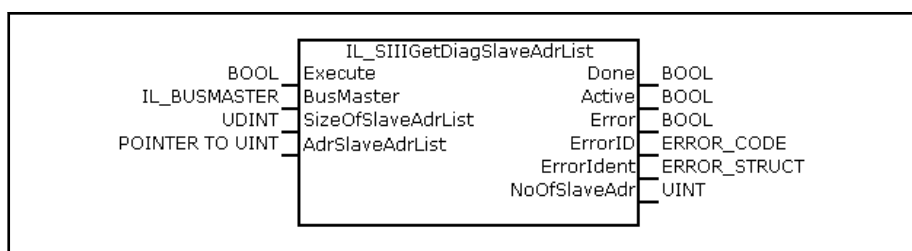


Abb. 11-16: Funktionsbaustein IL_SIIIGetDiagSlaveAdrList

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flanken-gesteuert)
	BusMaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable "BusMaster" unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 429
	SizeOfSlaveAdrList	UDINT	Größe des Puffers an der Adresse "ValueAdr" in Bytes, der für den Empfang der Daten zur Verfügung gestellt wird (SIZEOF(SlaveAdrList))
	AdrSlaveAdrList	POINTER TO UINT	Adresse des Puffers, der für die Liste der Slave Adressen zur Verfügung gestellt wird (ADR(SlaveAdrList)).
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig.
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig.
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCT	Detaildiagnose
	NoOfSlaveAdr	UINT	Anzahl der in den ValueAdr Puffer kopierten Bytes.

Tab. 11-52: Schnittstelle IL_SIIIGetDiagSlaveAdrList

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	BYTE	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
SizeOfSlaveAdrList	UDINT	0	16#FFFFFFFF	0	Steigende Flanke an Execute
AdrSlaveAdrList	POINTER TO UINT	n.def.	n.def.	0	Steigende Flanke an Execute

Tab. 11-53: Min-/Max und Default-Werte der Eingänge IL_SIIIGetDiagSlaveAdrList

Fehlerbehandlung

Der Funktionsbaustein nutzt zur Darstellung von Fehlern des Sercos-Service-Kanals die Fehlertabelle SERCOS_TABLE (ERROR_TABLE = 16#0010).

Zusätzlich sind folgende Fehler des Bausteines in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additional2	
INPUT_RANGE_ERROR	16#0002	16#0001	Eingangswert des Parameters "BusMaster" ist ungleich IL_BUSMASTER_0 und außerhalb des Eingangswertebereiches.
INPUT_RANGE_ERROR	16#0002	16#0006	Eingangswert des Parameters "AdrSlaveAdrList" ist 0
STATE_MACHINE_ERROR	16#0006	16#0001	Fehler in der Zustandsmaschine
INPUT_INVALID_ERROR	16#2907	benötigter Wert für "Size-OfSlaveAdr-List"	Eingangswert des Parameters "SizeOfSlaveAdrList" ist kleiner als benötigt

Tab. 11-54: Fehlercodes Funktionsbaustein IL_SIIIGetDiagSlaveAdrList in F_RELATED_TABLE := 16#0170

11.5.3 IL_SIIIGetLiveSlaveAdrList

Kurzbeschreibung Mit dem Funktionsbaustein IL_SIIIGetLiveSlaveAdrList erhält man eine Liste der Sercos-Adressen aller Sercos-Teilnehmer, die am Bus gefunden wurden.

Zielsystem	Bibliothek/ Ordner
MLC	RIL_SercosIII/ Diagnostics
MTX	RIL_SercosIII/ Diagnostics
MLD (IL 2G)	RIL_SercosIII/ Diagnostics

Tab. 11-55: Referenztable

Schnittstellenbeschreibung

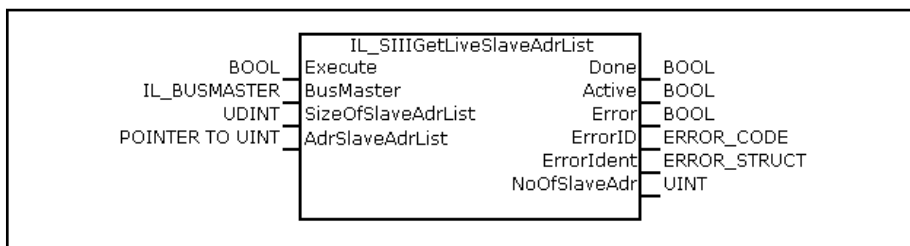


Abb. 11-17: Funktionsbaustein IL_SIIIGetLiveSlaveAdrList

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable "BusMaster" unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 429
	SizeOfSlaveAdrList	UDINT	Größe des Puffers an der Adresse "ValueAdr" in Bytes, der für den Empfang der Daten zur Verfügung gestellt wird (SIZEOF(SlaveAdrList))
	AdrSlaveAdrList	POINTER TO UINT	Adresse des Puffers, der für die Liste der Slave-Adressen zur Verfügung gestellt wird (ADR(SlaveAdrList)).
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig.

E/A-Typ	Name	Datentyp	Beschreibung
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig.
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCT	Detaildiagnose
	NoOfSlaveAdr	UINT	Anzahl der in den "ValueAdr"-Puffer kopierten Bytes.

Tab. 11-56: Schnittstelle IL_SIIIGetLiveSlaveAdrList

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	BYTE	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
SizeOfSlaveAdrList	UDINT	0	16#FFFFFFFF	0	Steigende Flanke an Execute
AdrSlaveAdrList	POINTER TO UINT	n.def.	n.def.	0	Steigende Flanke an Execute

Tab. 11-57: Min-/Max und Default-Werte der Eingänge IL_SIIIGetLiveSlaveAdrList

Fehlerbehandlung

Der Funktionsbaustein nutzt zur Darstellung von Fehlern des Sercos-Service-Kanals die Fehlertabelle SERCOS_TABLE (ERROR_TABLE = 16#0010).

Zusätzlich sind folgende Fehler des Bausteines in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additional2	
INPUT_RANGE_ERROR	16#0002	16#0001	Eingangswert des Parameters "BusMaster" ist ungleich IL_BUSMASTER_0 und außerhalb des Eingangswertebereiches.
INPUT_RANGE_ERROR	16#0002	16#0006	Eingangswert des Parameters "AdrSlaveAdrList" ist 0
STATE_MACHINE_ERROR	16#0006	16#0001	Fehler in der Zustandsmaschine
INPUT_INVALID_ERROR	16#2907	benötigter Wert für "SizeOfSlaveAdrList"	Eingangswert des Parameters "SizeOfSlaveAdrList" ist kleiner als benötigt

Tab. 11-58: Fehlercodes FB IL_SIIIGetLiveSlaveAdrList in F_RELATED_TABLE := 16#0170

11.5.4 IL_SIIIMasterState

Kurzbeschreibung

Mit dem Funktionsbaustein IL_SIIIMasterState erhält man den Status des Sercos-Masters.

Zielsystem	Bibliothek/ Ordner
MLC	RIL_SercosIII/ Diagnostics
MTX	RIL_SercosIII/ Diagnostics
MLD (IL 2G)	RIL_SercosIII/ Diagnostics

Tab. 11-59: Referenztablelle

Schnittstellenbeschreibung

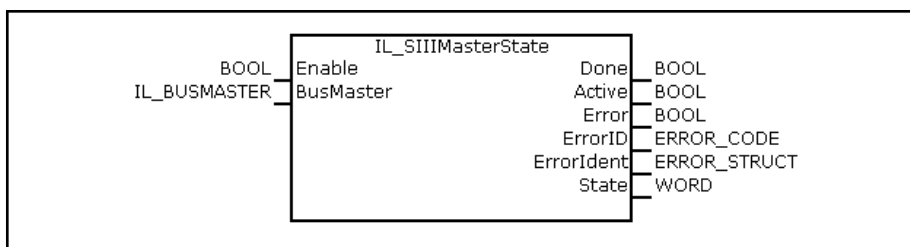


Abb. 11-18: Funktionsbaustein IL_SIIIMasterState

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (permanent, pegelgesteuert)
	BusMaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable "BusMaster" unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 429
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCT	Detaildiagnose
	State	WORD	Status des Sercos-Masters

Tab. 11-60: Schnittstelle IL_SIIIMasterState

Im Ausgang "State" ist eine Grobklassifizierung von Diagnosen binär kodiert. Die Bits sind in der RIL_FeldbusTypes.library als Globale Konstanten "FieldbusStates" enthalten.

Ausgang "State"

BIT-Nr.	Name	Beschreibung
0	IL_FBM_BUS_OFF	Feldbus nicht bereit. Der Sercos-Bus befindet sich in einer Sercos-Phase ungleich Sercos-Phase 4. Es findet keine zyklische Kommunikation statt.
1	IL_FBM_MASTER_FAILURE	Fehler des Busmasters (bus master failure)
2	IL_FBM_SLAVE_NOT_REACHABLE	Ein oder mehrere Feldbus-Slaves sind nicht erreichbar (slave not reachable)
3	IL_FBM_SLAVE_DIAGNOSIS	Ein oder mehrere Feldbus-Slaves melden Diagnose (slave diag)
4	IL_FBM_SLAVE_ERROR	Ein oder mehrere Feldbus-Slaves melden Fehler (slave error)

Min-/Max und Default-Werte

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	BYTE	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute

Tab. 11-61: Min-/Max und Default-Werte der Eingänge IL_SIIIMasterState

Fehlerbehandlung Der Funktionsbaustein nutzt zur Darstellung von Fehlern des Sercos-Service-Kanals die Fehlertabelle SERCOS_TABLE (ERROR_TABLE = 16#0010).
Zusätzlich sind folgende Fehler des Bausteines in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additional2	
INPUT_RANGE_ERROR	16#0002	16#0001	Eingangswert des Parameters "BusMaster" ist ungleich IL_BUSMASTER_0 und außerhalb des Eingangswertebereiches.
STATE_MACHINE_ERROR	16#0006	16#0001	Fehler in der Zustandsmaschine

Tab. 11-62: Fehlercodes FB IL_SIIIMasterState in F_RELATED_TABLE := 16#0170

11.5.5 IL_SIIIMasterStateDetails

Kurzbeschreibung Mit dem Funktionsbaustein IL_SIIIMasterStateDetails erhält man den detaillierten Status des Sercos-Masters mit Fehlerzählern und Topologie-Informationen.

Zielsystem	Bibliothek/ Ordner
MLC	RIL_SercosIII/Diagnostics
MLD (IL 2G)	RIL_SercosIII/Diagnostics

Tab. 11-63: Referenztable



IL_SIIIMasterStateDetails ist für das System MTX nicht verfügbar.

Schnittstellenbeschreibung

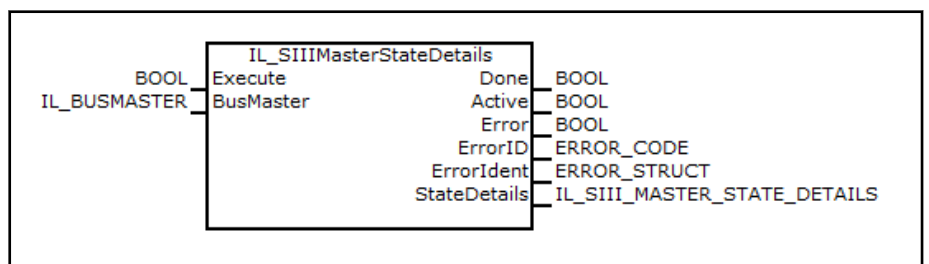


Abb. 11-19: Funktionsbaustein IL_SIIIMasterStateDetails

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (permanent, pegelgesteuert)
	BusMaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable "BusMaster" unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 429
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCT	Detaildiagnose
	StateDetails	IL_SI- II_MASTER_STATE _DETAILS	Detaillierter Status des Masters siehe auch IL_SIII_MASTER_STATE_DETAILS auf Seite 398

Tab. 11-64: Schnittstelle IL_SIIIMasterStateDetails

Min-/Max und Default-Werte

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	BYTE	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute

Tab. 11-65: Min-/Max und Default-Werte der Eingänge IL_SIIIMasterStateDetails

Fehlerbehandlung

Der Funktionsbaustein nutzt zur Darstellung von Fehlern des Sercos-Servicekanals die Fehlertabelle SERCOS_TABLE (ERROR_TABLE = 16#0010).

Zusätzlich sind Fehler der intern genutzten Funktionsbausteine IL_SIIIMasterState und IL_SIIIGetLiveSlaveAdrList, und folgende Fehler des Bausteines in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additional2	
STATE_MACHINE_ERROR	16#0006	16#0001	Fehler in der Zustandsmaschine

Tab. 11-66: Fehlercodes FB IL_SIIIMasterStateDetails in F_RELATED_TABLE := 16#0170

11.5.6 IL_SIIIRemoteSlaveState

Kurzbeschreibung

Mit dem Funktionsbaustein IL_SIIIRemoteSlaveState erhält man den Status eines am Sercos-Master betriebenen Slaves.

Zielsystem	Bibliothek/ Ordner
MLC	RIL_SercosIII/ Diagnostics
MTX	RIL_SercosIII/ Diagnostics
MLD (IL 2G)	RIL_SercosIII/ Diagnostics

Tab. 11-67: Referenztafel

Schnittstellenbeschreibung

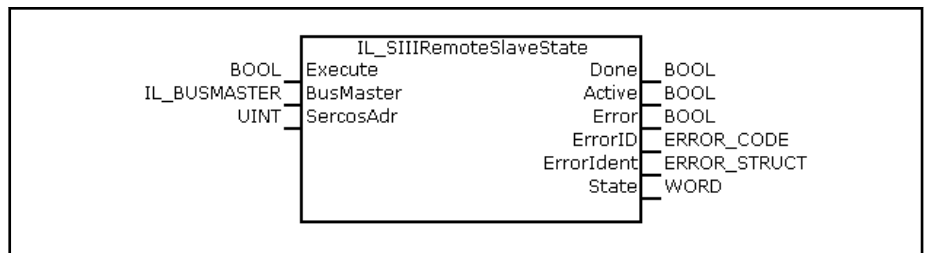


Abb. 11-20: Funktionsbaustein IL_SIIIRemoteSlaveState

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable "BusMaster" unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 429
	SercosAdr	UINT	Sercos III-Adresse des Slave-Teilnehmers.
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCT	Detalldiagnose
	State	WORD	Status eines am Sercos-Master betriebenen Slaves

Tab. 11-68: Schnittstelle IL_SIIIRemoteSlaveState

Ausgang "State" Im Ausgang "State" ist eine Grobklassifizierung von Diagnosen binär kodiert. Die Bits sind in der RIL_FieldbusTypes.library als Globale Konstanten "FieldbusStates" enthalten.

BIT-Nr.	Name	Beschreibung
0	IL_RBS_SLAVE_NOT_REACHABLE	Slave nicht existent
1	IL_RBS_SLAVE_NOT_READY	Slave nicht bereit
2	IL_RBS_SLAVE_CONFIG_ERROR	Slave-Konfigurationsfehler
3	IL_RBS_SLAVE_DIAGNOSIS	Slave-Diagnose verfügbar
4	IL_RBS_SLAVE_ERROR	Slave-Fehler verfügbar
5	IL_RBS_CON_OFF	Mindestens eine zyklische Verbindung läuft nicht

Min-/Max und Default-Werte

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	BYTE	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
SercosAdr	UINT	1	511	0	Steigende Flanke an Execute

Tab. 11-69: Min-/Max und Default-Werte der Eingänge IL_SIIIRemoteSlaveState

Fehlerbehandlung Der Funktionsbaustein nutzt zur Darstellung von Fehlern des Sercos-Service-Kanals die Fehlertabelle SERCOS_TABLE (ERROR_TABLE = 16#0010).
Zusätzlich sind folgende Fehler des Bausteines in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additional2	
INPUT_RANGE_ERROR	16#0002	16#0001	Eingangswert des Parameters "BusMaster" ist ungleich IL_BUSMASTER_0 und außerhalb des Eingangswertebereiches.
INPUT_RANGE_ERROR	16#0002	16#0002	Eingangswert des Parameters "SercosAdr" ist 0 oder > 511
STATE_MACHINE_ERROR	16#0006	16#0001	Fehler in der Zustandsmaschine
INPUT_INVALID_ERROR	16#2906	16#0000	Eingangswert des Parameters "SercosAdr" adressiert einen Slave, der nicht in der Konfiguration vorhanden ist.

Tab. 11-70: Fehlercodes FB IL_SIIIRemoteSlaveState in F_RELATED_TABLE := 16#0170

11.5.7 IL_SIIIRemoteSlaveStateDetails

Kurzbeschreibung Mit dem Funktionsbaustein IL_SIIIRemoteSlaveStateDetails erhält man den (Detail-)Status eines am Sercos-Master betriebenen Slaves.

Zielsystem	Bibliothek/ Ordner
MLC	RIL_SercosIII/ Diagnostics
MTX	RIL_SercosIII/ Diagnostics
MLD (IL 2G)	RIL_SercosIII/ Diagnostics

Tab. 11-71: Referenztable

Schnittstellenbeschreibung

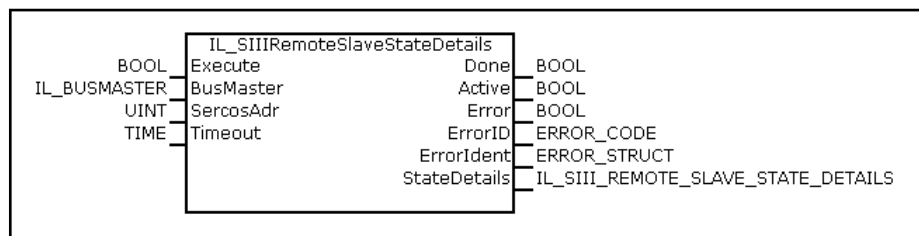


Abb. 11-21: Funktionsbaustein IL_SIIIRemoteSlaveStateDetails

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flanken-gesteuert)
	BusMaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable "BusMaster" unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 429
	SercosAdr	UINT	Sercos III-Adresse des Slave-Teilnehmers.
	Timeout	TIME	Timeout-Überwachung des FBs in ms. T#0ms = Timeout-Überwachung ist nicht aktiv.
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCT	Detailldiagnose
	StateDetails	IL_SIII_REMO- TE_SLAVE_STATE_ DETAILS	Detaillierter Status eines am Sercos-Master betriebenen Slaves IL_SIII_REMOTE_SLAVE_STATE_DETAILS, Seite 399

Tab. 11-72: Schnittstelle IL_SIIIRemoteSlaveStateDetails

Im Ausgang "State" ist eine Grobklassifizierung von Diagnosen binär kodiert. Die Bits sind in der RIL_FeldbusTypes.library als Globale Konstanten "FieldbusStates" enthalten.

Min-/Max und Default-Werte

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	BYTE	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Execute
SercosAdr	UINT	1	511	0	Steigende Flanke an Execute

Tab. 11-73: Min-/Max und Default-Werte der Eingänge IL_SIIIRemoteSlaveStateDetails

Fehlerbehandlung

Der Funktionsbaustein nutzt zur Darstellung von Fehlern des Sercos-Service-Kanals die Fehlertabelle SERCOS_TABLE (ERROR_TABLE = 16#0010). Zusätzlich sind folgende Fehler des Bausteines in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additional2	
INPUT_RANGE_ERROR	16#0002	16#0001	Eingangswert des Parameters "BusMaster" ist ungleich "IL_BUSMASTER_0" und außerhalb des Eingangswertebereiches.
INPUT_RANGE_ERROR	16#0002	16#0002	Eingangswert des Parameters "SercosAdr" ist 0 oder > 511
STATE_MACHINE_ERROR	16#0006	16#0001	Fehler in der Zustandsmaschine
INPUT_INVALID_ERROR	16#2906	16#0000	Eingangswert des Parameters "SercosAdr" adressiert einen Slave, der nicht in der Konfiguration vorhanden ist.

Tab. 11-74: Fehlercodes FB IL_SIIIRemoteSlaveStateDetails in F_RELATED_TABLE := 16#0170

11.5.8 IL_SIII_MASTER_STATE_DETAILS

Kurzbeschreibung Die Struktur enthält den detaillierten Status eines Sercos-Master.

Strukturelement	Datentyp	Beschreibung
CommunicationCOS	DWORD	Siehe Ausgang Ausgang State, Seite 392 des Funktionsbausteins "IL_SIIIMasterState".
CommunicationState	IL_FBUS_COMMUNICATION_STATE	Status der Kommunikation. CP0/CP1=Not configured CP2=STOP CP3=IDLE CP4=OPERATE. IL_FBUS_COMMUNICATION_STATE, Seite 190
Version	UINT	Version der Datenstruktur der Diagnoseinfo
ErrorCounterPri	UDINT	Fehlerzähler Primär (Port 1, siehe Port-Bezeichnungen der verschiedenen Steuerungen) der Steuerung
ErrorCounterSec	UDINT	Fehlerzähler Sekundär (Port 2, siehe Port-Bezeichnungen der verschiedenen Steuerungen) der Steuerung
NumOfActiveSlaves	UDINT	Anzahl aktiver Slaves, d. h. Slaves, zu denen der Master eine Kommunikationsverbindung aufgebaut hat
SercosAdrEndLine1	UINT	Sercos-Adresse des Slaves am Ende der Linie an Port 1 (siehe Tabelle unten, Port-Bezeichnungen der verschiedenen Steuerungen). 0 = Ring-Topologie oder an dieser Linie ist kein Gerät angeschlossen
SercosAdrEndLine2	UINT	Sercos-Adresse des Slaves am Ende der Linie an Port 2 (siehe Tabelle unten, Port-Bezeichnungen der verschiedenen Steuerungen). 0 = Ring-Topologie oder an dieser Linie ist kein Gerät angeschlossen
RingbreakTopoAdr	UINT	Topologische Adresse der Ringunterbrechung 16#FFFF = Ring-Topologie

Tab. 11-75: Elemente der Struktur IL_SIII_MASTER_STATE_DETAILS

Port	XM	VPx	MLD-M (CSH02, HCS01)
1	XF1	XF01	X25
2	XF2	XF02	X24

Tab. 11-76: Port-Bezeichnungen der verschiedenen Steuerungen

11.5.9 IL_SIII_REMOTE_SLAVE_STATE_DETAILS

Kurzbeschreibung Die Struktur enthält den detaillierten Status eines am Sercos-Master betriebenen Slaves

Nachstehende Tabelle zeigt ihre Unterstrukturen:

Unterstrukturen und deren Abhängigkeiten

IL_SIII_REMOTE_SLAVE_STATE_EX, Seite 401		
	IL_SIII_REMOTE_SLAVE_STATE_EX_BITS, Seite 401	
IL_SIII_DEVICE_STATUS, Seite 402		
	IL_SIII_DEVICE_STATUS_UNION, Seite 402	
	IL_SIII_DEVICE_STATUS_UNION_BITS, Seite 403	
	IL_SIII_TOPOLOGY_PORT_STATUS, Seite 403	
IL_SIII_DIAGNOSTIC_NUMBER, Seite 404		
	IL_SIII_DIAGNOSTIC_NUMBER_R, Seite 405	
	IL_SIII_DIAGNOSTIC_NUMBER_SOURCE_GROUP, Seite 405	
	IL_SIII_DIAGNOSTIC_NUMBER_SOURCE_TYPE, Seite 405	
	IL_SIII_DIAGNOSTIC_NUMBER_CLASS, Seite 406	
IL_SIII_DEVICE_TYPE, Seite 406		
IL_SIII_DRIVE_DIAG, Seite 406		
	IL_SIII_DRV_ERROR, Seite 407	
	IL_SIII_DRV_ERROR_BITS, Seite 407	
	IL_SIII_DRV_WARNING, Seite 408	
	IL_SIII_DRV_WARNING_BITS, Seite 408	
	IL_SIII_DRV_MESSAGE, Seite 409	
	IL_SIII_DRV_MESSAGE_BITS, Seite 409	
IL_SIII_IO_DIAG, Seite 410		
	IL_SIII_IO_DIAG_STATUS_CODE_TYPE, Seite 411	
	IL_SIII_DIAGNOSTIC_NUMBER_CLASS, Seite 406	
	IL_SIII_IO_DIAG_STATUS_CODE, Seite 411	

Strukturelement	Datentyp	Beschreibung
MasterDiagNo	UDINT	Masterdiagnosenummer, die der Master für den Slave meldet. WRONG_VENDORCODE 1 Der Master stellt in Sercos-Phase 2 fest, dass die Herstellerkennung der Gerätebeschreibung des Slaves nicht mit der Herstellerkennung (VendorCode S-1300.0.3) übereinstimmt die vom Slave gelesen wird. WRONG_DEVICEID 2 Der Master stellt in Sercos-Phase 2 fest, dass die Gerätekennung der Gerätebeschreibung des Slaves nicht mit der Herstellerkennung (VendorCode S-1300.0.5) übereinstimmt die vom Slave gelesen wird. WRONG_MODULENBR 3 Der Master stellt in Sercos-Phase 2 fest, dass die Anzahl der im Indraworks Projekt vorgegebenen IO Module nicht der am Slave ermittelten Module (List of module type codes S-0-1500.x.03) entspricht. WRONG_MODULEID 4 Der Master stellt in Sercos-Phase 2 fest, dass eine im Indraworks Projekt vorgegebenen IO Modulkennung nicht der am Slave ermittelten Modulkennung (List of module type codes S-0-1500.x.03) entspricht. SLAVEERROR 5 Slave-Teilnehmer meldet über den Bus einen Fehler. Siehe Strukturelement DiagNumber. SLAVEWARN 6 Slave-Teilnehmer meldet über den Bus einen Fehler. Siehe Strukturelement DiagNumber. ERR_SWITCH_P3 7 Fehler beim Schalten von Sercos-Phase 2 nach Sercos-Phase 3. ERR_SWITCH_P4 8 Fehler beim Schalten von Sercos-Phase 2 nach Sercos-Phase 3. ERR_REPLACEFG 9 Fehler beim Mechanismus Replace Function Groups. ERR_WRITEPARAM_P2 16#A Fehler beim Schreiben eines Parameters in Phase 2. ERR_WRITEPARAM_P3 16#B Fehler beim Schreiben eines Parameters in Phase 3. ERR_WRITEPARAM_P4 16#C Fehler beim Schreiben eines Parameters in Phase 4. WRONG_ATLENGTH 16#D Falsche At Länge. WRONG_MDTLENGTH 16#E Falsche Mdt Länge. CON_TIMEOUT 16#F Master stellt Verbindungstimeout fest. CON_ERROR 16#10 Slave meldet Verbindungsfehler fest. NEWCONFIG 16#11 Neue Konfiguration von Indraworks vorhanden, dei noch nicht aktiv ist. Busneustart notwendig. NOT_ON_BUS 16#12 Slave ist nicht am Bus vorhanden.

Strukturelement	Datentyp	Beschreibung
MasterDiagNoEx	UDINT	Zusatzfehlernummer zur MasterDiagNo
MasterDiagText	STRING	Hinweistext zur Masterdiagnosenummer (nur Englisch)
MasterDiagSolutionText	STRING	Text, der einen Hinweis zur Lösung des Fehlerbildes gibt (nur Englisch)
StateEx	IL_SIII_REMOTE_SLAVE_STATE_EX	Alle für den Slave anstehenden Fehler bitcodiert IL_SIII_REMOTE_SLAVE_STATE_EX, Seite 401
DeviceStatus	IL_SIII_DEVICE_STATUS	Entspricht Parameter Device Status S-0-1045 IL_SIII_DEVICE_STATUS, Seite 402
DiagNumber	IL_SIII_DIAGNOSTIC_NUMBER	Entspricht Sercos-Parameter S-0-0390 des Sercos-Slaves IL_SIII_DIAGNOSTIC_NUMBER, Seite 404
DiagText	STRING	Entspricht Sercos-Parameter Diagnosetext S-0-0095 des Sercos-Slaves
DeviceType	IL_SIII_DEVICE_TYPE	Zeigt, welchen Gerätetyp der Slave hat IL_SIII_DEVICE_TYPE, Seite 406
DriveDiag	IL_SIII_DRIVE_DIAG	Zeigt Antriebs Diagnoseinformationen IL_SIII_DRIVE_DIAG, Seite 406
IoDiag	IL_SIII_IO_DIAG	Zeigt IO-Diagnoseinformationen IL_SIII_IO_DIAG, Seite 410

11.5.10 IL_SIII_REMOTE_SLAVE_STATE_EX

Kurzbeschreibung Die Struktur IL_SIII_REMOTE_SLAVE_STATE_EX enthält alle für den Slave anstehenden Fehler bitcodiert.

	Datentyp	Beschreibung
StateBits	IL_SIII_REMOTE_SLAVE_STATE_EX_BITS	Bitcodiertes DWORD, das verschiedene Fehlerzustände und Status anzeigt. IL_SIII_REMOTE_SLAVE_STATE_EX_BITS, Seite 401
StateDWord	DWORD	

Tab. 11-77: Elemente der Struktur IL_SIII_REMOTE_SLAVE_STATE_EX

11.5.11 IL_SIII_REMOTE_SLAVE_STATE_EX_BITS

Kurzbeschreibung Die Struktur IL_SIII_REMOTE_SLAVE_STATE_EX_BITS enthält alle für den Slave möglichen Fehler als bitcodiertes DWORD, das verschiedene Fehlerzustände und Status anzeigt

Strukturelement	Typ	Beschreibung
NewConfig	BIT	Bitcodiertes DWORD, das verschiedene Fehlerzustände und Status anzeigt
NotOnBus	BIT	Slave ist nicht am Bus vorhanden.
ErrVendorCode	BIT	Der Master stellt in Sercos Phase 2 fest, dass die Herstellerkennung der Gerätebeschreibung des Slaves nicht mit der Herstellerkennung (VendorCode S-1300.0.3) übereinstimmt die vom Slave gelesen wird.
ErrDeviceId	BIT	Der Master stellt in Sercos Phase 2 fest, dass die Geräteerkennung der Gerätebeschreibung des Slaves nicht mit der Herstellerkennung (VendorCode S-1300.0.5) übereinstimmt die vom Slave gelesen wird.

Strukturelement	Typ	Beschreibung
ErrModuletypeNbr	BIT	Der Master stellt in Sercos Phase 2 fest, dass die Anzahl der im Indraworks Projekt vorgegebenen IO Module nicht der am Slave ermittelten Module (List of module type codes S-0-1500.x.03) entspricht.
ErrModuletype	BIT	Der Master stellt in Sercos Phase 2 fest, dass eine im Indraworks Projekt vorgegebenen IO Modulkennung nicht der am Slave ermittelten Modulkennung (List of module type codes S-0-1500.x.03) entspricht.
ErrReplaceFg	BIT	Fehler beim Mechanismus Replace Function Groups.
ErrParamP2	BIT	Fehler beim Schreiben eines Parameters in Phase 2.
ErrParamP3	BIT	Fehler beim Schreiben eines Parameters in Phase 3
ErrMdtLength	BIT	Falsche MDT-Länge
ErrAtLength	BIT	Falsche AT-Länge
ErrCl1Err	BIT	Slave Teilnehmer meldet über den Bus einen Fehler. Siehe Strukturelement DiagNumber.
ErrCl2Warn	BIT	Slave Teilnehmer meldet über den Bus einen Fehler. Siehe Strukturelement DiagNumber.
ConMdtErr	BIT	Slave meldet Verbindungsfehler fest.
ConAtTimeout	BIT	Master stellt Verbindungstimeout fest.
ConOff	BIT	Mindestens eine Sercos Verbindung ist nicht im Zustand „consuming“

Tab. 11-78: Elemente der Struktur IL_SIII_REMOTE_SLAVE_STATE_EX_BITS

11.5.12 IL_SIII_DEVICE_STATUS

Kurzbeschreibung Die Struktur IL_SIII_DEVICE_STATUS entspricht Parameter Device Status S-0-1045, ergänzt um die Topologie-Statusinformationen beider Ports.

Strukturelement	Datentyp	Beschreibung
Status	IL_SIII_DEVICE_STATUS_UNION	BitStruktur des Sercos-Parameters S-0-1045 Device Status (S-Dev) IL_SIII_DEVICE_STATUS_UNION , Seite 402
TopologyPortStatus	IL_SIII_TOPOLOGY_PORT_STATUS	unterstützt die Anzeige des Status beider Ports IL_SIII_TOPOLOGY_PORT_STATUS , Seite 403

Tab. 11-79: Elemente der Struktur IL_SIII_DEVICE_STATUS

11.5.13 IL_SIII_DEVICE_STATUS_UNION

Kurzbeschreibung Die Union IL_SIII_DEVICE_STATUS_UNION überlagert die StatusBits und das StatusWord des Parameters Device Status S-0-1045 mit dem Ziel eines bequemeren Datenzugriffs.

Element	Datentyp	Beschreibung
StatusBits	IL_SIII_DEVICE_STATUS_UNION_BITS	BitStruktur des Sercos-Parameters S-0-1045 Device Status (S-Dev) IL_SIII_DEVICE_STATUS_UNION_BITS , Seite 403
StatusWord	WORD	

Tab. 11-80: Elemente der UNION IL_SIII_DEVICE_STATUS_UNION

11.5.14 IL_SIII_DEVICE_STATUS_UNION_BITS

Kurzbeschreibung Die Struktur IL_SIII_DEVICE_STATUS_UNION_BITS enthält die einzelnen Bits des Parameters Device Status S-0-1045.

Strukturelement	Typ	Beschreibung
CommunicationWarning	BIT	Die Anzahl der erlaubten MST-Ausfälle hat die Hälfte der im Parameter S-0-1003 festgelegten überschritten
TopologyHs	BIT	Das Umschalten des Bits signalisiert eine Änderung des Topologiestatus
TopologPortStatus_3	BIT	
TopologPortStatus_2	BIT	
TopologPortStatus_1	BIT	
TopologPortStatus_0	BIT	
ErrorConnection	BIT	Der Slave hat einen Fehler in einer Verbindung festgestellt
SlaveValid	BIT	Der Inhalte des DeviceStatus ist gültig
ErrorOfDevice	BIT	Slave meldet einen Fehler
WarningOfDevice	BIT	Slave meldet eine Warnung
ProcedureCommandAck	BIT	Quittierung eines Prozedurkommandos
ParameterizationLevel	BIT	Parametrierlevel ist aktiv
LoopbackWithoutFwd	BIT	Sercos III-Telegramm wird nicht an den inaktiven Port übertragen
Reserved_2	BIT	reserviert
Reserved_1	BIT	reserviert
Reserved_0	BIT	reserviert

Tab. 11-81: Elemente der Struktur IL_SIII_DEVICE_STATUS_UNION_BITS

11.5.15 IL_SIII_TOPOLOGY_PORT_STATUS

Kurzbeschreibung Der Aufzählungstyp IL_SIII_TOPOLOGY_PORT_STATUS (UINT) unterstützt die Anzeige des Status beider Ports.

Element	Wert	Beschreibung
FAST_FORWARD_ON_BOTH_PORTS	16#00	Sercos-Gerät angeschlossen.
LOOPBACK_WITH_FORWARD_OF_P_TELEGRAMS_WITHOUT_LINK	16#04	Kein Link am inaktiven Port. Kein Gerät angeschlossen.
LOOPBACK_WITHOUT_FORWARD_OF_P_TELEGRAMS_WITHOUT_LINK	16#1C	Kein Link am inaktiven Port. Kein Gerät angeschlossen.

Element	Wert	Beschreibung
LOOPBACK_WITH_FORWARD_OF_P_TELEGRAMS_WITH_LINK	16#05	Link am inaktiven Port. Kein Gerät angeschlossen.
LOOPBACK_WITHOUT_FORWARD_OF_P_TELEGRAMS_WITH_LINK	16#1D	Link am inaktiven Port. Kein Gerät angeschlossen.
LOOPBACK_WITH_FORWARD_OF_P_TELEGRAMS_WITH_P_LINK	16#40	Sercos-Gerät angeschlossen.
LOOPBACK_WITH_FORWARD_OF_P_TELEGRAMS_WITH_S_LINK	16#41	Sercos-Gerät angeschlossen.
LOOPBACK_WITHOUT_FORWARD_OF_P_TELEGRAMS_WITH_S_LINK	16#42	Sercos-Gerät angeschlossen.
LOOPBACK_WITH_FORWARD_OF_S_TELEGRAMS_WITHOUT_LINK	16#43	Kein Link am inaktiven Port. Kein Gerät angeschlossen
LOOPBACK_WITHOUT_FORWARD_OF_S_TELEGRAMS_WITHOUT_LINK	16#44	Kein Link am inaktiven Port. Kein Gerät angeschlossen
LOOPBACK_WITH_FORWARD_OF_S_TELEGRAMS_WITH_LINK	16#45	Link am inaktiven Port. Kein Gerät angeschlossen
LOOPBACK_WITHOUT_FORWARD_OF_S_TELEGRAMS_WITH_LINK	16#46	Link am inaktiven Port. Kein Gerät angeschlossen
LOOPBACK_WITH_FORWARD_OF_S_TELEGRAMS_WITH_P_LINK	16#47	Sercos-Gerät angeschlossen
LOOPBACK_WITHOUT_FORWARD_OF_S_TELEGRAMS_WITH_P_LINK	16#48	Sercos-Gerät angeschlossen
LOOPBACK_WITH_FORWARD_OF_S_TELEGRAMS_WITH_S_LINK	16#49	Sercos-Gerät angeschlossen

Tab. 11-82: Elemente des Aufzählungstyps *IL_SIII_TOPOLOGY_PORT_STATUS*

11.5.16 IL_SIII_DIAGNOSTIC_NUMBER

Kurzbeschreibung Der Überlagerungstyp *IL_SIII_DIAGNOSTIC_NUMBER* unterstützt den Zugriff auf die einzelnen Abschnitte der vollständigen Sercos-Diagnosenummer (S-0-0390). Alternativ kann der Anwender auch auf die nichtstrukturierte Information als *DWORD* zugreifen.

Strukturelement	Datentyp	Beschreibung
Number	IL_SIII_DIAGNOSTIC_NUMBER_R	Entspricht Parameter S-0-0390. IL_SIII_DIAGNOSTIC_NUMBER_R, Seite 405
NumberDWord	DWORD	

Tab. 11-83: Elemente des Überlagerungstyps *IL_SIII_DIAGNOSTIC_NUMBER*

Im Parameter S-0-0390 wird die aktuelle Diagnosenummer angezeigt. Dadurch ist es der Steuerung möglich, eigene Diagnosen anhand dieser Nummer zu generieren (z.B. in Sprachen, in denen die Diagnosetexte nicht im Antrieb abgelegt sind).

Ab MPx-18VRS werden Sercos-konforme Diagnose-Codes verwendet (Bit 24 bis 31). Um ab der MPx-18VRS ein kompatibles Verhalten zu den vorherigen Versionen zu bekommen, muss im Parameter P-0-0006 das Bit 0 auf "1" gesetzt werden.

11.5.17 IL_SIII_DIAGNOSTIC_NUMBER_R

Kurzbeschreibung Die Struktur IL_SIII_DIAGNOSTIC_NUMBER_R erlaubt den Zugriff auf die einzelnen Elemente der vollständigen Diagnosenummer.

Strukturelement	Datentyp	Beschreibung
SourceGroup	IL_SIII_DIAGNOSTIC_NUMBER_SOURCE_GROUP	beschreibt die Verwendung der Bits der vollständigen Diagnosenummer IL_SIII_DIAGNOSTIC_NUMBER_SOURCE_GROUP , Seite 405
SourceType	IL_SIII_DIAGNOSTIC_NUMBER_SOURCE_TYPE	Quelle der Diagnose IL_SIII_DIAGNOSTIC_NUMBER_SOURCE_TYPE , Seite 405
Class	IL_SIII_DIAGNOSTIC_NUMBER_CLASS	Priorität der anstehenden Diagnose IL_SIII_DIAGNOSTIC_NUMBER_R , Seite 405
StatusCode	UINT	Fehlernummer

Tab. 11-84: Elemente der Struktur IL_SIII_DIAGNOSTIC_NUMBER_R

11.5.18 IL_SIII_DIAGNOSTIC_NUMBER_SOURCE_GROUP

Kurzbeschreibung Der Aufzählungstyp IL_SIII_DIAGNOSTIC_NUMBER_SOURCE_GROUP beschreibt die Verwendung der Bits der vollständigen Diagnosenummer.

Element	Wert	Beschreibung
MANUFACTURER_SPECIFIC_15_0	0	Herstellerspezifischer StatusCode Bits 29-16 (SourceType und Class) sind durch Sercos definiert Bits 15-0, der StatusCode, ist herstellerspezifisch
MANUFACTURER_SPECIFIC_29_0	1	Vollständig herstellerspezifisch Bits 29-0 sind durch den Gerätehersteller spezifiziert
RESERVED	2	Reserviert
STANDARD	3	Standard Bits 29-0 sind durch Sercos definiert

Tab. 11-85: Elemente des Aufzählungstyps IL_SIII_DIAGNOSTIC_NUMBER_SOURCE_GROUP

11.5.19 IL_SIII_DIAGNOSTIC_NUMBER_SOURCE_TYPE

Kurzbeschreibung Der Aufzählungstyp IL_SIII_DIAGNOSTIC_NUMBER_SOURCE_TYPE gibt Auskunft über die Herkunft der Diagnose.

Element	Wert	Beschreibung
ST_FSP_DRIVE	0	StatusCode ist definiert durch Antriebs-Profil
ST_FSP_IO	1	StatusCode ist definiert durch IO-Profil
ST_GDP	2	StatusCode ist definiert durch GDP-Profil
ST_SCP	3	StatusCode ist definiert durch SCP-Profil
ST_CSoS	4	StatusCode ist definiert durch CSoS-Spezifikation

Element	Wert	Beschreibung
ST_FSP_ENCODER	5	StatusCode ist definiert durch Encoder-Profil
ST_SAFETY_APPLICATION	6	StatusCode ist definiert durch Safety-Applikation

Tab. 11-86: Elemente des Aufzählungstyps IL_SIII_DIAGNOSTIC_NUMBER_SOURCE_TYPE

11.5.20 IL_SIII_DIAGNOSTIC_NUMBER_CLASS

Kurzbeschreibung Der Aufzählungstyp IL_SIII_DIAGNOSTIC_NUMBER_CLASS zeigt die Priorität der Diagnosen.

Element	Wert	Beschreibung
RESERVED_0	16#00	reserviert: 16#00...16#09
OPERATIONAL_STATE	16#0A	Meldung (Message), Priorität 4. Gibt Auskunft über einen allgemeinen Betriebszustand, z. B. Antrieb HALT...
RESERVED_1	16#0B	reserviert
PROCEDUR_CMD_STATE	16#0C	Kommando-Fehler, Priorität 3. Zeigt an, dass bei der Abarbeitung eines Kommandos ein Fehler aufgetreten ist.
RESERVED_2	16#0D	reserviert
WARNING_C2D	16#0E	Warnung, Priorität 2
ERROR_C1D	16#0F	Fehler mit höchster Priorität (1). Fehler muss bestätigt werden Siehe auch S-0-0099, C0500 Reset Zustandsklasse 1

Tab. 11-87: Elemente des Aufzählungstyps IL_SIII_DIAGNOSTIC_NUMBER_CLASS

11.5.21 IL_SIII_DEVICE_TYPE

Kurzbeschreibung Der Aufzählungstyp IL_SIII_DEVICE_TYPE (UDINT) zählt die unterschiedlichen Gerätetypen auf.

Element	Wert	Beschreibung
FSP_IO	1	Gerätetyp nach IO-Profil
FSP_DRIVE	2	Gerätetyp nach Antriebs-Profil
FSP_ENCODER	3	Gerätetyp nach Encoder-Profil
FSP_MANUFATURER_SPECIFIC	16#00008000	Herstellerspezifischer Gerätetyp
FSP_UNKNOWN	16#FFFFFFFF	Unbekannter Gerätetyp

Tab. 11-88: Elemente des Aufzählungstyps IL_SIII_DEVICE_TYPE

11.5.22 IL_SIII_DRIVE_DIAG

Kurzbeschreibung Die Struktur IL_SIII_DRIVE_DIAG fasst die möglichen Diagnosebits von Fehlern, Warnungen und Meldungen zusammen.

Strukturelement	Datentyp	Beschreibung
DrvError	IL_SIII_DRV_ERROR	UNION der Fehlerbits, siehe S-0-0011 IL_SIII_DRV_ERROR, Seite 407
DrvWarning	IL_SIII_DRV_WARNING	UNION der Warnungsbits, siehe S-0-0012 IL_SIII_DRV_WARNING, Seite 408
DrvMessage	IL_SIII_DRV_MESSAGE	UNION der Meldungsbits, siehe S-0-0013 IL_SIII_DRV_MESSAGE, Seite 409

Tab. 11-89: Elemente der Struktur IL_SIII_DRIVE_DIAG

11.5.23 IL_SIII_DRV_ERROR

Kurzbeschreibung Der Überlagerungstyp (UNION) IL_SIII_DRV_ERROR unterstützt den Zugriff und die Anzeige der Fehlerbits nach S-0-0011.

Strukturelement	Datentyp	Beschreibung
ErrorBits	IL_SIII_DRV_ERROR_BITS	Entspricht Parameter S-0-0011 IL_SIII_DRV_ERROR_BITS, Seite 407
ErrorWord	WORD	

Tab. 11-90: Elemente des Überlagerungstyps IL_SIII_DRV_ERROR

11.5.24 IL_SIII_DRV_ERROR_BITS

Kurzbeschreibung Die Struktur IL_SIII_DRV_ERROR_BITS enthält die einzelnen Bits des Parameters S-0-0011.

Strukturelement	Typ	Beschreibung
ManufacturerSpecificError	BIT	Herstellerspezifischer Fehlercode, siehe S-0-0129
Reserved	BIT	
TravelRangelsExceeded	BIT	Lagegrenzwerte überschritten, siehe S-0-0049, S-0-0050 oder S-0-0478, Lage-/Fahrbereichsgrenze-Status, wenn aktiviert
NotUsed	BIT	
ExcessivePositionDeviation	BIT	siehe S-0-0159, Überwachungsfenster, oder S-0-0391, Überwachungsfenster Geber 2 oder Differenz zwischen Positionssollwerten zu groß
PowerSupplyPhaseError	BIT	Netzspannungsfehler
UnderVoltageError	BIT	Unterspannungsfehler, Netzspannung oder DC-Bus-Spannung, Grenzwert ist herstellerspezifisch
OverVoltageError	BIT	Überspannungsfehler, Netzspannung oder DC-Bus-Spannung, Grenzwert ist herstellerspezifisch
OverCurrentError	BIT	Überstromfehler
ErrorInTheCommutationSystem	BIT	Fehler im "Commutation System"
FeedbackErrorEncoder	BIT	Motorgeber: S-0-0390, Diagnose-Nummer, enthält 0xC00F8022 Externer Geber: S-0-0390, Diagnose-Nummer, enthält 0xC00F2042
ControlVoltageError	BIT	Ausfall Steuerspannung, S-0-0390, Diagnose-Nummer, enthält 0xC00F8070
CoolingErrorShutDown	BIT	Siehe S-0-0205

Strukturelement	Typ	Beschreibung
MotorOverTemperatureShut-Down	BIT	Siehe S-0-0204, Motor-Abschalttemperatur überschritten
AmplifierOverTemperatureShutDown	BIT	Siehe S-0-0203
OverloadShutDown	BIT	Siehe S-0-0114

Tab. 11-91: Elemente der Struktur IL_SIII_DRV_ERROR_BITS

11.5.25 IL_SIII_DRV_WARNING

Kurzbeschreibung Der Überlagerungstyp (UNION) IL_SIII_DRV_WARNING unterstützt den Zugriff und die Anzeige der Warnungsbits nach S-0-0012.

Strukturelement	Datentyp	Beschreibung
WarningBits	IL_SIII_DRV_WARNING_BITS	Entspricht Parameter S-0-0012 IL_SIII_DRV_WARNING_BITS, Seite 408
WarningWord	WORD	

Tab. 11-92: Elemente des Überlagerungstyps IL_SIII_DRV_WARNING

11.5.26 IL_SIII_DRV_WARNING_BITS

Kurzbeschreibung Die Struktur IL_SIII_DRV_WARNING_BITS enthält die einzelnen Bits des Parameters S-0-0012.

Strukturelement	Typ	Beschreibung
ManufacturerSpecificWarning	BIT	Herstellerspezifische Warnung, ohne/ mit Systemabschaltung, siehe S-0-0181
Reserved_14	BIT	
Reserved_13	BIT	
CommunicationWarning	BIT	Warnung Kommunikation, ohne/ mit Systemabschaltung
ExcessiveVelocityDeviation	BIT	Warnung Geschwindigkeitsüberschreitung, ohne/ mit Systemabschaltung, siehe S-0-0377
Reserved_10	BIT	
UndervoltageWarning	BIT	Warnung Unterspannung, ohne/ mit Systemabschaltung
Reserved_8	BIT	
Reserved_7	BIT	
Reserved_6	BIT	
Reserved_5	BIT	
Reserved_4	BIT	
CoolingErrorWarning	BIT	Warnung Kühlung, ohne/ mit Systemabschaltung,
MotorOverTemperatureWarning	BIT	Warnung Motorübertemperatur, ohne/ mit Systemabschaltung, siehe S-0-0312

Strukturelement	Typ	Beschreibung
AmplifierOverTemperatureWarning	BIT	Warnung Verstärkerübertemperatur, ohne/ mit Systemabschaltung, siehe S-0-0311
OverloadWarning	BIT	Warnung Überlastung von Verstärker, Motor, Stromversorgung, ohne/ mit Systemabschaltung, siehe S-0-0310

Tab. 11-93: Elemente der Struktur IL_SIII_DRV_WARNING_BITS

11.5.27 IL_SIII_DRV_MESSAGE

Kurzbeschreibung Der Überlagerungstyp (UNION) IL_SIII_DRV_MESSAGE unterstützt den Zugriff und die Anzeige der Meldungsbits nach S-0-0013.

Strukturelement	Datentyp	Beschreibung
MessageBits	IL_SIII_DRV_MESSAGE_BITS	Entspricht Parameter S-0-0013 IL_SIII_DRV_MESSAGE_BITS, Seite 409
MessageWord	WORD	

Tab. 11-94: Elemente des Überlagerungstyps IL_SIII_DRV_MESSAGE

11.5.28 IL_SIII_DRV_MESSAGE_BITS

Kurzbeschreibung Die Struktur IL_SIII_DRV_MESSAGE_BITS enthält die einzelnen Bits des Parameters S-0-0013.

Strukturelement	Typ	Beschreibung
ManufacturerSpecificOperationStatus	BIT	Siehe S-0-0182
PositionFeedbackValueStatusSee	BIT	Siehe S-0-0403, BIT 0, "Status Lageistwerte"
InterpolatorHalted	BIT	Siehe S-0-0343, "Meldung Interpolator angehalten"
TargetPositionAttained	BIT	Siehe S-0-0342, "Zielposition erreicht"
InCoarsePosition	BIT	Siehe S-0-0341, "Meldung In-Position grob"
MaximumSpindleSpeedExceede	BIT	Siehe S-0-0340, nfeedback >= maximum spindle speed
MinimumSpindleSpeedUndershot	BIT	Siehe S-0-0339, nfeedback <= minimum spindle speed
TargetPositionReached	BIT	Siehe S-0-0338, "Meldung In Zielposition" Position feedback = Active target position S-0-0430 – (S-0-0051 OR S-0-0053) < S-0-0057
PowerSupplyExceeded	BIT	Siehe S-0-0337, "P >= Px" Betrag des Leistungsistwerts hat einen einstellbaren Schwellenwert überschritten
InPosition	BIT	Siehe S-0-0336, "Meldung In-Position"
VelocityLimitExceeded	BIT	Siehe S-0-0335, "Meldung n_soll > n_grenz" Sollwertbetrag der Geschwindigkeit ist größer als der einstellbare Geschwindigkeitsgrenzwert

Strukturelement	Typ	Beschreibung
TorqueLimitExceeded	BIT	Siehe S-0-0334, "Meldung Md >= Mdgrenz" Istwert des Drehmoments bzw. der Kraft hat den Grenzwert erreicht
TorqueThresholdExceeded	BIT	Siehe S-0-0333, "Meldung Md >= Mdx" Istwert des Drehmoments bzw. der Kraft hat einen einstellbaren Schwellenwert überschritten
VelocityThresholdUndershot	BIT	Siehe S-0-0332, "Meldung n_ist < nx" Istwert der Geschwindigkeit hat einen einstellbaren Schwellenwert unterschritten
VelocityInStandstillWindow	BIT	S-0-0331, "Meldung n_ist = 0" Istwert der Geschwindigkeit einen einstellbaren Schwellenwert unterschritten
VelocityInCommandWindow	BIT	S-0-0330, "Meldung n_ist = n_soll" Istwert der Geschwindigkeit hat den Sollwert innerhalb eines Toleranzfensters erreicht

Tab. 11-95: Elemente der Struktur IL_SIII_DRV_MESSAGE_BITS

11.5.29 IL_SIII_IO_DIAG

Kurzbeschreibung Die Struktur IL_SIII_IO_DIAG zeigt IO-Diagnoseinformationen an.

Strukturelement	Datentyp	Beschreibung
StatusCodeTyp	IL_SIII_IO_DIAG_STATUS_CODE_TYPE	gibt die Herkunft des "Status Code" an. IL_SIII_IO_DIAG_STATUS_CODE_TYPE, Seite 411
DiagClass	IL_SIII_DIAGNOSTIC_NUMBER_CLASS	Priorität der anstehenden Diagnose IL_SIII_DIAGNOSTIC_NUMBER_CLASS, Seite 406
StatusCode	IL_SIII_IO_DIAG_STATUS_CODE	"Status Code", der im IO-Profil spezifiziert ist IL_SIII_IO_DIAG_STATUS_CODE, Seite 411
IoFunctiongroup	UINT	IO-Funktionsgruppe, die die Diagnosemeldung verursacht hat z. B. "1503 - Digital output", S-0-1500.0.32, Bits 39-24
IoSlotnumber	BYTE	IO-Slotnummer des Moduls, dem die Funktionsgruppe zugeordnet ist (Strukturinstanz). S-0-1500.0.32, Bits 23-16
IoSubbusSlaveIndex	BYTE	Subbus-Slave, der die Diagnosemeldung verursacht hat S-0-1500.0.32, Bits 15-8
IoChannelNumber	BYTE	Kanal, der die Diagnosemeldung verursacht hat S-0-1500.0.32, Bits 7-0
IoOutputsActive	BOOL	Ausgänge sind aktiv S-0-1500.0.02, BIT 15
IoInputsValid	BOOL	Eingänge sind gültig S-0-1500.0.02, BIT 14

Tab. 11-96: Elemente der Struktur IL_SIII_IO_DIAG

11.5.30 IL_SIII_IO_DIAG_STATUS_CODE_TYPE

Kurzbeschreibung Der Aufzählungstyp IL_SIII_IO_DIAG_STATUS_CODE_TYPE gibt die Herkunft des "StatusCode" an.

Element	Wert	Beschreibung
IO_PROFILE_STATUS_CODE	0	StatusCode ist im IO-Profil definiert
MANUFACTURER_STATUS_CODE	1	StatusCode ist herstellerspezifisch

Tab. 11-97: Elemente des Aufzählungstyps IL_SIII_IO_DIAG_STATUS_CODE_TYPE

11.5.31 IL_SIII_IO_DIAG_STATUS_CODE

Kurzbeschreibung Der Aufzählungstyp IL_SIII_IO_DIAG_STATUS_CODE zeigt den "StatusCode" an, der im IO-Profil spezifiziert ist.

Der Sercos-IO-StatusCode ist enthalten in:

- S-0-0390, Diagnose-Nummer
Quellentyp (Bit 29-24) | Klasse (Bit 19-16) | StatusCode (Bit 15-0)
z.B. 16#C10E2350 ⇒ C1 | 0E | 2350
- S-0-1500.0.32 I/O diagnostic message
Klasse (Bit 59-56) | StatusCode (Bit 55-40) | Funktionsgruppe (Bit 39-24) | Slotnummer (Bit 23-16) | Subbus-Slave (Bit 15-8) | Kanalnummer (Bit 7-0)
z.B. 16#0E235005DE010008 ⇒ 0E | 2350 | 05DE | 01 | 00 | 08

Wert	Beschreibung
Statuscodes mit den Diagnoseklassen Betriebszustand (0xC10A), Warnung (0xC10E), Fehler (0xC10F)	
16#0	Kein Fehler
16#1000	Allgemeiner Fehler
16#2000	Strom
16#2100	Strom auf der Geräteeingangsseite
16#2110	Kurzschluss/Erdschluss
16#2120	Erdschluss
16#2130	Kurzschluss
16#2136	Kurzschluss an VCC
16#2137	Kurzschluss an Erde
16#2140	Kurzschluss in Peripherie
16#2141	Querschuss Ausgang und andere Spannungen
16#2142	Querschuss Ausgang und einer Taktleitung
16#2146	Kurzschluss Eingang an VCC
16#2147	Kurzschluss Eingang an Erde
16#2150	Offene Signalleitungen, nicht verbunden
16#2160	Drahtbruch Signalleitung
16#2180	Eingang inaktiv, nicht aktiviert

Wert	Beschreibung
Statuscodes mit den Diagnoseklassen Betriebszustand (0xC10A), Warnung (0xC10E), Fehler (0xC10F)	
16#2181	Eingangsschaltung fehlerhaft (z.B. load)
16#2200	Strom innerhalb des Gerätes
16#2213	Überstrom beim Hochlauf
16#2214	Überstrom im Betrieb
16#2220	Dauerüberstrom
16#2230	Kurzschluss/ Erdschluss
16#2240	Erdschluss
16#2250	Kurzschluss
16#2300	Strom auf der Geräteausgangsseite
16#2310	Dauerüberstrom
16#2320	Kurzschluss/Erdschluss
16#2330	Erdschluss
16#2340	Kurzschluss
16#2344	Ausgang überlastet
16#2345	Sensorversorgung überlastet
16#2346	Kurzschluss an VCC
16#2347	Kurzschluss an Erde
16#2350	Drahtbruch
16#2360	Drahtbruch
16#2370	Querschluss
16#2371	Querschluss Ausgang
16#2372	Querschluss Ausgang
16#2380	Ausgang nicht aktiviert
16#2381	Ausgangsschaltung fehlerhaft (z.B. load)
16#3000	Spannung
16#3100	Netzspannung (Spannung an der Geräteeingangsseite -Spannungsversorgung)
16#3110	Netzüberspannung
16#3120	Netzunterspannung
16#3130	Phasenfehler
16#3134	Phasensequenz
16#3140	Netzfrequenz
16#3141	Netzfrequenz zu hoch
16#3142	Netzfrequenz zu niedrig
16#3150	Polarität Versorgungsspannung verkehrt
16#3180	Signaleingang Allgemein

Wert	Beschreibung
Statuscodes mit den Diagnoseklassen Betriebszustand (0xC10A), Warnung (0xC10E), Fehler (0xC10F)	
16#3181	Signalleitung offen
16#3182	Drahtbruch
16#3183	Unerwarteter Signalwechsel
16#3184	Bereichsüberschreitung Spannung
16#3185	Signalspannung außerhalb des definierten Bereiches
16#3186	Bereichsunterschreitung Spannung
16#3200	Spannung innerhalb des Gerätes
16#3210	Überspannung innerhalb des Gerätes
16#3220	Unterspannung innerhalb des Gerätes
16#3230	Ladefehler
16#3300	Ausgangsspannung (Spannung an der Geräteausgangsseite, z. B. Versorgungs- oder Arbeitsspannung für externe Teile)
16#3310	Ausgangsüberspannung
16#3320	Ausgangsunterspannung
16#3400	Versorgungsspannung
16#3401	Versorgungsspannung Überspannung
16#3402	Versorgungsspannung fehlt
16#3403	Versorgungsspannung Unterspannung
16#3410	Sensorversorgung
16#3411	Sensorversorgung Unterspannung
16#3412	Sensorversorgung fehlt
16#3413	Sensorversorgung Überspannung
16#3420	Aktorversorgung
16#3421	Aktorversorgung Unterspannung
16#3422	Aktorversorgung fehlt
16#3423	Aktorversorgung Überspannung
16#4000	Temperatur
16#4100	Umgebungstemperatur
16#4110	Umgebungsübertemperatur
16#4120	Umgebungsuntertemperatur
16#4130	Temperatur Ansaugluft
16#4140	Temperatur Abluft
16#4200	Gerätetemperatur
16#4210	Geräteübertemperatur
16#4220	Geräteuntertemperatur

Wert	Beschreibung
Statuscodes mit den Diagnoseklassen Betriebszustand (0xC10A), Warnung (0xC10E), Fehler (0xC10F)	
16#4300	Externe Temperatur (z.B. des Antriebs)
16#4310	Externe Übertemperatur (z.B. des Antriebs)
16#4320	Externe Untertemperatur (z.B. des Antriebs)
16#4400	Anschlussstemperatur
16#4410	Anschlussübertemperatur
16#4420	Anschlussuntertemperatur
16#5000	Gerätehardware (nur im Geräteinneren)
16#5010	Bauteilfehler
16#5100	Versorgung
16#5110	Niederspannungsversorgung (für alle internen Spannungen 24V, 5V, 3,3V, 3,0V, 2,5V, 1,8V)
16#5120	Luftversorgung
16#5130	Farbversorgung
16#5140	Zwischenkreisversorgung
16#5150	Initiatorversorgung (Fehler in der Versorgung eines externen Sensors, der aus dem Geräteinneren versorgt wurde)
16#5151	Kurzschluss
16#5160	Versorgung der peripheren Geräte (Fehler der Versorgung für externe Geräte, die aus dem Geräteinneren versorgt werden)
16#5200	Steuergerät/Messkreis
16#5210	Messkreise
16#5220	Computerschaltkreise
16#5230	Kommunikation (innerhalb des Gerätes)
16#5300	Bedieneinheit
16#5400	Leistungsteil
16#5410	Ausgangsstufen
16#5420	Chopper (Unterbrecher)
16#5430	Eingangsstufen
16#5440	Schütze
16#5450	Sicherungen
16#5500	Kommunikation mit Erweiterungsmodulen
16#5600	Interner Speicher
16#5610	RAM
16#5620	EPROM
16#5630	EEPROM
16#5640	Flash Speicher
16#6000	Gerätesoftware

Wert	Beschreibung
Statuscodes mit den Diagnoseklassen Betriebszustand (0xC10A), Warnung (0xC10E), Fehler (0xC10F)	
16#6010	Software-Reset (Watchdog)
16#6100	Interne Software (Firmware)
16#6110	Flexware fehlt
16#6120	Flexware inkonsistent
16#6130	Applikation noch nicht betriebsbereit, Bootvorgang aktiv
16#6200	Applikationssoftware
16#6300	Datensatz fehlerhaft
16#6310	Fehlender Parameter
16#6320	Parameterfehler
16#6330	Parameter noch nicht initialisiert
16#6800	Konfiguration
16#6810	Prozessdaten mehrfach gemapped
16#7000	Erweiterungsmodul(e) defekt
16#7100	Leistung
16#7140	Relais, Schütz Fehler
16#7150	Sicherung Fehler
16#7200	Messkreis
16#7300	Sensor
16#7400	Computerschaltkreis
16#7500	Kommunikation
16#7600	Datenspeicher
16#7610	RAM
16#7620	EPROM
16#7630	EEPROM
16#7640	Flash Speicher
16#7700	Leitungsbruch / Kabelfehler
16#7710	Drahtbruch Kabel zum Sensor
16#8000	Überwachung
16#8100	Kommunikation
16#8110	Prozessdatenüberwachung
16#8120	Hostüberwachung
16#8121	Prozessdatenkanal Handshake TimeOut
16#8150	Puffer Bearbeitung
16#8151	Sendepuffer
16#8152	Sendepuffer voll

Wert	Beschreibung
Statuscodes mit den Diagnoseklassen Betriebszustand (0xC10A), Warnung (0xC10E), Fehler (0xC10F)	
16#8153	Sendepuffer Überlauf
16#8159	Empfangspuffer
16#815A	Empfangspuffer voll
16#815B	Empfangspuffer Überlauf
16#8200	Regelung
16#8210	Hohe Systemabweichung (Soll > Ist, herstellerspezifisch), Abweichung länger als zulässig vorhanden
16#8211	Maximaler Stellwert erreicht
16#8220	Systemabweichung (Soll > Ist, herstellerspezifisch), Abweichung länger als zulässig vorhanden
16#8221	Maximaler Stellwert erreicht
16#8900	Sensoren
16#8910	Gemessener Wert oberhalb des Wertebereiches
16#8920	Gemessener Wert unterhalb des Wertebereiches
16#8930	Zusätzlicher Anschluss für externen Sensor
16#8A00	Aktor
16#8B00	Vorbeugende Wartung notwendig (Zustandsüberwachung)
16#8F00	Safety Fehler
16#8F01	Symmetrieverletzung
16#8F08	Helltest, Ausgang kann nicht geschaltet werden, wenn inaktiv
16#8F09	Dunkeltest, Ausgang kann nicht abgeschaltet werden, wenn aktiv
16#8F0F	Fehlerzustand, Modul arbeitet nicht mehr
16#9000	Externer Fehler
16#A000	Subbus-Slave
16#A001	Subbus-Slave fehlt
16#A002	Falscher Subbus-Slave vorhanden
16#A003	Kompatibler Ersatz (der erwartete Subbus-Slave wurde durch einen kompatiblen ersetzt)
16#A004	Fehler Anzahl der Subbus-Geräte (zu viele Geräte)
16#A010	Gerätefehler
16#A012	Anwendung auf dem Subbus-Slave nicht bereit
16#A013	Subbus-Geräte-Reset
16#A014	Subbus-Gerät Parameterfehler
16#A020	Kommunikationsfehler
16#A021	Subbus-Fehler - Timeout (Break)
16#A022	Mehrfacher Übertragungsfehler auf dem Subbus
16#A023	Subbus E/A-Datenkommunikationsfehler

Wert	Beschreibung
Statuscodes mit den Diagnoseklassen Betriebszustand (0xC10A), Warnung (0xC10E), Fehler (0xC10F)	
16#A024	Datenkommunikationsfehler beim Subbus-Management
16#A030	Subbus-Konfigurationsfehler
16#A031	E/A-Datenkonfiguration doppelt abgebildet
16#A040	Allgemeiner Fehler
16#A041	Subbus-Hardwarefehler
16#A042	Subbus-Firmwarefehler
16#A043	Subbus läuft asynchron (im Bezug auf den sercos Buszyklus)
16#B000	Lokalbus (Systembus der modularen Station)
16#B001	Lokalbus-Slave fehlt
16#B002	Falscher Lokalbus angeschlossen
16#B003	Kompatibler Ersatz (der erwartete Lokalbus-Slave wurde durch einen kompatiblen Slave ersetzt)
16#B004	Fehler bei der Anzahl der Lokalbus-Teilnehmer (zur viele Teilnehmer)
16#B010	Teilnehmerfehler
16#B012	Anwendung auf dem Lokalbus-Slave nicht bereit
16#B013	Lokalbus-Teilnehmerreset
16#B014	Lokalbus-Teilnehmer Parametrierung fehlt oder fehlerhaft
16#B020	Kommunikationsfehler
16#B021	Lokalbusfehler - Timeout (Unterbrechung)
16#B022	Mehrfacher Übertragungsfehler auf dem Lokalbus
16#B023	Lokalbus E/A-Datenkommunikationsfehler
16#B024	Datenkommunikationsfehler beim Lokalbus-Management
16#B025	Ringtopologie unterbrochen
16#B030	Konfigurationsfehler auf dem Lokalbus
16#B031	E/A-Datenkonfiguration doppelt abgebildet
16#B032	Lokalbus asynchron zum überlagerten Bus
16#B040	Allgemeiner Fehler
16#B041	Hardwarefehler auf dem Lokalbus
16#B042	Firmwarefehler auf dem Lokalbus
16#B043	Lokalbus läuft asynchron (im Bezug auf den sercos Buszyklus)
16#F000	Zusätzliche Funktionen

Value	Description
Status codes with diagnosis class "procedure command specific state" (0xC10C)	
16#A100	S-0-1508.x.67 Sub-bus Tunnel Command
16#A101	S-0-1508.x.67 Sub-bus Tunnel Command failed. Unsupported Sub-bus Tunnel PDU version

Value	Description
Status codes with diagnosis class "procedure command specific state" (0xC10C)	
16#A102	S-0-1508.x.67 Sub-bus Tunnel Command failed. Unsupported Sub-bus SDU Type
16#A103	S-0-1508.x.67 Sub-bus Tunnel Command failed. Unsupported Sub-bus Tunnel SDU version
16#A104	S-0-1508.x.67 Sub-bus Tunnel Command failed. Sub-bus slave index wrong
16#A105	S-0-1508.x.67 Sub-bus Tunnel Command failed. Wrong length of SDU
16#A106	S-0-1508.x.67 Sub-bus Tunnel Command failed. Wrong length of Command SDU
16#A107	S-0-1508.x.67 Sub-bus Tunnel Command failed. Answer PDU too large
16#A108	S-0-1508.x.67 Sub-bus Tunnel Command failed. Buffer overflow
16#A120	The procedure command S-0-1508.x.80 Sub-bus reconfiguration procedure command is in process
16#A200	The procedure command S-0-1500.x.21 Local bus reconfiguration procedure command is in process
16#BC00	S-0-1500.x.12 Rearrangement of IO resource
16#BC01	Forbidden insertion/replacement - An attempt was made to insert/replace a generic IO function group with an invalid/forbidden slot number
16#BC02	Forbidden insertion/replacement - An attempt was made to insert/replace the function group IO bus coupler (S-0-1500.x.0)
16#BC03	Forbidden insertion/replacement - An attempt was made to insert/replace an undefined generic IO function group
16#BC04	Forbidden insertion/replacement - The list of inserted/replaced function groups (S-0-1500.x.4/11) contains duplicate entries
16#BC05	Forbidden insertion/replacement - The list of inserted/replaced function groups (S-0-1500.x.4/11) is not sorted in an increasing order (slot number, data block number)
16#BC0F	Forbidden insertion/replacement - Manufacturer-specific reason

11.5.32 IL_FBUS_COMMUNICATION_STATE

Kurzbeschreibung Der Aufzählungstyp **IL_FBUS_COMMUNICATION_STATE** dient dazu, den Status der Feldbus-Kommunikation darzustellen.

Element	Wert	Beschreibung
COMM_STATE_NOT_CONFIGURED	16#00000001	Kommunikation ist nicht konfiguriert
COMM_STATE_STOP	16#00000002	Kommunikation befindet sich im Stopp-Zustand
COMM_STATE_IDLE	16#00000003	Kommunikation befindet sich im Idle-Zustand
COMM_STATE_OPERATE	16#00000004	Kommunikation im zyklischen Betrieb

Tab. 11-98: Aufzählungstyp **IL_FBUS_COMMUNICATION_STATE**

11.6 Hilfsfunktionen (Utilities)

11.6.1 IL_SIIIAttribute

Kurzbeschreibung Die Funktion **IL_SIIIAttribute** dient der Aufschlüsselung der Elemente des Sercos-Attributes.

Zielsystem	Bibliothek/ Ordner
MLC	RIL_SercosIII/ Utilities
MTX	RIL_SercosIII/ Utilities
MLD (IL 2G)	RIL_SercosIII/ Utilities

Tab. 11-99: Referenztabelle

Schnittstellenbeschreibung

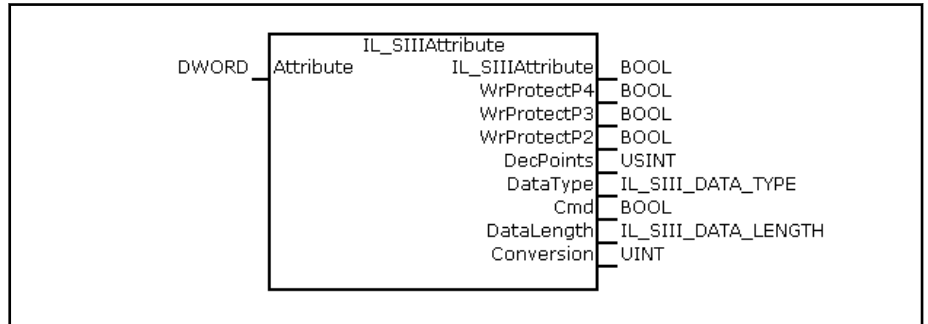


Abb. 11-22: Funktion IL_SIIIAttribute

	Name	Datentyp	Beschreibung
VAR_INPUT	Attribute	DWORD	Sercos-Attribut
VAR_OUPUT	IL_SIIIAttribute	BOOL	Immer TRUE
	WrProtectP4	BOOL	Betriebsdatum des Parameters im Betriebsmodus (Sercos-Phase 4 - P4) schreibgeschützt
	WrProtectP3	BOOL	Betriebsdatum des Parameters in Sercos-Phase 3 (P3) schreibgeschützt
	WrProtectP2	BOOL	Betriebsdatum des Parameters im Parametriermodus (Sercos-Phase 2 - P2) schreibgeschützt
	DecPoints	USINT	Anzahl der Nachkommastellen des Betriebsdatums des Parameters
	DataType	IL_SIII_DATA_TYPE	Datentyp und Anzeigeformat des Betriebsdatums des Parameters IL_SIII_DATA_TYPE, Seite 427
	Cmd	BOOL	Betriebsdatum ruft Kommandofunktion auf
	DataLength	IL_SIII_DATA_LENGTH	Datenlänge des Betriebsdatums des Parameters IL_SIII_DATA_LENGTH, Seite 427,
	Conversion	UINT	Bewertungsfaktor des Betriebsdatums des Parameters

Tab. 11-100: Schnittstelle IL_SIIIAttribute

Sercos-Attribut Die folgende Tabelle gibt den Aufbau des Sercos-Attributs wieder.

Bit	Schnittstellenvariable	Beschreibung
30:	WrProtectP4	Schreibschutz des Parameter-Betriebsdatums im Betriebsmodus (Sercos-Phase 4)
		0: Betriebsdatum ist beschreibbar 1: Betriebsdatum ist schreibgeschützt

Bit	Schnittstellenvariable	Beschreibung	
29:	WrProtectP3	Schreibschutz des Betriebsdatums des Parameters (Sercos-Phase 3)	
		0: Betriebsdatum ist beschreibbar 1: Betriebsdatum ist schreibgeschützt	
28:	WrProtectP2	Schreibschutz des Parameter-Betriebsdatums im Parametriermodus (Sercos-Phase 2)	
		0: Betriebsdatum ist beschreibbar 1: Betriebsdatum ist schreibgeschützt	
27..24:	DecPoints	Die Nachkommastellen legen die Stellung des Kommas für die Anzeige und Eingabe des betreffenden Betriebsdatums fest.	
		0000: Keine Nachkommastellen 1111: 15 Nachkommastellen (Maximum)	
23:		Reserviert	
22..20:	DataType	Datentyp und Anzeigeformat wird benutzt, um das Betriebsdatum sowie den minimalen - und maximalen Eingabewert in das richtige Anzeigeformat umzuwandeln.	
		Datentyp	Anzeigeformat
		000: Binärzahl	binär
		001: Vorzeichenlose ganze Zahl	Dezimalzahl ohne Vorzeichen
		010: Ganze Zahl	Dezimalzahl mit Vorzeichen
		011: Vorzeichenlose ganze Zahl	hexadezimal
		100: Erweiterter Zeichensatz	Text
		101: Vorzeichenlose ganze Zahl	Parameter-Identnummerformat (Sercos-Norm, 2 Byte)
		110: Gleitkommazahl	Dezimalzahl mit Vorzeichen
		111: Vorzeichenlose ganze Zahl	Parameter-Identnummerformat (Sercos-Norm, 4 Byte)
19:	Cmd	Die Funktion des Betriebsdatums zeigt an, dass mit diesem Betriebsdatum eine Kommandofunktion im Antrieb aufgerufen wird.	
		0: Betriebsdatum oder Parameter 1: Kommando	

Bit	Schnittstellenvariable	Beschreibung
18..16:	DataLength	Die Datenlänge wird von der Steuerung für das ordnungsgemäße Beenden der Datenübertragung über den Servicekanal benötigt. 000: reserviert 001: Betriebsdatum ist 2 Byte lang 010: Betriebsdatum ist 4 Byte lang 011: reserviert 100: Variable Länge mit 1-Byte-Daten 101: Variable Länge mit 2-Byte-Daten 110: Variable Länge mit 4-Byte-Daten 111: reserviert
15..0:	Conversion	Der Bewertungsfaktor ist eine vorzeichenlose ganze Zahl, die dazu dient numerische Daten in ein Anzeigeformat umzuwandeln. Der Bewertungsfaktor ist auf den Wert von 1 gesetzt, wenn er für die Datenanzeige nicht verwendet wird.

Tab. 11-101: Elemente des Sercos-Attributs

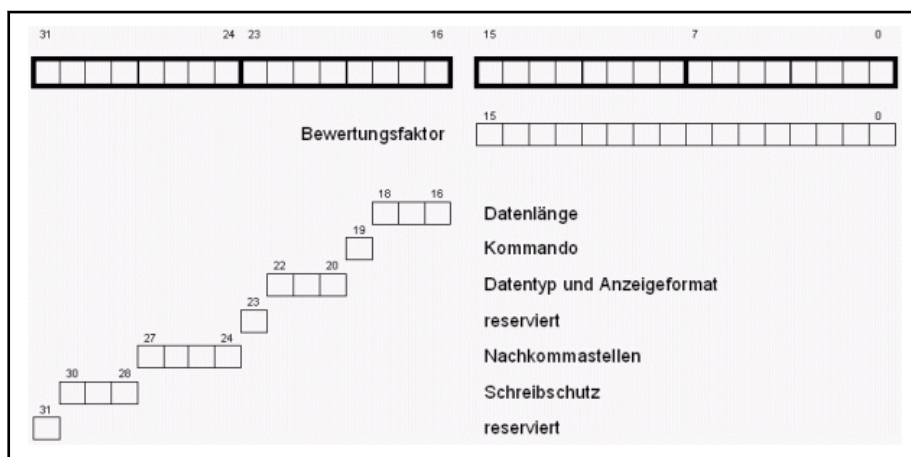


Abb. 11-23: Aufbau des Sercos-Attributs

Funktionsbeschreibung Die Funktion IL_SIIIAttribute interpretiert das am Eingang "Attribute" angelegte Sercos-Attribut. Die Gültigkeit der Elemente an den entsprechenden Ausgängen wird durch den Ausgang "IL_SIIIAttribute" signalisiert.

Fehlerbehandlung Für diese Funktion ist keine Fehlerbehandlung verfügbar.

11.6.2 IL_SIIIElementsToldn

Kurzbeschreibung Die Funktion IL_SIIIElementsToldn fasst die Einzelelemente einer IDN in einen MB_IDN-Wert zusammen.

Zielsystem	Bibliothek/ Ordner
MLC	RIL_SercosIII/ Utilities
MTX	RIL_SercosIII/ Utilities
MLD (IL 2G)	RIL_SercosIII/ Utilities

Tab. 11-102: Referenztable

Schnittstellenbeschreibung

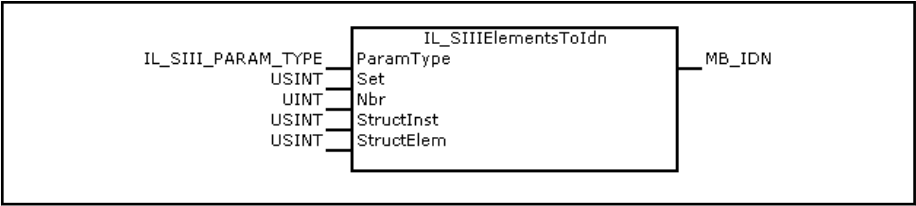


Abb. 11-24: Funktion IL_SIIIElementsToIdn

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	ParamType	IL_SIII_PARAM_TY- PE	Standard (S) oder produktspezifische (P)-IDN IL_SIII_PARAM_TYPE auf Seite 428
	Set	USINT	Parametersatz
	Nbr	UINT	Datenblocknummer
	StructInst	USINT	Strukturinstanz
	StructElem	USINT	Strukturelement
Rückgabewert		MB_IDN	Sercos-IDN Bei ungültigem Eingangsparameter ist IL_SIIIElementsToIdn = 0

Tab. 11-103: Schnittstelle IL_SIIIElementsToIdn

11.6.3 IL_SIIIGetSercosPhase

Kurzbeschreibung Mit der Funktion IL_SIIIGetSercosPhase kann die Sercos-Phase eines Sercos-Busmasters gelesen werden.

Zielsystem	Bibliothek/ Ordner
MLC	RIL_SercosIII/Utilities
MTX	RIL_SercosIII/Utilities
MLD (IL 2G)	RIL_SercosIII/Utilities

Tab. 11-104: Referenztablelle

Schnittstellenbeschreibung

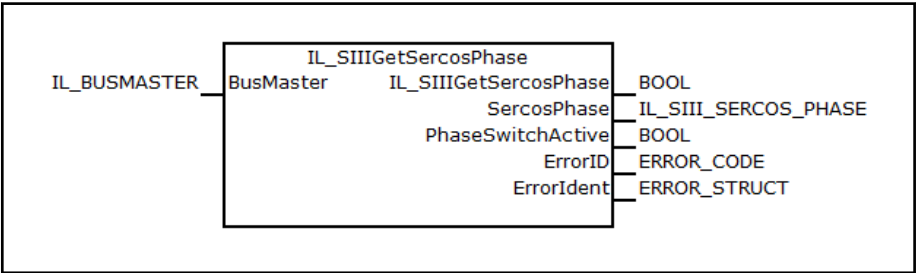


Abb. 11-25: Funktion IL_SIIIGetSercosPhase

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	BusMaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable BusMaster unterschieden. Wird nur ein Master betrieben, so hat dieser die Master ID = IL_BUSMASTER_0. IL_BUSMASTER, Seite 429
VAR_OUTPUT	IL_SIIIGetSercosPhase	BOOL	Rückgabewert der Funktion: TRUE: es trat ein Fehler auf, FALSE: Fehlerfreie Abarbeitung
	SercosPhase	IL_SI-II_SERCOS_PHASE	Aktive Sercos-Phase IL_SIII_SERCOS_PHASE, Seite 428
	PhaseSwitchActive	BOOL	TRUE: Sercos-Phasenschaltung derzeit aktiv, FALSE: Keine Sercos-Phasenschaltung aktiv.
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall.
	ErrorIdent	ERROR_STRUCT	Detaildiagnose

Tab. 11-105: Schnittstelle IL_SIIIGetSercosPhase

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an Execute. Je nach Zielsystem kann es sein, dass nicht alle Werte erlaubt sind.

Tab. 11-106: Min-/Max und Default-Werte der Eingänge von IL_SIIIGetSercosPhase

Funktionsbeschreibung

Je nach Zielsystem können mehrere Sercos-Master vorhanden sein. Die Zuordnung des jeweiligen Masters zu den Werten aus "IL_BUSMASTER" ist der Dokumentation des jeweiligen Zielsystems zu entnehmen. Gibt es bei dem Zielsystem nur einen Sercos-Master, so besitzt dieser den Wert "IL_BUSMASTER_0".

Der Baustein versucht die Sercos-Phase des jeweiligen Bus Masters zu lesen. Ist der Sercos-Master noch nicht vollständig initialisiert, oder die Phase kann nicht gelesen werden, dann wird ein Fehler erzeugt. Dabei wird für die Phase der Wert "IL_SIII_PHASE_INIT" und für den Ausgang "PhaseSwitchActive" der Wert FALSE zurückgegeben.

Der Ausgang "PhaseSwitchActive" zeigt an, ob gerade eine Phasenschaltung aktiv ist. Dabei wird dieser Ausgang während dem gesamten Phasenschalten aktiv gesetzt. Wenn z. B. von Sercos-Phase 2 nach Sercos-Phase 4 geschaltet wird, so ist dieser Ausgang bereits in Sercos-Phase 2 aktiv, aber auch in Sercos-Phase 3 und bis zum Erreichen der Sercos-Phase 4.

Liest man also kontinuierlich die Sercos-Phase mit dieser Funktion, kann man das Phasenschalten verfolgen. Erst wenn der Ausgang "PhaseSwitchActive" FALSE annimmt ist die endgültige Phase erreicht.

Fehlerbehandlung

Der Funktionsbaustein generiert Fehler aus der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) :

ErrorID	Additional1	Additional2	
INPUT_RANGE_ERROR	16#00000002	16#00000001	Der angegebene Wert für Eingang BusMaster ist ungültig.
STATE_MACHINE_ERROR	16#00000006	16#xxxxxxx	Es ist ein interner Fehler aufgetreten, Steuerung neu starten. Bei mehrmaligem Auftreten den Service kontaktieren und Additional2 Fehlercode für weitere Diagnose melden.
SYSTEM_ERROR	16#00000016	16#xxxxxxx	Es ist ein interner Fehler aufgetreten, Steuerung neu starten. Bei mehrmaligem Auftreten den Service kontaktieren und Additional2 Fehlercode für weitere Diagnose melden.
RESOURCE_ERROR	16#00002912	16#00000000	Die Funktion wird auf diesem Zielsystem nicht unterstützt.
RESOURCE_ERROR	16#00002915	16#00000000	Die Funktionalität ist in der Steuerung noch nicht verfügbar, z. B. ist der Sercos-Bus noch nicht initialisiert.

Tab. 11-107: Fehlercodes der Funktion IL_SIIIGetSercosPhase in F_RELATED_TABLE := 16#0170

11.6.4 IL_SIIISetSercosPhase

Kurzbeschreibung Mit dem Funktionsbaustein IL_SIIISetSercosPhase kann die Sercos-Phase eines Sercos-Busmasters gesetzt werden.

Zielsystem	Bibliothek/ Ordner
MLC	RIL_SercosIII/Utilities
MTX	RIL_SercosIII/Utilities
MLD (IL 2G)	RIL_SercosIII/Utilities

Tab. 11-108: Referenztable

Schnittstellenbeschreibung

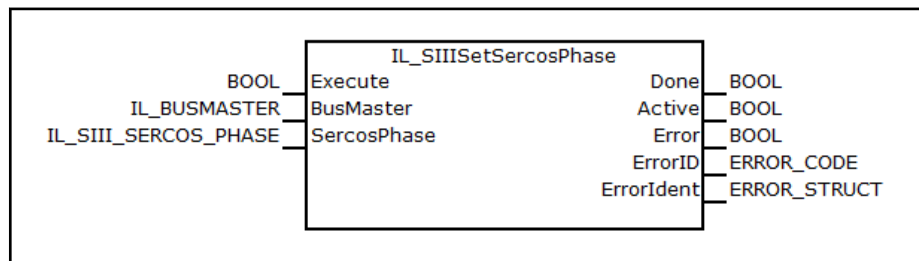


Abb. 11-26: Funktionsbaustein IL_SIIISetSercosPhase

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (einmalig, flankengesteuert)
	BusMaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable BusMaster unterschieden. Wird nur ein Master betrieben, so hat dieser die Master ID = IL_BUSMASTER_0. IL_BUSMASTER, Seite 429
	SercosPhase	IL_SI-II_SERCOS_PHASE	Sercos-Phase in die geschaltet werden soll IL_SIII_SERCOS_PHASE, Seite 428
VAR_OUTPUT	Done	BOOL	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten gültig
	Active	BOOL	Bearbeitung läuft, Ausgangsdaten sind ungültig

E/A-Typ	Name	Datentyp	Beschreibung
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCT	Detalldiagnose

Tab. 11-109: Schnittstelle IL_SIIISetSercosPhase

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke an Execute. Je nach Zielsystem kann es sein, dass nicht alle Werte erlaubt sind
SercosPhase	IL_SII_SERCOS_PHASE	IL_SIII_PHASE_0	IL_SIII_PHASE_NRT	IL_SIII_PHASE_4	Steigende Flanke an Execute. Je nach Zielsystem kann es sein, dass nicht alle Werte erlaubt sind

Tab. 11-110: Min-/Max und Default-Werte der Eingänge IL_SIIISetSercosPhase

Fehlerbehandlung Der Funktionsbaustein generiert Fehler aus der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) :

ErrorID	Additional1	Additional2	
INPUT_RANGE_ERROR	16#00000002	16#00000001	Der angegeben Wert für Eingang BusMaster ist ungültig
STATE_MACHINE_ERROR	16#00000006	16#xxxxxxx	Es ist ein interner Fehler aufgetreten, Steuerung neu starten. Bei mehrmaligem Auftreten den Service kontaktieren und Additional2 Fehlercode für weitere Diagnose melden.
SYSTEM_ERROR	16#00000016	16#00002914	Es ist ein interner Fehler aufgetreten, Steuerung neu starten. Bei mehrmaligem Auftreten den Service kontaktieren
ACCESS_ERROR	16#00002910	16#00000000	Die Phasenschaltung ist bereits aktiv, es kann nicht noch ein Phasenschalten kommandiert werden
INPUT_RANGE_ERROR	16#00002911	16#00000000	Der angegebene Wert für Eingang SercosPhase ist ungültig oder das System unterstützt diese Phase nicht
RESOURCE_ERROR	16#00002912	16#00000000	Der Baustein wird auf diesem Zielsystem nicht unterstützt
RESOURCE_ERROR	16#00002915	16#00000000	Die Funktionalität ist in der Steuerung noch nicht verfügbar, z. B. ist der Sercos-Bus noch nicht initialisiert
RESOURCE_ERROR	16#00002916	16#00000000	Phasenschaltung ist abgeschlossen, aber die gewünschte Phase wurde nicht erreicht. Dies kann durch einen Fehler während des Phasenschaltens erfolgen
ACCESS_ERROR	16#00002917	16#00000000	Die Phasenschaltung kann im Moment nicht vorgenommen werden. Dies kann z. B. dadurch entstehen, wenn sich Antriebe noch in Leistung befinden

Tab. 11-111: Fehlercodes Funktionsbaustein IL_SIIISetSercosPhase in F_RELATED_TABLE := 16#0170

Funktionsbeschreibung Beim Eingang "SercosPhase" ist der Wert "IL_SIII_PHASE_INIT" nicht erlaubt. Es wird dann der Fehlercode 16#00002911 zurückgeliefert. Dieser

Fehler kann aber auch auftreten, wenn ein Zielsystem die geforderte Phase nicht unterstützt.

Je nach Zielsystem können mehrere Sercos-Master vorhanden sein. Die Zuordnung des jeweiligen Masters zu den Werten aus "IL_BUSMASTER" ist der Dokumentation des jeweiligen Zielsystems zu entnehmen. Gibt es bei dem Zielsystem nur einen Sercos-Master, so besitzt dieser den Wert "IL_BUSMASTER_0".

Der Baustein versucht die Sercos-Phase des jeweiligen Busmasters zu schalten. Bei erfolgreicher Ausführung wird der "Done"-Ausgang gesetzt, ansonsten der "Error"-Ausgang. Die Dauer, bis einer der beiden Ausgänge gesetzt wird, variiert dabei stark in Abhängigkeit der Anzahl an Teilnehmern, die am betreffenden Bus angeschlossen sind. Wird hier eine Zeitüberwachung gewünscht, so ist dies durch den Anwender selbst umzusetzen.

11.6.5 IL_SIIStringToIdn

Kurzbeschreibung Die Funktion IL_SIIStringToIdn wandelt einen String der Form "S-0-0001.0.0" oder in Kurzformen "S-0-0001" oder "S1" in einen MB_IDN-Wert.

Zielsystem	Bibliothek/ Ordner
MLC	RIL_SercosIII/ Utilities
MTX	RIL_SercosIII/ Utilities
MLD (IL 2G)	RIL_SercosIII/ Utilities

Tab. 11-112: Referenztablelle

Schnittstellenbeschreibung

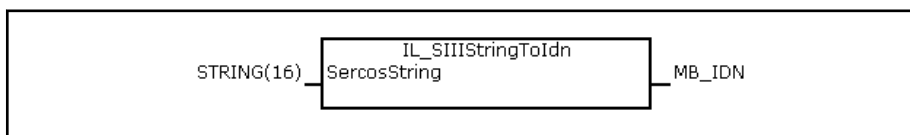


Abb. 11-27: Funktion IL_SIIStringToIdn

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	SercosString	STRING(16)	Parameterstring "S-0-0001.0.0" oder in Kurzformen "S-0-0001" oder "S1"
Rückgabewert		MB_IDN	Sercos IDN Bei ungültigem Eingangsparameter ist IL_SIIStringToIdn = -1

Tab. 11-113: Schnittstelle IL_SIIStringToIdn

Funktionsbeschreibung Die Funktion IL_SIIStringToIdn wandelt einen Sercos-Parameterstring mit folgenden Stringelementen (Formatierung: "X-Y-ZZZZ.SI.SE" oder "X-Y-ZZZZ" oder "XZ"):

- SercosType "X": "S", "P", "s", "p" (bei MLC auch "A", "C", "K", "M", "N", "O", "a", "c", "k", "m", "n", "o")
- SercosSet "Y": "0"..."7" (optional nur bei "XZ")
- SercosNo "ZZZZ": "0"..."4095" (nur bei "X-Y-ZZZZ" und "XZ" ohne führende Nullen möglich)
- Strukturinstanz "SI": "0"..."255" (optional, zusammen mit SE)
- Strukturelement "SE": "0"..."255" (optional, zusammen mit SI)
- Trennzeichen: "-" bzw. "."

11.6.6 IL_SIII_DATA_LENGTH

Kurzbeschreibung Der Aufzählungstyp unterstützt die Auswertung des Sercos-Attributes bezüglich Datenlänge und Unterscheidung Listen- oder Einzelparameter.

Bibliothek	Ordner
RIL_SercosIII	Utilities

Tab. 11-114: Bibliothekszuordnung

Programm:

```

TYPE IL_SIII_DATA_LENGTH :
(
  FIX_2_BYTE := 16#01, (* Operation data is two octets long *)
  FIX_4_BYTE := 16#02, (* Operation data is four octets long *)
  FIX_8_BYTE := 16#03, (* Operation data is eight octets long *)
  VAR_1_BYTE := 16#04, (* Variable length with one-octet data strings *)
  VAR_2_BYTE := 16#05, (* Variable length with two-octet data strings *)
  VAR_4_BYTE := 16#06, (* Variable length with four-octet data strings *)
  VAR_8_BYTE := 16#07 (* Variable length with eight-octet data strings *)
)USINT;
END_TYPE

```

Funktionsbeschreibung Der Aufzählungstyp unterstützt die Auswertung des Sercos-Attributes bezüglich Datenlänge (`_x_Byte` -> `x=Datenlänge`) und Unterscheidung Listen- (`VAR_x_BYTE`) oder Einzelparameter (`FIX_x_BYTE`).

Der Datentyp wird am Ausgang "DataLength" der Funktion [IL_SIIIAttribute](#), Seite 418, verwendet.

11.6.7 IL_SIII_DATA_TYPE

Kurzbeschreibung Der Aufzählungstyp unterstützt die Auswertung des Sercos-Attributes bezüglich Datentyp und Anzeigeformat.

Bibliothek	Ordner
RIL_SercosIII	Utilities

Tab. 11-115: Bibliothekszuordnung

Programm:

```

TYPE IL_SIII_DATA_TYPE :
(
  BINARY := 16#00, (* binary number *)
  UDEC := 16#01, (* unsigned integer, display Decimal *)
  DEC := 16#02, (* signed integer, display Decimal *)
  HEX := 16#03, (* unsigned integer, display Hexadecimal *)
  TEXT := 16#04, (* extended character set *)
  IDN := 16#05, (* unsigned integer, display IDN "S-0-0000.0.0" *)
  FLOAT := 16#06, (* floating-point number *)
  STIME := 16#07 (* Sercos time *)
)USINT;
END_TYPE

```

Funktionsbeschreibung Der Aufzählungstyp unterstützt die Auswertung des Sercos-Attributes bezüglich Datentyp und Anzeigeformat.

Der Datentyp wird am Ausgang "DataType" der Funktion [IL_SIIIAttribute](#), Seite 418, verwendet.

11.6.8 IL_SIII_ELEMENT

Kurzbeschreibung

Der Aufzählungstyp `IL_SIII_ELEMENT` beschreibt die Sercos-Elemente eines Parameters.

Bibliothek	Ordner
RIL_SercosIII	Utilities

Tab. 11-116: Bibliothekszuordnung

Name	Wert	Beschreibung
IL_STATUS	1	Service Kanal Data Status
IL_NAME	2	Name
IL_ATTRIBUTE	3	Attribute
IL_UNIT	4	Einheit
IL_MINVALUE	5	Minimalwert
IL_MAXVALUE	6	Maximalwert
IL_OPDATA	7	Datum

Tab. 11-117: Elemente des Aufzählungstyps IL_SIII_ELEMENT

11.6.9 IL_SIII_PARAM_TYPE

Kurzbeschreibung Der Aufzählungstyp IL_SIII_PARAM_TYPE unterscheidet zwischen Standard-IDN und produktspezifischer IDN.

Bibliothek	Ordner
RIL_SercosIII	Utilities

Tab. 11-118: Bibliothekszuordnung

Name	Wert	Beschreibung
IL_S_PARAM	0	Standard-IDN
IL_P_PARAM	1	Produktspezifische IDN

Tab. 11-119: Elemente des Aufzählungstyps IL_SIII_PARAM_TYPE

11.6.10 IL_SIII_SERCOS_PHASE

Typdefinitionen Die folgende Definition beschreibt den Aufzählungstyp IL_SIII_SERCOS_PHASE.

Die Typ-Definitionen sind in der Bibliothek enthalten. Der Wert IL_SIII_PHASE_INIT wird immer dann angegeben, wenn die Sercos-Phase nicht ermittelt werden kann, z. B. wenn der Sercos-Bus noch nicht initialisiert ist oder aufgrund eines Fehlers die Phase nicht gelesen werden kann.

Der Datentyp "IL_SIII_SERCOS_PHASE" ist wie folgt definiert:

Aufzählungselement	Wert	Beschreibung
IL_SIII_PHASE_0	0	Sercos-Phase 0
IL_SIII_PHASE_1	1	Sercos-Phase 1
IL_SIII_PHASE_2	2	Sercos-Phase 2
IL_SIII_PHASE_3	3	Sercos-Phase 3
IL_SIII_PHASE_4	4	Sercos-Phase 4

Aufzählungselement	Wert	Beschreibung
IL_SIII_PHASE_NRT	7	Keine Sercos-Phase aktiv, IP-Kommunikation möglich
IL_SIII_PHASE_INIT	16#FFFFFFFF	Ungültige Sercos-Phase oder Phase nicht ermittelbar

Tab. 11-120: Definition der Aufzählungselemente des Datentyps
IL_SIII_SERCOS_PHASE

11.6.11 IL_BUSMASTER

Kurzbeschreibung Der Aufzählungstyp IL_BUSMASTER aus der RIL_FieldbusTypes.library dient dazu, den jeweiligen Sercos III-Master auszuwählen.

Momentan wird nur der Onboard-Master unterstützt.

Element	Wert	Beschreibung
IL_BUSMASTER_0	0	OnBoard Master
IL_BUSMASTER_1	1	nicht unterstützt
IL_BUSMASTER_2	2	nicht unterstützt
IL_BUSMASTER_3	3	nicht unterstützt
IL_BUSMASTER_4	4	nicht unterstützt

Tab. 11-121: Aufzählungstyp IL_BUSMASTER

12 RMB_SercosIII_Util.library

12.1 Einführung und Übersicht

Die Bibliothek RMB_SercosIII_Util.library enthält Funktionsbausteine zur Ansteuerung von Sercos III Teilnehmern.

Zielsysteme Die Bibliothek kann auf folgenden Systemen eingesetzt werden:

Zielsystem	Bibliothek
MLC/MTX	RMB_SercosIII_Util
MLD mit IndraLogic 2G ab 20VRS	RMB_SercosIII_Util

Antrieb als SPS-Gerät

In den Systemen MLC und MLD werden Antriebsachsen über Funktionsbausteine nach "PLCopen" angesteuert.

Neben diesen Funktionsbausteinen existiert darüber hinaus das "AchsInterface". Das Achs-Interface bündelt und erweitert die PLCopen-Bewegungsfunktionsbausteine und stellt ein einfach zu bedienendes Interface für die Antriebsfunktionalität zur Verfügung.

Die Achsanzahl ist begrenzt und die Motion-Achsen beanspruchen Ressourcen der Steuerung.

Die folgenden Funktionsbausteine "PlcDrive" dienen zur vereinfachten Achsansteuerung über den Sercos-Bus. Dabei wird der Antrieb als SPS-Gerät angelegt (siehe auch Antrieb als SPS-Gerät).

Es wird keine Motion-Achsinstanz belegt und die benötigten Ressourcen der Steuerung sind minimal. Es werden die Betriebsarten des Antriebes (Interpolation im Antrieb) genutzt. Die Anforderungen an den jeweiligen Antrieb sind minimal. Antriebe anderer Hersteller (z.B. ohne Wichtungssystem) können über diese Funktionsbausteine genutzt werden.

Einschränkungen bei Nutzung dieser Funktionsbausteine:

- Für diese Achsen ist nur die Nutzung **einer** Betriebsart vorgesehen. Diese ist über das Engineeringtool des Gerätes einzustellen.
- Für die Konfiguration der zyklischen Kommunikation und Diagnose der SPS-Geräte stehen in IndraWorks Dialoge zur Verfügung. Die Inbetriebnahme des Gerätes ist allerdings nicht direkt in IndraWorks auf dem Geräteknoten möglich. Für Rexrothgeräte kann über eine Direktverbindung IndraWorks Ds genutzt werden (siehe Hilfe Antrieb als SPS-Gerät). Für Antriebe anderer Hersteller ist das jeweilige Tool zu verwenden.



Bei Rexroth-Geräten wird die Parametrierung mit "IndraWorks Ds" empfohlen. Dort sind gerätespezifische Inbetriebnahmedialoge realisiert.

Über den Eintrag "IndraWorks Ds hier" (unterhalb des Sercos-Knoten, Geräteknoten, rechte Maustaste) kann "IndraWorks Ds" innerhalb der Engineering-Suite IndraWorks gestartet werden (Routing von der Steuerung zum Gerät erfolgt automatisch).



Ein Beispielprogramm "AchsInterface für Sercos SPS-Geräte" ist als GAT Plug-In "SercosPlcDriveInterface" verfügbar. Dieses Beispielprogramm kann auch ohne GAT verwendet werden. Siehe dazu die Plug-In Dokumentation.

Das Beispielprogramm umfasst folgende Funktionsbausteine bzw. Betriebsarten:

- MB_SIIIPlcDriveVelocityType01 - Geschwindigkeitsregelung
- MB_SIIIPlcDriveTorqueType01 - Momentenregelung
- MB_SIIIPlcDrivePosInterpolationType01 - Antriebsinterne Interpolation (nur in Version 1 des Plug-In)
- MB_SIIIPlcDrivePositioningType01 - Antriebsgeführte Positionierung (ab Version 2 des Plug-In)
- MB_SIIIPlcDriveEfcFrequencyType01 - Frequenzsteuerung

Übersicht über den Inhalt der Bibliothek

Im Ordner "Drives" sind Funktionsbausteine für die Sercos Grund-Betriebsarten "Geschwindigkeitsregelung", "Momentenregelung", "Antriebsinterne Interpolation" und "Antriebsgeführtes Positionieren" realisiert.

Ordner "PowerSupply": Funktionsbausteine für die Ansteuerung von Versorger-Geräten mit Sercos.

Ordner "RexrothEFC": ein Funktionsbaustein für die Betriebsart "Frequenzsteuerung" und Hilfsfunktionen für den Rexroth Frequenzumrichter EFC mit Sercos.

Im Ordner "RexrothHydraulicDrives" sind Funktionsbausteine für die Hydraulik-Betriebsarten "Volumenstromregelung", "Druck-/Volumenstromregelung", "Druck-/Schwenkwinkelregelung", "Ventildirektsteuerung", "Drehmoment-/Ventildirektsteuerung", "Drehmoment-/Volumenstromregelung", "Druckregelung-/Ventildirektsteuerung" und "Geschwindigkeitsregelung mit ablösender Momentenregelung" realisiert.

Ordner "RexrothSytronix": Funktionsbausteine für die Ansteuerung von Sytronix-Geräten DRn 5020 und FcP5020 mit Sercos.

Anwendungsgebiete und Funktionsbausteine von RMB_SercosIII_Util

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine	Beschreibung
Drives (FBs zur vereinfachten Achsansteuerung)	
MB_SIIIPlcDriveType01, Seite 436	Basis-Funktionsbaustein, der durch die weiteren Funktionsbausteine erweitert wird.
MB_SIIIPlcDriveVelocityType01, Seite 441	Erweitert MB_SIIIPlcDriveType01. Der Baustein ist implementiert für die Betriebsart "Geschwindigkeitsregelung"
MB_SIIIPlcDriveTorqueType01, Seite 444	Erweitert MB_SIIIPlcDriveType01. Der Baustein ist implementiert für die Betriebsart "Momentenregelung"
MB_SIIIPlcDrivePositioningType01, Seite 448	Erweitert MB_SIIIPlcDriveType01. Der Baustein ist implementiert für die Betriebsart "Antriebsgeführtes Positionieren"
MB_SIIIPlcDrivePosInterpolationType01, Seite 454	Erweitert MB_SIIIPlcDriveType01. Der Baustein ist implementiert für die Betriebsart "Antriebsinterne Interpolation"
Mit Datentypen:	
MB_SIII_CONTROLWORD, Seite 458	Entspricht dem Parameter Master-Steuerwort S-0-0134

MB_SIII_STATUSWORD, Seite 459	Entspricht dem Parameter Master-Statuswort S-0-0135
MB_SIII_DRIVE_SHUTDOWN_MODE, Seite 460	Ermöglicht die Auswahl des Verhaltens beim Abschalten der Antriebsfreigabe
MB_SIII_HOMING_MODE, Seite 460	Ermöglicht die Auswahl des Verhaltens beim Herstellen des Maßbezuges
MB_SIII_POSCTRLWORD, Seite 460	Entspricht dem Parameter Positioniersteuerwort S-0-0346
MB_SIII_IN_PROC_VELOCITY, Seite 461	Enthält die Prozess-Eingangsdaten des Funktionsbausteins MB_SIIIPlcDriveVelocityType01
MB_SIII_OUT_PROC_VELOCITY, Seite 461	Enthält die Prozess-Ausgangsdaten des Funktionsbausteins MB_SIIIPlcDriveVelocityType01
MB_SIII_IN_PROC_TORQUE, Seite 462	Enthält die Prozess-Eingangsdaten des Funktionsbausteins MB_SIIIPlcDriveTorqueType01
MB_SIII_OUT_PROC_TORQUE, Seite 462	Enthält die Prozess-Ausgangsdaten des Funktionsbausteins MB_SIIIPlcDriveTorqueType01
MB_SIII_IN_PROC_POSITIONING, Seite 462	Enthält die Prozess-Eingangsdaten des Funktionsbausteins MB_SIIIPlcDrivePositioningType01
MB_SIII_OUT_PROC_POSITIONING, Seite 462	Enthält die Prozess-Ausgangsdaten des Funktionsbausteins MB_SIIIPlcDrivePositioningType01
MB_SIII_IN_PROC_POS_INTERPOLATION, Seite 463	Enthält die Prozess-Eingangsdaten des Funktionsbausteins MB_SIIIPlcDrivePosInterpolationType01
MB_SIII_OUT_PROC_POS_INTERPOLATION, Seite 463	Enthält die Prozess-Ausgangsdaten des Funktionsbausteins MB_SIIIPlcDrivePosInterpolationType01
PowerSupply	
MB_SIIIPowerSupplyDriveIntegrated, Seite 465	Kann den internen Zustand der Stromversorgung identifizieren, die Übergänge zwischen den Zuständen der Stromversorgung handhaben, die Stromversorgung in den gewünschten Zustand bringen und die Betriebsart steuern.
MB_SIIIPowerSupplyStandalone, Seite 473	Die Funktion entspricht MB_SIIIPowerSupplyDriveIntegrated. Zusätzlich kann der Funktionsbaustein einen Stromversorgungsfehler zurücksetzen und Diagnoseinformationen über den Zustand der Stromversorgung bereitstellen
MB_SIIIPowerSupplyDcDcIslandGrid, Seite 480	Funktion entspricht MB_SIIIPowerSupplyStandalone für Versorger-Geräte für die erweiterten Versorgerfunktionen DC/DC-Steller oder Inselnetzbetrieb
Mit Datentypen:	
MB_POWERSUPPLY_ACTUAL_MODE, Seite 487	Aufzählung der Zustände der Zustandsmaschine einer Stromversorgung
MB_POWERSUPPLY_ERROR_MODE, Seite 488	Aufzählung, in welchen Zustand die Stromversorgung schaltet, wenn sie einen internen Fehlerzustand verlässt
MB_POWERSUPPLY_MODE, Seite 487	Aufzählung der Betriebsarten einer Stromversorgung

MB_POWERSUPPLY_TARGET_MODE, Seite 489	Aufzählung der verfügbaren Zielmodi für die Funktionsbausteine MB_SIIIPowerSupplyDriveIntegrated und MB_SIIIPowerSupplyStandalone. Der Zielmodus entscheidet über den zu erreichenden Endzustand durch die Stromversorgung, sowie über den zurückzulegenden Weg zum Erreichen des Endzustandes
MB_PS_TARGET_MODE_DC_DC_ISLAND_GRID, Seite 490	Aufzählung der verfügbaren Zielmodi für den Funktionsbaustein MB_SIIIPowerSupplyDcDcIslandGrid. Der Zielmodus entscheidet über den zu erreichenden Endzustand durch die Stromversorgung, sowie über den zurückzulegenden Weg zum Erreichen des Endzustandes
MB_POWERSUPPLY_FUNCTION, Seite 490	Aufzählung der erweiterten Versorgerfunktionen für den Funktionsbaustein MB_SIIIPowerSupplyDcDcIslandGrid
RexrothEFC	
MB_SIIIEfcCodeToIdn, Seite 491	Rechnet einen Funktions-/Parametercode des EFC in die zugehörige Sercos-Parameter IDN um.
MB_SIIIPlcDriveEfcFrequencyType01, Seite 492	Dient dazu einen Rexroth Frequenzumrichter EFC in der Betriebsart "Frequenzsteuerung" zu betreiben.
Mit Datentypen:	
MB_EFC_FUNCTION_CODE, Seite 496	Beschreibt den Funktionscode des Rexroth Frequenzumrichter EFC
MB_SIII_IN_PROC_EFC_FREQ, Seite 496	Enthält die Prozess-Eingangsdaten des Funktionsbausteins MB_SIIIPlcDriveEfcFrequencyType01
MB_SIII_OUT_PROC_EFC_FREQ, Seite 496	Enthält die Prozess-Ausgangsdaten des Funktionsbausteins MB_SIIIPlcDriveEfcFrequencyType01
MB_SIII_EFC_CONTROLWORD, Seite 497	Entspricht dem Parameter P-0-1098, Register H0.00 "Register für Kommunikationssteuerung" des EFC
MB_SIII_EFC_STATUSWORD, Seite 497	Entspricht dem Parameter P-0-1098.0.1, Register H0.01 "Register für Rückmeldung Kommunikationsstatus" des EFC
RexrothHydraulicDrives	
MB_SIIIPlcDrivePressureFlowControl, Seite 503	Realisiert die Betriebsart "Druck- und Volumenstromregelung" für Rexroth HydraulicDrives
MB_SIIIPlcDrivePressureSwivelAngleControl, Seite 508	Realisiert die Betriebsart "Druck- und Schwenkwinkelregelung" für Rexroth HydraulicDrives
MB_SIIIPlcDrivePressureSwivelAngleSlave, Seite 513	Realisiert die Betriebsart "Druck- und Schwenkwinkelregelung (für Slave)" für Rexroth HydraulicDrives
MB_SIIIPlcDriveDirectValveControl, Seite 517	Realisiert die Betriebsart "Ventildirektsteuerung" für Rexroth HydraulicDrives
MB_SIIIPlcDriveTorqueDirectValveControl, Seite 521	Realisiert die Betriebsart "Ventildirektsteuerung" für Rexroth HydraulicDrives
MB_SIIIPlcDriveTorqueFlowControl, Seite 527	Realisiert die Betriebsart "Drehmoment-/Kraftregelung/Ventildirektsteuerung" für Rexroth HydraulicDrives
MB_SIIIPlcDrivePressureDirectValveControl, Seite 532	Realisiert die Betriebsart "Druckregelung/Ventildirektsteuerung" für Rexroth HydraulicDrives
MB_SIIIPlcDriveVelocityAltCtrl, Seite 537	Realisiert die Betriebsart "Geschwindigkeitsregelung mit ablösender Kraftregelung" für Rexroth HydraulicDrives

Mit Datentypen:	
MB_SIII_HYDR_CONTROL_MODE, Seite 542	Aufzählungstyp, enthält die möglichen Betriebsarten von Rexroth HydraulicDrives
MB_SIII_IN_PROC_FLOW, Seite 543	Eingangsstruktur für den Eingang "InProc" des Funktionsbaustein MB_SIIIPlcDriveFlowControl
MB_SIII_OUT_PROC_FLOW, Seite 543	Ausgangsstruktur für den Ausgang "OutProc" des Funktionsbaustein MB_SIIIPlcDriveFlowControl
MB_SIII_IN_PROC_PRESSURE_FLOW, Seite 543	Eingangsstruktur für den Eingang "InProc" des Funktionsbaustein MB_SIIIPlcDrivePressureFlowControl
MB_SIII_OUT_PROC_PRESSURE_FLOW, Seite 544	Ausgangsstruktur für den Ausgang "OutProc" des Funktionsbaustein MB_SIIIPlcDrivePressureFlowControl
MB_SIII_IN_PROC_SWIVEL, Seite 545	Eingangsstruktur für den Eingang "InProc" des Funktionsbaustein MB_SIIIPlcDrivePressureSwivelAngleControl
MB_SIII_OUT_PROC_SWIVEL, Seite 545	Ausgangsstruktur für den Ausgang "OutProc" des Funktionsbaustein MB_SIIIPlcDrivePressureSwivelAngleControl
MB_SIII_IN_PROC_SWIVEL_SLAVE, Seite 546	Eingangsstruktur für den Eingang "InProc" des Funktionsbaustein MB_SIIIPlcDrivePressureSwivelAngleSlave
MB_SIII_OUT_PROC_SWIVEL_SLAVE, Seite 546	Ausgangsstruktur für den Ausgang "OutProc" des Funktionsbaustein MB_SIIIPlcDrivePressureSwivelAngleSlave
MB_SIII_IN_PROC_SPOOL, Seite 547	Eingangsstruktur für den Eingang "InProc" des Funktionsbaustein MB_SIIIPlcDriveDirectValveControl
MB_SIII_OUT_PROC_SPOOL, Seite 547	Ausgangsstruktur für den Ausgang "OutProc" des Funktionsbaustein MB_SIIIPlcDriveDirectValveControl
MB_SIII_IN_PROC_TORQUE_SPOOL, Seite 548	Eingangsstruktur für den Eingang "InProc" des Funktionsbaustein MB_SIIIPlcDriveTorqueDirectValveControl
MB_SIII_OUT_PROC_TORQUE_SPOOL, Seite 548	Ausgangsstruktur für den Ausgang "OutProc" des Funktionsbaustein MB_SIIIPlcDriveTorqueDirectValveControl
MB_SIII_IN_PROC_TORQUE_FLOW, Seite 548	Eingangsstruktur für den Eingang "InProc" des Funktionsbaustein MB_SIIIPlcDriveTorqueFlowControl
MB_SIII_OUT_PROC_TORQUE_FLOW, Seite 549	Ausgangsstruktur für den Ausgang "OutProc" des Funktionsbaustein MB_SIIIPlcDriveTorqueFlowControl
MB_SIII_IN_PROC_PRESSURE_SPOOL, Seite 549	Eingangsstruktur für den Eingang "InProc" des Funktionsbaustein MB_SIIIPlcDrivePressureDirectValveControl
MB_SIII_OUT_PROC_PRESSURE_SPOOL, Seite 550	Ausgangsstruktur für den Ausgang "OutProc" des Funktionsbaustein MB_SIIIPlcDrivePressureDirectValveControl
MB_SIII_IN_PROC_VEL_ALT_CTRL, Seite 550	Eingangsstruktur für den Eingang "InProc" des Funktionsbaustein MB_SIIIPlcDriveVelocityAltCtrl
MB_SIII_OUT_PROC_VEL_ALT_CTRL, Seite 551	Ausgangsstruktur für den Ausgang "OutProc" des Funktionsbaustein MB_SIIIPlcDriveVelocityAltCtrl
RexrothSytronix	
MB_SIIIDRn5020, Seite 551	Ermöglicht die Druck- und Schwenkwinkelsteuerung des auf EFC 5610 basierenden Sytronix DRn 5020 über Sercos-Kommunikation

MB_SIIIFcP5020, Seite 557	Ermöglicht die Drucksteuerung und Volumenstromregelung des auf EFC 5610 basierenden Sytronix FcP 5020 über Sercos-Kommunikation
Mit Datentypen:	
MB_SIII_DRN_ENABLE, Seite 561	Aufzählung wird vom Eingang DRnEnable des Funktionsbaustein MB_SIII_DRn5020 verwendet
MB_SIII_DRN_VAR_VEL_ENABLE, Seite 562	Aufzählung wird vom Eingang DRnVarVelocityEnable des Funktionsbaustein MB_SIII_DRn5020 verwendet
MB_SIII_FCP_MASTERSLAVE, Seite 562	Aufzählung wird vom Eingang MasterSlave des Funktionsbaustein MB_SIII_FcP5020 verwendet
MB_SIII_FCP_PARAGROUP, Seite 562	Aufzählung wird vom Eingang pQParameterGroup und vom Ausgang pQParameterGroupStatus des Funktionsbaustein MB_SIII_FcP5020 verwendet
MB_SIII_FCP_DUALPUMPSTATUS, Seite 562	Aufzählung wird vom Ausgang TwoPointDualPumpStatus des Funktionsbaustein MB_SIII_FcP5020 verwendet

Tab. 12-1: RMB_SercosIII_Util-Bibliothek

12.2 Drives

12.2.1 MB_SIIIPlcDriveType01

Kurzbeschreibung Der Funktionsbaustein MB_SIIIPlcDriveType01 bildet die Basis für die folgenden Funktionsbausteine, die einer bestimmten Betriebsart zugeordnet sind. Es wird immer die Hauptbetriebsart aktiviert.



Der Funktionsbaustein "MB_SIIIPlcDrive" wurde mit Version 14V16 in "MB_SIIIPlcDriveType01" umbenannt und die Schnittstelle wurde optimiert.

Zielsystem	Bibliothek
MLC/MTX ab 14V10	RMB_SercosIII_Util
MLD mit IndraLogic 2G ab 20VRS	RMB_SercosIII_Util

Tab. 12-2: Referenztabelle

Schnittstellenbeschreibung

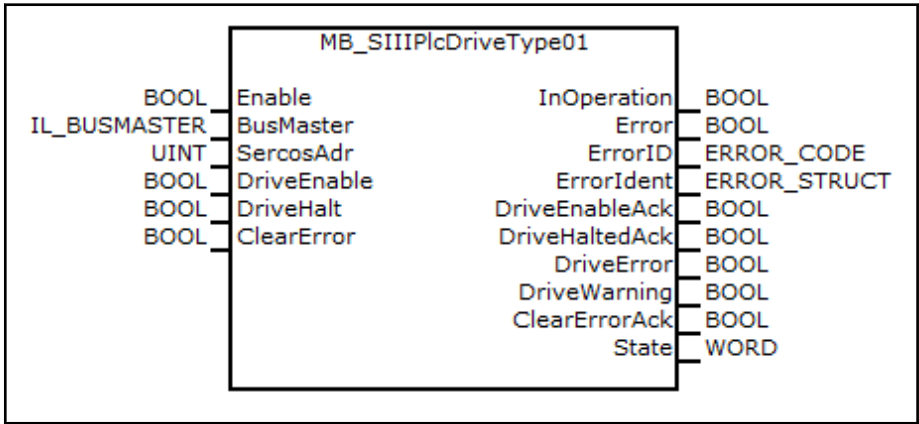


Abb. 12-1: Funktionsbaustein MB_SIIIPlcDriveType01

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (pegelgesteuert).
	Busmaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable BusMaster unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 429
	SercosAdr	UINT	Sercos III-Adresse des Slave-Teilnehmers.
	DriveEnable	BOOL	Freigabe des Antriebes (AH/AF) S-0-0134, Master-Steuerwort, Reglerfreigabe, Bit 14
	DriveHalt	BOOL	Stoppen des Antriebes (AH) S-0-0134, Master-Steuerwort, Antrieb Halt, Bit 13
	ClearError	BOOL	Löschen eines Fehlers im Antrieb durch das Kommando Fehler löschen (S-0-0099, C0500 Reset Zustandsklasse 1). Nach gelöschtem Fehler wird die angewählte Freigabe wieder aktiviert.
VAR_OUT-PUT	InOperation	BOOL	Baustein arbeitet
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCTURE	Detailldiagnose Fehlerbehandlung, Seite 440
	DriveEnableAck	BOOL	Antriebsfreigabe quittiert, S-0-0135, Antriebs-Status, Bit 14/15
	DriveHaltedAck	BOOL	Antrieb Halt quittiert, S-0-0135, Antriebs-Status, Bit 4
	DriveError	BOOL	Antriebsfehler, S-0-0135, Antriebs-Status, Bit 13
	DriveWarning	BOOL	Antriebswarnung, S-0-0135, Antriebs-Status, Bit 12
	ClearErrorAck	BOOL	Bestätigung, dass das Kommando Fehler löschen ausgeführt wurde, S-0-0099, C0500 Reset Zustandsklasse 1
	State	WORD	Aktueller Status des Antriebes. Siehe unten.
PROPERTY			
IN	DriveShutdown-ModeProp	MB_SII_DRIVE_SHUTDOWN_MODE	Auswahl für das Abschalten der Antriebsfreigabe bei negativer Flanke an "DriveEnable". Optionen: Abbremsen oder Momentenfreischaltung. MB_SIII_DRIVE_SHUTDOWN_MODE auf Seite 460
IN	StatusWord	WORD	Eingangsvariable %IW des Statuswortes S-0-0135 am Element "wControlword" anschließen.
OUT	ControlWord	WORD	Ausgangsvariable %QW des Steuerwortes S-0-0134 am Element "wControlword" anschließen.

Tab. 12-3: Schnittstelle MB_SIIPlcDriveType01

Ausgang "State"

Im Ausgang "State" ist eine Grobklassifizierung von Diagnosen binär kodiert. Die Bits sind in der RIL_FieldbusTypes.library als Globale Konstanten "FieldbusStates" enthalten.

BIT-Nr.	Name	Beschreibung
0	IL_RBS_SLAVE_NOT_REACHABLE	Slave nicht existent
1	IL_RBS_SLAVE_NOT_READY	Slave nicht bereit
2	IL_RBS_SLAVE_CONFIG_ERROR	Slave-Konfigurationsfehler
3	IL_RBS_SLAVE_DIAGNOSIS	Slave-Diagnose verfügbar
4	IL_RBS_SLAVE_ERROR	Slave-Fehler verfügbar
5	IL_RBS_CON_OFF	Mindestens eine zyklische Verbindung läuft nicht

Min-/Max und Default-Werte

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
Busmaster	IL_BUSMASTER			0	Steigende Flanke an Enable
SercosAdr	UINT	0 (MLD: lokale Achse) 1 (andere Systeme)	Systemspezi-fisch	0	Kontinuierlich
DriveEnable	BOOL			FALSE	Kontinuierlich
DriveHalt	BOOL			FALSE	Kontinuierlich
ClearError	BOOL			FALSE	Kontinuierlich
DriveShutdown-ModeProp	MB_SIII_DRIVE_SHUT-DOWN_MODE			DRIVE_SHUT-DOWN_TOR-QUE_MAX	Schreiben der Property
StatusWord	WORD			0	Schreiben der Property

Tab. 12-4: Min-/Max und Default-Werte der Eingänge MB_SIIIPlcDriveType01

Funktionsbeschreibung

Der Funktionsbaustein interpretiert das Statuswort S-0-0135 des Antriebes und bildet das Steuerwort S-0-0134, über das der Antrieb die nötigen Befehle erhält. Diese beiden Parameter müssen zyklisch konfiguriert werden.

Der Eingang "ClearError" aktiviert das Kommando "Fehler löschen" S-0-0099.

Mit dem Eingang "DriveShutdownModeProp" wird das Verhalten beim Rücksetzen des Einganges "DriveEnable" definiert. Die Ansteuerung der Bits 14 und 15 im S-0-0134 wird damit beeinflusst.



Einige Ein- und Ausgänge sind als "Property" implementiert, um diese in den abgeleiteten Funktionsbausteinen flexibler anzuwenden.

Der Eingang "DriveShutDownMode" wird in einem abgeleiteten Funktionsbaustein nur dann implementiert, wenn das entsprechende Gerät die Modi unterstützt.

Die Prozessdaten (StatusWord und ControlWord) werden in den abgeleiteten Funktionsbausteinen in Strukturen zusammengefasst.

Konfiguration der zyklischen Daten im Sercos:

Um die zyklische Kommunikation des Funktionsbausteins mit dem Antrieb aufzubauen müssen Parameter auf die jeweiligen Ein- und Ausgangsvariablen kopiert werden. Die Konfiguration der zyklisch übertragenen Parameter erfolgt im Projektextplorer unterhalb der Sercosknoten. Dazu den Knoten "Drive" unterhalb des jeweiligen Gerätes anwählen.

Hier müssen zunächst im Reiter "Allgemeine Ein- und Ausgänge" die zyklischen Parameter konfiguriert werden.

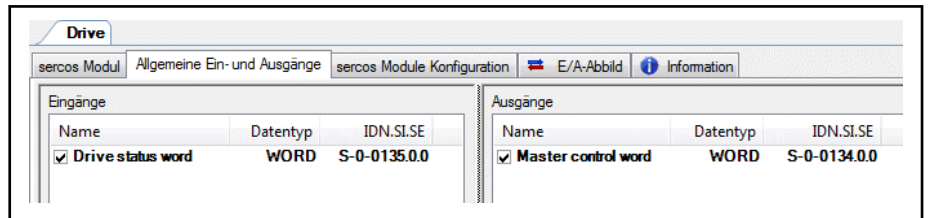


Abb. 12-2: Konfiguration der zyklischen Parameter

Anschließend werden im Reiter "E/A-Abbild" die entsprechenden Ein- und Ausgangsvariablen zugewiesen. Da Status- und Kontrollwort als "Property" implementiert sind, ist es notwendig Zwischenvariablen anzulegen und diese im E/A-Abbild zu verknüpfen. Das Statuswort wird dann **vor** dem Aufruf des Funktionsbausteins auf die Property "Statusword" kopiert. Das Kontrollwort wird **nach** dem Aufruf des Funktionsbausteins auf die Property "Controlword" kopiert.

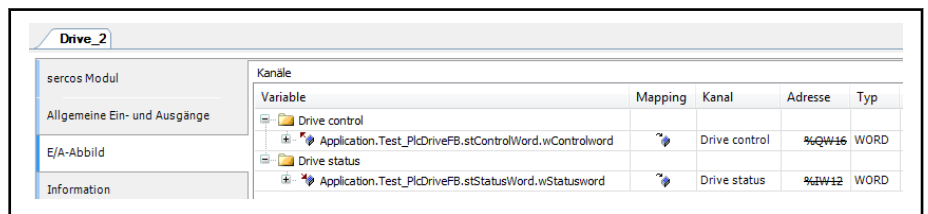


Abb. 12-3: Mappen der Ein- und Ausgangsvariablen

Parametrierung des Antriebes:



Bei Rexroth-Geräten wird die Parametrierung mit "IndraWorks Ds" empfohlen. Dort sind gerätespezifische Inbetriebnahmedialoge realisiert.

Über den Eintrag "IndraWorks Ds hier" (unterhalb des Sercos-Knoten, Geräteknoten, rechte Maustaste) kann "IndraWorks Ds" innerhalb der Engineering-Suite IndraWorks gestartet werden (Routing von der Steuerung zum Gerät erfolgt automatisch).

Der Funktionsbaustein MB_SIIIPlcDriveType01 erfordert nur die Parametrierung der Hauptbetriebsart S-0-0032. Je nach gewählter Betriebsart sind weitere Parameter zu konfigurieren.

Eine Grundparametrierung des Antriebes (Grenzwerte usw.) ist auf jeden Fall erforderlich.

Nutzung des MB_SIIIPlcDrive-Type01 als Basis für eigene Funktionsbausteine:

Zur Implementation eigener Funktionsbausteine für weitere Betriebsarten kann der MB_SIIIPlcDriveType01 verwendet werden.

In diesem Beispiel wird der optionale Eingang "DriveShutdownMode" auch implementiert.

Dazu sind die beiden folgenden Strukturen zu implementieren:

MY_SIII_IN_PROC_Xxx

```
TYPE MY_SIII_IN_PROC_Xxx :
STRUCT
  Statusword: MB_SIII_STATUSWORD;           // S-0-0135 status word
```

```

        MyActValue: DINT:= 16#0000000000000000; // S-0-xxxx actual value
        // define more inputs here
    END_STRUCT
END_TYPE

```

MY_SIII_OUT_PROC_Xxx

```

TYPE MY_SIII_OUT_PROC_Xxx :
STRUCT
    Controlword: MB_SIII_CONTROLWORD; // S-0-0134 control word
    MYCmdValue: DINT:= 16#0000000000000000; // S-0-xxxx setpoint
    // define more outputs here
END_STRUCT
END_TYPE

```

Umsetzung des abgeleiteten Funktionsbausteins:

MY_SIIISpecialPlcDrive (Deklaration)

```

FUNCTION BLOCK MY_SIIISpecialPlcDrive EXTENDS MB_SIIIPlcDriveType01
VAR_INPUT
    DriveShutdownMode: MB_SIII_DRIVE_SHUTDOWN_MODE := DRIVE_SHUTDOWN_TORQUE_MAX;
    // Shut down torque free or immediately stop

    MyCmd: REAL; // setpoint
    InProc: MY_SIII_IN_PROC_Xxx; // Input structure
END_VAR
VAR_OUTPUT
    MyAct: REAL; // actual value
    OutProc: MY_SIII_OUT_PROC_Xxx; // Output structure
END_VAR

```

MY_SIIISpecialPlcDrive (Implementation)

```

// The following inputs and outputs have to be copied to properties of the base FB,
// because we do not want these in- and outputs visible. Statusword and Controlword are here structures.
// DriveShutdownMode is not used in all FBs
SUPER^.StatusWord := InProc.Statusword.wStatusword;
SUPER^.DriveShutdownModeProp := DriveShutdownMode;
SUPER^();
OutProc.Controlword.wControlword := SUPER^.ControlWord;

```

Dazu sind die beiden folgenden Methoden zu implementieren:

Initialisierung:

```
METHOD PUBLIC mCalcInit: BOOL
```

Mit `mCalcInit := FALSE`; wird der Abschluss der Initialisierung gemeldet.

Zyklischer Aufruf:

```
METHOD mCalcInOp
```

Fehlerbehandlung Zusätzlich zu den Fehlern der intern genutzten Funktionsbausteine IL_SIII-Command, IL_SIIIMasterState und IL_SIIIRemoteSlaveState sind folgende Fehler des Bausteines in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#00000002	SercosAdr	Eingangswert des Parameters "SercosAdr" ist außerhalb des gültigen Bereiches
STATE_MACHINE_ERROR	16#00000006	State	Fehler in der Zustandsmaschine
RESOURCE_ERROR	16#00002980	16#00000000	Sercos-Bus ist nicht im Betriebsmodus P4
RESOURCE_ERROR	16#00002981	State	Gerät ausgefallen während "InOperation". Siehe oben.

Tab. 12-5: Fehlercodes FB MB_SIIIPlcDriveType01

12.2.2 MB_SIIIPlcDriveVelocityType01

Kurzbeschreibung Der Funktionsbaustein MB_SIIIPlcDriveVelocityType01 dient dazu einen Antrieb in der Betriebsart "Geschwindigkeitsregelung" zu betreiben.



Der Funktionsbaustein "MB_SIIIVelocityPlcDrive" wurde mit Version 14V16 in "MB_SIIIPlcDriveVelocityType01" umbenannt und die Schnittstelle wurde optimiert.



Ein Beispielprogramm "AchsInterface" zu diesem Funktionsbaustein ist als GAT Plug-In "SercosPlcDriveInterface" verfügbar. Dieses Beispielprogramm kann auch ohne GAT verwendet werden. Siehe dazu die Plug-In Dokumentation.

Zielsystem	Bibliothek
MLC/MTX ab 14V10	RMB_SercosIII_Util
MLD mit IndraLogic 2G ab 20VRS	RMB_SercosIII_Util

Tab. 12-6: Referenztablelle

Schnittstellenbeschreibung

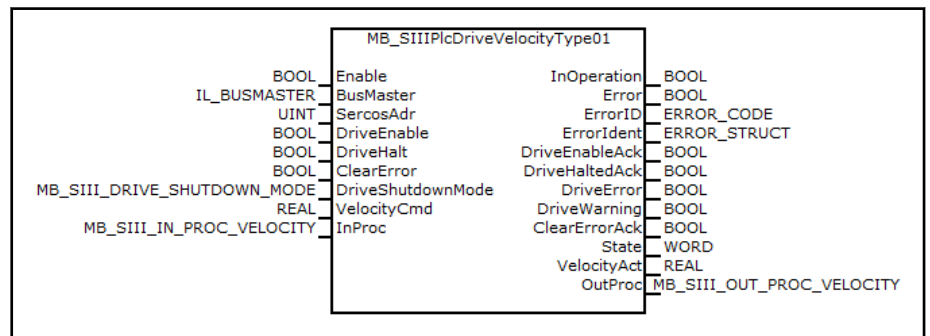


Abb. 12-4: Funktionsbaustein MB_SIIIPlcDriveVelocityType01

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (pegelgesteuert).
	Busmaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable BusMaster unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 429
	SercosAdr	UINT	Sercos III-Adresse des Slave-Teilnehmers.
	DriveEnable	BOOL	Freigabe des Antriebes (AH/AF) S-0-0134, Master-Steuerwort, Reglerfreigabe, Bit 14
	DriveHalt	BOOL	Stoppen des Antriebes (AH) S-0-0134, Master-Steuerwort, Antrieb Halt, Bit 13
	ClearError	BOOL	Löschen eines Fehlers im Antrieb durch das Kommando Fehler löschen (S-0-0099, C0500 Reset Zustandsklasse 1). Nach gelöschtem Fehler wird die angewählte Freigabe wieder aktiviert.

E/A-Typ	Name	Datentyp	Beschreibung
	DriveShutdown-Mode	MB_SI-II_DRIVE_SHUT-DOWN_MODE	Auswahl für das Abschalten der Antriebsfreigabe bei negativer Flanke an "DriveEnable". Optionen: Abbremsen oder Momentenfreischaltung. MB_SIII_DRIVE_SHUTDOWN_MODE auf Seite 460
	VelocityCmd	REAL	Geschwindigkeits-Sollwert. Einheit z.B. in [U/min], je nach Parametrierung.
	InProc	MB_SI-II_IN_PROC_VELOCITY	Prozess-Eingangsdaten %I anschließen. MB_SIII_IN_PROC_VELOCITY auf Seite 461
VAR_OUT-PUT	InOperation	BOOL	Baustein arbeitet
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCTURE	Detaildiagnose Fehlerbehandlung auf Seite 444
	DriveEnableAck	BOOL	Antriebsfreigabe quittiert, S-0-0135, Antriebs-Status, Bit 14/15
	DriveHaltedAck	BOOL	Antrieb Halt quittiert, S-0-0135, Antriebs-Status, Bit 4
	DriveError	BOOL	Antriebsfehler, S-0-0135, Antriebs-Status, Bit 13
	DriveWarning	BOOL	Antriebswarnung, S-0-0135, Antriebs-Status, Bit 12
	ClearErrorAck	BOOL	Bestätigung, dass das Kommando Fehler löschen ausgeführt wurde, S-0-0099, C0500 Reset Zustandsklasse 1
	State	WORD	Aktueller Status des Antriebes.
	VelocityAct	REAL	Geschwindigkeits-Istwert. Einheit z.B. in [U/min], je nach Parametrierung.
	OutProc	MB_SI-II_OUT_PROC_VELOCITY	Prozess-Ausgangsdaten %Q anschließen. MB_SIII_OUT_PROC_VELOCITY auf Seite 461

Tab. 12-7: Schnittstelle MB_SIIIPlcDriveVelocityType01

Ausgang "State" Im Ausgang "State" ist eine Grobklassifizierung von Diagnosen binär kodiert. Die Bits sind in der RIL_FeldbusTypes.library als Globale Konstanten "FieldbusStates" enthalten.

BIT-Nr.	Name	Beschreibung
0	IL_RBS_SLAVE_NOT_REACHABLE	Slave nicht existent
1	IL_RBS_SLAVE_NOT_READY	Slave nicht bereit
2	IL_RBS_SLAVE_CONFIG_ERROR	Slave-Konfigurationsfehler
3	IL_RBS_SLAVE_DIAGNOSIS	Slave-Diagnose verfügbar
4	IL_RBS_SLAVE_ERROR	Slave-Fehler verfügbar
5	IL_RBS_CON_OFF	Mindestens eine zyklische Verbindung läuft nicht

Min-/Max und Default-Werte

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
Busmaster	IL_BUSMASTER			0	Steigende Flanke an Enable
SercosAdr	UINT	0 (MLD: lokale Achse) 1 (andere Systeme)	Systemspezi- fisch	0	Kontinuierlich
DriveEnable	BOOL			FALSE	Kontinuierlich
DriveHalt	BOOL			FALSE	Kontinuierlich
ClearError	BOOL			FALSE	Kontinuierlich
DriveShutdown-Mode	MB_SIII_DRIVE_SHUT-DOWN_MODE			DRIVE_SHUT-DOWN_TOR-QUE_MAX	Kontinuierlich
VelocityCmd	REAL			0.0	Kontinuierlich
InProc	MB_SIII_IN_PROC_VELO-CITY			0	Kontinuierlich

Tab. 12-8: Min-/Max und Default-Werte der Eingänge MB_SIIIPlcDriveVelocityType01

Funktionsbeschreibung

Der Funktionsbaustein MB_SIIIPlcDriveVelocityType01 erweitert den Funktionsbaustein MB_SIIIPlcDriveType01 und besitzt von diesem alle Ein- und Ausgänge sowie dessen Funktionalität.

Zusätzlich hat er noch Ein- und Ausgänge für die Ist- und Sollwerte dieser Betriebsart:

- Der Eingang "VelocityCmd" gibt den Geschwindigkeits-Sollwert der Achse an. Der Wert wird zyklisch über den Ausgang "OutProc.Velocity" und den Parameter S-0-0036 an den Antrieb übergeben.
- Der Ausgang "VelocityAct" gibt den Geschwindigkeits-Istwert der Achse aus. Der Wert wird zyklisch über den Parameter S-0-0036 und den Eingang "InProc.Velocity" vom Antrieb gelesen.

Konfiguration der zyklischen Daten im Sercos:

Um die zyklische Kommunikation des Funktionsbausteins mit dem Antrieb aufzubauen müssen Parameter auf die jeweiligen Ein- und Ausgangsvariablen kopiert werden. Die Konfiguration der zyklisch übertragenen Parameter erfolgt im Projektextplorer unterhalb der Sercosknotens. Dazu den Knoten "Drive" unterhalb des jeweiligen Gerätes anwählen.

Hier müssen zunächst im Reiter „Allgemeine Ein- und Ausgänge“ die zyklischen Parameter konfiguriert werden.

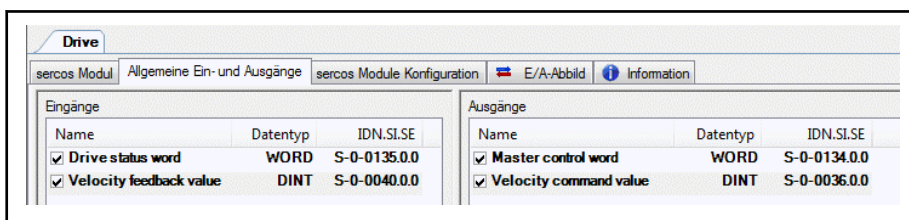


Abb. 12-5: Konfiguration der zyklischen Parameter

Anschließend werden im Reiter "E/A-Abbild" die entsprechenden Ein- und Ausgangsvariablen des Funktionsbausteins zugewiesen.

Drive_2					
sercos Modul					
Allgemeine Ein- und Ausgänge					
E/A-Abbild					
Information					
Kanäle		Variable	Mapping	Kanal	Adresse Typ
		Drive control			
		Application.Test_VelocityPlcDriveFB_SMC.FBTest1.OutProc.Controlword.wControlword		Drive control	400016 WORD
		Velocity command value			
		Application.Test_VelocityPlcDriveFB_SMC.FBTest1.OutProc.Velocity		Velocity command value	400020 DINT
		Drive status			
		Application.Test_VelocityPlcDriveFB_SMC.FBTest1.InProc.Statusword.wStatusword		Drive status	400024 WORD
		Velocity feedback value 1			
		Application.Test_VelocityPlcDriveFB_SMC.FBTest1.InProc.Velocity		Velocity feedback value 1	400028 DINT

Abb. 12-6: Mappen der Ein- und Ausgangsvariablen

Parametrierung des Antriebes:



Bei Rexroth-Geräten wird die Parametrierung mit "IndraWorks Ds" empfohlen. Dort sind gerätespezifische Inbetriebnahmedialoge realisiert.

Über den Eintrag "IndraWorks Ds hier" (unterhalb des Sercos-Knoten, Geräteknoten, rechte Maustaste) kann "IndraWorks Ds" innerhalb der Engineering-Suite IndraWorks gestartet werden (Routing von der Steuerung zum Gerät erfolgt automatisch).

Der Funktionsbaustein MB_SIIIVelocityPlcDriveType01 erfordert die Parametrierung der Hauptbetriebsart S-0-0032 auf "Geschwindigkeitsregelung" (Wert 2#0000.0000.0000.0010).

Die weitere Parametrierung für die Betriebsart "Geschwindigkeitsregelung" (Beschleunigung usw.) ist mit z.B. IndraWorks Ds auszuführen.

Eine Grundparametrierung des Antriebes (Grenzwerte usw.) ist auf jeden Fall erforderlich.

Fehlerbehandlung

Zusätzlich zu den Fehlern der intern genutzten Funktionsbausteine MB_SII-IPlcDriveType01 und IL_SIIISvcRead sind folgende Fehler des Bausteines in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#00000006	State	Fehler in der Zustandsmaschine
INPUT_RANGE_ERROR	16#00002982	Wert des Attributes	Attribut des Parameters S-0-0036 oder S-0-0040 ist nicht 4 Byte dezimal mit Vorzeichen.

Tab. 12-9: Fehlercodes FB MB_SIIIPlcDriveVelocityType01

12.2.3 MB_SIIIPlcDriveTorqueType01

Kurzbeschreibung

Der Funktionsbaustein MB_SIIIPlcDriveTorqueType01 dient dazu einen Antrieb in der Betriebsart "Momentenregelung" zu betreiben.



Der Funktionsbaustein " MB_SIIITorquePlcDrive" wurde mit Version 14V16 in " MB_SIIIPlcDriveTorqueType01" umbenannt und die Schnittstelle wurde optimiert.



Ein Beispielprogramm "AchsInterface" zu diesem Funktionsbaustein ist als GAT Plug-In "SercosPlcDriveInterface" verfügbar. Dieses Beispielprogramm kann auch ohne GAT verwendet werden. Siehe dazu die Plug-In Dokumentation.

Zielsystem	Bibliothek
MLC/MTX ab 14V10	RMB_SercosIII_Util
MLD mit IndraLogic 2G ab 20VRS	RMB_SercosIII_Util

Tab. 12-10: Referenztabelle

Schnittstellenbeschreibung

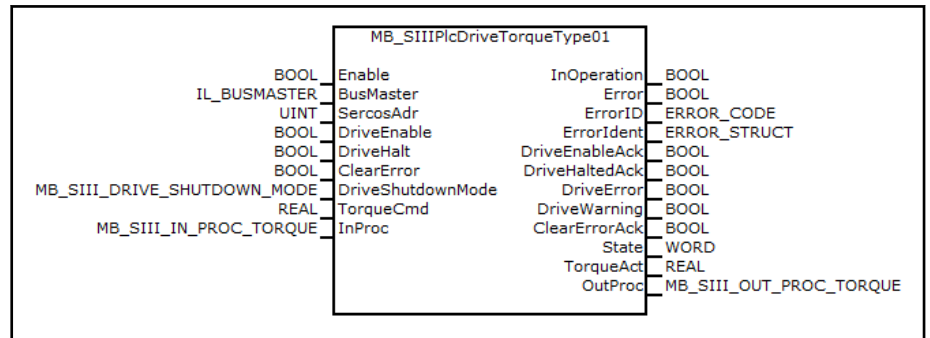


Abb. 12-7: Funktionsbaustein MB_SIIIPlcDriveTorqueType01

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (pegelgesteuert)
	Busmaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable BusMaster unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0.
	SercosAdr	UINT	Sercos III-Adresse des Slave-Teilnehmers.
	DriveEnable	BOOL	Freigabe des Antriebes (AH/AF) S-0-0134, Master-Steuerwort, Reglerfreigabe, Bit 14
	DriveHalt	BOOL	Stoppen des Antriebes (AH) S-0-0134, Master-Steuerwort, Antrieb Halt, Bit 13
	ClearError	BOOL	Löschen eines Fehlers im Antrieb durch das Kommando Fehler löschen (S-0-0099, C0500 Reset Zustandsklasse 1). Nach gelöschtem Fehler wird die angewählte Freigabe wieder aktiviert.
	DriveShutdown-Mode	MB_SIII_DRIVE_SHUTDOWN_MODE	Auswahl für das Abschalten der Antriebsfreigabe bei negativer Flanke an "DriveEnable". Optionen: Abbremsen oder Momentenfreischaltung. MB_SIII_DRIVE_SHUTDOWN_MODE auf Seite 460
	TorqueCmd	REAL	Momenten-Sollwert. Einheit z.B. in [%], je nach Parametrierung.
	InProc	MB_SIII_IN_PROC_TORQUE	Prozess-Eingangsdaten %I anschließen. MB_SIII_IN_PROC_TORQUE auf Seite 462
VAR_OUTPUT	InOperation	BOOL	Baustein arbeitet
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall

E/A-Typ	Name	Datentyp	Beschreibung
	ErrorIdent	ERROR_STRUCTURE	Detaildiagnose Fehlerbehandlung auf Seite 448
	DriveEnableAck	BOOL	Antriebsfreigabe quittiert, S-0-0135, Antriebs-Status, Bit 14/15
	DriveHaltedAck	BOOL	Antrieb Halt quittiert, S-0-0135, Antriebs-Status, Bit 4
	DriveError	BOOL	Antriebsfehler, S-0-0135, Antriebs-Status, Bit 13
	DriveWarning	BOOL	Antriebswarnung, S-0-0135, Antriebs-Status, Bit 12
	ClearErrorAck	BOOL	Bestätigung, dass das Kommando Fehler löschen ausgeführt wurde, S-0-0099, C0500 Reset Zustandsklasse 1
	State	WORD	Aktueller Status des Antriebes. Siehe unten.
	TorqueAct	REAL	Momenten-Istwert. Einheit z.B. in [%], je nach Parametrierung.
	OutProc	MB_SII_OUT_PROC_TORQUE	Prozess-Ausgangsdaten %Q anschließen. MB_SIII_OUT_PROC_TORQUE auf Seite 462

Tab. 12-11: Schnittstelle MB_SIIPlcDriveTorqueType01

Ausgang "State" Im Ausgang "State" ist eine Grobklassifizierung von Diagnosen binär kodiert. Die Bits sind in der RIL_FeldbusTypes.library als Globale Konstanten "FieldbusStates" enthalten.

BIT-Nr.	Name	Beschreibung
0	IL_RBS_SLAVE_NOT_REACHABLE	Slave nicht existent
1	IL_RBS_SLAVE_NOT_READY	Slave nicht bereit
2	IL_RBS_SLAVE_CONFIG_ERROR	Slave-Konfigurationsfehler
3	IL_RBS_SLAVE_DIAGNOSIS	Slave-Diagnose verfügbar
4	IL_RBS_SLAVE_ERROR	Slave-Fehler verfügbar
5	IL_RBS_CON_OFF	Mindestens eine zyklische Verbindung läuft nicht

Min-/Max und Default-Werte

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
Busmaster	IL_BUSMASTER			0	Steigende Flanke an Enable
SercosAdr	UINT	0 (MLD: lokale Achse) 1 (andere Systeme)	Systemspezifisch	0	Kontinuierlich
DriveEnable	BOOL			FALSE	Kontinuierlich
DriveHalt	BOOL			FALSE	Kontinuierlich
ClearError	BOOL			FALSE	Kontinuierlich
DriveShutdown-Mode	MB_SIII_DRIVE_SHUTDOWN_MODE			DRIVE_SHUTDOWN_TORQUE_MAX	Kontinuierlich

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
TorqueCmd	REAL			0.0	Kontinuierlich
InProc	MB_SIII_IN_PROC_TORQUE			0	Kontinuierlich

Tab. 12-12: Min-/Max und Default-Werte der Eingänge MB_SIIIPlcDriveTorqueType01

Funktionsbeschreibung

Der Funktionsbaustein MB_SIIIPlcDriveTorqueType01 erweitert den Funktionsbaustein MB_SIIIPlcDriveType01 und besitzt von diesem alle Ein- und Ausgänge sowie dessen Funktionalität.

Zusätzlich hat er noch Ein- und Ausgänge für die Ist- und Sollwerte dieser Betriebsart:

- Der Eingang "TorqueCmd" gibt den Momenten-Sollwert der Achse an. Der Wert wird zyklisch über den Ausgang "OutProc.Torque" und den Parameter S-0-0080 an den Antrieb übergeben.
- Der Ausgang "TorqueAct" gibt den Momenten-Istwert der Achse aus. Der Wert wird zyklisch über den Parameter S-0-0084 und den Eingang "InProc.Torque" vom Antrieb gelesen.

Konfiguration der zyklischen Daten im Sercos:

Um die zyklische Kommunikation des Funktionsbausteins mit dem Antrieb aufzubauen müssen Parameter auf die jeweiligen Ein- und Ausgangsvariablen kopiert werden. Die Konfiguration der zyklisch übertragenen Parameter erfolgt im Projektexplorer unterhalb der Sercosknoten. Dazu den Knoten "Drive" unterhalb des jeweiligen Gerätes anwählen.

Hier müssen zunächst im Reiter "Allgemeine Ein- und Ausgänge" die zyklischen Parameter konfiguriert werden.

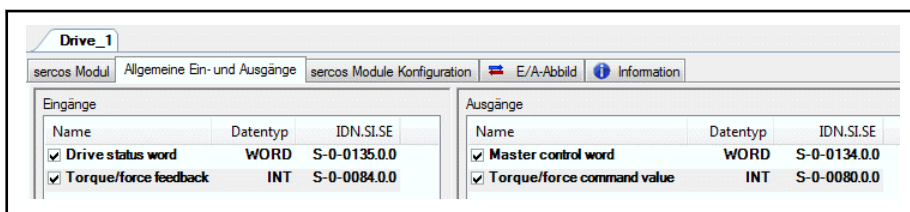


Abb. 12-8: Konfiguration der zyklischen Parameter

Anschließend werden im Reiter „E/A-Abbild“ die entsprechenden Ein- und Ausgangsvariablen des Funktionsbausteins zugewiesen.

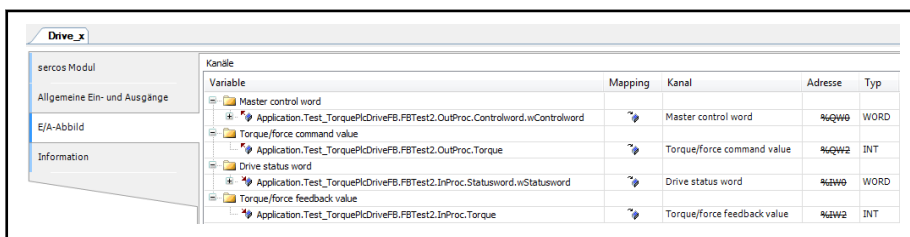


Abb. 12-9: Mappen der Ein- und Ausgangsvariablen

Parametrierung des Antriebes:

Bei Rexroth-Geräten wird die Parametrierung mit "IndraWorks Ds" empfohlen. Dort sind gerätespezifische Inbetriebnahmedialoge realisiert.

Über den Eintrag "IndraWorks Ds hier" (unterhalb des Sercos-Knoten, Geräteknoten, rechte Maustaste) kann "IndraWorks Ds" innerhalb der Engineering-Suite IndraWorks gestartet werden (Routing von der Steuerung zum Gerät erfolgt automatisch).

Der Funktionsbaustein MB_SIIIPlcDriveTorqueType01 erfordert die Parametrierung der Hauptbetriebsart S-0-0032 auf "Momentenregelung" (Wert 2#0000.0000.0000.0001).

Die weitere Parametrierung für die Betriebsart "Momentenregelung" (Rampe usw.) ist mit z.B. IndraWorks Ds auszuführen.

Eine Grundparametrierung des Antriebes (Grenzwerte usw.) ist auf jeden Fall erforderlich.

Fehlerbehandlung

Zusätzlich zu den Fehlern der intern genutzten Funktionsbausteine MB_SIIIPlcDriveType01 und IL_SIIISvcRead sind folgende Fehler des Bausteines in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#00000006	State	Fehler in der Zustandsmaschine
INPUT_RANGE_ERROR	16#00002984	Wert des Attributes	Attribut des Parameters S-0-0080 oder S-0-0084 ist nicht 2 Byte dezimal mit Vorzeichen.

Tab. 12-13: Fehlercodes FB MB_SIIIPlcDriveTorqueType01

12.2.4 MB_SIIIPlcDrivePositioningType01

Kurzbeschreibung

Der Funktionsbaustein MB_SIIIPlcDrivePositioningType01 dient dazu einen Antrieb in der Betriebsart "Antriebsgeführte Positionierung" zu betreiben.



Ein Beispielprogramm "AchsInterface" zu diesem Funktionsbaustein ist als GAT Plug-In "SercosPlcDriveInterface" verfügbar. Dieses Beispielprogramm kann auch ohne GAT verwendet werden.

Zielsystem	Bibliothek
MLC/MTX ab 14V22	RMB_SercosIII_Util
MLD mit IndraLogic 2G ab 20VRS	RMB_SercosIII_Util

Tab. 12-14: Referenztablelle

Schnittstellenbeschreibung

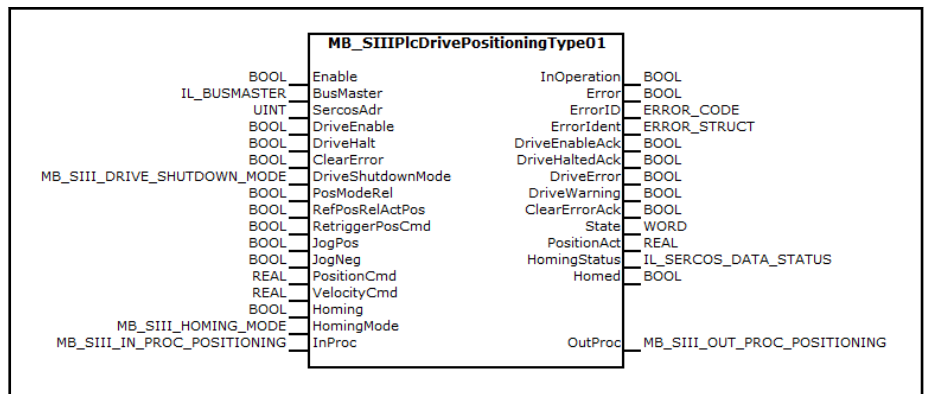


Abb. 12-10: Funktionsbaustein MB_SIIIPlcDrivePositioningType01

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (pegelgesteuert)
	Busmaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable BusMaster unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0 siehe auch IL_BUSMASTER auf Seite 429
	SercosAdr	UINT	Sercos III-Adresse des Slave-Teilnehmers
	DriveEnable	BOOL	Freigabe des Antriebes (AH/AF)
	DriveHalt	BOOL	Stoppen des Antriebes (AH)
	ClearError	BOOL	Löschen eines Fehlers im Antrieb durch das Kommando Fehler löschen S-0-0099. Nach gelöschtem Fehler wird die angewählte Freigabe wieder aktiviert
	DriveShutdown-Mode	MB_SI-II_DRIVE_SHUT-DOWN_MODE	Auswahl für das Abschalten der Antriebsfreigabe bei negativer Flanke an "DriveEnable" Optionen: Abbremsen oder Momentenfreischaltung MB_SIII_DRIVE_SHUTDOWN_MODE auf Seite 460
	PosModeRel	BOOL	Art des Positioniersollwerts FALSE: Absolut TRUE: Relativ
	RefPosRelAct-Pos	BOOL	Bezugspunkt für Positioniersollwerte FALSE: Letzte wirksame Zielposition (S-0-0430) TRUE: Aktiver Lageistwert (S-0-0386)
	Retrigger-PosCmd	BOOL	Letzten Positionierbefehl erneut durchführen
	JogPos	BOOL	Endlosfahren in positiver Richtung
	JogNeg	BOOL	Endlosfahren in negativer Richtung
	PositionCmd	REAL	Sollposition. Einheit z.B. in [Grad], je nach Parametrierung

E/A-Typ	Name	Datentyp	Beschreibung
	VelocityCmd	REAL	Positioniergeschwindigkeit. Einheit z.B. in [U/min], je nach Parametrierung
	Homing	BOOL	Herstellen des Maßbezuges starten
	HomingMode	MB_SIII_HOMING_MODE	S-0-0148, C0600 Kommando Antriebsgeführtes Referenzieren, oder S-0-0447, C0300 Kommando Absolutmaß setzen MB_SIII_HOMING_MODE auf Seite 460
	InProc	MB_SI-II_OUT_PROC_POSITIONING	Prozesseingangsdaten %I anschließen
VAR_OUT-PUT	InOperation	BOOL	Baustein arbeitet
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCT	Detaildiagnose "Fehlerbehandlung" auf Seite 454
	DriveEnableAck	BOOL	Antriebsfreigabe quittiert, aus S-0-0135, Antriebs-Status, Bit 14/15
	DriveHaltedAck	BOOL	Antrieb Halt quittiert, aus S-0-0135, Antriebs-Status, Bit 4
	DriveError	BOOL	Antriebsfehler, aus S-0-0135, Antriebs-Status, Bit 13
	DriveWarning	BOOL	Antriebswarnung, aus S-0-0135, Antriebs-Status, Bit 12
	ClearErrorAck	BOOL	Bestätigung, dass das Kommando Fehler löschen ausgeführt wurde: S-0-0099, C0500 Reset Zustandsklasse 1, Bit 13
	State	WORD	Aktueller Status des Antriebes. Siehe Ausgang State des Funktionsbaustein IL_SIIIRemoteSlaveState
	PositionAct	REAL	Positions-Istwert. Einheit z.B. in [Grad], je nach Parametrierung
	HomingStatus	IL_SERCOS_DATA_STATUS	Status des Referenzierungskommandos IL_SERCOS_DATA_STATUS auf Seite 372
	Homed	BOOL	Antrieb in Referenz, aus Statuswort S-0-0135, Antriebs-Status, Bit 4
	OutProc	MB_SI-II_IN_PROC_POSITIONING	Prozess-Ausgangsdaten %Q anschließen

Tab. 12-15: Schnittstelle MB_SIIIPcDrivePositioningType01

Ausgang "State" Im Ausgang "State" ist eine Grobklassifizierung von Diagnosen binär kodiert. Die Bits sind in der RIL_FeldbusTypes.library als Globale Konstanten "FieldbusStates" enthalten.

BIT-Nr.	Name	Beschreibung
0	IL_RBS_SLAVE_NOT_REACHABLE	Slave nicht existent
1	IL_RBS_SLAVE_NOT_READY	Slave nicht bereit
2	IL_RBS_SLAVE_CONFIG_ERROR	Slave-Konfigurationsfehler

BIT-Nr.	Name	Beschreibung
3	IL_RBS_SLAVE_DIAGNOSIS	Slave-Diagnose verfügbar
4	IL_RBS_SLAVE_ERROR	Slave-Fehler verfügbar
5	IL_RBS_CON_OFF	Mindestens eine zyklische Verbindung läuft nicht

Min-/Max- und Default-Werte

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
Busmaster	IL_BUSMASTER			0	Steigende Flanke an Enable
SercosAdr	UINT	0 (MLD: lokale Achse) 1 (andere Systeme)	Systemspezifisch	0	Kontinuierlich
DriveEnable	BOOL			FALSE	Kontinuierlich
DriveHalt	BOOL			FALSE	Kontinuierlich
ClearError	BOOL			FALSE	Kontinuierlich
DriveShutdown-Mode	MB_SI-II_DRIVE_SHUT-DOWN_MODE			DRIVE_SHUT-DOWN_TORQUE_MAX	Kontinuierlich
PosModeRel	BOOL			FALSE	Kontinuierlich
RefPosRelAct-Pos	BOOL			FALSE	Kontinuierlich
Retrigger-PosCmd	BOOL			FALSE	Kontinuierlich
JogPos	BOOL			FALSE	Kontinuierlich
JogNeg	BOOL			FALSE	Kontinuierlich
PositionCmd	REAL			0.0	Kontinuierlich
VelocityCmd	REAL			0.0	Kontinuierlich
Homing	BOOL			FALSE	Kontinuierlich
HomingMode	MB_SIII_HOMING_MODE			DRIVE_CONTROL-LED_HOMING	Kontinuierlich
InProc	MB_SI-II_IN_PROC_POSITIONING			0	Kontinuierlich

Tab. 12-16: Min-/Max- und Default-Werte der Eingänge MB_SIIIPlcDrivePositioningType01

Funktionsbeschreibung

Der Funktionsbaustein MB_SIIIPlcDrivePositioningType01 erweitert den Funktionsbaustein MB_SIIIPlcDriveType01 und besitzt von diesem Funktionsbaustein alle Ein- und Ausgänge sowie dessen Funktionalität.

Zusätzlich hat er noch Ein- und Ausgänge für die Betriebsarten Antriebsgeführtes Positionieren und Tippen:

- Eingang "PositionCmd": Positionssollwert der Achse. Der Wert wird zyklisch über den Ausgang "OutProc.Position" und den Parameter S-0-0282, Positioniersollwert, an den Antrieb übergeben. Bei einer Änderung dieses Wertes wird der Positionierbefehl sofort ausgeführt. Um einen Positionierbefehl mit dem gleichen Wert erneut auszuführen, wird eine Flanke am Eingang "RetriggerPosCmd" benötigt.
- Eingang "VelocityCmd": Positionier-Geschwindigkeit der Achse. Der Wert wird zyklisch über den Ausgang "OutProc.Velocity" und den Parameter S-0-0259, Positioniergeschwindigkeit, an den Antrieb übergeben.
- Eingang "PosModeRel":
 - FALSE = absolutes Positionieren;
 - TRUE = relatives Positionieren (Bezugspunkt für Positioniersollwert siehe "RefPosRelActPos")
- Eingang "RefPosRelActPos":
 - FALSE = additiv. Bezugspunkt für Positioniersollwert letzte wirksame Zielposition S-0-0340
 - TRUE = relativ. Bezugspunkt für Positioniersollwert aktiver Lageistwert S-0-0386

Die drei möglichen Positionierarten mit den beiden Eingängen "PosModeRel" und "RefPosRelActPos" sind im "Zeitdiagramm Positionierung" unterhalb dargestellt.

- PosModeRel = FALSE. RefPosRelActPos = FALSE: **Absolut** Positionieren. Das heißt, dass jeder Positionierbefehl immer in Bezug zur Referenz verfahren wird.
- PosModeRel = TRUE. RefPosRelActPos = FALSE: **Additiv** Positionieren. Hierbei wird zum Parameter S-0-0430 Wirksame Zielposition des Antriebs der Positionierbefehl aufaddiert
- PosModeRel = TRUE. RefPosRelActPos = TRUE: **Relativ** Positionieren. Die Bewegung ist bezogen auf den Parameter S-0-0386 Aktiver Lageistwert

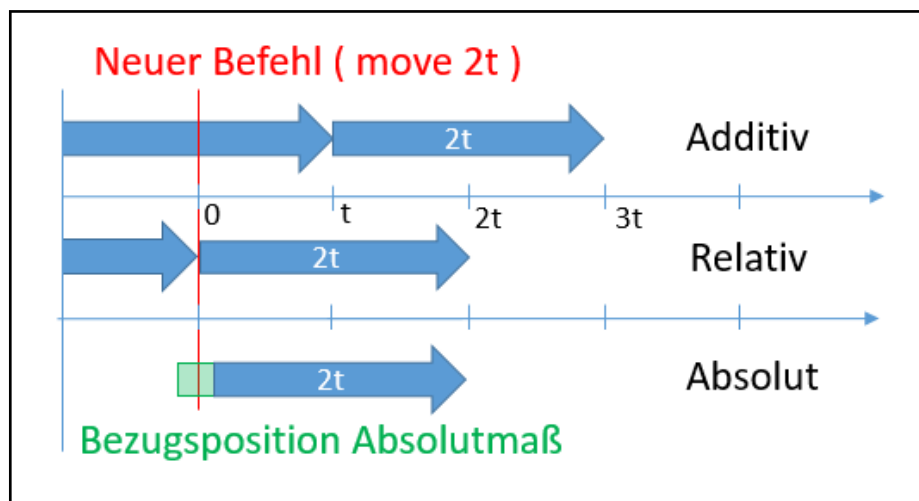


Abb. 12-11: Zeitdiagramm Positionierung

- Eingang "RetriggerPosCmd": FALSE -> TRUE = Erneutes Ausführen des letzten Positionierbefehl
- Eingang "JogPos": TRUE = Endlosfahren in positiver Richtung
- Eingang "JogNeg": TRUE = Endlosfahren in negativer Richtung

- Ausgang "PositionAct": Geschwindigkeitswert der Achse. Der Wert wird zyklisch über einen der Parameter S-0-0386 (Aktiver Lageistwert), S-0-0051 (Lageistwert Geber 1) oder S-0-0053 (Lageistwert Geber 2) und den Eingang "InProc.Position" vom Antrieb gelesen.

Für das Herstellen des Maßbezuges gibt es die Ein- und Ausgänge:

- Eingang HomingMode: S-0-0148, C0600 Kommando Antriebsgeführtes Referenzieren oder S-0-0447, C0300 Kommando Absolutmaß setzen
- Eingang "Homing": Starten des über "HomingMode" angewählten Kommandos
- Ausgang "HomingStatus": Status des Referenzierungskommandos
- Ausgang "Homed": Status Lageistwerte, TRUE = Antrieb in Referenz

Konfiguration der zyklischen Daten im Sercos:

Um die zyklische Kommunikation des Funktionsbausteins mit dem Antrieb aufzubauen, müssen Parameter auf die jeweiligen Ein- und Ausgangsvariablen kopiert werden. Die Konfiguration der zyklisch übertragenen Parameter erfolgt im Projektextplorer unterhalb des Sercos-Knotens. Dazu den Knoten "Drive" unterhalb des jeweiligen Gerätes anwählen.

Hier müssen zunächst im Reiter "Allgemeine Ein- und Ausgänge" die zyklischen Parameter konfiguriert werden.

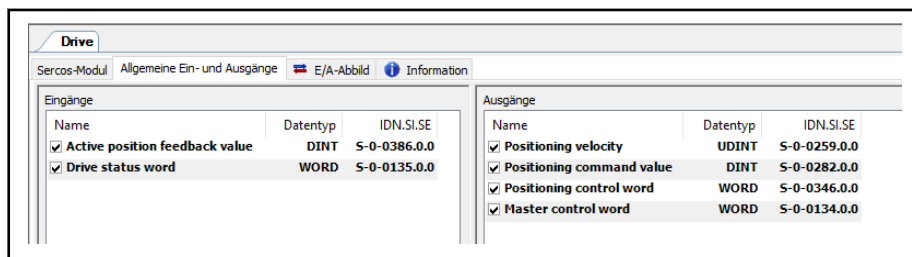


Abb. 12-12: Konfiguration der zyklischen Parameter

Anschließend werden im Reiter "E/A-Abbild" die entsprechenden Ein- und Ausgangsvariablen des Funktionsbausteins zugewiesen.

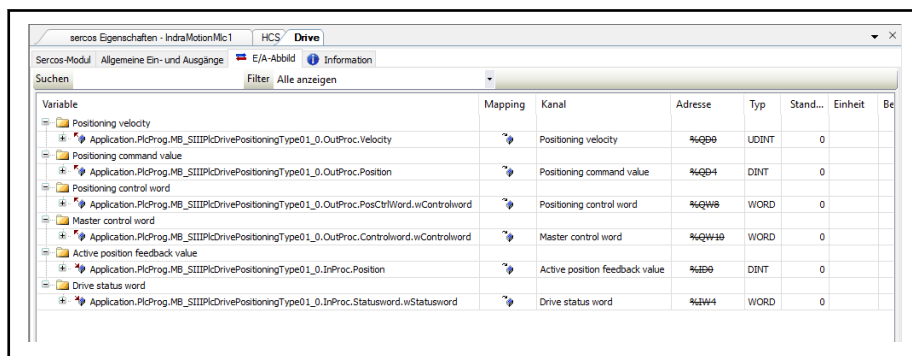


Abb. 12-13: Mappen der Ein- und Ausgangsvariablen

Parametrierung des Antriebes:



Bei Rexroth-Geräten wird die Parametrierung mit "IndraWorks Ds" empfohlen. Dort sind gerätespezifische Inbetriebnahmedialoge realisiert.

Über den Eintrag "IndraWorks Ds hier" (unterhalb des Sercos-Knotens, Geräteknoten, rechte Maustaste) kann "IndraWorks Ds" innerhalb der Engineering-Suite IndraWorks gestartet werden (Routing von der Steuerung zum Gerät erfolgt automatisch).

Der Funktionsbaustein MB_SIIIPositioningPlcDrive erfordert die Parametrierung der Hauptbetriebsart S-0-0032 auf "Antriebsgeführtes Positionieren" (Wert 0b0000.00xx.0010.xxxx).

Die weitere Parametrierung für die Betriebsart "Antriebsgeführtes Positionieren" (Beschleunigung usw.) ist mit z.B. IndraWorks Ds auszuführen.

Eine Grundparametrierung des Antriebes (Grenzwerte usw.) ist auf jeden Fall erforderlich.

Fehlerbehandlung Zusätzlich zu den Fehlern der intern genutzten Funktionsbausteine MB_SIIIPlcDriveType01 und IL_SIIISvcRead sind folgende Fehler des Bausteines in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additio- nal2	Beschreibung
STATE_MACHINE_ERROR	16#00000006	State	Fehler in der Zustandsmaschine
INPUT_RANGE_ERROR	16#00002983	Wert des Attributes	Attribut des Parameters S-0-0386, S-0-0282 oder S-0-0259 ist nicht 4 Byte dezimal mit Vorzeichen

Tab. 12-17: Fehlercodes Funktionsbaustein MB_SIIIPlcDrivePositioningType01

12.2.5 MB_SIIIPlcDrivePosInterpolationType01

Kurzbeschreibung Der Funktionsbaustein MB_SIIIPlcDrivePosInterpolationType01 dient dazu einen Antrieb in der Betriebsart "Antriebsinterne Interpolation" zu betreiben.

 Der Funktionsbaustein "MB_SIIIPositioningPlcDrive" wurde mit Version 14V16 in "MB_SIIIPlcDrivePosInterpolationType01" umbenannt, die Schnittstelle wurde optimiert.

 Ein Beispielprogramm "AchsInterface" zu diesem Funktionsbaustein ist als GAT Plug-In "SercosPlcDriveInterface" verfügbar. Dieses Beispielprogramm kann auch ohne GAT verwendet werden. Siehe dazu die Plug-In Dokumentation.

Zielsystem	Bibliothek
MLC/MTX ab 14V10	RMB_SercosIII_Util
MLD mit IndraLogic 2G ab 20VRS	RMB_SercosIII_Util

Tab. 12-18: Referenztable

Schnittstellenbeschreibung

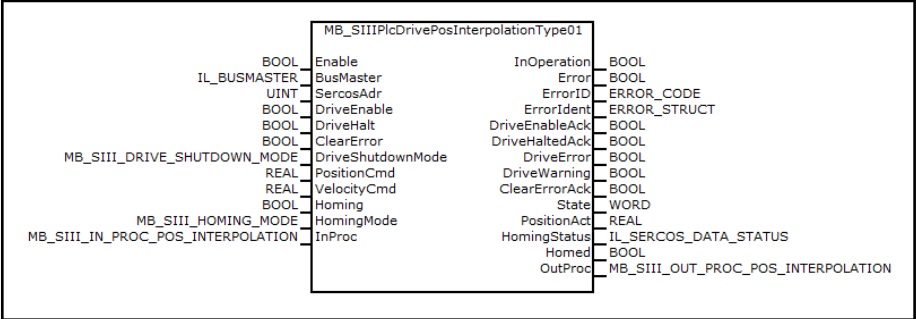


Abb. 12-14: Funktionsbaustein MB_SIIIPlcDrivePosInterpolationType01

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (pegelgesteuert).
	Busmaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable BusMaster unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 429
	SercosAdr	UINT	Sercos III-Adresse des Slave-Teilnehmers.
	DriveEnable	BOOL	Freigabe des Antriebes (AH/AF)
	DriveHalt	BOOL	Stoppen des Antriebes (AH)
	ClearError	BOOL	Löschen eines Fehlers im Antrieb durch das Kommando Fehler löschen S-0-0099. Nach gelöschtem Fehler wird die angewählte Freigabe wieder aktiviert.
	DriveShutdown-Mode	MB_SII_DRIVE_SHUTDOWN_MODE	Auswahl für das Abschalten der Antriebsfreigabe bei negativer Flanke an "DriveEnable". Optionen: Abbremsen oder Momentenfreischaltung. MB_SIII_DRIVE_SHUTDOWN_MODE auf Seite 460
	PositionCmd	REAL	Sollposition. Einheit z.B. in [Grad], je nach Parametrierung.
	VelocityCmd	REAL	Positioniergeschwindigkeit. Einheit z.B. in [U/min], je nach Parametrierung.
	Homing	BOOL	Herstellen des Maßbezuges starten
	HomingMode	MB_SIII_HOMING_MODE	S-0-0148, C0600 Kommando Antriebsgeführtes Referenzieren, oder S-0-0447, C0300 Kommando Absolutmaß setzen MB_SIII_HOMING_MODE auf Seite 460
	InProc	MB_SII_IN_PROC_POSITION_INTERPOLATION	Prozess-Eingangsdaten %I anschließen.
VAR_OUTPUT	InOperation	BOOL	Baustein arbeitet
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCT	Detailldiagnose "Fehlerbehandlung" auf Seite 458
	DriveEnableAck	BOOL	Antriebsfreigabe quittiert, aus S-0-0135, Antriebs-Status, Bit 14/15.
	DriveHaltedAck	BOOL	Antrieb Halt quittiert, aus S-0-0135, Antriebs-Status, Bit 4.
	DriveError	BOOL	Antriebsfehler, aus S-0-0135, Antriebs-Status, Bit 13.
	DriveWarning	BOOL	Antriebswarnung, aus S-0-0135, Antriebs-Status, Bit 12.
	ClearErrorAck	BOOL	Bestätigung, dass das Kommando Fehler löschen ausgeführt wurde: S-0-0099, C0500 Reset Zustandsklasse 1, Bit 13.

E/A-Typ	Name	Datentyp	Beschreibung
	State	WORD	Aktueller Status des Antriebes. Siehe Ausgang State des Funktionsbaustein IL_SIIIRemoteSlaveState.
	PositionAct	REAL	Positions-Istwert. Einheit z.B. in [Grad], je nach Parametrierung.
	HomingStatus	IL_SERCOS_DATA_STATUS	Status des Referenzierungskommandos IL_SERCOS_DATA_STATUS auf Seite 372
	Homed	BOOL	Antrieb in Referenz, aus Statuswort S-0-0135, Antriebs-Status, Bit 4.
	OutProc	MB_SI-II_OUT_PROC_POS_INTERPOLATION	Prozess-Ausgangsdaten %Q anschließen.

Tab. 12-19: Schnittstelle MB_SIIIPlcDrivePosInterpolationType01

Ausgang "State" Im Ausgang "State" ist eine Grobklassifizierung von Diagnosen binär kodiert. Die Bits sind in der RIL_FeldbusTypes.library als Globale Konstanten "FieldbusStates" enthalten.

BIT-Nr.	Name	Beschreibung
0	IL_RBS_SLAVE_NOT_REACHABLE	Slave nicht existent
1	IL_RBS_SLAVE_NOT_READY	Slave nicht bereit
2	IL_RBS_SLAVE_CONFIG_ERROR	Slave-Konfigurationsfehler
3	IL_RBS_SLAVE_DIAGNOSIS	Slave-Diagnose verfügbar
4	IL_RBS_SLAVE_ERROR	Slave-Fehler verfügbar
5	IL_RBS_CON_OFF	Mindestens eine zyklische Verbindung läuft nicht

Min-/Max und Default-Werte

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
Busmaster	IL_BUSMASTER			0	Steigende Flanke an Enable
SercosAdr	UINT	0 (MLD: lokale Achse) 1 (andere Systeme)	Systemspezifisch	0	Kontinuierlich
DriveEnable	BOOL			FALSE	Kontinuierlich
DriveHalt	BOOL			FALSE	Kontinuierlich
DriveShutdown-Mode	MB_SI-II_DRIVE_SHUT-DOWN_MODE			DRIVE_SHUT-DOWN_TORQUE_MAX	Kontinuierlich
ClearError	BOOL			FALSE	Kontinuierlich
PositionCmd	REAL			0.0	Kontinuierlich
VelocityCmd	REAL			0.0	Kontinuierlich
Homing	BOOL			FALSE	Kontinuierlich

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
HomingMode	MB_SIII_HOMING_MODE			DRIVE_CONTROL-LED_HOMING	Kontinuierlich
InProc	MB_SII_IN_PROC_POS_INTERPOLATION			0	Kontinuierlich

Tab. 12-20: Min-/Max und Default-Werte der Eingänge MB_SIIIPlcDrivePosInterpolationType01

Funktionsbeschreibung

Der Funktionsbaustein MB_SIIIPlcDrivePosInterpolationType01 erweitert den Funktionsbaustein MB_SIIIPlcDriveType01 und besitzt von diesem alle Ein- und Ausgänge sowie dessen Funktionalität.

Zusätzlich hat er noch Ein- und Ausgänge für die Ist- und Sollwerte dieser Betriebsart:

- Der Eingang "PositionCmd" gibt den Positions-Sollwert der Achse an. Der Wert wird zyklisch über den Ausgang "OutProc.Position" und den Parameter S-0-0258, Zielposition, an den Antrieb übergeben.
- Der Eingang "VelocityCmd" gibt die Positionier-Geschwindigkeit der Achse an. Der Wert wird zyklisch über den Ausgang "OutProc.Velocity" und den Parameter S-0-0259, Positioniergeschwindigkeit, an den Antrieb übergeben.
- Der Ausgang "PositionAct" gibt den Geschwindigkeits-Istwert der Achse aus. Der Wert wird zyklisch über den Parameter S-0-0051 (Lageistwert Geber 1) oder S-0-0053 (Lageistwert Geber 2) und den Eingang "In-Proc.Position" vom Antrieb gelesen.

Für das Herstellen des Maßbezuges gibt es die Ein- und Ausgänge:

- Eingang HomingMode: S-0-0148, C0600 Kommando Antriebsgeführtes Referenzieren oder S-0-0447, C0300 Kommando Absolutmaß setzen
- Eingang "Homing": Starten des über "HomingMode" angewählten Kommandos
- Ausgang "HomingStatus": Status des Referenzierungskommandos
- Ausgang "Homed": Status Lageistwerte, TRUE = Antrieb in Referenz

Die hier genutzte Betriebsart "Antriebsinterne Interpolation" unterstützt nur absolutes Positionieren.

Für Relatives Positionieren wird der Sollwert "PositionCmd" mit Hilfe des Istwertes "PositionAct" berechnet.

Für Additives Positionieren wird der Parameter S-0-0430, Wirksame Zielposition, zusätzlich zyklisch aufgelegt und der Sollwert "PositionCmd" mit Hilfe des S-0-0430 berechnet.

Konfiguration der zyklischen Daten im Sercos:

Um die zyklische Kommunikation des Funktionsbausteins mit dem Antrieb aufzubauen müssen Parameter auf die jeweiligen Ein- und Ausgangsvariablen kopiert werden. Die Konfiguration der zyklisch übertragenen Parameter erfolgt im Projektextplorer unterhalb der Sercosknotens. Dazu den Knoten "Drive" unterhalb des jeweiligen Gerätes anwählen.

Hier müssen zunächst im Reiter "Allgemeine Ein- und Ausgänge" die zyklischen Parameter konfiguriert werden.

Die Variable „wControlword“ liegt auf dem gleichen Speicher wie die Bits und dient dem Mapping im E/A-Abbild des Antriebes.

Strukturelement	Typ	Min-Wert	Max-Wert	Default-Wert	Beschreibung
wControlword	WORD			0	Komplettes Kontrollwort
OpModeBit8	BIT			FALSE	Bits 8, 9 und 11 geben an, welche Betriebsart gewählt ist. Es wird immer die Hauptbetriebsart aktiviert.
OpModeBit9	BIT			FALSE	
OpModeBit11	BIT			FALSE	
Drive_STOP	BIT			FALSE	1->0 Antrieb stopp (nur wenn Bit 14 und 15 = 1)
Drive_Release	BIT			FALSE	1->0 Unverzögerte Drehmoment-Ab-schaltung
Drive_ON	BIT			FALSE	1->0 Bestmögliche Stillsetzung (nur mit Bit 14 = 1)

12.2.7 MB_SIII_STATUSWORD

Kurzbeschreibung Die Datenstruktur MB_SIII_STATUSWORD entspricht dem Parameter Master-Statuswort S-0-0135. Die Bits können im Anwendungsprogramm nur gelesen werden.

Die Variable „wStatusword“ liegt auf dem gleichen Speicher wie die Bits und dient dem Mapping im E/A-Abbild des Antriebes.

Strukturelement	Typ	Min-Wert	Max-Wert	Default-Wert	Beschreibung
wStatusword	WORD			0	Komplettes Statuswort
StatusCmdValueProcessing	BIT			FALSE	Status der Sollwertverarbeitung
StatusDriveHalt	BIT			FALSE	Status Antrieb-HALT
Homed	BIT			FALSE	Status Lageistwerte
ActOpModeBit8	BIT			FALSE	Ist-Betriebsart 000: Hauptbetriebsart aktiv 001: 1 Neben-Betriebsart aktiv 010: 2 Neben-Betriebsart, usw.
ActOpModeBit9	BIT			FALSE	
ActOpModeBit10	BIT			FALSE	
Warning	BIT			FALSE	Warnung ZKL 2
Error	BIT			FALSE	Antriebsfehler ZKL 1
EnabledBit14	BIT			FALSE	Betriebsbereitschaft: 00: nicht bereit zur Leistungszu-schaltung (z.B. P2); 01: bereit zur Leistungszuschaltung (bb); 10: Steuer u. Leistungsteil betriebs-bereit (Ab); 11: In Betrieb, momentenbehaftet (z.B. AF)
EnabledBit15	BIT			FALSE	

Tab. 12-22: Elemente der Struktur MB_SIII_STATUSWORD

12.2.8 MB_SIII_DRIVE_SHUTDOWN_MODE

Kurzbeschreibung Der Aufzählungstyp MB_SIII_DRIVE_SHUTDOWN_MODE ermöglicht die Auswahl des Verhaltens beim Abschalten der Antriebsfreigabe.

Element	Wert	Beschreibung
DRIVE_SHUTDOWN_TORQUE_MAX	0	Abbremsen mit maximaler Beschleunigung
DRIVE_SHUTDOWN_TORQUE_FREE	1	Momentenfreischaltung (Austrudeln)

Tab. 12-23: Elemente des Aufzählungstyps MB_SIII_DRIVE_SHUTDOWN_MODE

12.2.9 MB_SIII_HOMING_MODE

Kurzbeschreibung Der Aufzählungstyp MB_SIII_HOMING_MODE ermöglicht die Auswahl des Verhaltens beim Herstellen des Maßbezuges.

Element	Wert	Beschreibung
DRIVE_CONTROLLED_HOMING	0148	S-0-0148, C0600 Kommando Antriebsgeführtes Referenzieren
SET_ABSOLUTE_POSITION	0447	S-0-0447, C0300 Kommando Absolutmaß setzen

Tab. 12-24: Elemente des Aufzählungstyps MB_SIII_HOMING_MODE

12.2.10 MB_SIII_POSCTRLWORD

Kurzbeschreibung Die Datenstruktur MB_SIII_POSCTRLWORD entspricht dem Parameter Master-Steuerwort S-0-0346. Die Bits können im Anwendungsprogramm nur gelesen werden.

Die Variable "wControlword" liegt auf dem gleichen Speicher wie die Bits und dient dem Mapping im E/A-Abbild des Antriebes.

Strukturelement	Typ	Min-Wert	Max - Wert	Default-Wert	Beschreibung
wControlword	WORD			0	Komplettes Kontrollwort
PosPointAcception	BIT			FALSE	Übernahme Positioniersollwert durch Toggeln
ActivatePos1	BIT			FALSE	00: Aktivierung der Positionierung Positionierung aktiv, Start durch Toggeln von Bit 0 Abbruch der Positionierung durch: 01: Endlosfahren in positiver Richtung (Tippen+) 10: Endlosfahren in negativer Richtung (Tippen-) 11: Anhalten der Achse (Positionierhalt)
ActivatePos2	BIT			FALSE	
PosTypeRel	BIT			FALSE	Art des Positioniersollwerts 0: Absolut 1: Relativ (abhängig von Bit 4)
RefPointActPos	BIT			FALSE	Bezugspunkt für Positioniersollwerte 0: Letzte wirksame Zielposition (S-0-0430) 1: Aktiver Lageistwert (S-0-0386)

Strukturelement	Typ	Min-Wert	Max-Wert	Default-Wert	Beschreibung
ImmediateBlock-Change	BIT			FALSE	Satzsofortwechsel 0: Aktuelle Zielposition wird noch angefahren bevor auf die neue Zielposition positioniert wird 1: Satzsofortwechsel, d.h. die neue Zielposition wird sofort angefahren
SeqBlockBehavior1	BIT			FALSE	Hinweis: Diese Bits werden vom Funktionsbaustein MB_SIIIPlcDrivePositioning nicht genutzt! Verhalten bei Folgesatz (Bit 5=0) 00: Halt bei Zielposition des Startsatzes 01: Überfahren der Zielposition des Startsatzes (Modus 1) 10: Überfahren der Zielposition des Startsatzes (Modus 2)
SeqBlockBehavior2	BIT			FALSE	

12.2.11 MB_SIII_IN_PROC_VELOCITY

Kurzbeschreibung Die Datenstruktur MB_SIII_IN_PROC_VELOCITY enthält die Prozess-Eingangsdaten des Funktionsbausteins [MB_SIIIPlcDriveVelocityType01 auf Seite 441](#).

Strukturelement	Typ	Default-Wert	Beschreibung
Statusword	MB_SIII_STATUSWORD		Eingangsvariable %IW des Statuswortes S-0-0135 am Element "wStatusword" anschließen. MB_SIII_STATUSWORD auf Seite 459
Velocity	DINT	16#0	Eingangsvariable %ID des Geschwindigkeits-Istwert S-0-0040 anschließen.

Tab. 12-25: Elemente der Struktur MB_SIII_IN_PROC_VELOCITY

12.2.12 MB_SIII_OUT_PROC_VELOCITY

Kurzbeschreibung Die Datenstruktur MB_SIII_OUT_PROC_VELOCITY enthält die Prozess-Ausgangsdaten des Funktionsbausteins [MB_SIIIPlcDriveVelocityType01 auf Seite 441](#).

Strukturelement	Typ	Default-Wert	Beschreibung
Controlword	MB_SIII_CONTROLWORD		Ausgangsvariable %QW des Steuerwortes S-0-0134 am Element "wControlword" anschließen. MB_SIII_CONTROLWORD auf Seite 458
Velocity	DINT	16#0	Ausgangsvariable %QD des Geschwindigkeits-Sollwert S-0-0036 anschließen.

Tab. 12-26: Elemente der Struktur MB_SIII_OUT_PROC_VELOCITY

12.2.13 MB_SIII_IN_PROC_TORQUE

Kurzbeschreibung Die Datenstruktur MB_SIII_IN_PROC_TORQUE enthält die Prozess-Eingangsdaten des Funktionsbausteins [MB_SIIIPlcDriveTorqueType01 auf Seite 444](#).

Strukturelement	Typ	Default-Wert	Beschreibung
Statusword	MB_SIII_STATUSWORD		Eingangsvariable %IW des Statuswortes S-0-0135 am Element "wStatusword" anschließen. MB_SIII_STATUSWORD auf Seite 459
Torque	INT	16#0	Eingangsvariable %IW des Momenten-Istwertes S-0-0084 anschließen.

Tab. 12-27: Elemente der Struktur MB_SIII_IN_PROC_TORQUE

12.2.14 MB_SIII_OUT_PROC_TORQUE

Kurzbeschreibung Die Datenstruktur MB_SIII_OUT_PROC_TORQUE enthält die Prozess-Ausgangsdaten des Funktionsbausteins [MB_SIIIPlcDriveTorqueType01 auf Seite 444](#).

Strukturelement	Typ	Default-Wert	Beschreibung
Controlword	MB_SIII_CONTROLWORD		Ausgangsvariable %QW des Steuerwortes S-0-0134 am Element "wControlword" anschließen. MB_SIII_CONTROLWORD auf Seite 458
Torque	INT	16#0	Ausgangsvariable %QW des Momenten-Sollwert S-0-0080 anschließen.

Tab. 12-28: Elemente der Struktur MB_SIII_OUT_PROC_TORQUE

12.2.15 MB_SIII_IN_PROC_POSITIONING

Kurzbeschreibung Die Datenstruktur MB_SIII_IN_PROC_POSITIONING enthält die Prozess-Eingangsdaten des Funktionsbausteins [MB_SIIIPlcDrivePositioningType01 auf Seite 448](#).

Strukturelement	Typ	Default-Wert	Beschreibung
Statusword	MB_SIII_STATUSWORD		Eingangsvariable %IW des Statuswortes S-0-0135 am Element "wStatusword" anschließen MB_SIII_STATUSWORD auf Seite 459
Position	DINT	16#0	Eingangsvariable %IW der aktuellen Position S-0-0386 oder S-0-0051 anschließen

Tab. 12-29: Elemente der Struktur MB_SIII_IN_PROC_POSITIONING

12.2.16 MB_SIII_OUT_PROC_POSITIONING

Kurzbeschreibung Die Datenstruktur MB_SIII_OUT_PROC_POSITIONING enthält die Prozess-Ausgangsdaten des Funktionsbausteins [MB_SIIIPlcDrivePosInterpolationType01 auf Seite 454](#).

Strukturelement	Typ	Default-Wert	Beschreibung
Controlword	MB_SIII_CONTROLWORD		Ausgangsvariable %QW des Steuerwortes S-0-0134 am Element "wControlword" anschließen MB_SIII_CONTROLWORD auf Seite 458
PosCtrlWord	MB_SIII_POSCTRLWORD		Ausgangsvariable %QW des Steuerwortes S-0-0346 am Element "wControlword" anschließen MB_SIII_POSCTRLWORD auf Seite 460
Position	DINT	16#0	Ausgangsvariable %QD des Positions-Sollwert S-0-0282 anschließen
Velocity	DINT	16#0	Ausgangsvariable %QD der Positionier-Geschwindigkeit S-0-0259 anschließen

Tab. 12-30: Elemente der Struktur MB_SIII_OUT_PROC_POSITIONING

12.2.17 MB_SIII_IN_PROC_POS_INTERPOLATION

Kurzbeschreibung Die Datenstruktur MB_SIII_IN_PROC_POS_INTERPOLATION enthält die Prozess-Eingangsdaten des Funktionsbausteins [MB_SIIIPlcDrivePosInterpolationType01 auf Seite 454](#).

Strukturelement	Typ	Default-Wert	Beschreibung
Statusword	MB_SIII_STATUSWORD		Eingangsvariable %IW des Statuswortes S-0-0135 am Element "wStatusword" anschließen. MB_SIII_STATUSWORD auf Seite 459
Position	DINT	16#0	Eingangsvariable %IW der aktuellen Position S-0-0051 anschließen.

Tab. 12-31: Elemente der Struktur MB_SIII_IN_PROC_POS_INTERPOLATION

12.2.18 MB_SIII_OUT_PROC_POS_INTERPOLATION

Kurzbeschreibung Die Datenstruktur MB_SIII_OUT_PROC_POS_INTERPOLATION enthält die Prozess-Ausgangsdaten des Funktionsbausteins [MB_SIIIPlcDrivePosInterpolationType01 auf Seite 454](#).

Strukturelement	Typ	Default-Wert	Beschreibung
Controlword	MB_SIII_CONTROLWORD		Ausgangsvariable %QW des Steuerwortes S-0-0134 am Element "wControlword" anschließen. MB_SIII_CONTROLWORD auf Seite 458
Position	DINT	16#0	Ausgangsvariable %QD des Positions-Sollwert S-0-0258 anschließen.
Velocity	DINT	16#0	Ausgangsvariable %QD der Positionier-Geschwindigkeit S-0-0259 anschließen.

Tab. 12-32: Elemente der Struktur MB_SIII_OUT_PROC_POS_INTERPOLATION

12.3 PowerSupply

Die PowerSupply-Funktionsbausteine steuern die Stromversorgung. Die Kommunikation zwischen der Steuerung und der Stromversorgung erfolgt über Sercos III. Diese Funktionsbausteine können den internen Status der Stromversorgung erkennen, Außerdem sorgen sie für einen reibungslosen Übergang zwischen den Zustandsmaschinen der Stromversorgung, setzen die Stromversorgung in den notwendigen Zustand, steuern die Betriebsart, löschen den Stromversorgungsfehler und lesen die Diagnoseinformationen der Stromversorgung aus.

Bitte beachten Sie die folgenden Regeln bei der Verwendung der Funktionsbausteine der Stromversorgung:

- Kontrolliert eine MLC einen Wechselrichter, der einen Antrieb und die notwendige Stromversorgung enthält und ist dieser Wechselrichter mit der MLC-Steuerung verbunden (d.h. eine Wechselrichterachse befindet sich unterhalb eines MLC-Steuerungsknotens), wird der Funktionsbaustein MB_SIIIPowerSupplyDriveIntegrated verwendet
- Wird eine der erweiterten Versorgerfunktionen **DC/DC-Steller** oder **Inselnetzbetrieb** genutzt, kommt der Funktionsbaustein MB_SIIIPowerSupplyDcDcIslandGrid zum Einsatz
- In allen anderen Fällen sollte der Funktionsbaustein MB_SIIIPowerSupplyStandalone bevorzugt werden

Ist z.B. die Stromversorgung KMV03 mit einer MLC verbunden, so sollte der Funktionsbaustein MB_SIIIPowerSupplyStandalone verwendet werden. MB_SIIIPowerSupplyStandalone ist üblicherweise der Funktionsbaustein, der am häufigsten verwendet wird.

Die folgende Abbildung zeigt die Zustandsmaschine, die durch die Funktionsbausteine der Stromversorgung kontrolliert wird.

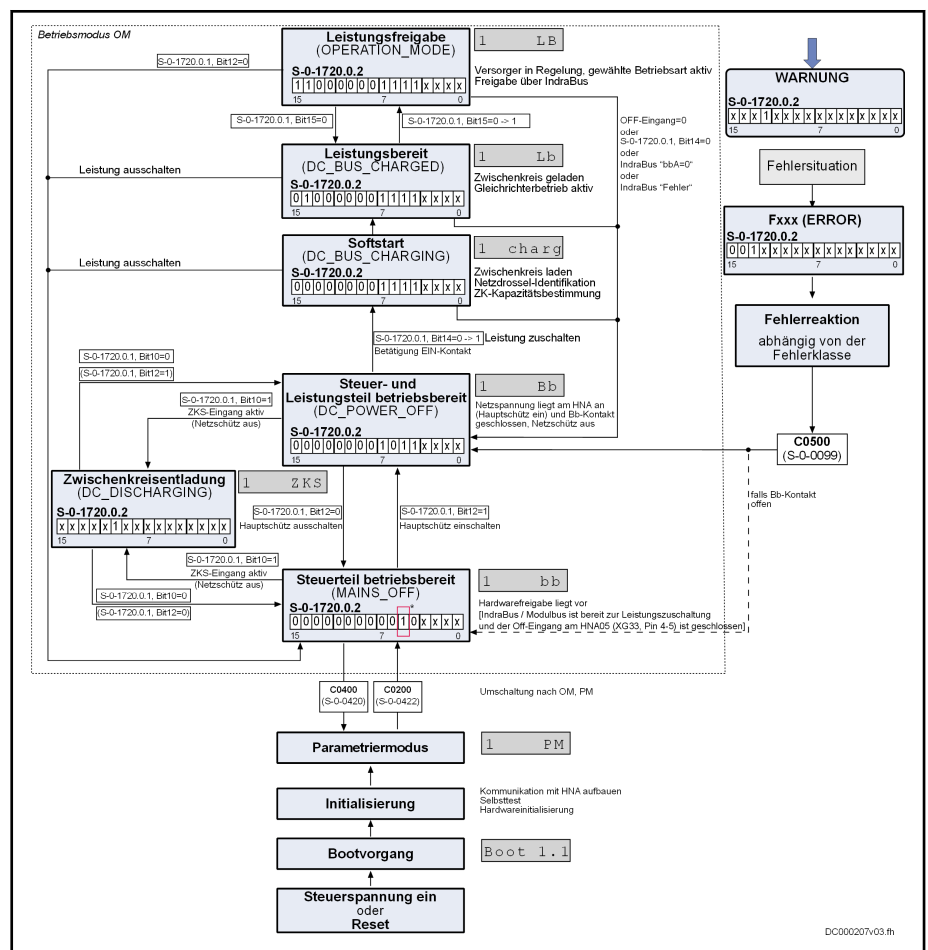


Abb. 12-17: *Gerätesteuerung (allgemeine Zustandsmaschine)*

Für mehr Informationen, siehe die Dokumentation der IndraDrive Stromversorgung für eine allgemeine Beschreibung der Stromversorgungsfunktionalität (IndraDrive Power Supply Basic PSB-21 Funktionen, DOK-INDRV*-PSB-21VRS**-APRS-DE-P).

12.3.1 MB_SIIIPowerSupplyDriveIntegrated

Kurzbeschreibung

Der Funktionsbaustein MB_SIIIPowerSupplyDriveIntegrated überwacht und steuert einen Versorger.

Die Daten zwischen dem Funktionsbaustein und dem Versorger werden über den Sercos-Feldbus ausgetauscht.

Der Funktionsbaustein kann den internen Zustand des Versorgers identifizieren, Übergänge der Zustandsmaschine des Versorgers auslösen, den Versorger in den gewünschten Zustand bringen und die Betriebsart steuern.

Der Funktionsbaustein überwacht das Statuswort "InProcStatus" des Versorgers kontinuierlich und aktualisiert das Steuerwort "OutProcCtrl" des Versorgers entsprechend seiner Eingangsbelegung.



Der Funktionsbaustein "MB_SIIIPowerSupplyType03" wurde mit Version 15V06 in "MB_SIIIPowerSupplyDriveIntegrated" umbenannt. Die Funktion ist unverändert.

Bibliothek:	RMB_SercosIII_Util
Gruppe:	"PowerSupply"
Zielsystem:	MTX 15VRS MLC 15VRS, Funktionspaket SPS MLD mit MPx20

Tab. 12-33: Referenztable des MB_SIIIPowerSupplyDriveIntegrated FB

Schnittstellenbeschreibung

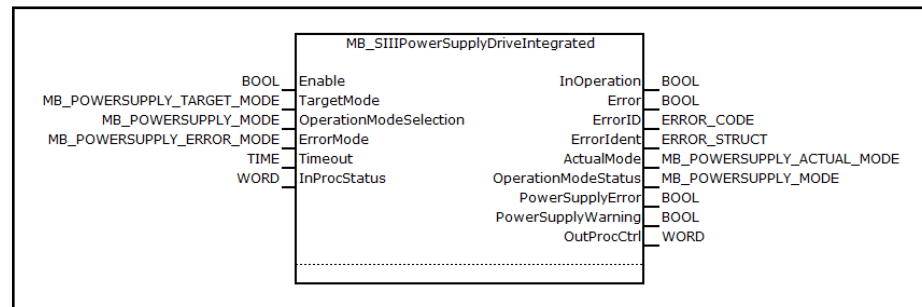


Abb. 12-18: Schnittstelle des MB_SIIIPowerSupplyDriveIntegrated FB

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe (zyklisch, zustandsgesteuert)
	TargetMode	MB_POWER_SUPPLY_TARGET_MODE	Vom Versorger zu erreichender Zielmodus MB_POWER_SUPPLY_TARGET_MODE auf Seite 489
	OperationModeSelection	MB_POWER_SUPPLY_MODE	Betriebsart für den Versorger MB_POWER_SUPPLY_MODE auf Seite 487
	ErrorMode	MB_POWER_SUPPLY_ERROR_MODE	Fehler beim Übergabe-Handling-Eingang für den Versorger MB_POWER_SUPPLY_ERROR_MODE auf Seite 488
	Timeout	TIME	Max. erlaubte Zeit von jedem "ActualMode" -> "TargetMode"
	InProcStatus	WORD	Statuswort des Versorgers (S-0-1720.0.2)
VAR_OUTPUT	InOperation	BOOL	Der Baustein arbeitet korrekt, die Ausgangsdaten sind gültig
	Error	BOOL	Zeigt an, dass ein Fehler aufgetreten ist (siehe ErrorID und ErrorIdent)
	ErrorID	ERROR_CODE	Fehlerursache
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerursache
	ActualMode	MB_POWER_SUPPLY_ACTUAL_MODE	Istzustand der Zustandsmaschine des Versorgers MB_POWER_SUPPLY_ACTUAL_MODE auf Seite 487
	OperationModeStatus	MB_POWER_SUPPLY_MODE	Istbetriebsart des Versorgers MB_POWER_SUPPLY_MODE auf Seite 487
	PowerSupplyError	BOOL	Interner Versorgerfehler

	Name	Typ	Kommentar
	PowerSupply-Warning	BOOL	Interne Versorgerwarnung
	OutProcCtrl	WORD	Steuerwort des Versorgers (S-0-1720.0.1)

Tab. 12-34: Schnittstellenbeschreibung des MB_SIIIPowerSupplyDriveIntegrated FB

Min./max. und Default-Werte und Anwendung des Eingangs

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Anwendung
Enable	BOOL			FALSE	Kontinuierlich
TargetMode	MB_POWERSUPPLY_TARGET_MODE			MAINS_OFF_NO_DISCHARGE	Kontinuierlich
OperationModeSelection	MB_POWERSUPPLY_MODE			OPERATION-MODE1	Kontinuierlich
ErrorMode	MB_POWERSUPPLY_ERROR_MODE			ERROR_TRANSITION_NO_DISCHARGE	Kontinuierlich
Timeout	TIME	n. def.	n. def.	T#10s	Kontinuierlich
InProcStatus	WORD	n. def.	n. def.	0	Kontinuierlich

Tab. 12-35: Min-/Max- und Default-Werte des MB_SIIIPowerSupplyDriveIntegrated FB

Funktionsbeschreibung

Der Funktionsbaustein überwacht und steuert den Versorger. Der Funktionsbaustein kann verschiedene Versorgerfunktionen realisieren, die in dem Steuerwort des Versorgers festgelegt sind.

Die Hauptaufgabe des Funktionsbausteins ist es, einen Zustandsübergang des Versorgers zum gewünschten Zielmodus durchzuführen.

Der zyklische Datenaustausch zwischen dem Funktionsbaustein und dem Versorger wird nachfolgend beschrieben.

- "InProcStatus" - Aktualisiert auf S-0-1720.0.2 (Versorgerstatuswort)
- "OutProcCtrl" - Aktualisiert auf S-0-1720.0.1 (Versorgersteuerwort)

Eine allgemeine Beschreibung der internen Zustandsmaschine des Versorgers finden Sie in der Dokumentation der IndraDrive-Versorgerfunktionen.

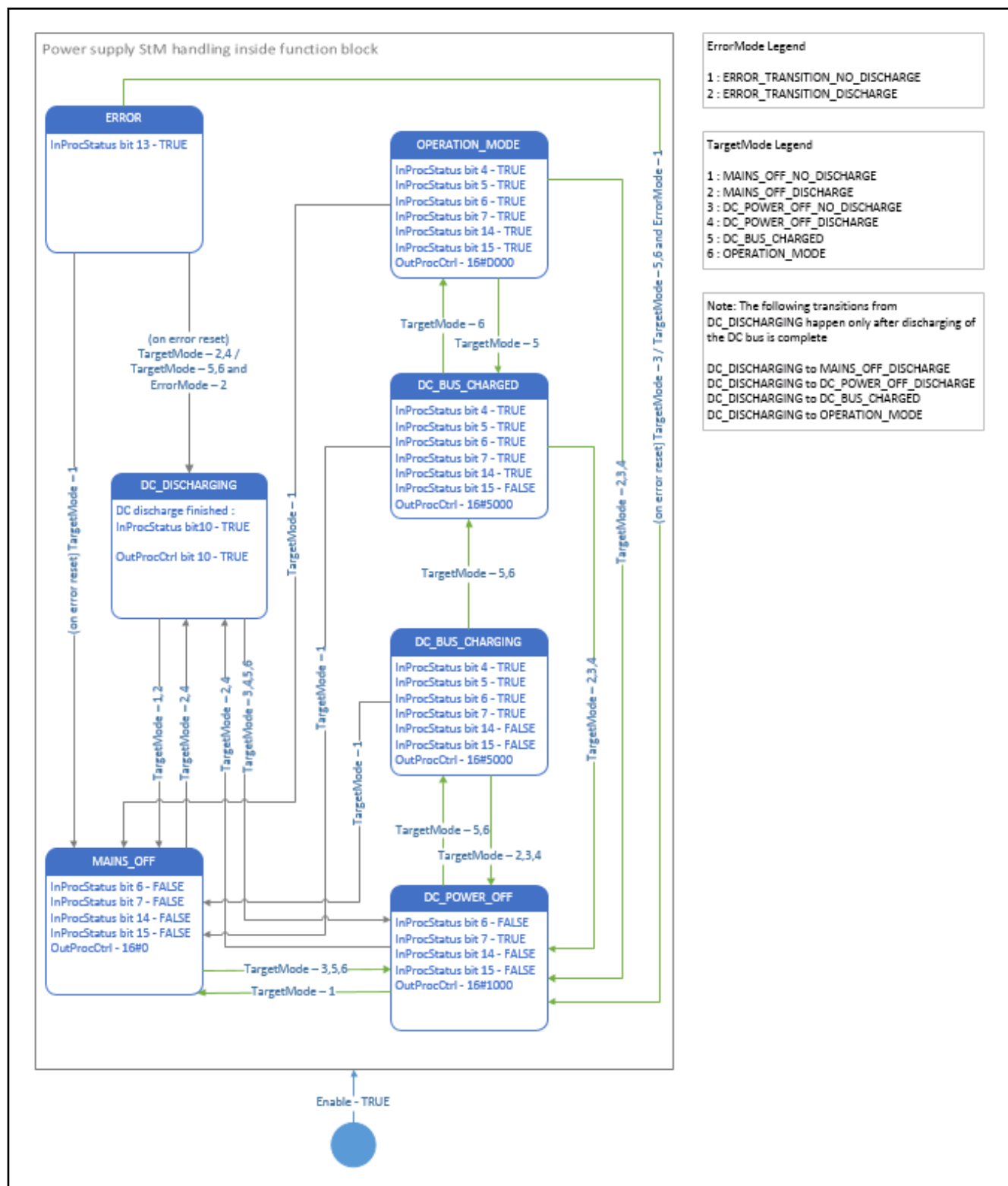


Abb. 12-19: Zustandsgraph eines Versorgers

Der Funktionsbaustein verwendet dieses Statuswort, um den aktuellen Zustand der Zustandsmaschine des Versorgers zu identifizieren.

Der von der Zustandsmaschine des Versorgers zu erreichende Zielmodus findet sich am Eingang "TargetMode". Der Funktionsbaustein führt den Über-

gang in den "TargetMode" aus, indem er die interne Zustandsmaschine des Versorgers steuert.

Im Fall eines Funktionsbausteinfehlers werden das Zustandshandling angehalten und der Fehler gelöscht.

Wenn der Versorger den Fehlerzustand erreicht, entscheiden Zielmodus und Fehlermodus über den nächsten Folgezustand des Versorgers.



Es kann zu widersprüchlichen Eingangskombinationen zwischen "ErrorMode" und "TargetMode" kommen.

In solchen Fällen dominiert "TargetMode" unabhängig vom Wert des Eingangs "ErrorMode". Siehe nachstehendes Beispiel.

- ActualMode = _ERROR
- TargetMode = DC_POWER_OFF_NO_DISCHARGE
- ErrorMode = ERROR_TRANSITION_DISCHARGE

Wenn der Versorger den Zustand "Error" verlässt, ist es für "TargetMode" notwendig, dass der Versorger in den Zustand "DC-Power-Off" geht ohne den DC-Bus zu entladen.

Für "ErrorMode" ist es hingegen notwendig, dass der Versorger den DC-Bus entlädt bevor in den nächsten Zustand geschaltet wird.

In diesem Fall dominiert "TargetMode" und der Versorger wird in den Modus "DC-Power-Off" geschaltet ohne den DC-Bus zu entladen.

Dies muss vom Programmierer selbst beim Erstellen von Applikationen beachtet werden.

Die Zeit, die der Versorger benötigt, um den Zielmodus (vom aktuellen Modus) zu erreichen, wird überwacht.

Wenn diese Zeit abläuft, setzt der Eingang "Timeout" einen Fehler und das Handling der Zustandsmaschine wird gestoppt.

Wird der Eingang "Enable" des Funktionsbausteins zurückgesetzt, erkennt der Funktionsbaustein den aktuellen Zustand des Versorgers und entscheidet, ob es notwendig ist, den Versorger (DC) abzuschalten.

Der Eingang "OperationModeSelection" wird sofort und kontinuierlich auf den Bits 8 und 9 des Steuerworts angezeigt.

Die Ausgänge des Funktionsbausteins werden während jedes SPS-Scans vom Eingang "InProcStatus" aktualisiert. Jede Statuswortänderung wird sofort an den Ausgängen ausgegeben.

Die Ausgänge "Error", "ErrorID" und "ErrorIdent" enthalten die notwendigen Informationen im Fehlerfall.

Fehlerbehandlung

Der Funktionsbaustein kann aufgrund ungültiger Eingänge oder abgelaufenem Timeout-Timer in einen Fehlerzustand übergehen. In beiden Fällen gibt der Funktionsbaustein einen Fehler aus und bricht die Operation ab.

Der Benutzer kann die Fehlerursache beheben, den Eingang "Enable" zurücksetzen und erneut setzen (Standardvorgehen beim Fehler zurücksetzen).

Der Funktionsbaustein arbeitet anschließend weiter. Zu den verfügbaren Fehlercodes, siehe nachfolgende Tabelle.

Zusätzlich zu den Fehlern der intern genutzten Funktionsbausteine IL_SIII-Command und IL_SIIIRemoteSlaveState, sind folgende Fehler des Bausteins

nes in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

STATE_MACHINE_ERROR	16#00000006	State	Fehler in der Zustandsmaschine
INPUT_INVALID_ERROR	16#00002990	TargetMode	Eingang TargetMode ist ungültig
INPUT_INVALID_ERROR	16#00002991	OperationModeSelection	Eingang OperationModeSelection ist ungültig
INPUT_INVALID_ERROR	16#00002992	ErrorMode	Eingang ErrorMode ist ungültig
STATE_MACHINE_ERROR	16#00002993	0	Zeitüberschreitung (Actual mode -> Target mode)

Tab. 12-36: Fehlercodes FB MB_SIIIPowerSupplyDriveIntegrated

Anwendungsbeispiel mit MB_SIIIPowerSupplyDriveIntegrated

An einem IndraDrive XCS mit integriertem Versorger sind zwei IndraDrive XMS angeschlossen.

Mit dem Funktionsbaustein MB_SIIIPowerSupplyDriveIntegrated wird der Versorger XCS gesteuert. Mit den Ausgängen des Funktionsbausteins werden die Wechselrichter XMS freigegeben (von "bb" nach "Ab" geschaltet).

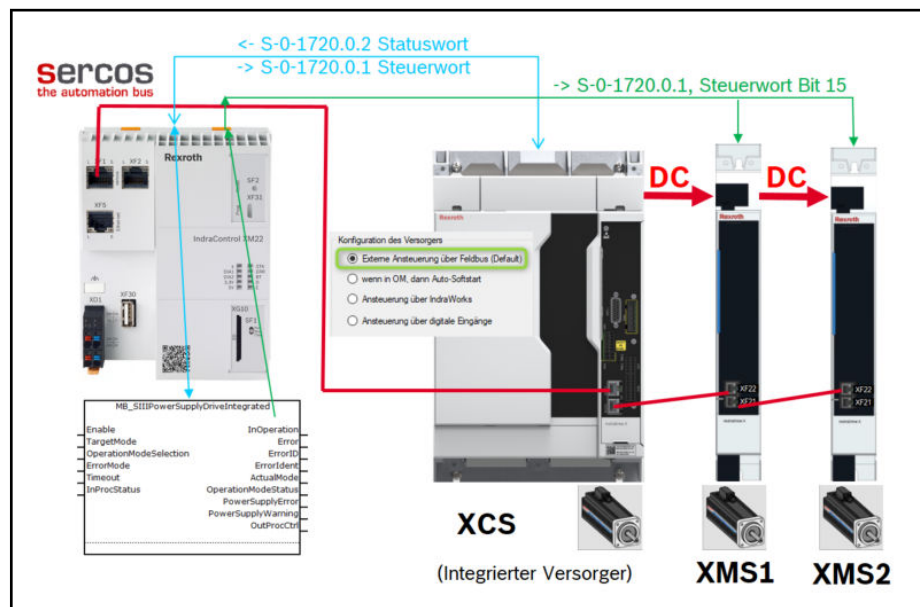


Abb. 12-20: Prinzipschaltbild: Integrierter Versorger XCS versorgt zwei Antriebe XMS

Voraussetzung: Der Versorger im XCS ist auf "Externe Ansteuerung über Feldbus" konfiguriert.

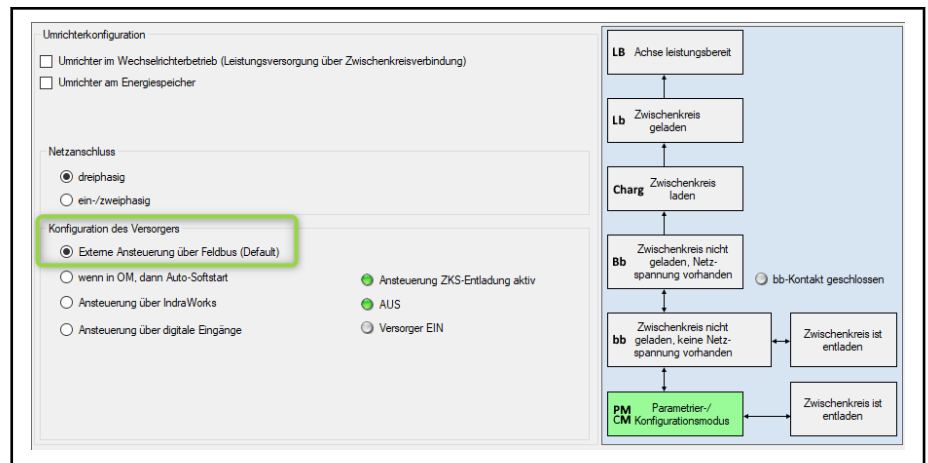


Abb. 12-21: Konfiguration des integrierten Versorgers

Die folgenden Parameter müssen beim XCS zyklisch konfiguriert werden:

- MDT: S-0-1720.0.1, "Versorger-Steuerwort"
- AT: S-0-1720.0.2, "Leistungsversorgung Statuswort"

Bei den XMS-Antrieben reicht es, das Bit 15 "Freigabe" des Parameters S-0-1720.0.1, "Versorger-Steuerwort" in das Signalsteuerwort zu konfigurieren.

Im folgenden SPS-Programmbeispiel wird davon ausgegangen, dass immer der Daten-Container "A" verwendet wird.

Statt dessen können auch alle anderen Container "B bis F" genutzt werden. Dabei ist das SPS-Programm entsprechend anzupassen.



Mit Hilfe des Funktionsbausteines "MB_GetCyclicParameterHandle" und den Funktionen "MB_ReadCyclicParameter" und "MB_WriteCyclicParameter" (siehe Bibliothek "ML_TechMotion-Base") kann das SPS-Programm unabhängig von den jeweils verwendeten Datencontainern "A bis F" implementiert werden.

Werden die Antriebe XCS und/oder XMS als SPS-Geräte eingebunden, sind die gleichen Parameter zyklisch zu konfigurieren und es müssen Variablen auf diese Parameter gemapped werden. Im folgenden Beispiel sind entsprechende Anweisungen auskommentiert enthalten und mit "drive as PLC device" gekennzeichnet.

Variablendefinition

```
fbMB_SIIIPowerSupply: MB_SIIIPowerSupplyDriveIntegrated;
bEnablePower: BOOL:=FALSE;
```

Eintritt in die Initialisierung (Beispiel GATcompact, Machine.actInit_entry)

```
// switch off power supply
bEnablePower := FALSE;
```

Ende der Initialisierung (Beispiel GATcompact, Machine.actInit_active)

```
IF (ImcStatus.Admin._OpModeAck = IMC_MODE_STATUS_BB) AND (TE_AxisInterfaceType01Prog.InitDone = TRUE) THEN
// *****
// Now we reached BB (operating mode)
// Add your code for writing the parameters (in BB) here
// *****
bEnablePower := TRUE;
IF fbMB_SIIIPowerSupply.ActualMode = MB_POWERSUPPLY_ACTUAL_MODE.OPERATION_MODE THEN
```

RMB_SercosIII_Util.library

```

    uiStateInit := uiSTATE_INIT_PREPARE_COLDSTART;
    bInitDone   := TRUE; // Set bInitDone if the initialization is done
  END_IF
END_IF

```

Zyklisch aufgerufen (Beispiel GATcompact, Machine.actCyclic_active)

```

//----- handle power-supply -----
//-----
// copy status word from bus to FB
fbMB_SIIIPowerSupply.InProcStatus := AxisData[XCS.AxisNo].dwUserActualDataA i.UINT; // motion axis
// fbMB_SIIIPowerSupply.InProcStatus := PlcDrive_InProcStatus; // drive as PLC device
// always switching direct to OPERATION_MODE
fbMB_SIIIPowerSupply(
  Enable:= bEnablePower,
  TargetMode:= MB_POWER_SUPPLY_TARGET_MODE.OPERATION_MODE,
  OperationModeSelection:= ,
  ErrorMode:= ,
  Timeout:= ,
  OperationModeStatus=> ,
  PowerSupplyError=> ,
  PowerSupplyWarning=> );
IF fbMB_SIIIPowerSupply.Error THEN
  bEnablePower := FALSE;
  mSetFBError(fbMB_SIIIPowerSupply.ErrorID, fbMB_SIIIPowerSupply.ErrorIdent);
END_IF;
// Copy control word from FB to bus
AxisData[XCS.AxisNo].dwUserCmdDataA q.UINT := fbMB_SIIIPowerSupply.OutProcCtrl; // motion axis
// PlcDrive_OutProcCtrl := fbMB_SIIIPowerSupply.OutProcCtrl; // drive as PLC device
// copy output of FB to power supply control word, bit 15 of all XMS
IF fbMB_SIIIPowerSupply.ActualMode = MB_POWER_SUPPLY_ACTUAL_MODE.OPERATION_MODE THEN
  // activate power in power supply control word
  AxisData[XMS1.AxisNo].dwUserCmdDataA q.UINT .15 := TRUE; // motion axis 1
  AxisData[XMS2.AxisNo].dwUserCmdDataA q.UINT .15 := TRUE; // motion axis 2
  // aruiPowerSupplyControlWord[1].15 :=TRUE; // drive as PLC device
ELSE
  // switch off power in power supply control word
  AxisData[XMS1.AxisNo].dwUserCmdDataA q.UINT .15 := FALSE;
  AxisData[XMS2.AxisNo].dwUserCmdDataA q.UINT .15 := FALSE;
  // aruiPowerSupplyControlWord[1].15 :=FALSE; // drive as PLC device
END_IF

```

In diesem Beispiel wird zur Vereinfachung am Ende der Initialisierung direkt in "OPERATION_MODE" geschaltet. Für Zwischenzustände siehe Abb. 12-19 "Zustandsgraph eines Versorgers" auf Seite 468. Diese Logik entspricht dem automatischen Start des Versorgers.

**Anwendungsbeispiel Automatik
ohne MB_SIIIPowerSupplyDriveIn-
tegrated**

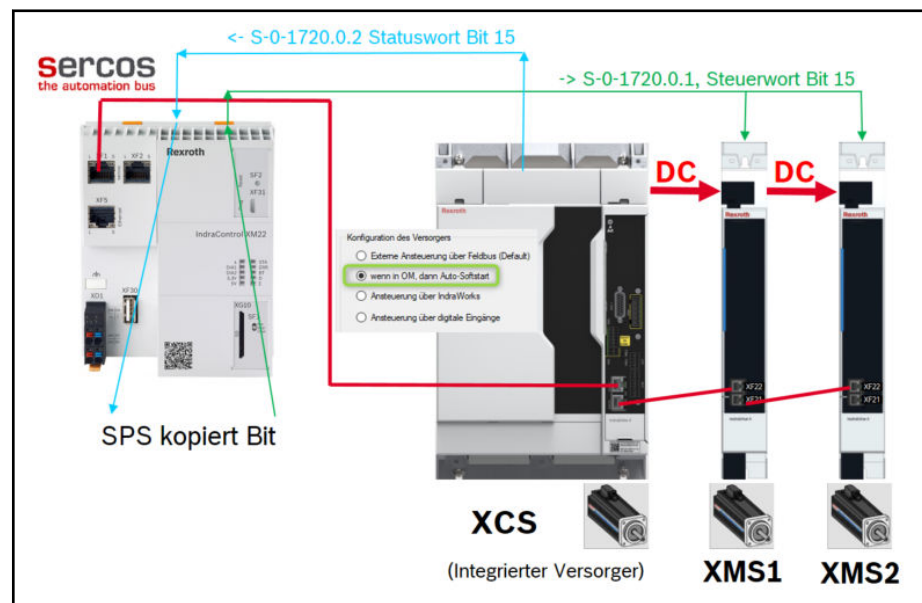


Abb. 12-22: Prinzipschaltbild: Integrierter Versorger XCS im Automatikmodus versorgt zwei Antriebe XMS

Alternativ zur Freigabe des Versorgers mit Hilfe des Funktionsbausteins MB_SIIIPowerSupplyDriveIntegrated kann der automatische Start des Versorgers genutzt werden.

Dazu wählen Sie im Dialog [Abb. 12-21 "Konfiguration des integrierten Versorgers"](#) auf Seite 471 die Option "wenn in OM, dann Auto-Softstart".

Beim XCS reicht es, das Bit 15 "Freigabe" des Parameters S-0-1720.0.2, "Leistungsversorgung Statuswort" in das Signalstatuswort zu konfigurieren.

In der SPS wird das Bit 15 des S-0-1720.0.2, "Leistungsversorgung Statuswort" vom XCS auf Bit 15 "Freigabe" des Parameter S-0-1720.0.1, "Versorger-Steuerwort" aller XMS-Antriebe kopiert.

Zyklisch aufgerufen (Beispiel GATcompact, Machine.actCyclic_active)

```
// Activate power for axis XMS, bit 15 of controlword 1702.0.1 and statusword 1720.0.2 is configured
AxisData[XMS1.AxisNo].wUserCmdDataBitA_q.15 := AxisData[XCS.AxisNo].wUserActualDataBitA_i.15;
AxisData[XMS2.AxisNo].wUserCmdDataBitA_q.15 := AxisData[XCS.AxisNo].wUserActualDataBitA_i.15;
```

12.3.2 MB_SIIIPowerSupplyStandalone

Kurzbeschreibung Der Funktionsbaustein MB_SIIIPowerSupplyStandalone überwacht und steuert einen Versorger.

Die Daten zwischen dem Funktionsbaustein und dem Versorger werden über den Sercos-Feldbus ausgetauscht.

Der Funktionsbaustein kann den internen Zustand des Versorgers identifizieren, Übergänge der Zustandsmaschine des Versorgers auslösen, den Versorger in den gewünschten Zustand bringen und die Betriebsart steuern.

Der Funktionsbaustein überwacht das Statuswort "InProcStatus" des Versorgers kontinuierlich und aktualisiert das Steuerwort "OutProcCtrl" des Versorgers entsprechend seiner Eingangsbelegung.



Der Funktionsbaustein "MB_SIIIPowerSupplyType04" wurde mit Version 15V06 in "MB_SIIIPowerSupplyStandalone" umbenannt. Die Funktion ist unverändert.

Bibliothek:	RMB_SercosIII_Util
Gruppe:	"PowerSupply"
Zielsystem:	MTX 15VRS MLC 15VRS, Funktionspaket SPS MLD mit MPx20

Tab. 12-37: Referenztable des MB_SIIIPowerSupplyStandalone FB

Schnittstellenbeschreibung

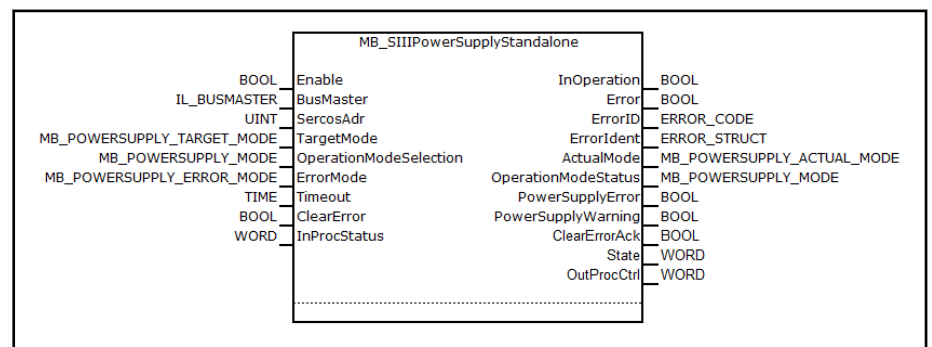


Abb. 12-23: Schnittstelle des MB_SIIIPowerSupplyStandalone FB

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe (zyklisch, zustandsgesteuert)
	BusMaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable „BusMaster“ unterschieden. Wird nur ein Master betrieben, ist die ID des Masters IL_BUSMASTER_0.
	SercosAdr	UINT	Sercos III-Adresse des Slave-Knotens
	TargetMode	MB_POWERSUPPLY_TARGET_MODE	Vom Versorger zu erreichender Zielmodus MB_POWERSUPPLY_TARGET_MODE auf Seite 489
	OperationModeSelection	MB_POWERSUPPLY_MODE	Betriebsart für den Versorger MB_POWERSUPPLY_MODE auf Seite 487
	ErrorMode	MB_POWERSUPPLY_ERROR_MODE	Fehler beim Übergabe-Handling-Eingang für den Versorger MB_POWERSUPPLY_ERROR_MODE auf Seite 488
	ClearError	BOOL	Löschen eines Fehlers im Antrieb durch das Kommando "Fehler löschen" S-0-0099. Nach gelöschtem Fehler wird die angewählte Freigabe wieder aktiviert.
	Timeout	TIME	Max. erlaubte Zeit von jedem "ActualMode" -> "TargetMode"
	InProcStatus	WORD	Statuswort des Versorgers (S-0-1720.0.2)
VAR_OUTPUT	InOperation	BOOL	Der Baustein arbeitet korrekt, die Ausgangsdaten sind gültig
	Error	BOOL	Zeigt an, dass ein Fehler aufgetreten ist (siehe ErrorID und ErrorIdent)
	ErrorID	ERROR_CODE	Fehlerursache
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerursache
	ActualMode	MB_POWERSUPPLY_ACTUAL_MODE	Istzustand der Zustandsmaschine des Versorgers MB_POWERSUPPLY_ACTUAL_MODE auf Seite 487
	OperationModeStatus	MB_POWERSUPPLY_MODE	Istbetriebsart des Versorgers MB_POWERSUPPLY_MODE auf Seite 487
	PowerSupplyError	BOOL	Interner Versorgerfehler
	PowerSupplyWarning	BOOL	Interne Versorgerwarnung
	ClearErrorAck	BOOL	Bestätigung, dass das Kommando "Fehler löschen" S-0-0099 ausgeführt wurde.
	State	WORD	Gerätestatus aus IL_SIIIRemoteSlaveState
	OutProcCtrl	WORD	Steuerwort des Versorgers (S-0-1720.0.1)

Tab. 12-38: Schnittstellenbeschreibung des MB_SIIIPowerSupplyStandalone FB

**Min./Max. und Default-Werte und
Anwendung der Eingänge**

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Anwendung
Enable	BOOL			FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	n. def.	n. def.	IL_BUSMASTER_0	
SercosAdr	UINT	0 (MLD: lokale Achse) 1 (andere Systeme)	Systemspezifisch	0	
TargetMode	MB_POWERSUPPLY_TARGET_MODE			MAINS_OFF_NO_DISCHARGE	Kontinuierlich
OperationModeSelection	MB_POWERSUPPLY_MODE			OPERATIONMODE1	Kontinuierlich
ErrorMode	MB_POWERSUPPLY_ERROR_MODE			ERROR_TRANSITION_NO_DISCHARGE	Kontinuierlich
ClearError	BOOL			FALSE	Kontinuierlich
Timeout	TIME	n. def.	n. def.	T#10s	Kontinuierlich
InProcStatus	WORD	n. def.	n. def.	0	Kontinuierlich

Tab. 12-39: Min-/Max- und Default-Werte des MB_SIIIPowerSupplyStandalone FB:

Funktionsbeschreibung

Der Funktionsbaustein überwacht und steuert den Versorger. Der Funktionsbaustein kann verschiedene Versorgerfunktionen, die in dem Steuerwort des Versorgers festgelegt sind, realisieren.

Die Hauptaufgabe des Funktionsbausteins ist es, einen Zustandsübergang des Versorgers zum gewünschten Zielmodus durchzuführen.

Der zyklische Datenaustausch zwischen dem Funktionsbaustein und dem Versorger wird nachfolgend beschrieben.

- "InProcStatus" – Aktualisiert auf S-0-1720.0.2 (Versorgerstatuswort)
- "OutProcCtrl" – Aktualisiert auf S-0-1720.0.1 (Versorgersteuerwort)

Eine allgemeine Beschreibung der internen Zustandsmaschine des Versorgers finden Sie in der Dokumentation der IndraDrive-Versorgerfunktionen.

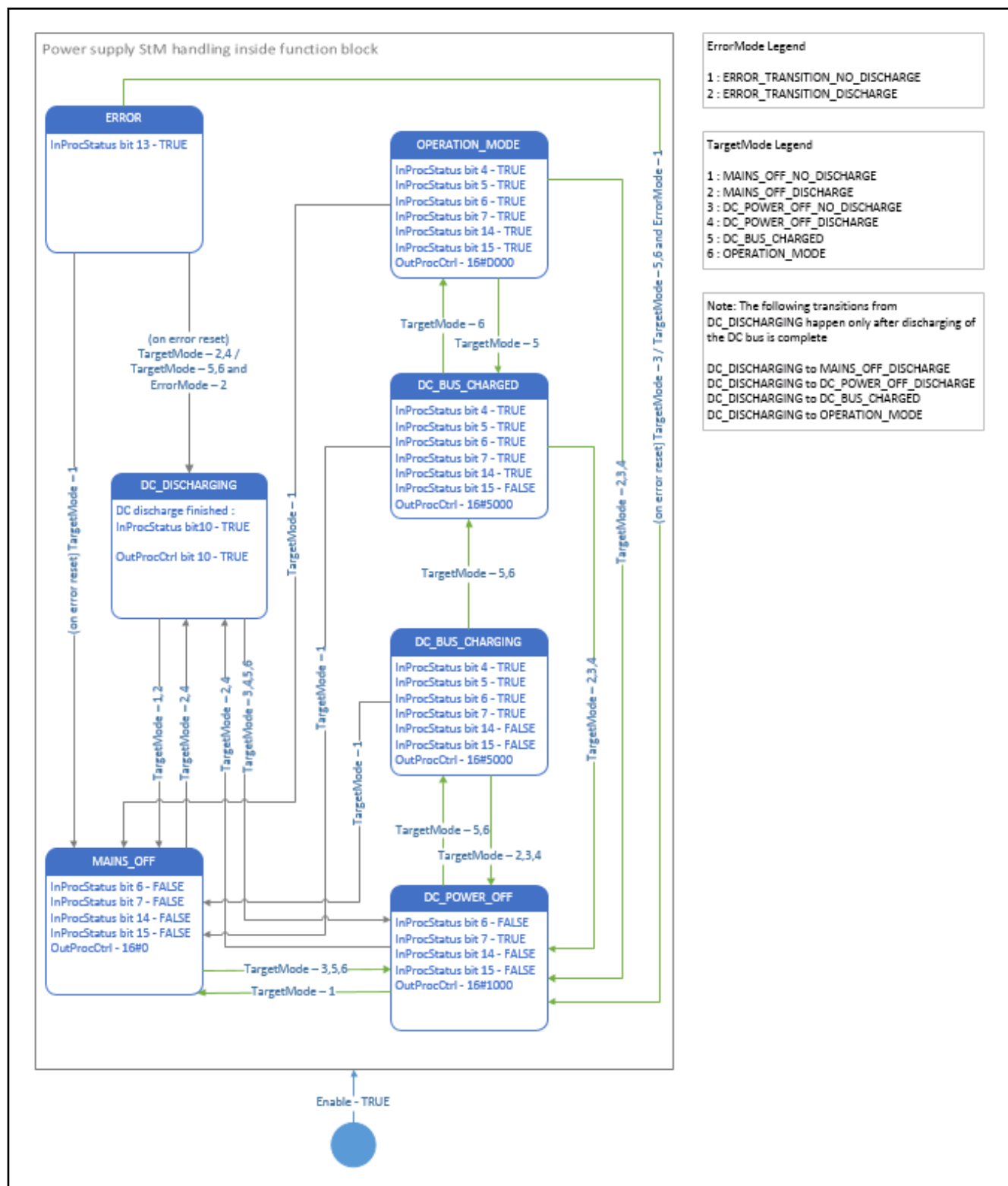


Abb. 12-24: Zustandsgraph eines Versorgers

Der Funktionsbaustein verwendet dieses Statuswort, um den aktuellen Zustand der Zustandsmaschine der Versorger zu identifizieren.

Der von der Zustandsmaschine des Versorgers zu erreichende Zielmodus ist bereits am Eingang "TargetMode" verfügbar.

Der zurückzulegende Weg zum Erreichen des Zielmodus ist dem Funktionsbaustein bekannt. Daher löst der Funktionsbaustein den Übergang der Versorger in den nächsten sofortigen Zustand aus. Dieses Übergehen in den nächsten sofortigen Zustand geht weiter bis die Versorger den Zielmodus nach verschiedenen Zuständen erreicht.

Im Fall eines Funktionsbausteinfehlers werden das Zustandshandling angehalten und der Fehler gelöscht.

Wenn der Versorger den Fehlerzustand erreicht, entscheiden Zielmodus und Fehlermodus über den nächsten Folgezustand des Versorgers.



Es kann zu widersprüchlichen Eingangskombinationen zwischen "ErrorMode" und "TargetMode" kommen.

In solchen Fällen dominiert "TargetMode" unabhängig vom Wert des Eingangs "ErrorMode". Siehe nachstehendes Beispiel.

- ActualMode = _ERROR
- TargetMode = DC_POWER_OFF_NO_DISCHARGE
- ErrorMode = ERROR_TRANSITION_DISCHARGE

Wenn der Versorger den Zustand "Error" verlässt, ist es für "TargetMode" notwendig, dass der Versorger in den Zustand "DC-Power-Off" geht ohne den DC-Bus zu entladen.

Für "ErrorMode" ist es hingegen notwendig, dass der Versorger den DC-Bus entlädt bevor in den nächsten Zustand geschaltet wird.

In diesem Fall dominiert "TargetMode" und der Versorger wird in den Modus "DC-Power-Off" geschaltet ohne den DC-Bus zu entladen.

Dies muss vom Programmierer selbst beim Erstellen von Applikationen beachtet werden.

Die Zeit, die der Versorger benötigt, um den Zielmodus (vom aktuellen Modus) zu erreichen, wird überwacht.

Wenn diese Zeit abläuft, setzt der Eingang "Timeout" einen Fehler und das Handling der Zustandsmaschine wird gestoppt.

Wird der Eingang "Enable" des Funktionsbausteins zurückgesetzt, erkennt der Funktionsbaustein den aktuellen Zustand des Versorgers und entscheidet, ob es notwendig ist der Versorger (DC) abzuschalten.

In diesem Fall wird das Steuerwort gesetzt, um den Versorger in den DC-Power-Off-Modus zu bringen (falls notwendig).

Die Ausgänge "Error", "ErrorID" und "ErrorIdent" enthalten die notwendigen Informationen im Fehlerfall.

Der Eingang zur Betriebsartenauswahl wird sofort und kontinuierlich auf den Bits 8 und 9 des Steuerworts angezeigt. Die Ausgänge des Funktionsbausteins werden während jedem SPS-Scan vom Eingang "InProcStatus" aktualisiert. Jede Statuswortänderung wird sofort an den Ausgängen ausgegeben.

Außerdem kann der Versorgersfehler über den Eingang "ClearError" gelöscht werden. Die steigende Flanke dieses Eingangs löst den "Fehler löschen"-Befehl S-0-0099 für die Versorger aus. Der Status des erfolgreichen Fehler löschen wird über den Ausgang "ClearErrorAck" des Funktionsbausteins ausgegeben.

Die Diagnoseinformation über den Versorger wird kontinuierlich am Funktionsbaustein "State" aktualisiert. Aktualisiert wird durch den Funktionsbaustein IL_SIIIRemoteSlaveState.

Fehlerbehandlung

Der Funktionsbaustein kann aufgrund ungültiger Eingänge oder abgelaufenem Timeout-Timer in einen Fehlerzustand übergehen. In beiden Fällen erzeugt der Funktionsbaustein einen Fehler und bricht die Operation ab.

Der Benutzer kann die Fehlerursache beheben, den Eingang "Enable" zurücksetzen und erneut setzen (Standardvorgehen beim Fehler zurücksetzen).

Der Funktionsbaustein arbeitet anschließend weiter. Zu den verfügbaren Fehlercodes, siehe nachfolgende Tabelle.

Zusätzlich zu den Fehlern der intern genutzten Funktionsbausteine IL_SIII-Command und IL_SIIIRemoteSlaveState, sind folgende Fehler des Bausteines in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

STATE_MACHINE_ERROR	16#00000006	State	Fehler in der Zustandsmaschine
INPUT_INVALID_ERROR	16#00002990	TargetMode	Eingang TargetMode ist ungültig
INPUT_INVALID_ERROR	16#00002991	OperationModeSelection	Eingang OperationModeSelection ist ungültig
INPUT_INVALID_ERROR	16#00002992	ErrorMode	Eingang ErrorMode ist ungültig
STATE_MACHINE_ERROR	16#00002993	0	Zeitüberschreitung (Actual mode -> Target mode)

Tab. 12-40: Fehlercodes FB MB_SIIIPowerSupplyStandalone

Anwendungsbeispiel mit MB_SIII-PowerSupplyStandalone

An einem Versorger XVR sind zwei IndraDrive XMS angeschlossen.

Mit dem Funktionsbaustein MB_SIIIPowerSupplyStandalone wird der Versorger XVR gesteuert. Mit den Ausgängen des Funktionsbausteins werden die Antriebe an den Wechselrichtern XMS freigegeben (von "bb" nach "Ab" geschaltet).

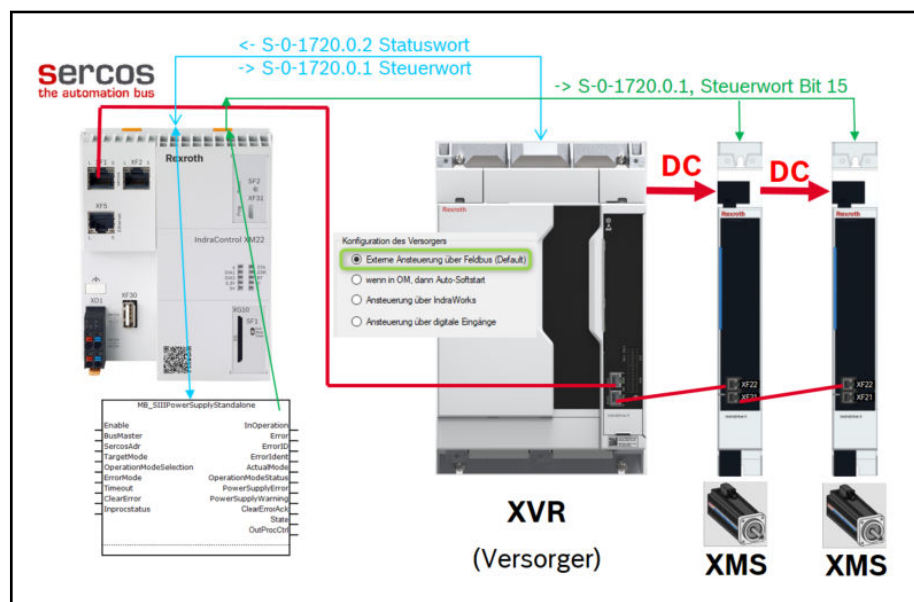


Abb. 12-25: Prinzipialschaltbild: Versorger XVR versorgt zwei Antriebe XMS



Das folgende SPS-Programmbeispiel kann auch für Versorger HMV und KMV verwendet werden. Dabei entfällt das Kopieren des Versorgerstatuswort auf das Versorgersteuerwort der Slaves. In diesem Fall erfolgt die Informationsübertragung über den Modulbus.

Die folgenden Parameter müssen beim Versorger XVR zyklisch konfiguriert und auf SPS Variablen gemapped werden:

- MDT: S-0-1720.0.1, "Versorger-Steuerwort"
- AT: S-0-1720.0.2, "Leistungsversorgung Statuswort"

Bei den XMS-Antrieben reicht es, das Bit 15 "Freigabe" des Parameters S-0-1720.0.1, "Versorger-Steuerwort" in das Signalsteuerwort zu konfigurieren.

Im folgenden SPS-Programmbeispiel wird davon ausgegangen, dass bei den XMS-Antrieben der Daten-Container "A" verwendet wird.

Statt dessen können auch alle anderen Container "B bis D" genutzt werden. Dabei ist das SPS-Programm entsprechend anzupassen.



Mit Hilfe des Funktionsbausteines "MB_GetCyclicParameterHandle" und den Funktionen "MB_ReadCyclicParameter" und "MB_WriteCyclicParameter" (siehe Bibliothek "ML_TechMotion-Base") kann das SPS-Programm unabhängig von den jeweils verwendeten Datencontainern "A bis D" implementiert werden.

Werden die Antriebe XMS als SPS-Geräte eingebunden, sind die gleichen Parameter zyklisch zu konfigurieren und es müssen Variablen auf diese Parameter gemapped werden.

Im folgenden Beispiel sind entsprechende Anweisungen auskommentiert enthalten und mit "drive as PLC device" gekennzeichnet.

Variablendefinition

```
fbMB_SIIIPowerSupply: MB_SIIIPowerSupplyStandalone;
bEnablePower: BOOL:=FALSE;
```

Eintritt in die Initialisierung (Beispiel GATcompact, Machine.actInit_entry)

```
// switch off power supply
bEnablePower := FALSE;
```

Ende der Initialisierung (Beispiel GATcompact, Machine.actInit_active)

```
IF (ImcStatus.Admin._OpModeAck = IMC_MODE_STATUS_BB) AND (TE_AxisInterfaceType01Prog.InitDone = TRUE) THEN
// *****
// Now we reached BB (operating mode)
// Add your code for writing the parameters (in BB) here
// *****
bEnablePower := TRUE;
IF fbMB_SIIIPowerSupply.ActualMode = MB_POWER_SUPPLY_ACTUAL_MODE.OPERATION_MODE THEN
  uiStateInit := uiSTATE_INIT_PREPARE_COLDSTART;
  bInitDone := TRUE; // Set bInitDone if the initialization is done
END_IF
END_IF
```

Zyklisch aufgerufen (Beispiel GATcompact, Machine.actCyclic_active)

```
//----- handle power-supply -----
//-----
// copy status word from bus to FB
fbMB_SIIIPowerSupply.InProcStatus := XVR_InProcStatus; // drive as PLC device
```

```

// always switching direct to OPERATION_MODE
fbMB_SIIIPowerSupply(
  Enable:= bEnablePower,
  BusMaster:= IL_BUSMASTER.IL_BUSMASTER_0,
  SercosAdr:= 42,
  TargetMode:= MB_POWER_SUPPLY_TARGET_MODE.OPERATION_MODE,
  OperationModeSelection:= ,
  ErrorMode:= ,
  Timeout:= ,
  ClearError:= PlcProg.fbMachine.ClearError,
  OperationModeStatus=> ,
  PowerSupplyError=> ,
  PowerSupplyWarning=> ,
  ClearErrorAck=> ,
  State=> );
IF fbMB_SIIIPowerSupply.Error THEN
  bEnablePower := FALSE;
  mSetFBError(fbMB_SIIIPowerSupply.ErrorID, fbMB_SIIIPowerSupply.ErrorIdent);
END_IF
// copy control word from FB to bus
XVR_OutProcCtrl := fbMB_SIIIPowerSupply.OutProcCtrl; // drive as PLC device
// copy output of FB to power supply control word, bit 15 of all XMS
IF fbMB_SIIIPowerSupply.ActualMode = MB_POWER_SUPPLY_ACTUAL_MODE.OPERATION_MODE THEN
  // activate power in power supply control word
  AxisData[XMS1.AxisNo].dwUserCmdDataA.q.UINT_15 := TRUE; // motion axis 1
  AxisData[XMS2.AxisNo].dwUserCmdDataA.q.UINT_15 := TRUE; // motion axis 2
  // aruiPowerSupplyControlWord[1].15 :=TRUE; // drive as PLC device
ELSE
  // switch off power in power supply control word
  AxisData[XMS1.AxisNo].dwUserCmdDataA.q.UINT_15 := FALSE;
  AxisData[XMS2.AxisNo].dwUserCmdDataA.q.UINT_15 := FALSE;
  // aruiPowerSupplyControlWord[1].15 :=FALSE; // drive as PLC device
END_IF

```

In diesem Beispiel wird zur Vereinfachung am Ende der Initialisierung direkt in "OPERATION_MODE" geschaltet. Für Zwischenzustände siehe [Abb. 12-24 "Zustandsgraph eines Versorgers"](#) auf Seite 476.

12.3.3 MB_SIIIPowerSupplyDcDcIslandGrid

Kurzbeschreibung

Der Funktionsbaustein MB_SIIIPowerSupplyDcDcIslandGrid überwacht und steuert einen Versorger mit einem der folgenden Funktionspakete:

- **DCE** -> "DC/DC-Steller" für DC/DC Stellerbetrieb
- **MSE** -> "Netzfunktion" für Inselnetzbetrieb

Die Daten zwischen dem Funktionsbaustein und dem Versorger werden über den Sercos-Feldbus ausgetauscht.

Der Funktionsbaustein kann den internen Zustand des Versorgers identifizieren, Übergänge der Zustandsmaschine des Versorgers auslösen, den Versorger in den gewünschten Zustand bringen und die Betriebsart steuern.

Der Funktionsbaustein überwacht das Statuswort "InProcStatus" des Versorgers kontinuierlich und aktualisiert das Steuerwort "OutProcCtrl" des Versorgers entsprechend seiner Eingangsbelegung.

Bibliothek:	RMB_SercosIII_Util
Gruppe:	"PowerSupply"
Zielsystem:	MTX 15VRS MLC 15VRS, Funktionspaket SPS MLD mit MPx20

Tab. 12-41: Referenztable des MB_SIIIPowerSupplyDcDcIslandGrid FB

Schnittstellenbeschreibung

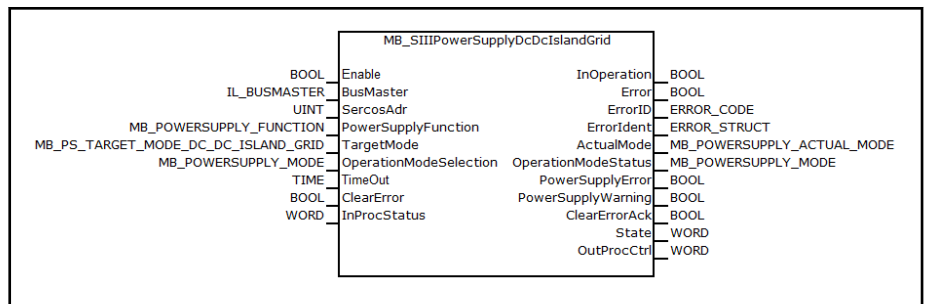


Abb. 12-26: Schnittstelle des MB_SIIIPowerSupplyDcDcIslandGrid FB

	Name	Typ	Kommentar
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe (zyklisch, zustandsgesteuert)
	BusMaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable „BusMaster“ unterschieden. Wird nur ein Master betrieben, ist die ID des Masters IL_BUSMASTER_0.
	SercosAdr	UINT	Sercos III-Adresse des Slave-Knotens
	PowerSupply-Function	MB_POWERSUPPLY_FUNCTION	Erweiterte Versorgerfunktion, zusätzliche Funktionspakete des Versorgers notwendig. MB_POWERSUPPLY_FUNCTION auf Seite 490
	TargetMode	MB_PS_TARGET_MODE_DC_DC_ISLAND_GRID	Vom Versorger zu erreichender Zielmodus MB_PS_TARGET_MODE_DC_DC_ISLAND_GRID auf Seite 490
	OperationModeSelection	MB_POWERSUPPLY_MODE	Betriebsart für den Versorger MB_POWERSUPPLY_MODE auf Seite 487
	Timeout	TIME	Max. erlaubte Zeit von jedem "ActualMode" -> "TargetMode"
	ClearError	BOOL	Löschen eines Fehlers im Antrieb durch das Kommando "Fehler löschen" S-0-0099. Nach gelöschtem Fehler wird die angewählte Freigabe wieder aktiviert.
	InProcStatus	WORD	Statuswort des Versorgers (S-0-1720.0.2)
VAR_OUTPUT	InOperation	BOOL	Der Baustein arbeitet korrekt, die Ausgangsdaten sind gültig
	Error	BOOL	Zeigt an, dass ein Fehler aufgetreten ist (siehe ErrorID und ErrorIdent)
	ErrorID	ERROR_CODE	Fehlerursache
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerursache
	ActualMode	MB_POWERSUPPLY_ACTUAL_MODE	Istzustand der Zustandsmaschine des Versorgers MB_POWERSUPPLY_ACTUAL_MODE auf Seite 487
	OperationMode-Status	MB_POWERSUPPLY_MODE	Istbetriebsart des Versorgers MB_POWERSUPPLY_MODE auf Seite 487

	Name	Typ	Kommentar
	PowerSupplyError	BOOL	Interner Versorgerfehler
	PowerSupplyWarning	BOOL	Interne Versorgerwarnung
	ClearErrorAck	BOOL	Bestätigung, dass das Kommando "Fehler löschen" S-0-0099 ausgeführt wurde.
	State	WORD	Gerätestatus aus IL_SIIIRemoteSlaveState
	OutProcCtrl	WORD	Steuerwort des Versorgers (S-0-1720.0.1)

Tab. 12-42: Schnittstellenbeschreibung des MB_SIIIPowerSupplyDcDclslandGrid FB

Min./Max. und Default-Werte und Anwendung der Eingänge

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Anwendung
Enable	BOOL			FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	n. def.	n. def.	IL_BUSMASTER_0	Steigende Flanke an Enable
SercosAdr	UINT	0 (MLD: lokale Achse) 1 (andere Systeme)	Systemspezifisch	0	Steigende Flanke an Enable
PowerSupplyFunction	MB_POWERSUPPLY_FUNCTION			PS_FUNCTION_DC_DC	Steigende Flanke an Enable
TargetMode	MB_PS_TARGET_MODE_DC_DC_ISLAND_GRID			MAINS_OFF	Kontinuierlich
OperationModeSelection	MB_POWERSUPPLY_MODE			OPERATION_MODE1	Kontinuierlich
ClearError	BOOL			FALSE	Kontinuierlich
Timeout	TIME	n. def.	n. def.	T#10s	Kontinuierlich
InProcStatus	WORD	n. def.	n. def.	0	Kontinuierlich

Tab. 12-43: Min-/Max- und Default-Werte des MB_SIIIPowerSupplyDcDclslandGrid FB

Funktionsbeschreibung

Der Funktionsbaustein überwacht und steuert den Versorger. Der Funktionsbaustein kann verschiedene Versorgerfunktionen realisieren, die im Steuerwort des Versorgers festgelegt sind.

Die Hauptaufgabe des Funktionsbausteins ist es, den Versorger zum gewünschten Zielmodus zu bringen.

Der zyklische Datenaustausch zwischen dem Funktionsbaustein und dem Versorger wird nachfolgend beschrieben.

- "InProcStatus" - Aktualisiert auf S-0-1720.0.2 (Versorgerstatuswort)
- "OutProcCtrl" - Aktualisiert auf S-0-1720.0.1 (Versorgersteuerwort)

Eine allgemeine Beschreibung der internen Zustandsmaschine des Versorgers finden Sie in der Dokumentation der IndraDrive-Versorgerfunktionen.

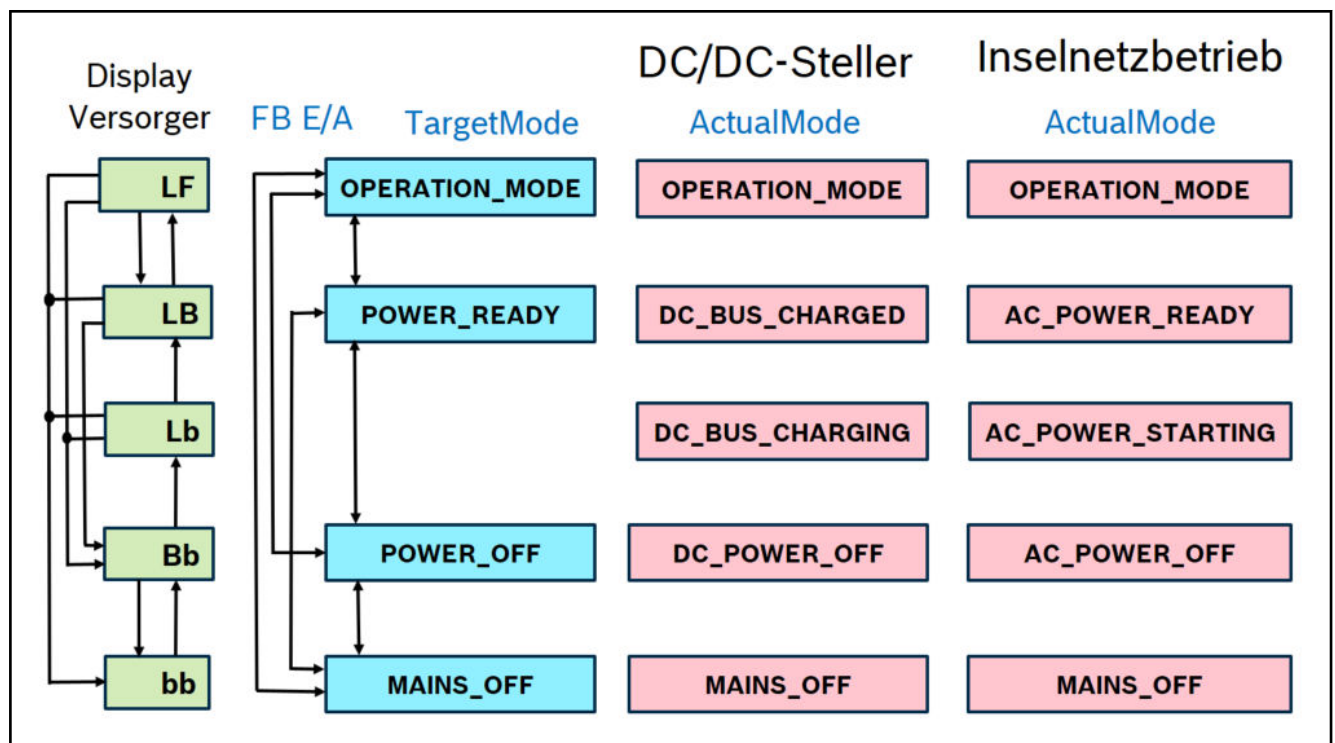


Abb. 12-27: Vereinfachter Zustandsgraph eines Versorgers mit den dazugehörigen Funktionsbaustein (FB) Ein-/Ausgängen

Der Funktionsbaustein verwendet das Versorgerstatuswort, um den aktuellen Zustand der Zustandsmaschine des Versorgers zu identifizieren.

Der aktuelle Zustand wird am Ausgang "ActualMode" in Abhängigkeit der am Eingang "PowerSupplyFunction" gewählten Funktion **DC/DC-Steller** oder **Inselnetzbetrieb** ausgegeben (siehe Zustandsgraph oben).

Der von der Zustandsmaschine des Versorgers zu erreichende Zielmodus wird über den Eingang "TargetMode" vorgegeben. Der zurückzulegende Pfad zum Erreichen des Zielmodus ist im Funktionsbaustein hinterlegt, der Funktionsbaustein führt die notwendigen Zustandsübergänge bis zum Zielmodus selbständig aus.

Im Fall eines Funktionsbausteinfehlers wird das Zustandshandling angehalten und ein Fehler am Ausgang "Error" signalisiert.

Die Zeit, die der Versorger benötigt, um vom aktuellen Modus den Zielmodus zu erreichen, wird mit Hilfe des Eingangs "Timeout" überwacht. Wenn diese Timeout-Zeit abläuft, gibt der Funktionsbaustein einen Fehler aus und das Handling der Zustandsmaschine wird gestoppt.

Wird der Eingang "Enable" des Funktionsbausteins zurückgesetzt, schaltet der Funktionsbaustein den Versorger ab. Das Steuerwort wird zurückgesetzt, um den Versorger in den Mains-Off-Modus zu bringen.

Die Ausgänge "Error", "ErrorID" und "ErrorIdent" enthalten im Fehlerfall die notwendigen Informationen.

Der Eingang zur Betriebsartenauswahl wird kontinuierlich auf die Bits 8 und 9 des Steuerworts ausgegeben. Die Ausgänge des Funktionsbausteins werden bei jedem Aufruf des Funktionsbausteins vom Eingang "InProcStatus" aktualisiert. Jede Statuswortänderung wird sofort an den Ausgängen ausgegeben.

Außerdem kann ein Versorgerfehler über den Eingang "ClearError" gelöscht werden. Die steigende Flanke dieses Eingangs löst den "Fehler löschen"-Befehl S-0-0099 für den Versorger aus. Der Status des erfolgreichen Fehler lö-

schen wird über den Ausgang "ClearErrorAck" des Funktionsbausteins ausgegeben.

Die Diagnoseinformation des Versorgers wird kontinuierlich am Funktionsbaustein Ausgang "State" aktualisiert.

Fehlerbehandlung

Der Funktionsbaustein kann aufgrund ungültiger Eingänge oder abgelaufenem Timeout-Timer in einen Fehlerzustand übergehen. In beiden Fällen erzeugt der Funktionsbaustein einen Fehler und bricht die Operation ab.

Der Benutzer kann die Fehlerursache beheben, den Eingang "Enable" zurücksetzen und erneut setzen (Standardvorgehen beim Fehler zurücksetzen).

Der Funktionsbaustein arbeitet anschließend weiter. Zu den verfügbaren Fehlercodes, siehe nachfolgende Tabelle.

Zusätzlich zu den Fehlern der intern genutzten Funktionsbausteine IL_SIII-Command, IL_SIIIReadParameter und IL_SIIIRemoteSlaveState, sind folgende Fehler des Bausteines in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additional2	Kommentar
STATE_MACHINE_ERROR	16#00000006	State	Fehler in der Zustandsmaschine
INPUT_INVALID_ERROR	16#00002990	TargetMode	Eingang TargetMode ist ungültig
INPUT_INVALID_ERROR	16#00002991	OperationModeSelection	Eingang OperationModeSelection ist ungültig
STATE_MACHINE_ERROR	16#00002993	0	Zeitüberschreitung (Actual mode -> Target mode)
INPUT_INVALID_ERROR	16#00002994	PowerSupplyFunction	Eingang PowerSupplyFunction ist ungültig

Tab. 12-44: Fehlercodes FB MB_SIIIPowerSupplyDcDcIslandGrid

Anwendungsbeispiel mit MB_SIII-PowerSupplyDcDcIslandGrid

Mit einem DC/DC-Steller soll eine Batterie geladen werden.
Mit dem Funktionsbaustein MB_SIIIPowerSupplyDcDcIslandGrid wird der Wechselrichter HMU05 gesteuert.

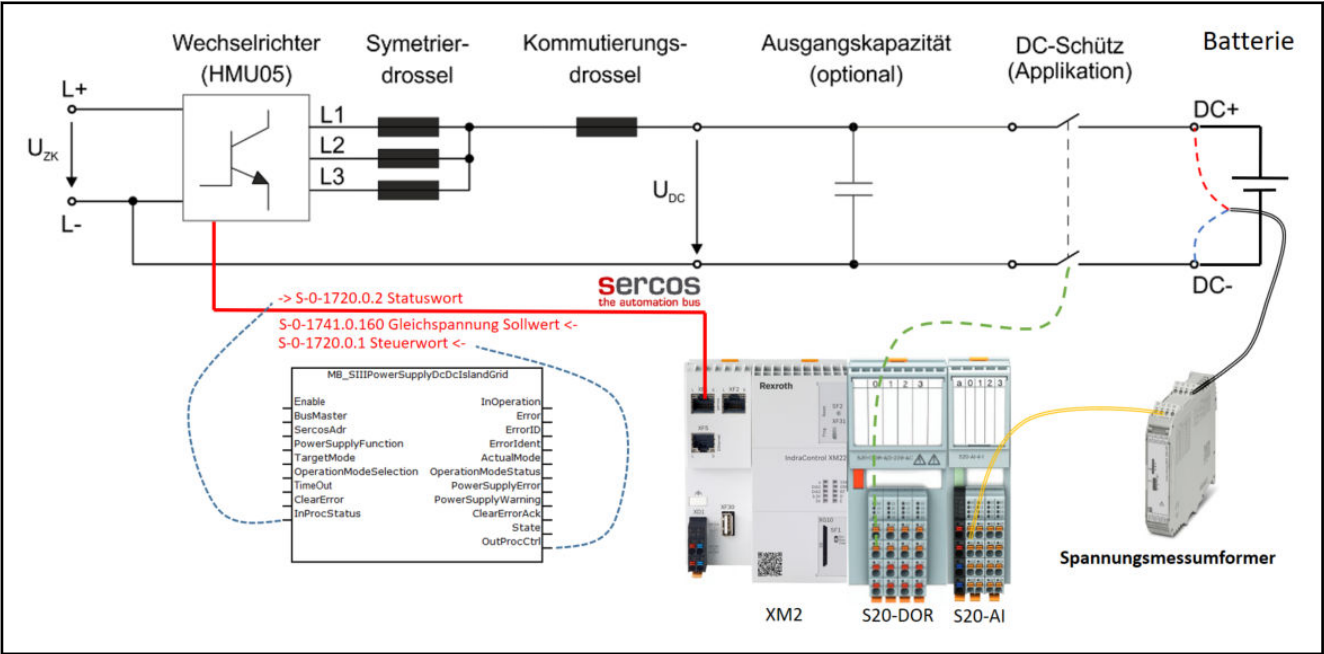


Abb. 12-28: Anwendungsbeispiel: Aufbau zum Laden einer Batterie

Die folgenden Parameter müssen beim HMU05 mindestens zyklisch konfiguriert werden:

- MDT: S-0-1720.0.1, "Versorger-Steuerwort"
- MDT: S-0-1741.0.160 "Gleichspannung Sollwert"
- AT: S-0-1720.0.2, "Leistungsversorgung Statuswort"

Am HMU05 ist zumindest die folgende Parametrierung auszuführen:

- Hauptbetriebsart: Gleichspannungsregelung
- Nebenbetriebsart 1: Gleichstromregelung
- Im S-0-1741.0.150 " DC/DC-Steller Steuerwort", Bit 0 = 1, d.h. Aufbau der DC-Ausgangsspannung vor Freigabe nicht erforderlich

Das Schalten des DC-Schützes wird hier beispielhaft mit einem lokal an einer Steuerung XM22 angeschlossenen S20 Relaismodul "S20-DOR" und die Messung der Batteriespannung mit einem lokal angeschlossenen S20 Analogmodul "S20-AI" ausgeführt. Es kann alternativ auch andere Hardware verwendet werden. Die Batteriespannung könnte über einen Analogeingang des HMU gemessen und der Wert über Sercos an die XM22 übertragen werden.

Ablauf im untenstehenden Programmbeispiel

Variable uiState	Beschreibung	Anwahl "TargetMode"	Zielzustand	
			ActualMode	Display Versorger
1	Anwahl Leistungsteil Aus, DC-Schütz offen	MAINS_OFF	MAINS_OFF	bb
2	Setzen des Spannungs-Sollwertes ($U_{\text{Soll, DC}}$) gleich der gemessenen Batteriespannung und Anwahl Hauptbetriebsart Spannungsregelung. Anwahl Leistungsteil betriebsbereit. Versorger prüft ob $U_{\text{ZK}} > U_{\text{Soll, DC}} * 1.01$? Noch keine DC Ausgangsspannung	POWER_OFF	DC_POWER_OFF	Bb
3	Anwahl Laden der DC Ausgangsspannung	OPERATION_MODE	OPERATION_MODE	LF
4	Wenn DC Ausgangsspannung geladen (damit ist $U_{\text{DC}} = \text{Batteriespannung}$) und die Hauptbetriebsart quittiert ist, wird wieder "Bb" vorgegeben.	POWER_OFF	DC Ausgangsspannung soll erhalten bleiben. Schnell weiter mit 5	
5	Schließen des DC-Schützes. Wenn DC-Schütz geschlossen, dann Anwahl Nebenbetriebsart 1: Stromregelung und Vorgabe "LF"	OPERATION_MODE	OPERATION_MODE	LF
6	Wenn Quittierung Nebenbetriebsart 1 und "LF", weiter mit Schritt "7"	OPERATION_MODE	OPERATION_MODE	LF
7	Fertig. Variable uiState auf "1" setzen um abzuschalten.	OPERATION_MODE	OPERATION_MODE	LF

Tab. 12-45: Beschreibung der Zustände

Der Ablauf wird über die Variable "uiState" gesteuert:

- uiState = "1" aktiviert den Funktionsbaustein und schaltet ab
- uiState = "2" aktiviert ebenfalls den Funktionsbaustein und startet den automatischen Ablauf



Im Schritt 4 wird in diesem Beispiel die Regelung ausgeschaltet und im Schritt 5 mit Schließen des Schützes mit Stromregelung wieder aktiviert. Alternativ könnte man je nach Applikation auch direkt in Stromregelung mit Sollwert 0 wechseln und den Schütz schließen.

Variablendefinition

```
fbPowerSupplyDcDcIslandGrid: MB_SIIIPowerSupplyDcDcIslandGrid;
wPS_Status : WORD;           // connect to S-0-1720.0.2 "Power supply status word"
wPS_Control: WORD;           // connect to S-0-1720.0.1 "Power supply control word"
uiState    : UINT := 0;      // set to 1 to Enable FB and stay in OFF or set to 2 for automatic start-up
```

Zyklisch aufgerufen

```
//----- handle power-supply -----
//-----
CASE uiState OF
  1: fbPowerSupplyDcDcIslandGrid.Enable := TRUE;
    // to do: Open contactor between output DC/DC converter and battery (S20-DOR)
    fbPowerSupplyDcDcIslandGrid.TargetMode := MB_PS_TARGET_MODE_DC_DC_ISLAND_GRID.MAINS_OFF; // all OFF

  2: fbPowerSupplyDcDcIslandGrid.Enable := TRUE;
    // to do: Open contactor between output DC/DC converter and battery (S20-DOR)
    // to do: Set S-0-1741.0.160 "DC voltage command value" to measured battery voltage (S20-AI)
    // Main operating mode Voltage control
    fbPowerSupplyDcDcIslandGrid.OperationModeSelection := MB_POWERSUPPLY_MODE.OPERATIONMODE1;
    // start automatic start-up. Switch to ready for operation Bb
    fbPowerSupplyDcDcIslandGrid.TargetMode := MB_PS_TARGET_MODE_DC_DC_ISLAND_GRID.POWER_OFF;
    IF fbPowerSupplyDcDcIslandGrid.ActualMode = MB_POWERSUPPLY_ACTUAL_MODE.DC_POWER_OFF AND
       TRUE (* to do: contactor is open? *)
    THEN
      uiState := uiState + 1;
    END_IF

  3: // Charging to LF
    fbPowerSupplyDcDcIslandGrid.TargetMode := MB_PS_TARGET_MODE_DC_DC_ISLAND_GRID.OPERATION_MODE;
    uiState := uiState + 1;

  4: // Charged to LF AND Ack Main op-mode
    IF fbPowerSupplyDcDcIslandGrid.ActualMode = MB_POWERSUPPLY_ACTUAL_MODE.OPERATION_MODE AND
       fbPowerSupplyDcDcIslandGrid.OperationModeStatus = MB_POWERSUPPLY_MODE.OPERATIONMODE1
    THEN
      // Back to "ready for operation" Bb
      fbPowerSupplyDcDcIslandGrid.TargetMode := MB_PS_TARGET_MODE_DC_DC_ISLAND_GRID.POWER_OFF;
      uiState := uiState + 1;
    END_IF

  5:
    // to do: Close contactor between output DC/DC converter and battery (S20-DOR)
    IF TRUE (* to do: contactor closed? *) THEN
      // activate secondary operation mode 1, (DC) current control
      fbPowerSupplyDcDcIslandGrid.OperationModeSelection := MB_POWERSUPPLY_MODE.OPERATIONMODE2;
      // Switch to operating LF
      fbPowerSupplyDcDcIslandGrid.TargetMode := MB_PS_TARGET_MODE_DC_DC_ISLAND_GRID.OPERATION_MODE;
      uiState := uiState + 1;
    END_IF

  6: // LF reached AND Ack 2nd op-mode
    IF fbPowerSupplyDcDcIslandGrid.ActualMode = MB_POWERSUPPLY_ACTUAL_MODE.OPERATION_MODE AND
       fbPowerSupplyDcDcIslandGrid.OperationModeStatus = MB_POWERSUPPLY_MODE.OPERATIONMODE2
    THEN
      uiState := uiState + 1;
    END_IF

  7: ; // Finished. Set uiState to 1 to switch power supply OFF

  99: ; // Error!! Add error-handling here
      // to do: Open contactor between output DC/DC converter and battery (S20-DOR)
END_CASE
// call function block. I/Os are handled above
fbPowerSupplyDcDcIslandGrid(
```

```

Enable:= ,
InOperation=> ,
Error=> ,
ErrorID=> ,
ErrorIdent=> ,
BusMaster:= ,
SercosAdr:= 42,
PowerSupplyFunction:= MB_POWERSUPPLY_FUNCTION.PS_FUNCTION_DC_DC,
TargetMode:= ,
OperationModeSelection:= ,
Timeout:= ,
ClearError:= ,
InProcStatus:= wPS_Status,
ActualMode=> ,
OperationModeStatus=> ,
PowerSupplyError=> ,
PowerSupplyWarning=> ,
ClearErrorAck=> ,
State=> ,
OutProcCtrl=> wPS_Control);
IF fbPowerSupplyDcDcIslandGrid.Error THEN
    uiState := 99;
END_IF

```

12.3.4 MB_POWERSUPPLY_MODE

Kurzbeschreibung Diese Aufzählung wird vom Funktionsbausteineingang "OperationModeSelection" und vom Funktionsbausteinausgang "OperationModeStatus" verwendet.

Jedes Element dieser Aufzählung entspricht einer der vier Betriebsarten des Versorgers.

Aufzählungselemente

Element	Beschreibung
OPERATIONMODE1	Betriebsart 1 (S-0-1709.0.1, Primäre Betriebsart)
OPERATIONMODE2	Betriebsart 2 (S-0-1709.0.2, Nebenbetriebsart 1)
OPERATIONMODE3	Betriebsart 3 (S-0-1709.0.3, Nebenbetriebsart 2)
OPERATIONMODE4	Betriebsart 4 (S-0-1709.0.4, Nebenbetriebsart 3)

Tab. 12-46: Elemente der Aufzählung MB_POWERSUPPLY_MODE

12.3.5 MB_POWERSUPPLY_ACTUAL_MODE

Kurzbeschreibung Diese Aufzählung wird vom Funktionsbausteinausgang "ActualMode" verwendet.

Jedes Element dieser Aufzählung stellt einen Zustand in der Zustandsmaschine des Versorgers dar.

Aufzählungselemente

Element	Beschreibung
MAINS_OFF	Der Versorger ist nicht ans Netz angeschlossen. Gibt es einen Ausgang, um den Schütz zu steuern, ist der Schütz ausgeschaltet.
DC_POWER_OFF	Der Schütz wird aktiviert. Die DC-Aufladung hat noch nicht begonnen.
DC_BUS_CHARGING	Die DC-Buskapazität wird gerade aufgeladen. Dieser Prozess kann, wenn nötig, durch eine Änderung des Eingangs "TargetMode" unterbrochen werden.
DC_DISCHARGING	Die DC-Buskapazität wird bis auf ein niedriges Spannungsniveau entladen. Dieser Prozess kann, wenn nötig, durch eine Änderung des Eingangs "TargetMode" unterbrochen werden.
DC_BUS_CHARGED	DC-Bus ist auf Spitzenwert des Netzes aufgeladen. Die Betriebsart ist nicht aktiv.

Element	Beschreibung
AC_POWER_OFF	Der Schütz wird aktiviert. Die Trafo-Magnetisierung und ggfs. Synchronisation hat noch nicht begonnen. Diesen Status gibt es nur bei Inselnetzbetrieb.
AC_POWER_STARTING	Trafo-Magnetisierung und ggfs. Synchronisation läuft. Diesen Status gibt es nur bei Inselnetzbetrieb.
AC_POWER_READY	Leistungsbereit. Ziel-Sollwertspannung erreicht und ggfs. synchron. Diesen Status gibt es nur bei Inselnetzbetrieb.
OPERATION_MODE	Die ausgewählte Betriebsart ist aktiv.
ERROR	Der Versorger hat einen internen Fehler erkannt und ist im Fehlerzustand
UNKNOWN_STATE	Statuswort des Versorgers entspricht keinem gültigen Zustand.

Tab. 12-47: Elemente der Aufzählung MB_POWERSUPPLY_ACTUAL_MODE



Es kann zu widersprüchlichen Eingangskombinationen zwischen "ErrorMode" und "TargetMode" kommen.

In solchen Fällen dominiert "TargetMode" unabhängig vom Wert des Eingangs "ErrorMode". Siehe nachstehendes Beispiel.

- ActualMode = _ERROR
- TargetMode = DC_POWER_OFF_NO_DISCHARGE
- ErrorMode = ERROR_TRANSITION_DISCHARGE

Wenn der Versorger den Zustand "Error" verlässt, ist es für "TargetMode" notwendig, dass der Versorger in den Zustand "DC-Power-Off" geht ohne den DC-Bus zu entladen.

Für "ErrorMode" ist es hingegen notwendig, dass der Versorger den DC-Bus entlädt bevor in den nächsten Zustand geschaltet wird.

In diesem Fall dominiert "TargetMode" und der Versorger wird in den Modus "DC-Power-Off" geschaltet ohne den DC-Bus zu entladen.

Dies muss vom Programmierer selbst beim Erstellen von Applikationen beachtet werden.

12.3.6 MB_POWERSUPPLY_ERROR_MODE

Kurzbeschreibung Dies Aufzählung wird vom Funktionsbausteineingang "ErrorMode" verwendet.

Jedes Element dieser Aufzählung ist entscheidend dafür, in welchen Zustand der Versorger schaltet, wenn er seinen internen Fehlerzustand verlässt.

Aufzählungselemente

Element	Beschreibung
ERROR_TRANSITION_NO_DISCHARGE	Der Funktionsbaustein schaltet entweder auf "Netz aus" (Mains Off) oder auf DC-Power-off. Dies hängt vom Zielmodus ab. Es kommt nicht zu einer DC-Entladung
ERROR_TRANSITION_DISCHARGE	Der Funktionsbaustein wechselt auf DC-Entladung und beendet diese, bevor in einen anderen Zustand geschaltet wird

Tab. 12-48: Elemente der Aufzählung MB_POWERSUPPLY_ERROR_MODE



Es kann zu widersprüchlichen Eingangskombinationen zwischen "ErrorMode" und "TargetMode" kommen.

In solchen Fällen dominiert "TargetMode" unabhängig vom Wert des Eingangs "ErrorMode". Siehe nachstehendes Beispiel.

- ActualMode = _ERROR
- TargetMode = DC_POWER_OFF_NO_DISCHARGE
- ErrorMode = ERROR_TRANSITION_DISCHARGE

Wenn der Versorger den Zustand "Error" verlässt, ist es für "TargetMode" notwendig, dass der Versorger in den Zustand "DC-Power-Off" geht ohne den DC-Bus zu entladen.

Für "ErrorMode" ist es hingegen notwendig, dass der Versorger den DC-Bus entlädt bevor in den nächsten Zustand geschaltet wird.

In diesem Fall dominiert "TargetMode" und der Versorger wird in den Modus "DC-Power-Off" geschaltet ohne den DC-Bus zu entladen.

Dies muss vom Programmierer selbst beim Erstellen von Applikationen beachtet werden.

12.3.7 MB_POWERSUPPLY_TARGET_MODE

Kurzbeschreibung Dies Aufzählung wird vom Funktionsbausteineingang "TargetMode" verwendet.

Jedes Element dieser Aufzählung entspricht einem der sechs verfügbaren Zielmodi.

Der Zielmodus entscheidet über den zu erreichenden Endzustand des Versorgers, sowie über den zurückzulegenden Weg zum Erreichen des Endzustandes.

Aufzählungselemente

Element	Beschreibung
MAINS_OFF_NO_DISCHARGE	Der Versorger soll sofort in den Zustand "Netz aus" (Mains Off) schalten
MAINS_OFF_DISCHARGE	Der Versorger soll den DC-Bus entladen und dann sofort in den Zustand "Netz aus" (Mains Off) schalten
DC_POWER_OFF_NO_DISCHARGE	Der Versorger soll - unabhängig vom momentanen "ActualMode" - in den DC-Power-Off-Zustand schalten
DC_POWER_OFF_DISCHARGE	Der Versorger soll den DC-Bus entladen und dann sofort in den Zustand "DC-Power-Off" schalten
DC_BUS_CHARGED	Der Versorger soll in den geladenen Zustand des DC-Busses schalten
OPERATION_MODE	Der Versorger soll in die Betriebsart "Operation_Mode" schalten

Tab. 12-49: Elemente der Aufzählung MB_POWERSUPPLY_TARGET_MODE



Es kann zu widersprüchlichen Eingangskombinationen zwischen "ErrorMode" und "TargetMode" kommen.

In solchen Fällen dominiert "TargetMode" unabhängig vom Wert des Eingangs "ErrorMode". Siehe nachstehendes Beispiel.

- ActualMode = _ERROR
- TargetMode = DC_POWER_OFF_NO_DISCHARGE
- ErrorMode = ERROR_TRANSITION_DISCHARGE

Wenn der Versorger den Zustand "Error" verlässt, ist es für "TargetMode" notwendig, dass der Versorger in den Zustand "DC-Power-Off" geht ohne den DC-Bus zu entladen.

Für "ErrorMode" ist es hingegen notwendig, dass der Versorger den DC-Bus entlädt bevor in den nächsten Zustand geschaltet wird.

In diesem Fall dominiert "TargetMode" und der Versorger wird in den Modus "DC-Power-Off" geschaltet ohne den DC-Bus zu entladen.

Dies muss vom Programmierer selbst beim Erstellen von Applikationen beachtet werden.

12.3.8 MB_PS_TARGET_MODE_DC_DC_ISLAND_GRID

Kurzbeschreibung Dies Aufzählung wird vom Eingang "TargetMode" des Funktionsbausteins "MB_SIIIPowerSupplyDcDcIslandGrid" verwendet.

Jedes Element dieser Aufzählung entspricht einem der verfügbaren Zielmodi.

Der Zielmodus entscheidet über den zu erreichenden Endzustand des Versorgers.

Aufzählungselemente

Element	Beschreibung
MAINS_OFF	Der Versorger soll sofort in den Zustand "Netz aus" (Mains Off) schalten
POWER_OFF	Der Versorger soll, unabhängig vom momentanen "ActualMode", in den Power-Off-Zustand schalten. DC_POWER_OFF oder AC_POWER_OFF je nach gewählter "PowerSupplyFunction"
POWER_READY	Der Versorger soll in den Ready-Zustand schalten. DC_BUS_CHARGED oder AC_POWER_READY je nach gewählter "PowerSupplyFunction"
OPERATION_MODE	Der Versorger soll in die Betriebsart "Operation_Mode" schalten

Tab. 12-50: Elemente der Aufzählung MB_PS_TARGET_MODE_DC_DC_ISLAND_GRID

12.3.9 MB_POWER_SUPPLY_FUNCTION

Kurzbeschreibung Dies Aufzählung wird vom Funktionsbausteineingang "PowerSupplyFunction" verwendet.

Jedes Element dieser Aufzählung entspricht einem möglichen Funktionspaket bzw. über die genutzte erweiterte Versorgerfunktion des Versorgers.

Je nach genutzter Versorgerfunktion meldet der Ausgang "ActualMode" andere Statuswerte.

Aufzählungselemente

Element	Beschreibung
PS_FUNCTION_DC_DC	DC/DC Stellerbetrieb, Funktionspaket DCE
PS_FUNCTION_ISLAND_GRID	Inselnetzbetrieb, Funktionspaket MSE -> "Netzfunktion".

Tab. 12-51: Elemente der Aufzählung MB_POWERSUPPLY_FUNCTION

12.4 RexrothEFC**12.4.1 MB_SIIIEfcCodeToIdn**

Kurzbeschreibung Die Funktion MB_SIIIEfcCodeToIdn rechnet einen Funktions-/Parametercode des EFC in die zugehörige Sercos-Parameter IDN um. Mit dieser IDN und den Funktionsbausteinen zum Parameter lesen/schreiben aus der Bibliothek RIL_SercosIII ist der Zugriff auf die Parameter des EFC über Sercos möglich.

Zielsystem	Bibliothek
MLC/MTX ab 14V14	RMB_SercosIII_Util
MLD mit IndraLogic 2G ab 20VRS	RMB_SercosIII_Util

Tab. 12-52: Referenztablelle

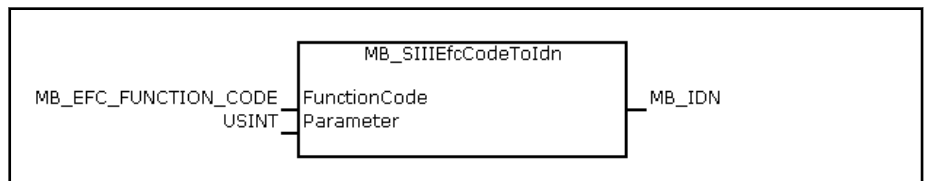
Schnittstellenbeschreibung

Abb. 12-29: Funktion MB_SIIIEfcCodeToIdn

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	FunctionCode	MB_EFC_FUNCTION_CODE	Funktionscode der Parameter des EFC z. B. H0 bei H0.10 MB_EFC_FUNCTION_CODE, Seite 496
	Parameter	USINT	Parametercode der Parameter des EFC z. B. 10 bei H0.10
Rückgabewert		MB_IDN	Sercos-IDN z.B. P-0-1098.0.10 bei H0.10

Beispiel für Datentyp UDEC (dezimal ohne Vorzeichen)

Im folgenden Beispiel wird der Parameter d0.11 "Ausgangsstrom" gelesen.

Variablendefinition

```
VAR
    fbSIIIRedRealParameter:    IL_SIIIRedRealParameter;
    rValue: REAL;
END_VAR
```

Programmcode d0.11 (P-0-1058.0.11) lesen

```
fbSIIIRedRealParameter.Execute:= TRUE;
fbSIIIRedRealParameter.SercosAdr:= 1;
fbSIIIRedRealParameter.Idn:= MB_SIIIEfcCodeToIDN(MB_EFC_FUNCTION_CODE.d0,11);

fbSIIIRedRealParameter();

IF fbSIIIRedRealParameter.Done = TRUE THEN
    rValue := fbSIIIRedRealParameter.Value;
END_IF
```

Nach erfolgreichem Lesen steht in **rValue** zum Beispiel der Wert "0,5".

12.4.2 MB_SIIIPlcDriveEfcFrequencyType01

Kurzbeschreibung Der Funktionsbaustein MB_SIIIPlcDriveEfcFrequencyType01 dient dazu einen Rexroth Frequenzumrichter EFC in der Betriebsart "Frequenzsteuerung" zu betreiben.



Der Funktionsbaustein "MB_SIIIEfcFrequencyPlcDrive" wurde mit Version 14V16 in "MB_SIIIPlcDriveEfcFrequencyType01" umbenannt und die Schnittstelle wurde optimiert.



Ein Beispielprogramm "AchsInterface" zu diesem Funktionsbaustein ist als GAT Plug-In "SercosPlcDriveInterface" verfügbar. Dieses Beispielprogramm kann auch ohne GAT verwendet werden. Siehe dazu die Plug-In Dokumentation.



Beim Rexroth Frequenzumrichter EFC werden die Parameter mit Funktionscodes bezeichnet, z.B. b0.00. Auf diese Parameter kann über das Sercos-Interface über P-Parameter zugegriffen werden.

Dokumentiert sind nur die Funktionscodes und nicht die P-Parameter.

Zum Zugriff auf die Parameter des EFC kann die Funktion [MB_SIIIEfcCodeToldn auf Seite 491](#), zusammen mit den Funktionsbausteinen zum Lesen und Schreiben von Parametern aus der Bibliothek RIL_SercosIII verwendet werden.

Zielsystem	Bibliothek
MLC/MTX ab 14V14	RMB_SercosIII_Util
MLD mit IndraLogic 2G ab 20VRS	RMB_SercosIII_Util

Tab. 12-53: Referenztablelle

Schnittstellenbeschreibung

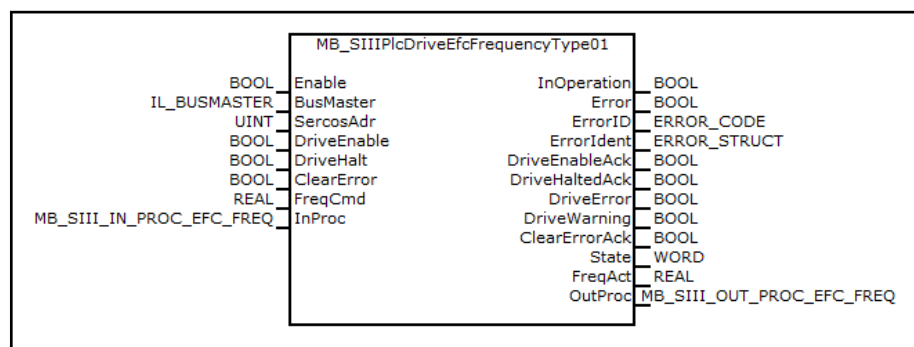


Abb. 12-30: Funktionsbaustein MB_SIIIPlcDriveEfcFrequencyType01

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (pegelgesteuert).
	Busmaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable BusMaster unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 429

E/A-Typ	Name	Datentyp	Beschreibung
	SercosAdr	UINT	Sercos III-Adresse des Slave-Teilnehmers.
	DriveEnable	BOOL	Freigabe des Antriebes (AH/AF) S-0-0134, Master-Steuerwort, Reglerfreigabe, Bit 14
	DriveHalt	BOOL	Stoppen des Antriebes (AH) S-0-0134, Master-Steuerwort, Antrieb Halt, Bit 13
	ClearError	BOOL	Löschen eines Fehlers im Antrieb durch das Kommando Fehler löschen (S-0-0099, C0500 Reset Zustandsklasse 1). Nach gelöschtem Fehler wird die angewählte Freigabe wieder aktiviert.
	FreqCmd	REAL	Frequenz-Sollwert. Einheit in [Hz]. Für Rückwärtsbetrieb negativen Wert angeben.
	InProc	MB_SI-II_IN_PROC_EFC_FREQ	Prozess-Eingangsdaten %I anschließen. MB_SIII_IN_PROC_EFC_FREQ auf Seite 496
VAR_OUT-PUT	InOperation	BOOL	Baustein arbeitet
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCTURE	Detaildiagnose Fehlerbehandlung, Seite 495
	DriveEnableAck	BOOL	Antriebsfreigabe quittiert, S-0-0135, Antriebs-Status, Bit 14/15
	DriveHaltedAck	BOOL	Antrieb Halt quittiert, S-0-0135, Antriebs-Status, Bit 4
	DriveError	BOOL	Antriebsfehler, S-0-0135, Antriebs-Status, Bit 13
	DriveWarning	BOOL	Antriebswarnung, S-0-0135, Antriebs-Status, Bit 12
	ClearErrorAck	BOOL	Bestätigung, dass das Kommando Fehler löschen ausgeführt wurde, S-0-0099, C0500 Reset Zustandsklasse 1
	State	WORD	Aktueller Status des Antriebes. Siehe unten.
	FreqAct	REAL	Aktuelle Ausgangsfrequenz. Einheit in [Hz]. Bei Rückwärtsbetrieb werden negative Werte ausgegeben.
	OutProc	MB_SI-II_OUT_PROC_EFC_FREQ	Prozess-Ausgangsdaten %Q anschließen. MB_SIII_OUT_PROC_EFC_FREQ auf Seite 496

Tab. 12-54: Schnittstelle MB_SIIIPlcDriveEfcFrequencyType01

Ausgang "State" Im Ausgang "State" ist eine Grobklassifizierung von Diagnosen binär kodiert. Die Bits sind in der RIL_FeldbusTypes.library als Globale Konstanten "FieldbusStates" enthalten.

BIT-Nr.	Name	Beschreibung
0	IL_RBS_SLAVE_NOT_REACHABLE	Slave nicht existent
1	IL_RBS_SLAVE_NOT_READY	Slave nicht bereit

BIT-Nr.	Name	Beschreibung
2	IL_RBS_SLAVE_CONFIG_ERROR	Slave-Konfigurationsfehler
3	IL_RBS_SLAVE_DIAGNOSIS	Slave-Diagnose verfügbar
4	IL_RBS_SLAVE_ERROR	Slave-Fehler verfügbar
5	IL_RBS_CON_OFF	Mindestens eine zyklische Verbindung läuft nicht

Min-/Max und Default-Werte

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
Busmaster	IL_BUSMASTER			0	Steigende Flanke an Enable
SercosAdr	UINT	0 (MLD: lokale Achse) 1 (andere Systeme)	Systemspezi- fisch	0	Kontinuierlich
DriveEnable	BOOL			FALSE	Kontinuierlich
DriveHalt	BOOL			FALSE	Kontinuierlich
ClearError	BOOL			FALSE	Kontinuierlich
FreqCmd	REAL	-400,0 (Definiert durch E0.08 und E0.09)	+400,0 (Definiert durch E0.08 und E0.09)	0.0	Kontinuierlich
InProc	MB_SI-II_IN_PROC_EFC_FREQ			0	Kontinuierlich

Tab. 12-55: Min-/Max und Default-Werte der Eingänge MB_SIIIPlcDriveEfcFrequencyType01

Funktionsbeschreibung

Der Funktionsbaustein MB_SIIIPlcDriveEfcFrequencyType01 erweitert den Funktionsbaustein MB_SIIIPlcDriveType01 und besitzt von diesem alle Ein- und Ausgänge sowie dessen Funktionalität.

Zusätzlich hat er noch Ein- und Ausgänge:

- für die Ist- und Sollwerte dieser Betriebsart:
 - Der Eingang „FreqCmd“ gibt den Frequenz-Sollwert der Achse an. Der Wert wird zyklisch über den Ausgang "OutProc.Frequency" und den Parameter P-0-1098.0.10 (H0.10) an den Antrieb übergeben. Der Wert wird begrenzt auf den Maximal Wert von P-0-1074.0.8 (E0.08) "Maximale Ausgangsfrequenz".
 - Der Ausgang "FreqAct" gibt den Frequenz-Istwert der Achse aus. Der Wert wird zyklisch über den Parameter P-0-1058.0.10 (d0.00) und den Eingang "InProc.Frequency" vom Antrieb gelesen.
- für zusätzliche spezielle Status- und Steuerworte für den EFC:
 - Der Eingang "InProc.EfcStatus" für Statusmeldungen des EFC, z. B. Verzögerung, Beschleunigung, Rückwärtsbetrieb können darüber abgefragt werden.
 - Der Ausgang "OutProc.EfcCtrl" zur Steuerung des EFC (z. B. Vorwärts-/Rückwärts-Umschaltung).

Konfiguration der zyklischen Daten im Sercos:

Um die zyklische Kommunikation des Funktionsbausteins mit dem Antrieb aufzubauen müssen Parameter auf die jeweiligen Ein- und Ausgangsvariablen kopiert werden. Die Konfiguration der zyklisch übertragenen Parameter erfolgt im Projektextplorer unterhalb der Sercosknotens. Dazu den Knoten „Drive“ unterhalb des jeweiligen Gerätes anwählen.

Hier müssen zunächst im Reiter "Allgemeine Ein- und Ausgänge" die zyklischen Parameter konfiguriert werden.

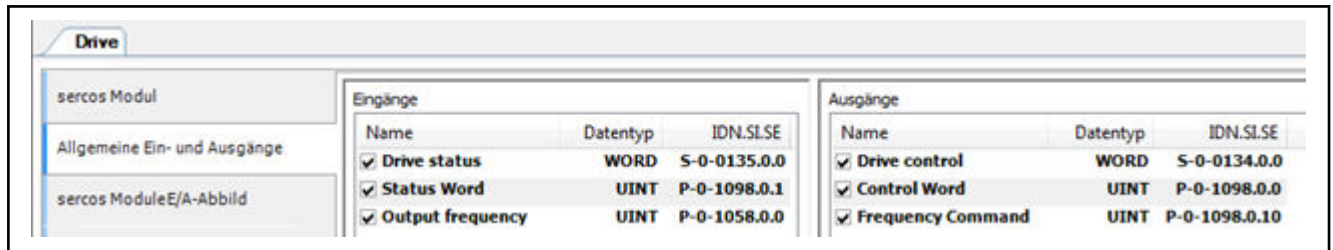


Abb. 12-31: Konfiguration der zyklischen Parameter

Anschließend werden im Reiter "E/A-Abbild" die entsprechenden Ein- und Ausgangsvariablen des Funktionsbausteins zugewiesen.

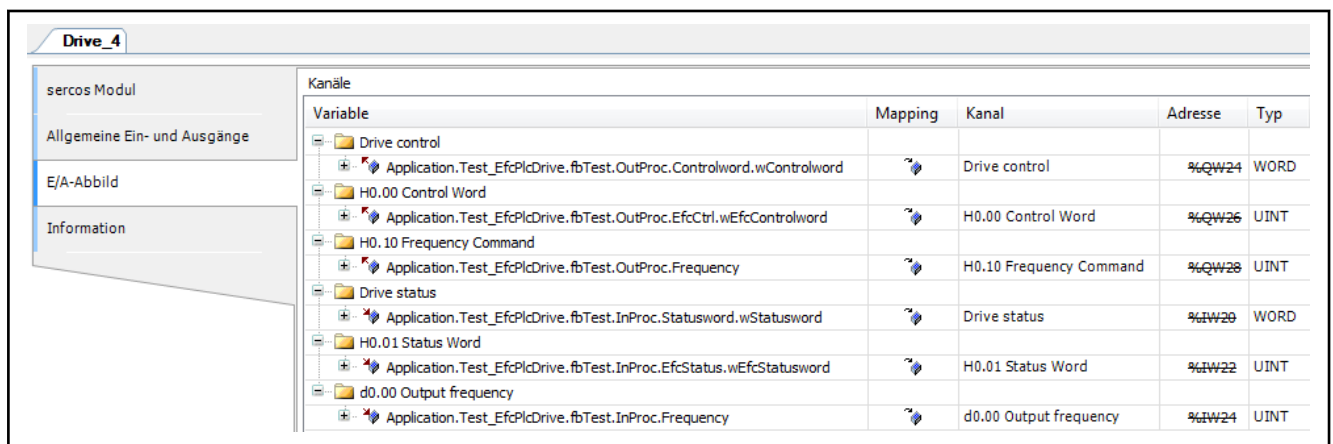


Abb. 12-32: Mappen der Ein- und Ausgangsvariablen

Parametrierung des Antriebes:

Bei Rexroth-Geräten wird die Parametrierung mit "IndraWorks Ds" empfohlen. Dort sind gerätespezifische Inbetriebnahmedialoge realisiert.

Über den Eintrag "IndraWorks Ds hier" (unterhalb des Sercos-Knotens, Geräteknoten, rechte Maustaste) kann "IndraWorks Ds" innerhalb der Engineering-Suite IndraWorks gestartet werden (Routing von der Steuerung zum Gerät erfolgt automatisch).

Der Funktionsbaustein MB_SIIPlcDriveEfcFrequencyType01 erfordert die Parametrierung der Hauptbetriebsart S-0-0032 auf "Frequenzsteuerung" (Wert 2#1000.0000.0000.0000). Die weitere Parametrierung für die Betriebsart "Frequenzsteuerung" ist mit z.B. IndraWorks Ds oder ConverterWorks auszuführen.

Eine Grundparametrierung des Rexroth Frequenzumrichter EFC (Grenzwerte usw.) ist auf jeden Fall erforderlich.

Fehlerbehandlung

Zusätzlich zu den Fehlern der intern genutzten Funktionsbausteine MB_SIIPlcDriveType01 und IL_SIIISvcRead sind folgende Fehler des Bausteines in

der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#00000006	State	Fehler in der Zustandsmaschine
INPUT_RANGE_ERROR	16#00002985	Wert des Attributes	Attribut des Parameters P-0-1098.0.10 oder P-0-1058.0.0 ist nicht 2 Byte dezimal ohne Vorzeichen.

Tab. 12-56: Fehlercodes FB MB_SIIIPlcDriveEfcFrequencyType01

12.4.3 MB_EFC_FUNCTION_CODE

Kurzbeschreibung Der Aufzählungstyp MB_EFC_FUNCTION_CODE beschreibt die Funktionscodes des EFC.

Name	Wert	Beschreibung
b0..b9	16#00..09	Funktionscode der Systemparameter b0 bis b9
d0..d9	16#10..19	Funktionscode der Überwachungsparameter d0 bis d9
C0..C9	16#20..29	Funktionscode der Leistungsparameter C0 bis C9
E0..E9	16#30..39	Funktionscode der Funktions-Regelungsparameter E0 bis E9
U0..U9	16#40..49	Funktionscode der Bedienfeld-Parameter U0 bis U9
F0..F9	16#50..59	Funktionscode der APP-Parameter F0 bis F9
H0..H9	16#60..69	Funktionscode der Option Karten-Parameter H0 bis H9

Tab. 12-57: Elemente des Aufzählungstyps MB_EFC_FUNCTION_CODE

12.4.4 MB_SIII_IN_PROC_EFC_FREQ

Kurzbeschreibung Die Datenstruktur MB_SIII_IN_PROC_EFC_FREQ enthält die Prozess-Eingangsdaten des Funktionsbausteins [MB_SIIIPlcDriveEfcFrequencyType01](#) auf Seite 492.

Strukturelement	Typ	Default-Wert	Beschreibung
Statusword	MB_SIII_STATUSWORD		Eingangsvariable %IW des Statuswortes S-0-0135 am Element "wStatusword" anschließen. MB_SIII_STATUSWORD auf Seite 459
EfcStatus	MB_SIII_EFC_STATUSWORD		Eingangsvariable %IW des Statuswortes P-0-1098.0.1 am Element "wEfcStatusword" anschließen. MB_SIII_EFC_STATUSWORD auf Seite 497
Frequency	UINT	16#0	Eingangsvariable %IW der aktuellen Ausgangsfrequenz P-0-1058.0.0 (d0.00) anschließen.

Tab. 12-58: Elemente der Struktur MB_SIII_IN_PROC_EFC_FREQ

12.4.5 MB_SIII_OUT_PROC_EFC_FREQ

Kurzbeschreibung Die Datenstruktur MB_SIII_OUT_PROC_EFC_FREQ enthält die Prozess-Ausgangsdaten des Funktionsbausteins [MB_SIIIPlcDriveEfcFrequencyType01](#) auf Seite 492.

Strukturelement	Typ	Default-Wert	Beschreibung
Controlword	MB_SIII_CONTROLWORD		Ausgangsvariable %QW des Steuerwortes S-0-0134 am Element "wControlword" anschließen. MB_SIII_CONTROLWORD auf Seite 458
EfcCtrl	MB_SIII_EFC_CONTROLWORD		Ausgangsvariable %QW des Steuerwortes P-0-1098 am Element "wControlword" anschließen. MB_SIII_EFC_CONTROLWORD auf Seite 497
Frequency	UINT	16#0	Ausgangsvariable %QW des Frequenzbefehl P-0-1098.0.10 (H0.10) anschließen.

Tab. 12-59: Elemente der Struktur MB_SIII_OUT_PROC_EFC_FREQ

12.4.6 MB_SIII_EFC_CONTROLWORD

Kurzbeschreibung Die Datenstruktur MB_SIII_EFC_CONTROLWORD entspricht dem Parameter P-0-1098, Register H0.00 "Register für Kommunikationssteuerung" des EFC. Die Bits können im Anwendungsprogramm nur gelesen werden.

Die Variable "wEfcControlword" liegt auf dem gleichen Speicher wie die Bits und dient dem Mapping im E/A-Abbild des EFC.

Strukturelement	Typ	Min-Wert	Max-Wert	Default-Wert	Beschreibung
wEfcControlword	WORD			0	Komplettes Kontrollwort
RunCommand	BIT			FALSE	1= Run-Befehl aktiv
Jogging	BIT			FALSE	1= Tippbetrieb aktiv (Schrittrichtung bestimmt durch Bit 2)
Reverse	BIT			FALSE	1= Rückwärts, 0 = Vorwärts
Stop	BIT			FALSE	1= Stopp gemäß Parametereinstellung
EStop	BIT			FALSE	1= Nothalt aktiv
FaultReset	BIT			FALSE	1= Fehler Rücksetzen aktiv
StopAccDec	BIT			FALSE	1= Stopp Beschl./Verz. aktiv (stoppt den internen Beschleunigungs-/Verzögerungsrampen-generator)
ControlWordActive	BIT			FALSE	1= Steuerwort aktiv

Tab. 12-60: Elemente der Struktur MB_SIII_EFC_CONTROLWORD

12.4.7 MB_SIII_EFC_STATUSWORD

Kurzbeschreibung Die Datenstruktur MB_SIII_EFC_STATUSWORD entspricht dem Parameter P-0-1098.0.1, Register H0.01 "Register für Rückmeldung Kommunikationsstatus" des EFC. Die Daten können im Anwendungsprogramm nur gelesen werden.

Die Variable "wEfcStatusword" liegt auf dem gleichen Speicher wie die Bits und dient dem Mapping im E/A-Abbild des EFC.

Strukturelement	Typ	Min-Wert	Max-Wert	Default-Wert	Beschreibung
wEfcStatusword	WORD			0	Komplettes Statuswort
Reverse	BIT			FALSE	1= Rückwärts, 0= Vorwärts
Running	BIT			FALSE	1= Start, 0= Stopp
Jogging	BIT			FALSE	1= Tippbetrieb
Accelerating	BIT			FALSE	1= Beschleunigung
Decelerating	BIT			FALSE	1= Verzögerung
Voltage	BIT			FALSE	1= Begrenzung Überspannung, 0= Normal
Current	BIT			FALSE	1= Begrenzung Überstrom, 0= Normal
Error	BIT			FALSE	1= Fehler
ErrorCode	BYTE			16#00	Fehlercode (entspricht E9.05)

Tab. 12-61: Elemente der Struktur MB_SIII_EFC_STATUSWORD

12.5 RexrothHydraulicDrives

12.5.1 MB_SIIIPlcDriveFlowControl

Kurzbeschreibung Der Funktionsbaustein MB_SIIIPlcDriveFlowControl erweitert den Funktionsbaustein MB_SIIIPlcDriveType01 und realisiert die Betriebsart **Volumenstromregelung**.



Details zu den möglichen Betriebsarten sind zu finden unter:

Rexroth HydraulicDrive HDx-20 Funktionen, Anwendungsbeschreibung, DOK-HYDRV*-HD*-20VRS**-APxx-DE-P, RD-30338-FK, Kapitel 6, Betriebsarten.

Es wird die Betriebsart im Gerät ein- bzw. ausgeschaltet sowie ein Volumenstrom als Sollwert mit eingestellter Wichtung vorgegeben. Der Volumenstromwert, der effektive Ventilsollwert, sowie Statusmeldungen werden angezeigt.

Zielsystem	Bibliothek
MLC/MTX ab 14V16	RMB_SercosIII_Util
MLD mit IndraLogic 2G ab HDx 20VRS bzw. MPx 20VRS	RMB_SercosIII_Util

Tab. 12-62: Referenztablelle

Schnittstellenbeschreibung

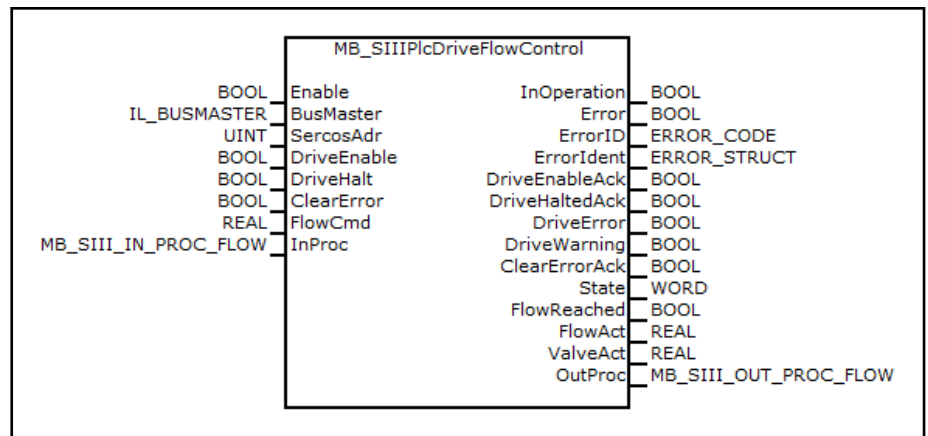


Abb. 12-33: Funktionsbaustein MB_SIIIPlcDriveFlowControl

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (pegelgesteuert).
	Busmaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable BusMaster unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 429
	SercosAdr	UINT	Sercos III-Adresse des Slave-Teilnehmers.
	DriveEnable	BOOL	Freigabe des Antriebes (AH/AF) S-0-0134, Master-Steuernwort, Reglerfreigabe, Bit 14
	DriveHalt	BOOL	Stoppen des Antriebes (AH) S-0-0134, Master-Steuernwort, Antrieb Halt, Bit 13
	ClearError	BOOL	Löschen eines Fehlers im Antrieb durch das Kommando Fehler löschen (S-0-0099, C0500 Reset Zustandsklasse 1). Nach gelöschtem Fehler wird die angewählte Freigabe wieder aktiviert.
	FlowCmd	REAL	S-0-0840, Volumenstrom-Sollwert, Wichtung lt. Attribut
	InProc	MB_SIII_IN_PROC_FLOW	Prozess-Eingangsdaten %I anschließen. MB_SIII_IN_PROC_FLOW, Seite 543
VAR_OUTPUT	InOperation	BOOL	Baustein arbeitet
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCT	Detaildiagnose Fehlerbehandlung, Seite 502
	DriveEnable-Ack	BOOL	Antriebsfreigabe quittiert.
	DriveHaltedAck	BOOL	Antrieb Halt quittiert. S-0-0135, Antriebs-Status, Bit 4

E/A-Typ	Name	Datentyp	Beschreibung
	DriveError	BOOL	Antriebsfehler. S-0-0135, Antriebs-Status, Bit 13
	DriveWarning	BOOL	Antriebswarnung. S-0-0135, Antriebs-Status, Bit 12
	ClearErrorAck	BOOL	Bestätigung, dass das Kommando Fehler löschen ausgeführt wurde. S-0-0099, C0500 Reset Zustandsklasse 1
	State	WORD	Device Status von IL_RemoteSlaveState
	FlowReached	BOOL	S-0-0849, Volumenstrom-Statuswort , Bit 1
	FlowAct	REAL	S-0-0842, Volumenstrom-Istwert , Wichtung lt. Attribut
	ValveAct	REAL	P-0-2914, Wirksamer Ventil-Sollwert, Wichtung lt. Attribut
	OutProc	MB_SIII_OUT_PROC_FLOW	Prozess-Ausgangsdaten %Q anschließen. MB_SIII_OUT_PROC_FLOW, Seite 543

Tab. 12-63: Schnittstelle MB_SIIIPlcDriveFlowControl

Ausgang "State" Im Ausgang "State" ist eine Grobklassifizierung von Diagnosen binär kodiert. Die Bits sind in der RIL_FeldbusTypes.library als Globale Konstanten "FieldbusStates" enthalten.

BIT-Nr.	Name	Beschreibung
0	IL_RBS_SLAVE_NOT_REACHABLE	Slave nicht existent
1	IL_RBS_SLAVE_NOT_READY	Slave nicht bereit
2	IL_RBS_SLAVE_CONFIG_ERROR	Slave-Konfigurationsfehler
3	IL_RBS_SLAVE_DIAGNOSIS	Slave-Diagnose verfügbar
4	IL_RBS_SLAVE_ERROR	Slave-Fehler verfügbar
5	IL_RBS_CON_OFF	Mindestens eine zyklische Verbindung läuft nicht

Min-/Max und Default-Werte

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER			FALSE	Kontinuierlich
SercosAdr	UINT	1	255	1	Steigende Flanke an Enable
DriveEnable	BOOL			FALSE	Kontinuierlich
DriveHalt	BOOL			FALSE	Kontinuierlich
ClearError	BOOL			FALSE	Kontinuierlich
FlowCmd	REAL	n.def.	n.def.	0.0	Kontinuierlich
InProc	MB_SIII_IN_PROC_FLOW				Kontinuierlich

Funktionsbeschreibung

Der Funktionsbaustein schaltet den Antrieb in die Betriebsart "Volumenstromregelung" (Betriebsarten-Code: 0x0052). Es wird ein Volumenstromsollwert am Eingang "FlowCmd" vorgegeben. Der Istwert des Volumenstromes wird am Ausgang "FlowAct" angezeigt. Die Umrechnung dieser Gleitkommagrößen (REAL) in die zyklischen Festpunktwerte des Gerätes (DINT) erfolgt intern im Baustein über die Strukturen "InProc" und "OutProc".

Das Erreichen des Volumenstromes wird am Ausgang „FlowReached“ angezeigt (S-0-0849, Volumenstrom-Statuswort, Bit 1).

Der wirksame Ventilsollwert (P-0-2914, Wirksamer Ventil-Sollwert) wird zu Diagnosezwecken am Ausgang "ValveAct" angezeigt. Zusätzlich wird eine (vorhandene) Warnung bzw. Fehler am Baustein angezeigt.

Konfiguration der zyklischen Daten für Sercos:

Um die zyklische Kommunikation des Funktionsbausteins mit dem Antrieb aufzubauen müssen Parameter auf die jeweiligen Ein- und Ausgangsvariablen kopiert werden. Die Konfiguration der zyklisch übertragenen Parameter erfolgt im Projektextplorer unterhalb der Sercos-Knotens. Dazu den Knoten "Drive" unterhalb des jeweiligen Gerätes anwählen.

Folgende Parameter müssen zyklisch konfiguriert werden:

Eingänge:

- S-0-0135, Antriebs-Status
- S-0-0144, Signal-Statuswort
- S-0-0842, Volumenstrom-Istwert
- P-0-2914, Wirksamer Ventil-Sollwert

Ausgänge:

- S-0-0134, Master-Steuerwort
- S-0-0840, Volumenstrom-Sollwert

Es müssen zunächst im Reiter „Allgemeine Ein- und Ausgänge“ die zyklischen Parameter konfiguriert werden.

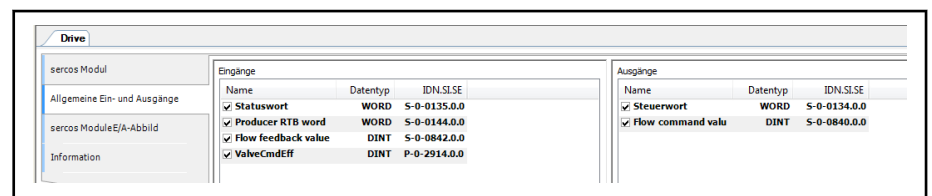


Abb. 12-34: Konfiguration der zyklischen Parameter

Anschließend werden im Reiter "E/A-Abbild" die entsprechenden Ein- und Ausgangsvariablen mit Instanzvariablen des SPS-Programmes verknüpft. Im unteren Bild ist dies eine globale Struktur HPC_DevMapping_gb mit den entsprechenden Strukturelementen. Diese wird dann im SPS-Programm auf die "InProc"- und "OutProc"-Strukturen des Funktionsbausteins kopiert.

Hier können aber auch direkt die "InProc"- und "OutProc"-Strukturen des Funktionsbausteins eingetragen werden.

Variable	Mapping	Kanal	Adresse	Typ	St
Steuernwort					
Application.HPC_DevMapping_gb.MasterControlWord		Steuernwort	%QW12	WORD	
Flow command value					
Application.HPC_DevMapping_gb.FlowCmdValue		Flow command value	%QD16	DINT	
Statuswort					
Application.HPC_DevMapping_gb.HydrStatusWord		Statuswort	%SW12	WORD	
Producer RTB word container					
Application.HPC_DevMapping_gb.ProducerRTBWord		Producer RTB word container	%SW14	WORD	
Flow feedback value					
Application.HPC_DevMapping_gb.FlowActValue		Flow feedback value	%ID16	DINT	
ValveCmdEff					
Application.HPC_DevMapping_gb.ValveCmdEff		ValveCmdEff	%ID20	DINT	

Abb. 12-35: Mappen der Ein- und Ausgangsvariablen

Parametrierung des Gerätes:

Bei Rexroth-Geräten wird die Parametrierung mit "IndraWorks Ds" empfohlen. Dort sind gerätespezifische Inbetriebnahmedialoge realisiert.

Über den Eintrag "IndraWorks Ds hier" (unterhalb des Sercos-Knoten, Geräteknoten, rechte Maustaste) kann "IndraWorks Ds" innerhalb der Engineering-Suite IndraWorks gestartet werden (Routing von der Steuerung zum Gerät erfolgt automatisch).

folgende Parameter müssen eingestellt werden:

- Im Parameter S-0-0032, Hauptbetriebsart muss „Volumenstromregelung“ (16#0052) eingetragen werden. Die Hauptbetriebsart wird ausgewählt.
- Im Parameter S-0-0144, Signal-Statuswort muss der Parameter S-0-0849 Volumenstrom-Statuswort, mit der Bitnummer 1 ("Volumenstromsoll = Volumenstromist") als Bit 1 eingetragen sein.

Status	Quellparameter	Bitnummer
Bit 0:	S-0-0000: <kein Signal>	0
Bit 1:	S-0-0849: Volumenstrom-Statuswort	1
Bit 2:	S-0-0000: <kein Signal>	0
Bit 3:	S-0-0000: <kein Signal>	0
Bit 4:	S-0-0000: <kein Signal>	0
Bit 5:	S-0-0000: <kein Signal>	0

Abb. 12-36: Parametrierung Signalstatuswort S-0-0144

- Parametrierung des Gerätes gemäß Applikation (Messsystem, Wichtung, Überwachung/Fehlerreaktion...)



Insbesondere Überprüfen, dass die vom Funktionsbaustein vorgegebenen Sollwerte nicht durch eine gerätespezifische Parametrierung überschrieben werden,

z.B. durch Parametrierung der Analogen Eingänge des Gerätes mit internen Sollwerten.

Fehlerbehandlung

Zusätzlich zu den Fehlern der intern genutzten Funktionsbausteine MB_SII-IPIcDriveType01 und IL_SIIISvcRead sind folgende Fehler des Bausteines in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#00000006	State	Fehler in der Zustandsmaschine
INPUT_RANGE_ERROR	16#000029B0	Wert des Attributes	Attribut der Parameter S-0-0840 oder P-0-2914 ist nicht 4 Byte dezimal mit Vorzeichen

Tab. 12-64: Fehlercodes FB MB_SIIIPlcDriveFlowControl

12.5.2 MB_SIIIPlcDrivePressureFlowControl

Kurzbeschreibung Der Funktionsbaustein MB_SIIIPlcDrivePressureFlowControl erweitert den Funktionsbaustein MB_SIIIPlcDriveType01 und realisiert die Betriebsart **Druck- und Volumenstromregelung**.



Details zu den möglichen Betriebsarten sind zu finden unter:

Rexroth HydraulicDrive HDx-20 Funktionen, Anwendungsbeschreibung, DOK-HYDRV*-HD*-20VRS**-APxx-DE-P, RD-30338-FK, Kapitel 6, Betriebsarten.

Es wird die Betriebsart im Gerät ein- bzw. ausgeschaltet sowie ein Volumenstrom und ein Druckwert als Sollwert mit passender Wichtung vorgegeben. Der Druckistwert, Volumenstromistwert und der effektive Ventilsollwert, sowie Statusmeldungen werden angezeigt. Die aktive Regelungsart wird am Ausgang "ControlMode" angezeigt.

Zielsystem	Bibliothek
MLC/MTX ab 14V16	RMB_SercosIII_Util
MLD mit IndraLogic 2G ab HDx 20VRS bzw. MPx 20VRS	RMB_SercosIII_Util

Tab. 12-65: Referenztablelle

Schnittstellenbeschreibung

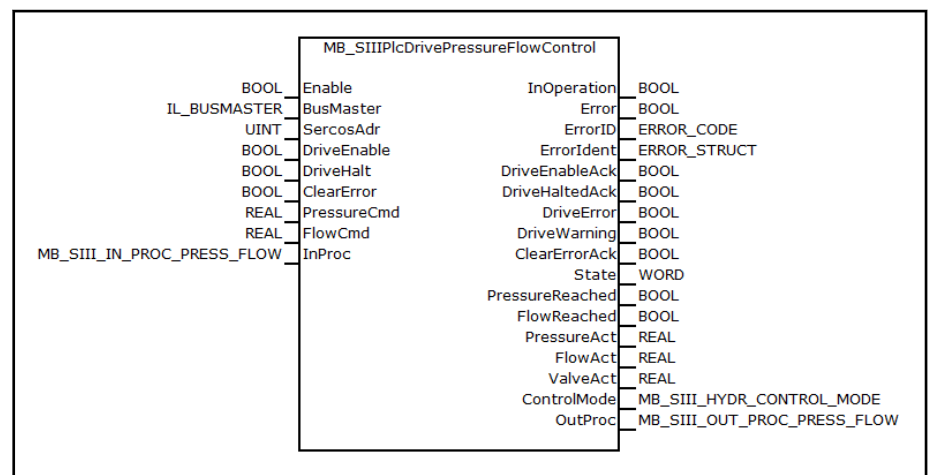


Abb. 12-37: Funktionsbaustein MB_SIIIPlcDrivePressureFlowControl

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (pegelgesteuert).
	Busmaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable BusMaster unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 429

E/A-Typ	Name	Datentyp	Beschreibung
	SercosAdr	UINT	Sercos III-Adresse des Slave-Teilnehmers.
	DriveEnable	BOOL	Freigabe des Antriebes (AH/AF) S-0-0134, Master-Steuerwort, Reglerfreigabe, Bit 14
	DriveHalt	BOOL	Stoppen des Antriebes (AH) S-0-0134, Master-Steuerwort, Antrieb Halt, Bit 13
	ClearError	BOOL	Löschen eines Fehlers im Antrieb durch das Kommando Fehler löschen (S-0-0099, C0500 Reset Zustandsklasse 1). Nach gelöschtem Fehler wird die angewählte Freigabe wieder aktiviert.
	PressureCmd	REAL	S-0-0800, Druck-Sollwert , Wichtung lt. Attribut
	FlowCmd	REAL	S-0-0840, Volumenstrom-Sollwert , Wichtung lt. Attribut
	InProc	MB_SI-II_IN_PROC_PRESS-URE_FLOW	Prozess-Eingangsdaten %I anschließen. MB_SIII_IN_PROC_PRESSURE_FLOW auf Seite 543
VAR_OUT-PUT	InOperation	BOOL	Baustein arbeitet
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCTURE	Detailldiagnose Fehlerbehandlung, Seite 507
	DriveEnableAck	BOOL	Antriebsfreigabe quittiert, S-0-0135, Antriebs-Status, Bit 14/15
	DriveHaltedAck	BOOL	Antrieb Halt quittiert, S-0-0135, Antriebs-Status, Bit 4
	DriveError	BOOL	Antriebsfehler, S-0-0135, Antriebs-Status, Bit 13
	DriveWarning	BOOL	Antriebswarnung, S-0-0135, Antriebs-Status, Bit 12
	ClearErrorAck	BOOL	Bestätigung, dass das Kommando Fehler löschen ausgeführt wurde, S-0-0099, C0500 Reset Zustandsklasse 1
	State	WORD	Aktueller Status des Antriebes. Siehe unten.
	PressureReached	BOOL	S-0-0830, Druck-Statuswort , Bit 1
	FlowReached	BOOL	S-0-0849, Volumenstrom-Statuswort , Bit 1
	PressureAct	REAL	S-0-0809, Druck-Istwert , Wichtung lt. Attribut
	FlowAct	REAL	S-0-0842, Volumenstrom-Istwert , Wichtung lt. Attribut
	ValveAct	REAL	P-0-2914, Wirksamer Ventil-Sollwert, Wichtung lt. Attribut

E/A-Typ	Name	Datentyp	Beschreibung
	ControlMode	MB_SI-II_HYDR_CONTROL_MODE	P-0-2951, Pumpenregler Statuswort, bzw. P-0-2801, p/Q-Regler Statuswort, bit 0 (1 Druckregler aktiv, 0 Volumenstromregler aktiv) MB_SIII_HYDR_CONTROL_MODE, Seite 542
	OutProc	MB_SI-II_OUT_PROC_PRESSURE_FLOW	Prozess-Ausgangsdaten %Q anschließen. MB_SIII_OUT_PROC_PRESSURE_FLOW, Seite 544

Tab. 12-66: Schnittstelle MB_SIIIPlcDrivePressureFlowControl

Ausgang "State"

Im Ausgang "State" ist eine Grobklassifizierung von Diagnosen binär kodiert. Die Bits sind in der RIL_FeldbusTypes.library als Globale Konstanten "FieldbusStates" enthalten.

BIT-Nr.	Name	Beschreibung
0	IL_RBS_SLAVE_NOT_REACHABLE	Slave nicht existent
1	IL_RBS_SLAVE_NOT_READY	Slave nicht bereit
2	IL_RBS_SLAVE_CONFIG_ERROR	Slave-Konfigurationsfehler
3	IL_RBS_SLAVE_DIAGNOSIS	Slave-Diagnose verfügbar
4	IL_RBS_SLAVE_ERROR	Slave-Fehler verfügbar
5	IL_RBS_CON_OFF	Mindestens eine zyklische Verbindung läuft nicht

Min-/Max und Default-Werte

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER			FALSE	Kontinuierlich
SercosAdr	UINT	1	255	1	Steigende Flanke an Enable
DriveEnable	BOOL			FALSE	Kontinuierlich
DriveHalt	BOOL			FALSE	Kontinuierlich
ClearError	BOOL			FALSE	Kontinuierlich
PressureCmd	REAL	n.def.	n.def.	0.0	Kontinuierlich
FlowCmd	REAL	n.def.	n.def.	0.0	Kontinuierlich
InProc	MB_SIII_IN_PROC_PRESSURE_FLOW				Kontinuierlich

Funktionsbeschreibung

Der Funktionsbaustein schaltet den Antrieb in die Betriebsart "Druckregelung / Volumenstromregelung" (Betriebsarten-Code: 16#0055). Es wird ein Drucksollwert am Eingang "PressureCmd" und Volumenstromsollwert am Eingang "FlowCmd" vorgegeben.

Der Istwert des Druckes wird am Ausgang "PressureAct" und der Istwert des Volumenstromes wird am Ausgang "FlowAct" angezeigt.

Die Umrechnung dieser Gleitkommagrößen (REAL) in die zyklischen Festpunktwerte des Gerätes (DINT) erfolgt intern im Baustein über die Strukturen "InProc" und "OutProc".

Das Erreichen des Druckistwertes wird am Ausgang "PressureReached" (S-0-0830, Druck-Statuswort, Bit 1) und das Erreichen des Volumenstromes wird am Ausgang "FlowReached" angezeigt (S-0-0849, Volumenstrom-Statuswort, Bit 1).

Der wirksame Ventilsollwert (P-0-2914, Wirksamer Ventil-Sollwert) wird zu Diagnosezwecken am Ausgang "ValveAct" angezeigt.

Die aktive Regelungsart wird am Ausgang "ControlMode" angezeigt (Druckregelung oder Volumenstromregelung).

Zusätzlich wird eine (vorhandene) Warnung bzw. Fehler am Baustein angezeigt.

Konfiguration der zyklischen Daten für Sercos

Um die zyklische Kommunikation des Funktionsbausteins mit dem Antrieb aufzubauen müssen Parameter auf die jeweiligen Ein- und Ausgangsvariablen kopiert werden. Die Konfiguration der zyklisch übertragenen Parameter erfolgt im Projektextplorer unterhalb der Sercos-Knotens. Dazu den Knoten "Drive" unterhalb des jeweiligen Gerätes anwählen.

Folgende Parameter müssen zyklisch konfiguriert werden:

Eingänge:

- S-0-0135, Antriebs-Status
- S-0-0144, Signal-Statuswort
- S-0-0842, Volumenstrom-Istwert
- S-0-0809, Druck-Istwert
- P-0-2914, Wirksamer Ventil-Sollwert

Ausgänge:

- S-0-0134, Master-Steuerwort
- S-0-0840, Volumenstrom-Sollwert
- S-0-0800, Druck-Sollwert

Es müssen zunächst im Reiter "Allgemeine Ein- und Ausgänge" die zyklischen Parameter konfiguriert werden.

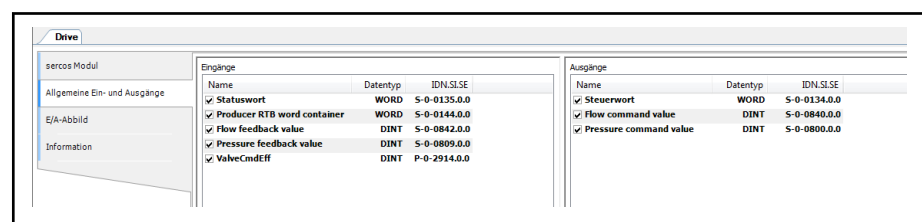


Abb. 12-38: Konfiguration der zyklischen Parameter

Anschließend werden im Reiter "E/A-Abbild" die entsprechenden Ein- und Ausgangsvariablen mit Instanzvariablen des SPS-Programmes verknüpft. Im unteren Bild ist dies eine globale Struktur HPC_DevMapping_gb mit den entsprechenden Strukturelementen. Diese wird dann im SPS-Programm auf die "InProc"- und "OutProc"-Strukturen des Funktionsbausteins kopiert.

Hier können aber auch direkt die "InProc"- und "OutProc"-Strukturen des Funktionsbausteins eingetragen werden.

Variable	Mapping	Kanal	Adresse	Typ	Standardwert	Einheit	Beschreib
Steuervort		Steuervort	%QW32	WORD			
Application.HPC_DevMapping_gb.MasterControlWord							
Flow command value		Flow command value	%QD16	DINT			
Application.HPC_DevMapping_gb.FlowCmdValue							
Pressure command value		Pressure command value	%QD20	DINT			
Application.HPC_DevMapping_gb.PressureCmdValue							
Statuswort		Statuswort	%DWH2	WORD			
Application.HPC_DevMapping_gb.HydrStatusWord							
Producer RTB word container		Producer RTB word container	%DWH4	WORD			
Application.HPC_DevMapping_gb.ProducerRTBWord							
Flow feedback value		Flow feedback value	%DDB16	DINT			
Application.HPC_DevMapping_gb.FlowActValue							
Pressure feedback value		Pressure feedback value	%DDB20	DINT			
Application.HPC_DevMapping_gb.PressureActValue							
ValveCmdEff		ValveCmdEff	%DDB24	DINT			
Application.HPC_DevMapping_gb.ValveCmdEff							

Abb. 12-39: Mappen der Ein- und Ausgangsvariablen

Parametrierung des Gerätes:

Bei Rexroth-Geräten wird die Parametrierung mit "IndraWorks Ds" empfohlen. Dort sind gerätespezifische Inbetriebnahmedialoge realisiert.

Über den Eintrag "IndraWorks Ds hier" (unterhalb des Sercos-Knoten, Geräteknoten, rechte Maustaste) kann "IndraWorks Ds" innerhalb der Engineering-Suite IndraWorks gestartet werden (Routing von der Steuerung zum Gerät erfolgt automatisch).

folgende Parameter müssen eingestellt werden:

- Im Parameter S-0-0032, Hauptbetriebsart muss "Druck- / Volumenstromregelung" (16#0055) eingetragen werden. Die Hauptbetriebsart wird angewählt.
- Parameter und Bits einstellen S-0-0144, Signal-Statuswort
 - Bit 0: S-0-0830, Druck-Statuswort, Bit 1
 - Bit 1: S-0-0849, Volumenstrom-Statuswort, Bit 1
 - Bit 2: P-0-2951, Pumpenregler Statuswort, Bit 0 für Funktionspaket Pumpenregelung

Status	Quellparameter	Bitnummer
Bit 0: ☐ S-0-0830: Druck-Statuswort	S-0-0830: Druck-Statuswort	1
Bit 1: ☐ S-0-0849: Volumenstrom-Statuswort	S-0-0849: Volumenstrom-Statuswort	1
Bit 2: ☐ P-0-2951: Pumpenregler Statuswort	P-0-2951: Pumpenregler Statuswort	0
Bit 3: ☐ S-0-0000: <kein Signal>	S-0-0000: <kein Signal>	0
Bit 4: ☐ S-0-0000: <kein Signal>	S-0-0000: <kein Signal>	0
Bit 5: ☐ S-0-0000: <kein Signal>	S-0-0000: <kein Signal>	0
Bit 6: ☐ S-0-0000: <kein Signal>	S-0-0000: <kein Signal>	0

Abb. 12-40: Parametrierung Signalstatuswort S-0-0144

- (Bit 2: P-0-2801, p/Q-Regler Statuswort, Bit 0 für Funktionspaket P/Q)
 - ...
- Parametrierung des Gerätes gemäß Applikation (Messsystem, Wichtung, Überwachung/Fehlerreaktion...)



Insbesondere Überprüfen, dass die vom Funktionsbaustein vorgegebenen Sollwerte nicht durch eine gerätespezifische Parametrierung überschrieben werden,

z.B. durch Parametrierung der Analogen Eingänge des Gerätes mit internen Sollwerten.

Fehlerbehandlung

Zusätzlich zu den Fehlern der intern genutzten Funktionsbausteine MB_SII-IPIcDriveType01 und IL_SIIISvcRead sind folgende Fehler des Bausteines in

der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#00000006	State	Fehler in der Zustandsmaschine
INPUT_RANGE_ERROR	16#000029B1	Wert des Attributes	Attribut der Parameter S-0-0800 oder S-0-0840 oder P-0-2914 ist nicht 4 Byte dezimal mit Vorzeichen

Tab. 12-67: Fehlercodes FB MB_SIIIPlcDrivePressureFlowControl

12.5.3 MB_SIIIPlcDrivePressureSwivelAngleControl

Kurzbeschreibung

Der Funktionsbaustein MB_SIIIPlcDrivePressureSwivelAngleControl erweitert den Funktionsbaustein MB_SIIIPlcDriveType01 und realisiert die Betriebsart **Druck- und Schwenkwinkelregelung**.



Details zu den möglichen Betriebsarten sind zu finden unter:
Rexroth HydraulicDrive HDx-20 Funktionen, Anwendungsbeschreibung, DOK-HYDRV*-HD*-20VRS**-APxx-DE-P, RD-30338-FK, Kapitel 6, Betriebsarten.

Es wird die Betriebsart im Gerät ein- bzw. ausgeschaltet sowie ein Druckwert und Schwenkwinkel als Sollwert mit passender Wichtung vorgegeben. Der Druckistwert, der Schwenkwinkelistwert und der effektive Ventilsollwert, sowie Statusmeldungen werden angezeigt.

Die aktive Regelungsart wird am Ausgang "ControlMode"angezeigt.

Desweiteren wird beim Einschalten des Bausteins die Master-Funktionalität im Parameter P-0-2950, Pumpenregler Steuerwortüberprüft und wenn nicht vorhanden eingestellt.

Zielsystem	Bibliothek
MLC/MTX/ ab 14V16	RMB_SercosIII_Util
MLD mit IndraLogic 2G ab HDx 20VRS bzw. MPx 20VRS	RMB_SercosIII_Util

Tab. 12-68: Referenztabelle

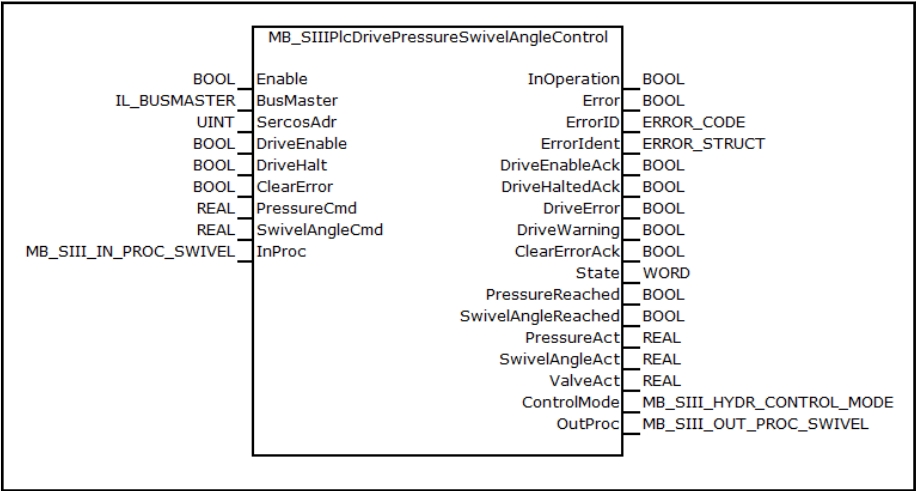


Abb. 12-41: Funktionsbaustein MB_SIIIPlcDrivePressureSwivelAngleControl

Schnittstellenbeschreibung

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (pegelgesteuert).
	Busmaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable BusMaster unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 429
	SercosAdr	UINT	Sercos III-Adresse des Slave-Teilnehmers.
	DriveEnable	BOOL	Freigabe des Antriebes (AH/AF) S-0-0134, Master-Steuerwort, Reglerfreigabe, Bit 14
	DriveHalt	BOOL	Stoppen des Antriebes (AH) S-0-0134, Master-Steuerwort, Antrieb Halt, Bit 13
	ClearError	BOOL	Löschen eines Fehlers im Antrieb durch das Kommando Fehler löschen (S-0-0099, C0500 Reset Zustandsklasse 1). Nach gelöschtem Fehler wird die angewählte Freigabe wieder aktiviert.
	PressureCmd	REAL	S-0-0800, Druck-Sollwert , Wichtung lt. Attribut
	SwivelAngleCmd	REAL	S-0-0880, Schwenkwinkel-Sollwert , Wichtung fix
	InProc	MB_SI-II_IN_PROC_SWIVEL	Prozess-Eingangsdaten %I anschließen. MB_SIII_IN_PROC_SWIVEL auf Seite 545
VAR_OUTPUT	InOperation	BOOL	Baustein arbeitet
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCTURE	Detaildiagnose Fehlerbehandlung, Seite 513
	DriveEnableAck	BOOL	Antriebsfreigabe quittiert.
	DriveHaltedAck	BOOL	Antrieb Halt quittiert. S-0-0135, Antriebs-Status, Bit 4
	DriveError	BOOL	Antriebsfehler. S-0-0135, Antriebs-Status, Bit 13
	DriveWarning	BOOL	Antriebswarnung. S-0-0135, Antriebs-Status, Bit 12
	ClearErrorAck	BOOL	Bestätigung, dass das Kommando Fehler löschen ausgeführt wurde. S-0-0099, C0500 Reset Zustandsklasse 1
	State	WORD	Aktueller Status des Antriebes. Siehe unten.
	PressureReached	BOOL	S-0-0830, Druck-Statuswort , Bit 1
	SwivelAngleReached	BOOL	S-0-0849, Volumenstrom-Statuswort , Bit 1

E/A-Typ	Name	Datentyp	Beschreibung
	PressureAct	REAL	S-0-0809, Druck-Istwert , Wichtung lt. Attribut
	SwivelAngleAct	REAL	S-0-0882, Schwenkwinkel-Istwert lt. Attribut
	ValveAct	REAL	P-0-2914, Wirksamer Ventil-Sollwert, Wichtung lt. Attribut
	ControlMode	MB_SI-II_HYDR_CONTROL_MODE	P-0-2951, Pumpenregler Statuswort bzw. P-0-2801, p/Q-Regler Statuswort, bit0 (1 Druckregler aktiv, 0 Volumenstromregler aktiv) MB_SIII_HYDR_CONTROL_MODE, Seite 542
	OutProc	MB_SI-II_OUT_PROC_SWIVEL	Prozess-Ausgangsdaten %Q anschließen. MB_SIII_OUT_PROC_SWIVEL auf Seite 545

Tab. 12-69: Schnittstelle MB_SIIIPlcDrivePressureSwivelAngleControl

Ausgang "State"

Im Ausgang "State" ist eine Grobklassifizierung von Diagnosen binär kodiert.

Die Bits sind in der RIL_FeldbusTypes.library als Globale Konstanten "FieldbusStates" enthalten.

BIT-Nr.	Name	Beschreibung
0	IL_RBS_SLAVE_NOT_REACHABLE	Slave nicht existent
1	IL_RBS_SLAVE_NOT_READY	Slave nicht bereit
2	IL_RBS_SLAVE_CONFIG_ERROR	Slave-Konfigurationsfehler
3	IL_RBS_SLAVE_DIAGNOSIS	Slave-Diagnose verfügbar
4	IL_RBS_SLAVE_ERROR	Slave-Fehler verfügbar
5	IL_RBS_CON_OFF	Mindestens eine zyklische Verbindung läuft nicht

Min-/Max und Default-Werte

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER			FALSE	Kontinuierlich
SercosAdr	UINT	1	255	1	Steigende Flanke an Enable
DriveEnable	BOOL			FALSE	Kontinuierlich
DriveHalt	BOOL			FALSE	Kontinuierlich
ClearError	BOOL			FALSE	Kontinuierlich
PressureCmd	REAL	n.def.	n.def.	0.0	Kontinuierlich
SwivelAngleCmd	REAL	n.def.	n.def.	0.0	Kontinuierlich
InProc	MB_SIII_IN_PROC_SWIVEL				Kontinuierlich

Funktionsbeschreibung

Der Funktionsbaustein schaltet eine Regelpumpe in die Betriebsart "Druck- / Schwenkwinkelregelung" (Betriebsarten-Code: 0x0056). Beim Einschalten des Funktionsbausteins (Über den Eingang "Enable") wird automatisch über den Parameter P-0-2950, Pumpenregler Steuerwort, Bit 5 = 0 überprüft, das für die Pumpe die Masterfunktionalität eingestellt ist. Es werden ein Druck-

sollwert am Eingang "PressureCmd" und Schwenkwinkelsollwert am Eingang "SwivelAngleCmd" vorgegeben.

Der Istwert des Druckes wird am Ausgang "PressureAct" und der Istwert des Schwenkwinkels wird am Ausgang "SwivelAngleAct" angezeigt.

Das Erreichen des Druckistwertes wird am Ausgang "PressureReached" (S-0-0830, Druck-Statuswort, Bit 1) und das Erreichen des Schwenkelwinkelwertes wird am Ausgang "SwivelAngleReached" angezeigt (S-0-0887, Schwenkwinkel Statuswort, Bit 0).

Der wirksame Ventilsollwert (P-0-2914, Wirksamer Ventil-Sollwert) wird zu Diagnosezwecken am Ausgang "ValveAct" angezeigt.

Die aktive Regelungsart wird am Ausgang "ControlMode" angezeigt (Druckregelung oder Schwenkwinkelregelung).

Zusätzlich wird eine (vorhandene) Warnung bzw. Fehler am Baustein angezeigt.

Konfiguration der zyklischen Daten für Sercos

Um die zyklische Kommunikation des Funktionsbausteins mit dem Antrieb aufzubauen müssen Parameter auf die jeweiligen Ein- und Ausgangsvariablen kopiert werden. Die Konfiguration der zyklisch übertragenen Parameter erfolgt im Projektextplorer unterhalb der Sercos-Knotens. Dazu den Knoten "Drive" unterhalb des jeweiligen Gerätes anwählen.

Folgende Parameter müssen zyklisch konfiguriert werden:

Eingänge:

- S-0-0135, Antriebs-Status
- S-0-0144, Signal-Statuswort
- S-0-0809, Druck-Istwert
- S-0-0882, Schwenkwinkel-Istwert
- P-0-2914, Wirksamer Ventil-Sollwert

Ausgänge:

- S-0-0134, Master-Steuerwort
- S-0-0800, Druck-Sollwert
- S-0-0880, Schwenkwinkel-Sollwert

Es müssen zunächst im Reiter „Allgemeine Ein- und Ausgänge“ die zyklischen Parameter konfiguriert werden.

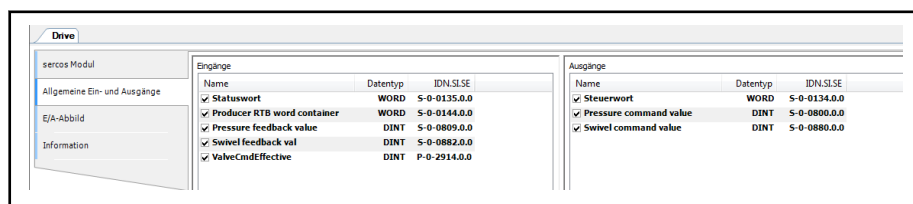


Abb. 12-42: Konfiguration der zyklischen Parameter

Anschließend werden im Reiter "E/A-Abbild" die entsprechenden Ein- und Ausgangsvariablen mit Instanzvariablen des SPS-Programmes verknüpft. Im unteren Bild ist dies eine globale Struktur HPC_DevMapping_gb mit den entsprechenden Strukturelementen. Diese wird dann im SPS-Programm auf die "InProc"- und "OutProc"-Strukturen des Funktionsbausteins kopiert.

Hier können aber auch direkt die "InProc"- und "OutProc"-Strukturen des Funktionsbausteins eingetragen werden.

Variable	Mapping	Kanal	Adresse	Typ
Drive control				
Application.HPC_DevMapping_gb.OutProcSwivel.Controlword.wControlword		Drive control	%QW24	WORD
Pressure command value				
Application.HPC_DevMapping_gb.OutProcSwivel.Pressure		Pressure command value	%QD28	DINT
Swivel command value				
Application.HPC_DevMapping_gb.OutProcSwivel.SwivelAngle		Swivel command value	%QD32	DINT
Drive status				
Application.HPC_DevMapping_gb.InProcSwivel.Statusword.wStatusword		Drive status	%IWS2	WORD
Producer RTB word container				
Application.HPC_DevMapping_gb.InProcSwivel.ConfStatus		Producer RTB word container	%IWS4	WORD
Pressure feedback value				
Application.HPC_DevMapping_gb.InProcSwivel.Pressure		Pressure feedback value	%ID56	DINT
Swivel feedback value				
Application.HPC_DevMapping_gb.InProcSwivel.SwivelAngle		Swivel feedback value	%ID60	DINT
ValveCmdEffective				
Application.HPC_DevMapping_gb.InProcSwivel.ValveEff		ValveCmdEffective	%ID64	DINT

Abb. 12-43: Mappen der Ein- und Ausgangsvariablen

Parametrierung des Gerätes:

Bei Rexroth-Geräten wird die Parametrierung mit "IndraWorks Ds" empfohlen. Dort sind gerätespezifische Inbetriebnahmedialoge realisiert.

Über den Eintrag "IndraWorks Ds hier" (unterhalb des Sercos-Knoten, Geräteknoten, rechte Maustaste) kann "IndraWorks Ds" innerhalb der Engineering-Suite IndraWorks gestartet werden (Routing von der Steuerung zum Gerät erfolgt automatisch).

folgende Parameter müssen eingestellt werden:

- Im Parameter S-0-0032 Hauptbetriebsart muss "Druck- / Schwenkwinkelregelung" (16#0056) eingetragen werden. Die Hauptbetriebsart wird angewählt.
- Im Parameter S-0-0144 Signal-Statuswort muss der Parameter S-0-0830, Druck-Statuswort, Bit 1 ("Drucksollwert erreicht") als Bit 0 eingetragen sein.
- Im Parameter S-0-0144 Signal-Statuswort muss der Parameter S-0-0887, Schwenkwinkel Statuswort, Bit 1 ("Schwenkwinkel-Sollwert erreicht") als Bit 3 eingetragen sein.
- Im Parameter S-0-0144 Signal-Statuswort muss der Parameter P-0-2951, Pumpenregler Statuswort, Bit 0 ("Drucksollwert erreicht") als Bit 4 eingetragen sein.
- Im Parameter S-0-0144 Signal-Statuswort muss der Parameter P-0-2951, Pumpenregler Statuswort, Bit 1 ("Drucksollwert erreicht") als Bit 5 eingetragen sein.

Status	Quelleparameter	Bitnummer
Bit 0:	S-0-0830: Druck-Statuswort	1
Bit 1:	S-0-0000: <kein Signal>	0
Bit 2:	S-0-0000: <kein Signal>	0
Bit 3:	S-0-0887: Schwenkwinkel Statuswort	1
Bit 4:	P-0-2951: Pumpenregler Statuswort	0
Bit 5:	P-0-2951: Pumpenregler Statuswort	1
Bit 6:	S-0-0000: <kein Signal>	0

Abb. 12-44: Parametrierung Signalstatuswort S-0-0144

- Parametrierung des Gerätes gemäß Applikation (Messsystem, Wichtung, Überwachung/Fehlerreaktion...)



Insbesondere Überprüfen, dass die vom Funktionsbaustein vorgegebenen Sollwerte nicht durch eine gerätespezifische Parametrierung überschrieben werden,

z.B. durch Parametrierung der Analogen Eingänge des Gerätes mit internen Sollwerten.

Fehlerbehandlung Zusätzlich zu den Fehlern der intern genutzten Funktionsbausteine MB_SIIIPlcDriveType01 und IL_SIIISvcRead sind folgende Fehler des Bausteines in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#00000006	State	Fehler in der Zustandsmaschine
INPUT_RANGE_ERROR	16#000029B2	Wert des Attributes	Attribut der Parameter S-0-0800 oder S-0-0880 oder P-0-2914 ist nicht 4 Byte dezimal mit Vorzeichen

Tab. 12-70: Fehlercodes FB MB_SIIIPlcPressureSwivelAngleControl

12.5.4 MB_SIIIPlcDrivePressureSwivelAngleSlave

Kurzbeschreibung Der Funktionsbaustein MB_SIIIPlcDrivePressureSwivelAngleSlave erweitert den Funktionsbaustein MB_SIIIPlcDriveType01 und realisiert die Betriebsart **Druck- und Schwenkwinkelregelung** (für Slave).



Details zu den möglichen Betriebsarten sind zu finden unter:

Rexroth HydraulicDrive HDx-20 Funktionen, Anwendungsbeschreibung, DOK-HYDRV*-HD*-20VRS**-APxx-DE-P, RD-30338-FK, Kapitel 6, Betriebsarten.

Es wird die Betriebsart im Gerät ein- bzw. ausgeschaltet sowie ein Druckwert und Schwenkwinkel als Sollwert mit passender Wichtung vorgegeben. Der Druckistwert, der Schwenkwinkelwert und der effektive Ventilsollwert, sowie Statusmeldungen werden angezeigt.

Desweiteren wird beim Einschalten des Bausteins die Slave-Funktionalität im Parameter P-0-2950, Pumpenregler Steuerwortautomatisch eingestellt.

Zielsystem	Bibliothek
MLC/MTX ab 14V16	RMB_SercosIII_Util
MLD mit IndraLogic 2G ab HDx 20VRS bzw. MPx 20VRS	RMB_SercosIII_Util

Tab. 12-71: Referenztablelle

Schnittstellenbeschreibung

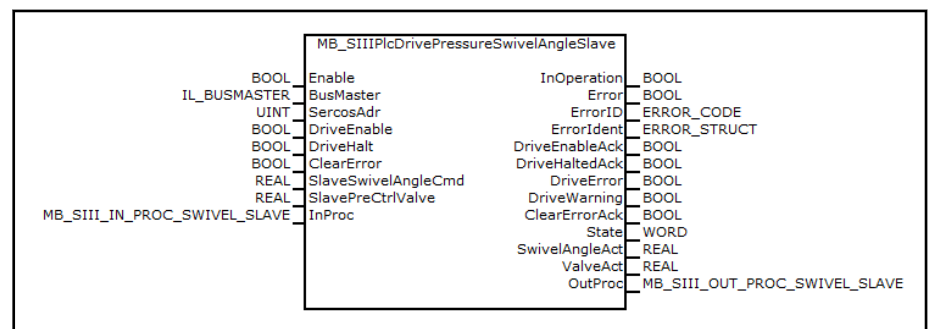


Abb. 12-45: Funktionsbaustein MB_SIIIPlcDrivePressureSwivelAngleSlave

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (pegelgesteuert).
	Busmaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable BusMaster unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 429
	SercosAdr	UINT	Sercos III-Adresse des Slave-Teilnehmers.
	DriveEnable	BOOL	Freigabe des Antriebes (AH/AF) S-0-0134, Master-Steuerwort, Reglerfreigabe, Bit 14
	DriveHalt	BOOL	Stoppen des Antriebes (AH) S-0-0134, Master-Steuerwort, Antrieb Halt, Bit 13
	ClearError	BOOL	Löschen eines Fehlers im Antrieb durch das Kommando Fehler löschen (S-0-0099, C0500 Reset Zustandsklasse 1). Nach gelöschtem Fehler wird die angewählte Freigabe wieder aktiviert.
	SlaveSwivelAngleCmd	REAL	P-0-2935, Schwenkwinkel-Sollwert für Master/Slave Funktion , Wichtung fix
	SlavePreCtrlValve	REAL	Vorsteuerung Slave P-0-2983, Stellgrößenanpassung Vorsteuerung Ventil
	InProc	MB_SII_IN_PROC_SWIVEL_SLAVE	Prozess-Eingangsdaten %I anschließen. MB_SIII_IN_PROC_SWIVEL_SLAVE auf Seite 546
VAR_OUTPUT	InOperation	BOOL	Baustein arbeitet
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCTURE	Detailldiagnose Fehlerbehandlung, Seite 517
	DriveEnableAck	BOOL	Antriebsfreigabe quittiert.
	DriveHaltedAck	BOOL	Antrieb Halt quittiert. S-0-0135, Antriebs-Status, Bit 4
	DriveError	BOOL	Antriebsfehler. S-0-0135, Antriebs-Status, Bit 13
	DriveWarning	BOOL	Antriebswarnung. S-0-0135, Antriebs-Status, Bit 12
	ClearErrorAck	BOOL	Bestätigung, dass das Kommando Fehler löschen ausgeführt wurde. S-0-0099, C0500 Reset Zustandsklasse 1
	State	WORD	Aktueller Status des Antriebes. Siehe unten.
	SwivelAngleAct	REAL	Gebildet aus S-0-0882, Schwenkwinkel-Istwert

E/A-Typ	Name	Datentyp	Beschreibung
	ValveAct	REAL	P-0-2914, Wirksamer Ventil-Sollwert, Wichtung lt. Attribut
	OutProc	MB_SI- II_OUT_PRO C_SWI- VEL_SLAVE	Prozess-Ausgangsdaten %Q anschließen. MB_SIII_OUT_PROC_SWIVEL_SLAVE, Seite 546

Tab. 12-72: Schnittstelle MB_SIIIPlcDrivePressureSwivelAngleSlave

Ausgang "State"

Im Ausgang "State" ist eine Grobklassifizierung von Diagnosen binär kodiert. Die Bits sind in der RIL_FeldbusTypes.library als Globale Konstanten "FieldbusStates" enthalten.

BIT-Nr.	Name	Beschreibung
0	IL_RBS_SLAVE_NOT_REACHABLE	Slave nicht existent
1	IL_RBS_SLAVE_NOT_READY	Slave nicht bereit
2	IL_RBS_SLAVE_CONFIG_ERROR	Slave-Konfigurationsfehler
3	IL_RBS_SLAVE_DIAGNOSIS	Slave-Diagnose verfügbar
4	IL_RBS_SLAVE_ERROR	Slave-Fehler verfügbar
5	IL_RBS_CON_OFF	Mindestens eine zyklische Verbindung läuft nicht

Min-/Max und Default-Werte

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER			FALSE	Kontinuierlich
SercosAdr	UINT	1	255	1	Steigende Flanke an Enable
DriveEnable	BOOL			FALSE	Kontinuierlich
DriveHalt	BOOL			FALSE	Kontinuierlich
ClearError	BOOL			FALSE	Kontinuierlich
SlaveSwivelAngleCmd	REAL	n.def.	n.def.	0.0	Kontinuierlich
SlavePreCtrlValve	REAL	n.def.	n.def.	0.0	Kontinuierlich
InProc	MB_SIII_IN_PROC_SWIVEL_SLAVE				Kontinuierlich

Funktionsbeschreibung

Der Funktionsbaustein schaltet in einem Mehrpumpensystem eine Regelpumpe mit Slavefunktionalität in die Betriebsart "Druck- / Schwenkwinkelregelung" (Betriebsarten-Code: 0x0056). Die Slavefunktionalität wird beim Einschalten des Funktionsbausteins (Über den Eingang "Enable") automatisch über den Parameter P-0-2950, Pumpenregler Steuerwort, Bit 5 = 1 eingestellt.

Es wird ein Schwenkwinkelsollwert am Eingang "SwivelAngleCmd" vorgegeben, der von einer Regelpumpe mit Masterfunktionalität stammt. Zusätzlich kann über den Eingang "SlavePreCtrlValve" der Ventilsollwert des Masterpumpe als Vorsteuerwort aufgeschaltet werden (P-0-2983, Stellgrößenanpassung Vorsteuerung Ventil).

Konfiguration der zyklischen Daten für Sercos

Um die zyklische Kommunikation des Funktionsbausteins mit dem Antrieb aufzubauen müssen Parameter auf die jeweiligen Ein- und Ausgangsvariablen kopiert werden. Die Konfiguration der zyklisch übertragenen Parameter erfolgt im Projektextplorer unterhalb der Sercos-Knotens. Dazu den Knoten "Drive" unterhalb des jeweiligen Gerätes anwählen.

Folgende Parameter müssen zyklisch konfiguriert werden:

Eingänge:

- S-0-0135, Antriebs-Status
- S-0-0882, Schwenkwinkel-Istwert
- P-0-2914, Wirksamer Ventil-Sollwert

Ausgänge:

- S-0-0134, Master-Steuerwort
- S-0-0880, Schwenkwinkel-Sollwert
- P-0-2935, Schwenkwinkel-Sollwert für Master/Slave Funktion
- P-0-2983, Stellgrößenanpassung Vorsteuerung Ventil

Es müssen zunächst im Reiter „Allgemeine Ein- und Ausgänge“ die zyklischen Parameter konfiguriert werden.

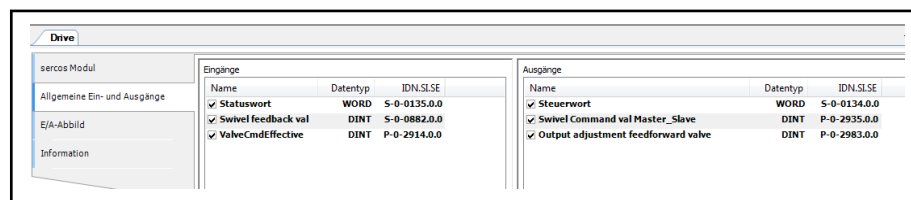


Abb. 12-46: Konfiguration der zyklischen Parameter

Anschließend werden im Reiter "E/A-Abbild" die entsprechenden Ein- und Ausgangsvariablen mit Instanzvariablen des SPS-Programmes verknüpft. Im unteren Bild ist dies eine globale Struktur HPC_DevMapping_gb mit den entsprechenden Strukturelementen. Diese wird dann im SPS-Programm auf die "InProc"- und "OutProc"-Strukturen des Funktionsbausteins kopiert.

Hier können aber auch direkt die "InProc"- und "OutProc"-Strukturen des Funktionsbausteins eingetragen werden.

Funktionsbeschreibung

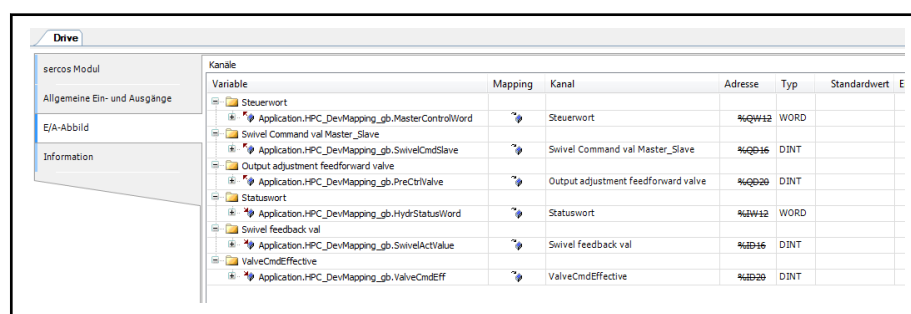


Abb. 12-47: Mappen der Ein- und Ausgangsvariablen

Parametrierung des Gerätes:



Bei Rexroth-Geräten wird die Parametrierung mit "IndraWorks Ds" empfohlen. Dort sind gerätespezifische Inbetriebnahmedialoge realisiert.

Über den Eintrag "IndraWorks Ds hier" (unterhalb des Sercos-Knoten, Geräteknoten, rechte Maustaste) kann "IndraWorks Ds" innerhalb der Engineering-Suite IndraWorks gestartet werden (Routing von der Steuerung zum Gerät erfolgt automatisch).

folgende Parameter müssen eingestellt werden:

- Im Parameter S-0-0032, Hauptbetriebsart muss "Druck- / Schwenkwinkelregelung" (16#0056) eingetragen werden. Die Hauptbetriebsart wird angewählt.
- Parametrierung des Gerätes gemäß Applikation (Messsystem, Wichtung, Überwachung/Fehlerreaktion...)



Insbesondere Überprüfen, dass die vom Funktionsbaustein vorgegebenen Sollwerte nicht durch eine gerätespezifische Parametrierung überschrieben werden,

z.B. durch Parametrierung der Analogen Eingänge des Gerätes mit internen Sollwerten.

Fehlerbehandlung Zusätzlich zu den Fehlern der intern genutzten Funktionsbausteine MB_SIIIPlcDriveType01 und IL_SIIISvcRead sind folgende Fehler des Bausteines in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#00000006	State	Fehler in der Zustandsmaschine
INPUT_RANGE_ERROR	16#000029B3	Wert des Attributes	Attribut der Parameter P-0-2983, P-0-2935 oder P-0-2914 ist nicht 4 Byte dezimal mit Vorzeichen

Tab. 12-73: Fehlercodes FB MB_SIIIPlcDrivePressureSwivelAngleSlave

12.5.5 MB_SIIIPlcDriveDirectValveControl

Kurzbeschreibung Der Funktionsbaustein MB_SIIIPlcDriveDirectValveControl erweitert den Funktionsbaustein MB_SIIIPlcDriveType01 und realisiert die Betriebsart **Ventildirektsteuerung**.

Wert im Betriebsartenparameter: 0x0051



Details zu den möglichen Betriebsarten sind zu finden unter:

Rexroth HydraulicDrive HDx-20 Funktionen, Anwendungsbeschreibung, DOK-HYDRV*-HD*-20VRS**-APxx-DE-P, RD-30338-FK, Kapitel 6, Betriebsarten.

Zielsystem	Bibliothek
MLC/MTX ab 14V20	RMB_SercosIII_Util
MLD mit IndraLogic 2G ab HDx 20VRS bzw. MPx 20VRS	RMB_SercosIII_Util

Tab. 12-74: Referenztablelle

Schnittstellenbeschreibung

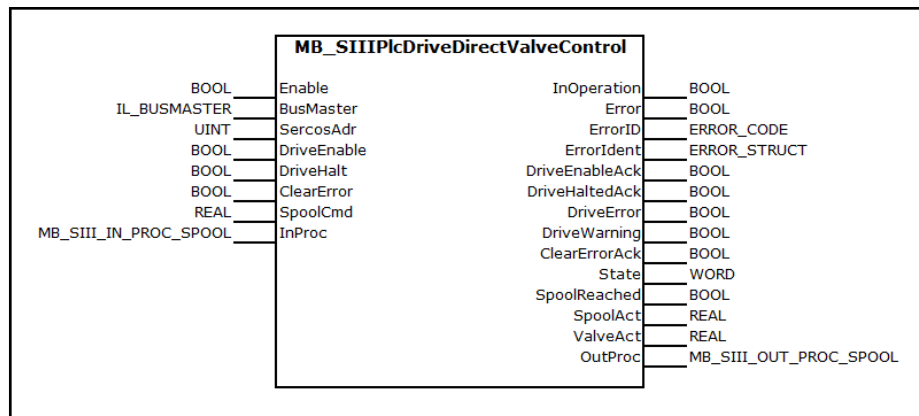


Abb. 12-48: Funktionsbaustein MB_SIIIPlcDriveDirectValveControl

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (pegelgesteuert).
	Busmaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable BusMaster unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 429
	SercosAdr	UINT	Sercos III-Adresse des Slave-Teilnehmers
	DriveEnable	BOOL	Freigabe des Antriebes (AH/AF) S-0-0134, Master-Steuerwort, Reglerfreigabe, Bit 14
	DriveHalt	BOOL	Stoppen des Antriebes (AH) S-0-0134, Master-Steuerwort, Antrieb Halt, Bit 13
	ClearError	BOOL	Löschen eines Fehlers im Antrieb durch das Kommando Fehler löschen (S-0-0099, C0500 Reset Zustandsklasse 1). Nach gelöschtem Fehler wird die angewählte Freigabe wieder aktiviert.
	SpoolCmd	REAL	S-0-0860, Ventil-Sollwert , Wichtung lt. Attribut
	InProc	MB_SIII_IN_PROC_SPOOL	Eingangsstruktur MB_SIII_IN_PROC_SPOOL , Seite 547
VAR_OUTPUT	InOperation	BOOL	Baustein arbeitet
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCT	Detailldiagnose Fehlerbehandlung , Seite 521
	DriveEnable-Ack	BOOL	Antriebsfreigabe quittiert
	DriveHaltedAck	BOOL	Antrieb Halt quittiert S-0-0135, Antriebs-Status, Bit 4

E/A-Typ	Name	Datentyp	Beschreibung
	DriveError	BOOL	Antriebsfehler S-0-0135, Antriebs-Status, Bit 13
	DriveWarning	BOOL	Antriebswarnung S-0-0135, Antriebs-Status, Bit 12
	ClearErrorAck	BOOL	Bestätigung, dass das Kommando Fehler löschen ausgeführt wurde S-0-0099, C0500 Reset Zustandsklasse 1
	State	WORD	Device Status von IL_RemoteSlaveState
	SpoolReached	BOOL	Ventilsollwert erreicht (S-0-0867, Ventil StatuswortBit 0)
	SpoolAct	REAL	S-0-0862, Ventil-Istwert , Wichtung lt. Attribut
	ValveAct	REAL	P-0-2914, Wirksamer Ventil-Sollwert , Wichtung lt. Attribut
	OutProc	MB_SI-II_OUT_PROC_SPOOL	Ausgangsstruktur MB_SIII_OUT_PROC_SPOOL, Seite 547

Tab. 12-75: Schnittstelle MB_SIIIPlcDriveDirectValveControl

Min-/Max und Default-Werte

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER			FALSE	Kontinuierlich
SercosAdr	UINT	1	255	1	Steigende Flanke an Enable
DriveEnable	BOOL			FALSE	Kontinuierlich
DriveHalt	BOOL			FALSE	Kontinuierlich
ClearError	BOOL			FALSE	Kontinuierlich
SpoolCmd	REAL	n. def.	n. def.	0.0	Kontinuierlich
InProc	MB_SIII_IN_PROC_SPOOL				Kontinuierlich

Funktionsbeschreibung

Der Funktionsbaustein schaltet den Antrieb in die Betriebsart "Ventildirektsteuerung" (Betriebsartencode: 16#0051). Es wird ein Ventilsollwert am Eingang "SpoolCmd" vorgegeben.

Der Ventilistwert wird am Ausgang "SpoolAct" angezeigt.

Die Umrechnung dieser Gleitkommagrößen (REAL) in die zyklischen Festpunktwerte des Gerätes (DINT) erfolgt intern im Baustein über die Strukturen "InProc" und "OutProc".

Das Erreichen des Ventilsollwertes wird am Ausgang "SpoolReached" (S-0-0867, Ventil Statuswort Bit 0) angezeigt.

Der wirksame Ventilsollwert (P-0-2914, Wirksamer Ventil-Sollwert) wird zu Diagnosezwecken am Ausgang "ValveAct" angezeigt.

Konfiguration der zyklischen Daten für Sercos:

Zusätzlich wird eine (vorhandene) Warnung bzw. ein Fehler am Baustein angezeigt.

Um die zyklische Kommunikation des Funktionsbausteins mit dem Antrieb aufzubauen, müssen Parameter auf die jeweiligen Ein- und Ausgangsvariablen kopiert werden. Die Konfiguration der zyklisch übertragenen Parameter erfolgt im Projektexplorer unterhalb des Sercos-Knotens. Dazu den Knoten "Drive" unterhalb des jeweiligen Gerätes anwählen.

Folgende Parameter müssen zyklisch konfiguriert werden:

Eingänge:

- S-0-0135, Antriebs-Status
- S-0-0144, Signal-Statuswort
- S-0-0862, Ventil-Istwert
- P-0-2914, Wirksamer Ventil-Sollwert

Ausgänge:

- S-0-0134, Master-Steuerwort
- S-0-0860, Ventil-Sollwert

Es müssen zunächst im Reiter "Allgemeine Ein- und Ausgänge" die zyklischen Parameter konfiguriert werden.

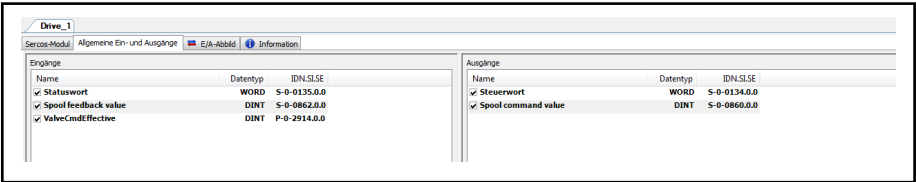


Abb. 12-49: Konfiguration der zyklischen Parameter

Anschließend werden im Reiter "E/A-Abbild" die entsprechenden Ein- und Ausgangsvariablen mit Instanzvariablen des SPS-Programms verknüpft. Im unteren Bild ist dies eine globale Struktur HPC_DevMapping_gb mit den entsprechenden Strukturelementen. Diese Struktur wird dann im SPS-Programm auf die "InProc"- und "OutProc"-Strukturen des Funktionsbausteins kopiert.

Hier können aber auch direkt die "InProc"- und "OutProc"-Strukturen des Funktionsbausteins eingetragen werden.

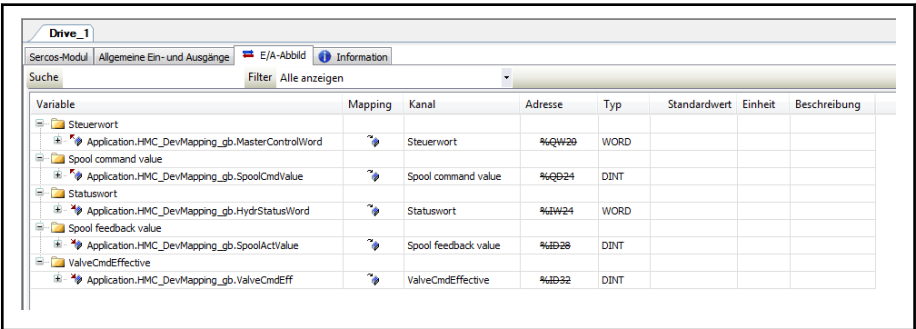


Abb. 12-50: Mappen der Ein- und Ausgangsvariablen

Parametrierung des Gerätes:

Bei Rexroth-Geräten wird die Parametrierung mit "IndraWorks Ds" empfohlen. Dort sind gerätespezifische Inbetriebnahmedialoge realisiert.

Über den Eintrag "IndraWorks Ds hier" (unterhalb des Sercos-Knoten, Geräteknoten, rechte Maustaste) kann "IndraWorks Ds" innerhalb der Engineering-Suite IndraWorks gestartet werden (Routing von der Steuerung zum Gerät erfolgt automatisch).

Folgende Parameter müssen eingestellt werden:

- Hauptbetriebsart auf Ventildirektsteuerung einstellen
 - S-0-0032 " Hauptbetriebsart" Wert "0x0051"
- Parameter und Bits einstellen S-0-0144 " Signal-Statuswort"
 - ...
 - Bit 6: S-0-0867 " Ventil Statuswort" Bit 0

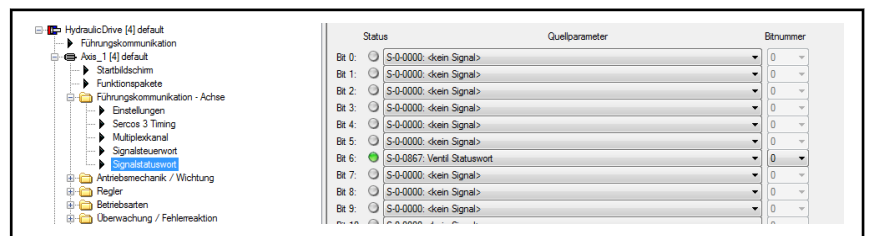


Abb. 12-51: Parametrierung Signalstatuswort S-0-0144

- Parametrierung des Gerätes gemäß Applikation (Messsystem, Wichtung, Überwachung/Fehlerreaktion...)



Insbesondere Überprüfen, dass die vom Funktionsbaustein vorgegebenen Sollwerte nicht durch eine gerätespezifische Parametrierung überschrieben werden,

z.B. durch Parametrierung der Analogen Eingänge des Gerätes mit internen Sollwerten.

Fehlerbehandlung

Zusätzlich zu den Fehlern der intern genutzten Funktionsbausteine MB_SIIPlcDriveType01 und IL_SIIISvcRead sind folgende Fehler des Bausteins in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#00000006	State	Fehler in der Zustandsmaschine
INPUT_RANGE_ERROR	16#000029B4	Wert des Attributes	Attribut der Parameter S-0-0860 oder P-0-2914 ist nicht 4 Byte dezimal mit Vorzeichen

Tab. 12-76: Fehlercodes FB MB_SIIPlcDriveDirectValveControl

12.5.6 MB_SIIPlcDriveTorqueDirectValveControl

Kurzbeschreibung

Der Funktionsbaustein MB_SIIPlcDriveTorqueDirectValveControl erweitert den Funktionsbaustein MB_SIIPlcDriveType01 und realisiert die Betriebsart **Drehmoment-/Kraftregelung/Ventildirektsteuerung**.

Wert im Betriebsartenparameter: 0x8051

Zielsystem	Bibliothek
MLC/MTX ab 14V20	RMB_SercosIII_Util
MLD mit IndraLogic 2G ab HDx 20VRS bzw. MPx 20VRS	RMB_SercosIII_Util

Tab. 12-77: Referenztable

Schnittstellenbeschreibung

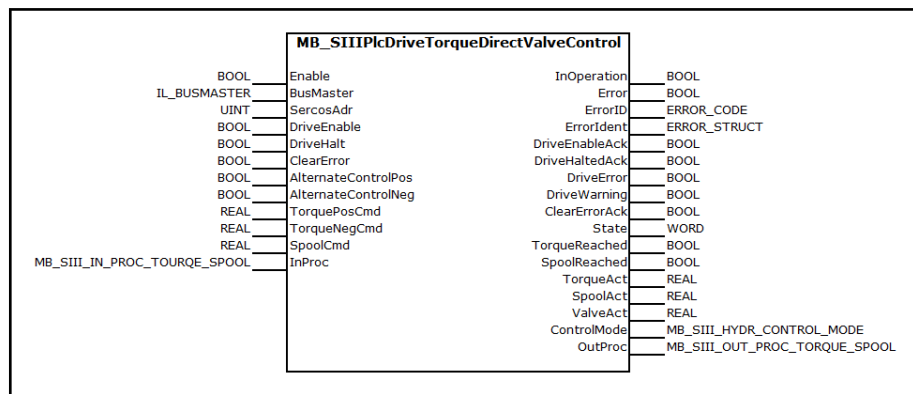


Abb. 12-52: Funktionsbaustein MB_SIIPlcDriveTorqueDirectValveControl

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (pegelgesteuert)
	Busmaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable BusMaster unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0 siehe auch IL_BUSMASTER auf Seite 429
	SercosAdr	UINT	Sercos III-Adresse des Slave-Teilnehmers
	DriveEnable	BOOL	Freigabe des Antriebes (AH/AF) S-0-0134, Master-Steuerwort, Reglerfreigabe, Bit 14
	DriveHalt	BOOL	Stoppen des Antriebes (AH) S-0-0134, Master-Steuerwort, Antrieb Halt, Bit 13
	ClearError	BOOL	Löschen eines Fehlers im Antrieb durch das Kommando Fehler löschen (S-0-0099, C0500 Reset Zustandsklasse 1). Nach gelöschtem Fehler wird die angewählte Freigabe wieder aktiviert
	AlternateControl Pos	BOOL	Ablösende Regelung in positiver Richtung aktivieren P-0-2854, " Ablösende Regelung Steuerwort 2" Bit3 (P-0-2853 Bit 0 muss gesetzt werden)
	AlternateControlNeg	BOOL	Ablösende Regelung in negativer Richtung aktivieren P-0-2854 " Ablösende Regelung Steuerwort 2" Bit4 (P-0-2853 Bit 0 muss gesetzt werden)
	TorquePosCmd	REAL	S-0-0550, Drehmoment-/Kraft-Sollwert positiv , Wichtung lt. Attribut
	TorqueNegCmd	REAL	S-0-0551, Drehmoment-/Kraft-Sollwert negativ , Wichtung lt. Attribut

E/A-Typ	Name	Datentyp	Beschreibung
	SpoolCmd	REAL	S-0-0860, Ventil-Sollwert , Wichtung lt. Attribut
	InProc	MB_SIII_IN_PROC_TORQUE_SPOOL	Eingangsstruktur MB_SIII_IN_PROC_TORQUE_SPOOL, Seite 548
VAR_OUTPUT	InOperation	BOOL	Baustein arbeitet
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCT	Detaildiagnose Fehlerbehandlung, Seite 526
	DriveEnable-Ack	BOOL	Antriebsfreigabe quittiert
	DriveHaltedAck	BOOL	Antrieb Halt quittiert S-0-0135, Antriebs-Status, Bit 4
	DriveError	BOOL	Antriebsfehler S-0-0135, Antriebs-Status, Bit 13
	DriveWarning	BOOL	Antriebswarnung S-0-0135, Antriebs-Status, Bit 12
	ClearErrorAck	BOOL	Bestätigung, dass das Kommando Fehler löschen ausgeführt wurde S-0-0099, C0500 Reset Zustandsklasse 1
	State	WORD	Gerätestatus von IL_SIIIRemoteSlaveState
	TorqueReached	BOOL	Kraftsollwert erreicht (S-0-0824, Meldung Drehmoment-/Kraftsollwert erreicht Bit 0)
	SpoolReached	BOOL	Ventilsollwert erreicht (S-0-0867, Ventil StatuswortBit 0)
	TorqueAct	REAL	S-0-0552, Drehmoment-/Kraft-Istwert , Wichtung lt. Attribut
	SpoolAct	REAL	S-0-0862, Ventil-Istwert , Wichtung lt. Attribut
	ValveAct	REAL	P-0-2914, Wirksamer Ventil-Sollwert , Wichtung lt. Attribut
	ControlMode	MB_SII-HYDR_CONTROL_MODE	P-0-2855, Ablösende Regelung Statuswort, Bit 0 (1 Kraftregelung Regler aktiv) MB_SIII_HYDR_CONTROL_MODE, Seite 542
	OutProc	MB_SIII_OUT_PROC_TORQUE_SPOOL	Ausgangsstruktur MB_SIII_OUT_PROC_TORQUE_SPOOL, Seite 548

Tab. 12-78: Schnittstelle MB_SIIIPlcDriveTorqueDirectValveControl

Min-/Max und Default-Werte

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER			FALSE	Kontinuierlich
SercosAdr	UINT	1	255	1	Steigende Flanke an Enable
DriveEnable	BOOL			FALSE	Kontinuierlich
DriveHalt	BOOL			FALSE	Kontinuierlich
ClearError	BOOL			FALSE	Kontinuierlich
AlternateControl Pos	BOOL			FALSE	Kontinuierlich
AlternateControlNeg	BOOL			FALSE	Kontinuierlich
TorquePosCmd	REAL	n. def.	n. def.	0.0	Kontinuierlich
TorqueNegCmd	REAL	n. def.	n. def.	0.0	Kontinuierlich
SpoolCmd	REAL	n. def.	n. def.	0.0	Kontinuierlich
InProc	MB_SIII_IN_PROC_TORQUE_SPOOL				Kontinuierlich

Funktionsbeschreibung

Der Funktionsbaustein schaltet den Antrieb in die Betriebsart "Drehmoment-/Kraftregelung/Ventildirektsteuerung" (Betriebsartencode: 16#8051). Es wird ein Ventilsollwert am Eingang "SpoolCmd" vorgegeben.

Der Ventilistwert wird am Ausgang "SpoolAct" angezeigt.

Für die ablösende Regelung können am Eingang "TorquePosCmd" ein positiver Drehmoment- bzw. Kraft-Sollwert und am Eingang "TorqueNegCmd" ein negativer Drehmoment bzw. Kraft-Sollwert vorgegeben werden.

Der Drehmoment- bzw. Kraftistwert wird am Ausgang "TorqueAct" angezeigt.

Die Umrechnung dieser Gleitkommagrößen (REAL) in die zyklischen Festpunktwerte des Gerätes (DINT) erfolgt intern im Baustein über die Strukturen "InProc" und "OutProc".

Das Erreichen des Ventilsollwertes wird am Ausgang "SpoolReached" (S-0-0867, Ventil StatuswortBit 0 angezeigt).

Das Erreichen des Drehmoment- Kraftsollwertes wird am Ausgang "TorqueReached" (S-0-0824, Meldung Drehmoment-/Kraftsollwert erreichtBit 0) angezeigt.

Der wirksame Ventilsollwert (P-0-2914, Wirksamer Ventil-Sollwert) wird zu Diagnosezwecken am Ausgang "ValveAct" angezeigt.

Die gerätespezifische Umschaltung zwischen Ventildirektsteuerung und Drehmoment- bzw. Kraftregelung wird am Ausgang "ControlMode" angezeigt. Dazu wird

P-0-2855, Ablösende Regelung Statuswortausgewertet.

Zusätzlich wird eine (vorhandene) Warnung bzw. Fehler am Baustein angezeigt.

Konfiguration der zyklischen Daten für Sercos:

Um die zyklische Kommunikation des Funktionsbausteins mit dem Antrieb aufzubauen, müssen Parameter auf die jeweiligen Ein- und Ausgangsvariablen kopiert werden. Die Konfiguration der zyklisch übertragenen Parameter erfolgt im Projektextplorer unterhalb des Sercos-Knotens. Dazu den Knoten "Drive" unterhalb des jeweiligen Gerätes anwählen.

Folgende Parameter müssen zyklisch konfiguriert werden:

Eingänge:

- S-0-0135, Antriebs-Status
- S-0-0144, Signal-Statuswort
- S-0-0552, Drehmoment-/Kraft-Istwert
- S-0-0862, Ventil-Istwert
- P-0-2914, Wirksamer Ventil-Sollwert

Ausgänge:

- S-0-0134, Master-Steuerwort
- S-0-0145, Signal-Steuerwort
- S-0-0550, Drehmoment-/Kraft-Sollwert positiv
- S-0-0551, Drehmoment-/Kraft-Sollwert negativ
- S-0-0860, Ventil-Sollwert

Es müssen zunächst im Reiter "Allgemeine Ein- und Ausgänge" die zyklischen Parameter konfiguriert werden.

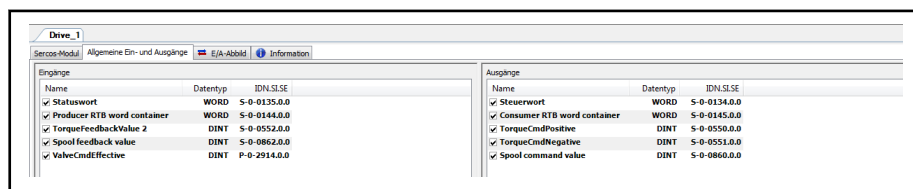


Abb. 12-53: Konfiguration der zyklischen Parameter

Anschließend werden im Reiter "E/A-Abbild" die entsprechenden Ein- und Ausgangsvariablen mit Instanzvariablen des SPS-Programms verknüpft. Im unteren Bild ist dies eine globale Struktur HPC_DevMapping_gb mit den entsprechenden Strukturelementen. Diese Struktur wird dann im SPS-Programm auf die "InProc"- und "OutProc"-Strukturen des Funktionsbausteins kopiert.

Hier können aber auch direkt die "InProc"- und "OutProc"-Strukturen des Funktionsbausteins eingetragen werden.

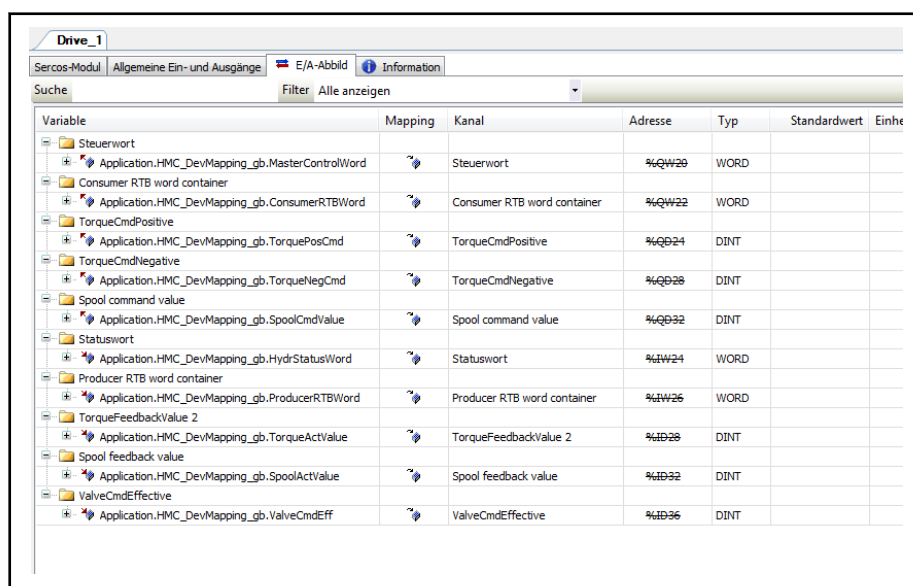


Abb. 12-54: Mappen der Ein- und Ausgangsvariablen

Parametrierung des Gerätes:



Bei Rexroth-Geräten wird die Parametrierung mit "IndraWorks Ds" empfohlen. Dort sind gerätespezifische Inbetriebnahmedialoge realisiert.

Über den Eintrag "IndraWorks Ds hier" (unterhalb des Sercos-Knoten, Geräteknoten, rechte Maustaste) kann "IndraWorks Ds" innerhalb der Engineering-Suite IndraWorks gestartet werden (Routing von der Steuerung zum Gerät erfolgt automatisch).

Folgende Parameter müssen eingestellt werden:

- Hauptbetriebsart auf Drehmoment-/Kraftregelung/Ventildirektsteuerung einstellen
 - S-0-0032 "Hauptbetriebsart" Wert "0x8051"
 - Bit 8: P-0-2853 Ablösende Regelung Steuerwort 1 Bit 0 setzen
- Parameter und Bits einstellen S-0-0144 "Signal-Statuswort"
 - ...
 - Bit 6: S-0-0867 "Ventil Statuswort" Bit 0
 - Bit 7: S-0-0824 "Meldung Drehmoment-/Kraftsollwert erreicht" Bit 0
 - Bit 8: P-0-2855 "Ablösende Regelung Statuswort" Bit 0

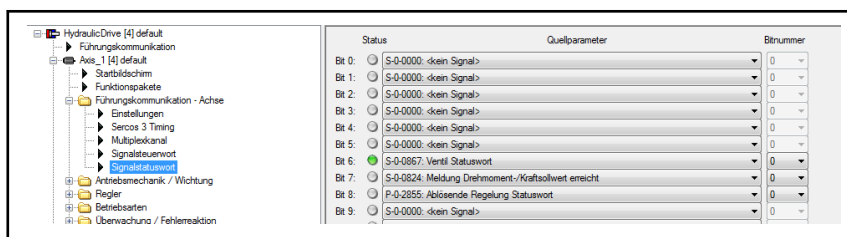


Abb. 12-55: Parametrierung Signalstatuswort S-0-0144

- Parameter und Bits einstellen S-0-0145 "Signal-Steuerwort"
 - Bit 0: P-0-2854 "Ablösende Regelung Steuerwort 2" Bit 3
 - Bit 1: P-0-2854 "Ablösende Regelung Steuerwort 2" Bit 4

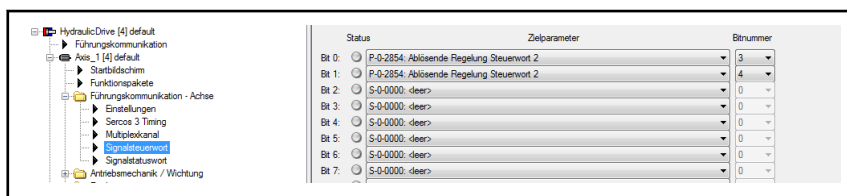


Abb. 12-56: Parametrierung Signalsteuerwort S-0-0145

- Parametrierung des Gerätes gemäß Applikation (Messsystem, Wichtung, Überwachung/Fehlerreaktion...)



Insbesondere überprüfen, dass die vom Funktionsbaustein vorgegebenen Sollwerte nicht durch eine gerätespezifische Parametrierung überschrieben werden, z.B. durch Parametrierung der Analogen Eingänge des Gerätes mit internen Sollwerten.

Fehlerbehandlung

Zusätzlich zu den Fehlern der intern genutzten Funktionsbausteine MB_SII-IPIcDriveType01 und IL_SIIISvcRead sind folgende Fehler des Bausteines in

der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#00000006	State	Fehler in der Zustandsmaschine
INPUT_RANGE_ERROR	16#000029B5	Wert des Attributes	Attribut der Parameter S-0-0860, S-0-0550, S-0-0551 oder P-0-2914 ist nicht 4 Byte dezimal mit Vorzeichen

Tab. 12-79: Fehlercodes MB_SIIIPlcDriveTorqueDirectValveControl

12.5.7 MB_SIIIPlcDriveTorqueFlowControl

Kurzbeschreibung Der Funktionsbaustein MB_SIIIPlcDriveTorqueFlowControl erweitert den Funktionsbaustein MB_SIIIPlcDriveType01 und realisiert die Betriebsart **Drehmoment-/Kraftregelung/Volumenstromregelung**.

Wert im Betriebsartenparameter: 0x8052

Zielsystem	Bibliothek
MLC/MTX ab 14V20	RMB_SercosIII_Util
MLD mit IndraLogic 2G ab HDx 20VRS bzw. MPx 20VRS	RMB_SercosIII_Util

Tab. 12-80: Referenztable

Schnittstellenbeschreibung

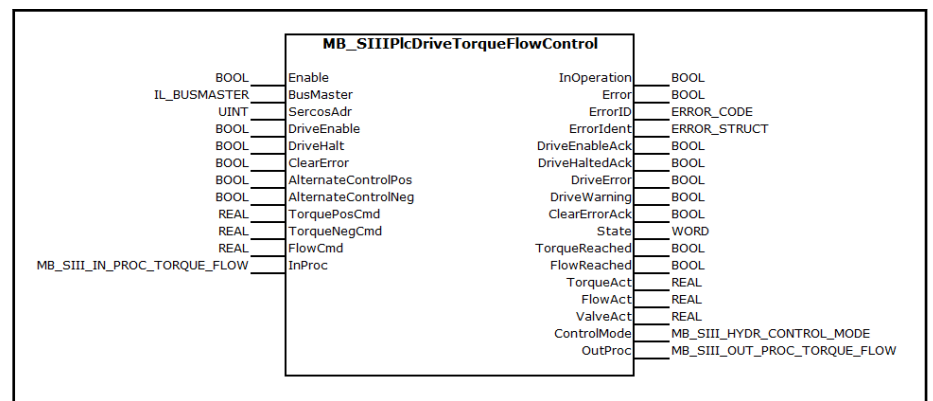


Abb. 12-57: Funktionsbaustein MB_SIIIPlcDriveTorqueFlowControl

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (pegelgesteuert)
	Busmaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable BusMaster unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0 siehe auch IL_BUSMASTER auf Seite 429
	SercosAdr	UINT	Sercos III-Adresse des Slave-Teilnehmers
	DriveEnable	BOOL	Freigabe des Antriebes (AH/AF) S-0-0134, Master-Steuerwort, Reglerfreigabe, Bit 14
	DriveHalt	BOOL	Stoppen des Antriebes (AH) S-0-0134, Master-Steuerwort, Antrieb Halt, Bit 13

E/A-Typ	Name	Datentyp	Beschreibung
	ClearError	BOOL	Löschen eines Fehlers im Antrieb durch das Kommando Fehler löschen (S-0-0099, C0500 Reset Zustandsklasse 1). Nach gelöschtem Fehler wird die angewählte Freigabe wieder aktiviert
	AlternateControlPos	BOOL	Ablösende Regelung in positiver Richtung aktivieren P-0-2854, " Ablösende Regelung Steuerwort 2" Bit3 (P-0-2853 Bit 0 muss gesetzt werden)
	AlternateControlNeg	BOOL	Ablösende Regelung in negativer Richtung aktivieren P-0-2854 " Ablösende Regelung Steuerwort 2" Bit4 (P-0-2853 Bit 0 muss gesetzt werden)
	TorquePosCmd	REAL	S-0-0550, Drehmoment-/Kraft-Sollwert positiv , Wichtung lt. Attribut
	TorqueNegCmd	REAL	S-0-0551, Drehmoment-/Kraft-Sollwert negativ , Wichtung lt. Attribut
	FlowCmd	REAL	S-0-0840, Volumenstrom-Sollwert , Wichtung lt. Attribut Volumenstrom-Sollwert für S-0-0840
	InProc	MB_SIII_IN_PROC_TORQUE_FLOW	Eingangsstruktur MB_SIII_IN_PROC_TORQUE_FLOW, Seite 548
VAR_OUTPUT	InOperation	BOOL	Baustein arbeitet
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCT	Detailldiagnose Fehlerbehandlung, Seite 532
	DriveEnableAck	BOOL	Antriebsfreigabe quittiert
	DriveHaltedAck	BOOL	Antrieb Halt quittiert S-0-0135, Antriebs-Status, Bit 4
	DriveError	BOOL	Antriebsfehler S-0-0135, Antriebs-Status, Bit 13
	DriveWarning	BOOL	Antriebswarnung S-0-0135, Antriebs-Status, Bit 12
	ClearErrorAck	BOOL	Bestätigung, dass das Kommando Fehler löschen ausgeführt wurde S-0-0099, C0500 Reset Zustandsklasse 1
	State	WORD	Gerätestatus von IL_SIIIRemoteSlaveState
	TorqueReached	BOOL	Kraftsollwert erreicht (S-0-0824, Meldung Drehmoment-/Kraftsollwert erreicht Bit 0)
	FlowReached	BOOL	Volumenstromsollwert erreicht (S-0-0849, Volumenstrom-Statuswort Bit 1)
	TorqueAct	REAL	S-0-0552, Drehmoment-/Kraft-Istwert , Wichtung lt. Attribut

E/A-Typ	Name	Datentyp	Beschreibung
	FlowAct	REAL	S-0-0862, Ventil-Istwert , Wichtung lt. Attribut Volumenstrom-Istwert von S-0-0862
	ValveAct	REAL	P-0-2914, Wirksamer Ventil-Sollwert , Wichtung lt. Attribut
	ControlMode	MB_SI-II_HYDR_CONTROL_MODE	P-0-2855, Ablösende Regelung Statuswort, Bit 0 (1 Kraftregelung Regler aktiv) MB_SIII_HYDR_CONTROL_MODE, Seite 542
	OutProc	MB_SIII_OUT_PROC_TORQUE_FLOW	Ausgangsstruktur MB_SIII_OUT_PROC_TORQUE_FLOW, Seite 549

Tab. 12-81: Schnittstelle MB_SIIIPlcDriveTorqueFlowControl

Min-/Max und Default-Werte

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER			FALSE	Kontinuierlich
SercosAdr	UINT	1	255	1	Steigende Flanke an Enable
DriveEnable	BOOL			FALSE	Kontinuierlich
DriveHalt	BOOL			FALSE	Kontinuierlich
ClearError	BOOL			FALSE	Kontinuierlich
AlternateControl Pos	BOOL			FALSE	Kontinuierlich
AlternateControlNeg	BOOL			FALSE	Kontinuierlich
TorquePosCmd	REAL	n. def.	n. def.	0.0	Kontinuierlich
TorqueNegCmd	REAL	n. def.	n. def.	0.0	Kontinuierlich
FlowCmd	REAL	n. def.	n. def.	0.0	Kontinuierlich
InProc	MB_SIII_IN_PROC_TORQUE_FLOW				Kontinuierlich

Funktionsbeschreibung

Der Funktionsbaustein schaltet den Antrieb in die Betriebsart "Drehmoment-/Kraftregelung / Volumenstrom" (Betriebsartencode: 16#8052). Es wird ein Volumenstromsollwert am Eingang "FlowCmd" vorgegeben.

Der Volumenstromistwert wird am Ausgang "FlowAct" angezeigt.

Für die ablösende Regelung können am Eingang "TorquePosCmd" ein positiver Drehmoment- bzw. Kraft-Sollwert und am Eingang "TorqueNegCmd" ein negativer Drehmoment bzw. Kraft-Sollwert vorgegeben werden.

Der Drehmoment- bzw. Kraftistwert wird am Ausgang "TorqueAct" angezeigt.

Die Umrechnung dieser Gleitkommagrößen (REAL) in die zyklischen Festpunktwerte des Gerätes (DINT) erfolgt intern im Baustein über die Strukturen "InProc" und "OutProc".

Das Erreichen des Volumenstromsollwertes wird am Ausgang "FlowReached" (S-0-0849, Volumenstrom-StatuswortBit 1 angezeigt).

Das Erreichen des Drehmoment- Kraftsollwertes wird am Ausgang "Torque-Reached" (S-0-0824, Meldung Drehmoment-/Kraftsollwert erreichtBit 0) angezeigt.

Der wirksame Ventilsollwert (P-0-2914, Wirksamer Ventil-Sollwert) wird zu Diagnosezwecken am Ausgang "ValveAct" angezeigt.

Die gerätespezifische Umschaltung zwischen Ventildirektsteuerung und Drehmoment- bzw. Kraftregelung wird am Ausgang "ControlMode" angezeigt. Dazu wird

P-0-2855, Ablösende Regelung Statuswortausgewertet.

Zusätzlich wird eine (vorhandene) Warnung bzw. Fehler am Baustein angezeigt.

Konfiguration der zyklischen Daten für Sercos:

Um die zyklische Kommunikation des Funktionsbausteins mit dem Antrieb aufzubauen, müssen Parameter auf die jeweiligen Ein- und Ausgangsvariablen kopiert werden. Die Konfiguration der zyklisch übertragenen Parameter erfolgt im Projektextplorer unterhalb des Sercos-Knotens. Dazu den Knoten "Drive" unterhalb des jeweiligen Gerätes anwählen.

Folgende Parameter müssen zyklisch konfiguriert werden:

Eingänge:

- S-0-0135, Antriebs-Status
- S-0-0144, Signal-Statuswort
- S-0-0552, Drehmoment-/Kraft-Istwert
- S-0-0842, Volumenstrom-Istwert
- P-0-2914, Wirksamer Ventil-Sollwert

Ausgänge:

- S-0-0134, Master-Steuerwort
- S-0-0145, Signal-Steuerwort
- S-0-0550, Drehmoment-/Kraft-Sollwert positiv
- S-0-0551, Drehmoment-/Kraft-Sollwert negativ
- S-0-0840, Volumenstrom-Sollwert

Es müssen zunächst im Reiter "Allgemeine Ein- und Ausgänge" die zyklischen Parameter konfiguriert werden.

Eingänge			Ausgänge		
Name	Datentyp	IDN,SI,SE	Name	Datentyp	IDN,SI,SE
☒ Statuswort	WORD	S-0-0135.0.0	☒ Steuerwort	WORD	S-0-0134.0.0
☒ Producer RTB word container	WORD	S-0-0144.0.0	☒ Consumer RTB word container	WORD	S-0-0145.0.0
☒ TorqueFeedbackValue 2	DINT	S-0-0552.0.0	☒ TorqueCmdPositive	DINT	S-0-0550.0.0
☒ Flow feedback value	DINT	S-0-0842.0.0	☒ TorqueCmdNegative	DINT	S-0-0551.0.0
☒ ValveCmdEffective	DINT	P-0-2914.0.0	☒ Flow command value	DINT	S-0-0840.0.0

Abb. 12-58: Konfiguration der zyklischen Parameter

Anschließend werden im Reiter "E/A-Abbild" die entsprechenden Ein- und Ausgangsvariablen mit Instanzvariablen des SPS-Programms verknüpft. Im unteren Bild ist dies eine globale Struktur HPC_DevMapping_gb mit den entsprechenden Strukturelementen. Diese wird dann im SPS-Programm auf die "InProc"- und "OutProc"-Strukturen des Funktionsbausteins kopiert.

Hier können aber auch direkt die "InProc"- und "OutProc"-Strukturen des Funktionsbausteins eingetragen werden.

Variable	Mapping	Kanal	Adresse	Typ	Standardwert	Einheit	Beschreibung
Steuervort	Application.HMC_DevMapping_gb.MasterControlWord	Steuervort	%QW20	WORD			
Consumer RTB word container	Application.HMC_DevMapping_gb.ConsumerRTBWord	Consumer RTB word container	%QW22	WORD			
TorqueCmdPositive	Application.HMC_DevMapping_gb.TorquePosCmd	TorqueCmdPositive	%QD24	DINT			
TorqueCmdNegative	Application.HMC_DevMapping_gb.TorqueNegCmd	TorqueCmdNegative	%QD26	DINT			
Flow command value	Application.HMC_DevMapping_gb.FlowCmdValue	Flow command value	%QD32	DINT			
Statuswort	Application.HMC_DevMapping_gb.HydrStatusWord	Statuswort	%tW24	WORD			
Producer RTB word container	Application.HMC_DevMapping_gb.ProducerRTBWord	Producer RTB word container	%tW26	WORD			
TorqueFeedbackValue 2	Application.HMC_DevMapping_gb.TorqueActValue	TorqueFeedbackValue 2	%tD28	DINT			
Flow feedback value	Application.HMC_DevMapping_gb.FlowActValue	Flow feedback value	%tD32	DINT			
ValveCmdEffective	Application.HMC_DevMapping_gb.ValveCmdEff	ValveCmdEffective	%tD36	DINT			

Abb. 12-59: Mappen der Ein- und Ausgangsvariablen

Parametrierung des Gerätes:

Bei Rexroth-Geräten wird die Parametrierung mit "IndraWorks Ds" empfohlen. Dort sind gerätespezifische Inbetriebnahmedialoge realisiert.

Über den Eintrag "IndraWorks Ds hier" (unterhalb des Sercos-Knoten, Geräteknoten, rechte Maustaste) kann "IndraWorks Ds" innerhalb der Engineering-Suite IndraWorks gestartet werden (Routing von der Steuerung zum Gerät erfolgt automatisch).

Folgende Parameter müssen eingestellt werden:

- Hauptbetriebsart auf Ventildirektsteuerung einstellen
 - S-0-0032 " Hauptbetriebsart" Wert "0x8052"
 - Bit 8: P-0-2853 Ablösende Regelung Steuerwort 1 Bit 0 setzen
- Parameter und Bits einstellen S-0-0144 " Signal-Statuswort"
 - ...
 - Bit 1 S-0-0849 " Volumenstrom-Statuswort" Bit 1
 - ...
 - Bit 7: S-0-0824 " Meldung Drehmoment-/Kraftsollwert erreicht" Bit 0
 - Bit 8: P-0-2855 " Ablösende Regelung Statuswort" Bit 0

Bit	Status	Quellparameter	Bitnummer
Bit 0:	☐ S-0-0000: <kein Signal>		0
Bit 1:	☐ S-0-0849: Volumenstrom-Statuswort		1
Bit 2:	☐ S-0-0000: <kein Signal>		0
Bit 3:	☐ S-0-0000: <kein Signal>		0
Bit 4:	☐ S-0-0000: <kein Signal>		0
Bit 5:	☐ S-0-0000: <kein Signal>		0
Bit 6:	☐ S-0-0000: <kein Signal>		0
Bit 7:	☐ S-0-0824: Meldung Drehmoment-/Kraftsollwert erreicht		0
Bit 8:	☐ P-0-2855: Ablösende Regelung Statuswort		0
Bit 9:	☐ S-0-0000: <kein Signal>		0

Abb. 12-60: Parametrierung Signalstatuswort S-0-0144

- Parameter und Bits einstellen S-0-0145 " Signal-Steuerwort"
 - Bit 0: P-0-2854 " Ablösende Regelung Steuerwort 2" Bit 3
 - Bit 1: P-0-2854 " Ablösende Regelung Steuerwort 2" Bit 4

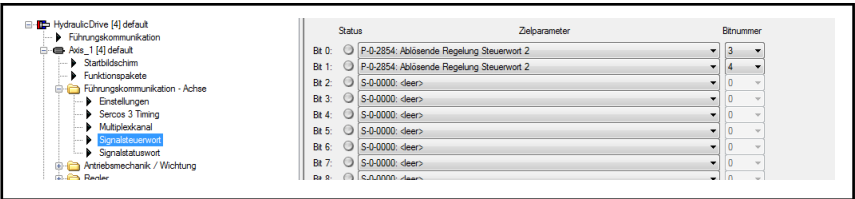


Abb. 12-61: Parametrierung Signalsteuerwort S-0-0145

- Parametrierung des Gerätes gemäß Applikation (Messsystem, Wichtung, Überwachung/Fehlerreaktion...)

 Insbesondere Überprüfen, dass die vom Funktionsbaustein vorgegebenen Sollwerte nicht durch eine gerätespezifische Parametrierung überschrieben werden,
z.B. durch Parametrierung der Analogen Eingänge des Gerätes mit internen Sollwerten.

Fehlerbehandlung Zusätzlich zu den Fehlern der intern genutzten Funktionsbausteine MB_SII-IPlcDriveType01 und IL_SIIISvcRead sind folgende Fehler des Bausteines in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#00000006	State	Fehler in der Zustandsmaschine
INPUT_RANGE_ERROR	16#000029B6	Wert des Attributes	Attribut der Parameter S-0-0840, S-0-0550, S-0-0551 oder P-0-2914 ist nicht 4 Byte dezimal mit Vorzeichen

Tab. 12-82: Fehlercodes MB_SIIIPlcDriveTorqueFlowControl

12.5.8 MB_SIIIPlcDrivePressureDirectValveControl

Kurzbeschreibung Der Funktionsbaustein MB_SIIIPlcDrivePressureDirectValveControl erweitert den Funktionsbaustein MB_SIIIPlcDriveType01 und realisiert die Betriebsart **Druckregelung/Ventildirektsteuerung**.
Wert im Betriebsartenparameter: 0x8055

 Details zu den möglichen Betriebsarten sind zu finden unter:
Rexroth HydraulicDrive HDx-20 Funktionen, Anwendungsbeschreibung, DOK-HYDRV*-HD*-20VRS**-APxx-DE-P, RD-30338-FK, Kapitel 6, Betriebsarten.

Zielsystem	Bibliothek
MLC/MTX ab 14V20	RMB_SercosIII_Util
MLD mit IndraLogic 2G ab HDx 20VRS bzw. MPx 20VRS	RMB_SercosIII_Util

Tab. 12-83: Referenztablelle

Schnittstellenbeschreibung

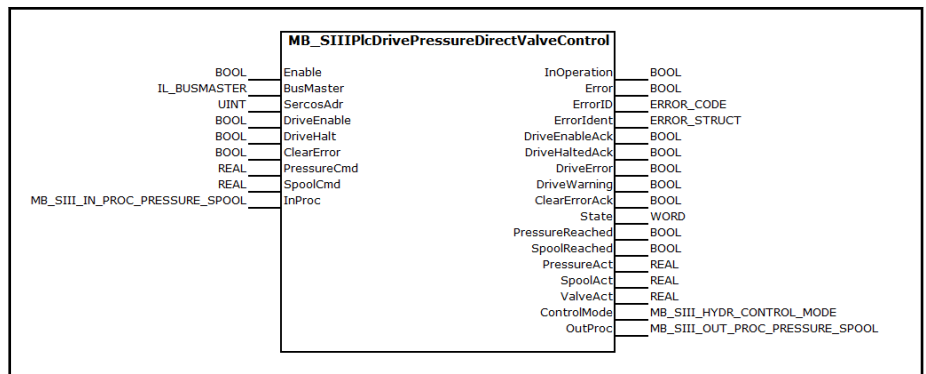


Abb. 12-62: Funktionsbaustein MB_SIIIPlcDrivePressureDirectValveControl

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (pegelgesteuert)
	Busmaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable BusMaster unterschieden. Wird nur ein Master betrieben so hat dieser die Master ID = IL_BUSMASTER_0 siehe auch IL_BUSMASTER auf Seite 429
	SercosAdr	UINT	Sercos III-Adresse des Slave-Teilnehmers
	DriveEnable	BOOL	Freigabe des Antriebes (AH/AF) S-0-0134, Master-Steuerwort, Reglerfreigabe, Bit 14
	DriveHalt	BOOL	Stoppen des Antriebes (AH) S-0-0134, Master-Steuerwort, Antrieb Halt, Bit 13
	ClearError	BOOL	Löschen eines Fehlers im Antrieb durch das Kommando Fehler löschen (S-0-0099, C0500 Reset Zustandsklasse 1). Nach gelöschtem Fehler wird die angewählte Freigabe wieder aktiviert
	PressureCmd	REAL	S-0-0800, Druck-Sollwert , Wichtung lt. Attribut
	SpoolCmd	REAL	S-0-0860, Ventil-Sollwert , Wichtung lt. Attribut
	InProc	MB_SIII_IN_PROC_PRESSURE_SPOOL	Eingangsstruktur MB_SIII_IN_PROC_PRESSURE_SPOOL , Seite 549
VAR_OUTPUT	InOperation	BOOL	Baustein arbeitet
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCT	Detaildiagnose Fehlerbehandlung , Seite 537
	DriveEnable-Ack	BOOL	Antriebsfreigabe quittiert
	DriveHaltedAck	BOOL	Antrieb Halt quittiert S-0-0135, Antriebs-Status, Bit 4

E/A-Typ	Name	Datentyp	Beschreibung
	DriveError	BOOL	Antriebsfehler S-0-0135, Antriebs-Status, Bit 13
	DriveWarning	BOOL	Antriebswarnung S-0-0135, Antriebs-Status, Bit 12
	ClearErrorAck	BOOL	Bestätigung, dass das Kommando Fehler löschen ausgeführt wurde S-0-0099, C0500 Reset Zustandsklasse 1
	State	WORD	Device Status von IL_SIIIRemoteSlaveState
	PressureReached	BOOL	Drucksollwert erreicht (S-0-0830, Druck-StatuswortBit 0)
	SpoolReached	BOOL	Ventilsollwert erreicht (S-0-0867, Ventil StatuswortBit 0)
	PressureAct	REAL	S-0-0809, Druck-Istwert , Wichtung lt. Attribut
	SpoolAct	REAL	S-0-0862, Ventil-Istwert , Wichtung lt. Attribut
	ValveAct	REAL	P-0-2914, Wirksamer Ventil-Sollwert , Wichtung lt. Attribut
	ControlMode	MB_SI-II_HYDR_CONTROL_MODE	P-0-2801, p/Q-Regler Statuswort, bit0 (1 Druckregler aktiv, 0 Ventildirektsteuerung aktiv) MB_SIII_HYDR_CONTROL_MODE, Seite 542
	OutProc	MB_SI-II_OUT_PROC_PRESSURE_SPOOL	Ausgangsstruktur MB_SIII_OUT_PROC_PRESSURE_SPOOL, Seite 550

Tab. 12-84: Schnittstelle MB_SIIIPlcDrivePressureDirectValveControl

Min-/Max und Default-Werte

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
Busmaster	IL_BUSMASTER			FALSE	Kontinuierlich
SercosAdr	UINT	1	255	1	Steigende Flanke an Enable
DriveEnable	BOOL			FALSE	Kontinuierlich
DriveHalt	BOOL			FALSE	Kontinuierlich
ClearError	BOOL			FALSE	Kontinuierlich
PressureCmd	REAL	n. def.	n. def.	0.0	Kontinuierlich
SpoolCmd	REAL	n. def.	n. def.	0.0	Kontinuierlich
InProc	MB_SIII_IN_PROC_PRESSURE_SPOOL				Kontinuierlich

Funktionsbeschreibung Der Funktionsbaustein schaltet den Antrieb in die Betriebsart "Druckregelung/Ventildirektsteuerung" (Betriebsartencode: 16#8055). Es wird ein Ventil-

sollwert am Eingang "SpoolCmd" und ein Drucksollwert am Eingang "PressureCmd" vorgegeben.

Der Ventilistwert wird am Ausgang "SpoolAct" angezeigt.

Der Druckistwert wird am Ausgang "PressureAct" angezeigt.

Die Umrechnung dieser Gleitkommagrößen (REAL) in die zyklischen Festpunktwerte des Gerätes (DINT) erfolgt intern im Baustein über die Strukturen "InProc" und "OutProc".

Das Erreichen des Ventilsollwertes wird am Ausgang "SpoolReached" (S-0-0867, Ventil StatuswortBit 0 angezeigt).

Das Erreichen des Drucksollwertes wird am Ausgang "PressureReached" (S-0-0830, Druck-StatuswortBit 0 angezeigt).

Der wirksame Ventilsollwert (P-0-2914, Wirksamer Ventil-Sollwert) wird zu Diagnosezwecken am Ausgang "ValveAct" angezeigt.

Die gerätespezifische Umschaltung zwischen Druckregelung und Ventildirektsteuerung wird am Ausgang "ControlMode" angezeigt. Dazu wird

P-0-2801, p/Q-Regler Statuswortausgewertet.

Zusätzlich wird eine (vorhandene) Warnung bzw. Fehler am Baustein angezeigt.

Konfiguration der zyklischen Daten für Sercos:

Um die zyklische Kommunikation des Funktionsbausteins mit dem Antrieb aufzubauen, müssen Parameter auf die jeweiligen Ein- und Ausgangsvariablen kopiert werden. Die Konfiguration der zyklisch übertragenen Parameter erfolgt im Projektextplorer unterhalb des Sercos-Knotens. Dazu den Knoten "Drive" unterhalb des jeweiligen Gerätes anwählen.

Folgende Parameter müssen zyklisch konfiguriert werden:

Eingänge:

- S-0-0135, Antriebs-Status
- S-0-0144, Signal-Statuswort
- S-0-0809, Druck-Istwert
- S-0-0862, Ventil-Istwert
- P-0-2914, Wirksamer Ventil-Sollwert

Ausgänge:

- S-0-0134, Master-Steuerwort
- S-0-0800, Druck-Sollwert
- S-0-0860, Ventil-Sollwert

Es müssen zunächst im Reiter "Allgemeine Ein- und Ausgänge" die zyklischen Parameter konfiguriert werden.

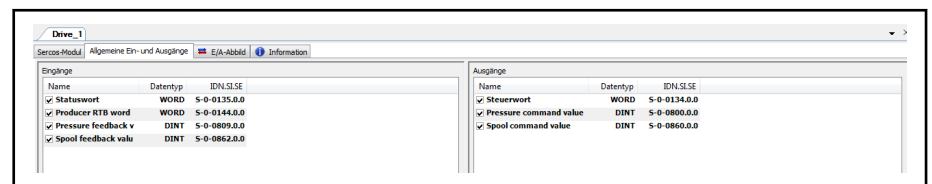


Abb. 12-63: Konfiguration der zyklischen Parameter

Anschließend werden im Reiter "E/A-Abbild" die entsprechenden Ein- und Ausgangsvariablen mit Instanzvariablen des SPS-Programms verknüpft. Im unteren Bild ist dies eine globale Struktur HPC_DevMapping_gb mit den entsprechenden Strukturelementen. Diese Struktur wird dann im SPS-Pro-

gramm auf die "InProc"- und "OutProc"-Strukturen des Funktionsbausteins kopiert.

Hier können aber auch direkt die "InProc"- und "OutProc"-Strukturen des Funktionsbausteins eingetragen werden.

Variable	Mapping	Kanal	Adresse	Typ	Standardwert	Einheit	Beschreibung
Steuervort							
Application.HMC_DevMapping_gb.MasterControlWord		Steuervort	%QW20	WORD			
Pressure command value							
Application.HMC_DevMapping_gb.PressureCmdValue		Pressure command value	%QD24	DINT			
Spool command value							
Application.HMC_DevMapping_gb.SpoolCmdValue		Spool command value	%QD28	DINT			
Statuswort							
Application.HMC_DevMapping_gb.HydrStatusWord		Statuswort	%QW24	WORD			
Producer RTB word container							
Application.HMC_DevMapping_gb.ProducerRTBWord		Producer RTB word container	%QW26	WORD			
Pressure feedback value							
Application.HMC_DevMapping_gb.PressureActValue		Pressure feedback value	%DD28	DINT			
Spool feedback value							
Application.HMC_DevMapping_gb.SpoolActValue		Spool feedback value	%DD32	DINT			

Abb. 12-64: Mappen der Ein- und Ausgangsvariablen

Parametrierung des Gerätes:



Bei Rexroth-Geräten wird die Parametrierung mit "IndraWorks Ds" empfohlen. Dort sind gerätespezifische Inbetriebnahmedialoge realisiert.

Über den Eintrag "IndraWorks Ds hier" (unterhalb des Sercos-Knoten, Geräteknoten, rechte Maustaste) kann "IndraWorks Ds" innerhalb der Engineering-Suite IndraWorks gestartet werden (Routing von der Steuerung zum Gerät erfolgt automatisch).

Folgende Parameter müssen eingestellt werden:

- Hauptbetriebsart auf Druckregelung/Ventildirektsteuerung einstellen
 - S-0-0032 " Hauptbetriebsart" Wert "0x8055"
- Parameter und Bits einstellen S-0-0144 " Signal-Statuswort"
 - Bit 0: S-0-0830 " Druck-Statuswort" Bit 1
 - ...
 - Bit 6: S-0-0867 " Ventil Statuswort" Bit 0

Status	Quellparameter	Bitnummer
Bit 0: ☐ S-0-0830: Druck-Statuswort		1
Bit 1: ☐ S-0-0000: <kein Signal>		0
Bit 2: ☐ S-0-0000: <kein Signal>		0
Bit 3: ☐ S-0-0000: <kein Signal>		0
Bit 4: ☐ S-0-0000: <kein Signal>		0
Bit 5: ☐ S-0-0000: <kein Signal>		0
Bit 6: ☒ S-0-0867: Ventil Statuswort		0
Bit 7: ☐ S-0-0000: <kein Signal>		0
Bit 8: ☐ S-0-0000: <kein Signal>		0
Bit 9: ☐ S-0-0000: <kein Signal>		0
Bit 10: ☐ S-0-0000: <kein Signal>		0
Bit 11: ☐ S-0-0000: <kein Signal>		0
Bit 12: ☐ S-0-0000: <kein Signal>		0
Bit 13: ☐ S-0-0000: <kein Signal>		0
Bit 14: ☐ S-0-0000: <kein Signal>		0
Bit 15: ☐ S-0-0000: <kein Signal>		0

Abb. 12-65: Parametrierung Signalstatuswort S-0-0144

- Parametrierung des Gerätes gemäß Applikation (Messsystem, Wichtung, Überwachung/Fehlerreaktion...)



Insbesondere Überprüfen, dass die vom Funktionsbaustein vorgegebenen Sollwerte nicht durch eine gerätespezifische Parametrierung überschrieben werden,

z.B. durch Parametrierung der Analogen Eingänge des Gerätes mit internen Sollwerten.

Fehlerbehandlung Zusätzlich zu den Fehlern der intern genutzten Funktionsbausteine MB_SIIIPlcDriveType01 und IL_SIIISvcRead sind folgende Fehler des Bausteines in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#00000006	State	Fehler in der Zustandsmaschine
INPUT_RANGE_ERROR	16#000029B7	Wert des Attributes	Attribut der Parameter S-0-0860, S-0-0800 oder P-0-2914 ist nicht 4 Byte dezimal mit Vorzeichen

Tab. 12-85: Fehlercodes MB_SIIIPlcDrivePressureDirectValveControl

12.5.9 MB_SIIIPlcDriveVelocityAltCtrl

Kurzbeschreibung Der Funktionsbaustein MB_SIIIPlcDriveVelocityAltCtrl erweitert den Funktionsbaustein MB_SIIIPlcDriveType01 und realisiert die Betriebsart **Geschwindigkeitsregelung** (mit ablösender Kraftregelung).

Wert im Betriebsartenparameter: 0x0002

Zielsystem	Bibliothek
MLC/MTX ab 14V20	RMB_SercosIII_Util
MLD mit IndraLogic 2G ab HDx 20VRS bzw. MPx 20VRS	RMB_SercosIII_Util

Tab. 12-86: Referenztablelle

Schnittstellenbeschreibung

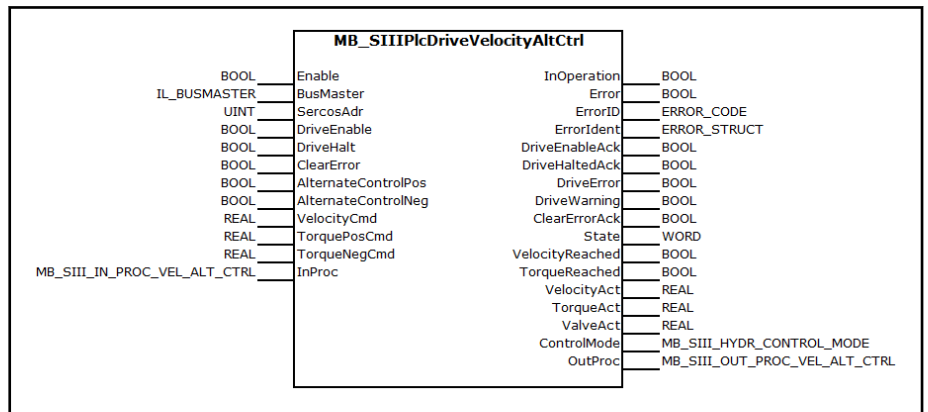


Abb. 12-66: Funktionsbaustein MB_SIIIPlcDriveVelocityAltCtrl

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (pegelgesteuert)
	Busmaster	IL_BUSMASTER	Master ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable BusMaster unterschieden. Wird nur ein Master betrieben, so hat dieser die Master ID = IL_BUSMASTER_0 siehe auch IL_BUSMASTER auf Seite 429
	SercosAdr	UINT	Sercos III-Adresse des Slave-Teilnehmers
	DriveEnable	BOOL	Freigabe des Antriebes (AH/AF) S-0-0134, Master-Steuerwort, Reglerfreigabe, Bit 14
	DriveHalt	BOOL	Stoppen des Antriebes (AH) S-0-0134, Master-Steuerwort, Antrieb Halt, Bit 13
	ClearError	BOOL	Löschen eines Fehlers im Antrieb durch das Kommando Fehler löschen (S-0-0099, C0500 Reset Zustandsklasse 1). Nach gelöschtem Fehler wird die angewählte Freigabe wieder aktiviert
	AlternateControlPos	BOOL	Ablösende Regelung in positiver Richtung aktivieren P-0-2854, " Ablösende Regelung Steuerwort 2" Bit3 (P-0-2853 Bit 0 muss gesetzt werden)
	AlternateControlNeg	BOOL	Ablösende Regelung in negativer Richtung aktivieren P-0-2854 " Ablösende Regelung Steuerwort 2" Bit4 (P-0-2853 Bit 0 muss gesetzt werden)
	VelocityCmd	REAL	S-0-0036, Geschwindigkeits-Sollwert
	TorquePosCmd	REAL	S-0-0550, Drehmoment-/Kraft-Sollwert positiv , Wichtung lt. Attribut
	TorqueNegCmd	REAL	S-0-0551, Drehmoment-/Kraft-Sollwert negativ , Wichtung lt. Attribut
	InProc	MB_SI-II_IN_PROC_VEL_ALT_CTRL	Prozess-Eingangsdaten %I anschließen MB_SIII_IN_PROC_VEL_ALT_CTRL, Seite 550
VAR_OUTPUT	InOperation	BOOL	Baustein arbeitet
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCT	Detailldiagnose Fehlerbehandlung, Seite 542
	DriveEnableAck	BOOL	Antriebsfreigabe quittiert
	DriveHaltedAck	BOOL	Antrieb Halt quittiert S-0-0135, Antriebs-Status, Bit 4
	DriveError	BOOL	Antriebsfehler S-0-0135, Antriebs-Status, Bit 13

E/A-Typ	Name	Datentyp	Beschreibung
	DriveWarning	BOOL	Antriebswarnung S-0-0135, Antriebs-Status, Bit 12
	ClearErrorAck	BOOL	Bestätigung, dass das Kommando Fehler löschen ausgeführt wurde S-0-0099, C0500 Reset Zustandsklasse 1
	State	WORD	Gerätestatus von IL_SIIIRemoteSlaveState
	VelocityReached	BOOL	Geschwindigkeitsistwert erreicht (S-0-0330)
	TorqueReached	BOOL	Kraftsollwert erreicht (S-0-0824, Meldung Drehmoment-/Kraftsollwert erreicht Bit 0)
	VelocityAct	REAL	S-0-0040, Geschwindigkeitsistwert Geber 1 , Wichtung lt. Attribut
	TorqueAct	REAL	S-0-0552, Drehmoment-/Kraft-Istwert , Wichtung lt. Attribut
	ValveAct	REAL	P-0-2914, Wirksamer Ventil-Sollwert , Wichtung lt. Attribut
	ControlMode	MB_SI-II_HYDR_CONTROL_MODE	P-0-2855, Ablösende Regelung Statuswort, Bit 0 (1 Kraftregelung Regler aktiv) MB_SIII_HYDR_CONTROL_MODE, Seite 542
	OutProc	MB_SI-II_OUT_PROC_VEL_ALT_CTRL	Ausgangsstruktur MB_SIII_OUT_PROC_VEL_ALT_CTRL, Seite 551

Tab. 12-87: Schnittstelle MB_SIIIPlcDriveVelocityAltCtrl

Min-/Max und Default-Werte

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER			FALSE	Kontinuierlich
SercosAdr	UINT	1	255	1	Steigende Flanke an Enable
DriveEnable	BOOL			FALSE	Kontinuierlich
DriveHalt	BOOL			FALSE	Kontinuierlich
ClearError	BOOL			FALSE	Kontinuierlich
AlternateControl Pos	BOOL			FALSE	Kontinuierlich
AlternateControlNeg	BOOL			FALSE	Kontinuierlich
VelocityCmd	REAL	n. def.	n. def.	0.0	Kontinuierlich
TorquePosCmd	REAL	n. def.	n. def.	0.0	Kontinuierlich
TorqueNegCmd	REAL	n. def.	n. def.	0.0	Kontinuierlich
InProc	MB_SIII_IN_PROC_VEL_ALT_CTRL				Kontinuierlich

Funktionsbeschreibung

Der Funktionsbaustein schaltet den Antrieb in die Betriebsart "Geschwindigkeitsregelung (mit ablösender Kraftregelung)" (Betriebsartencode: 16#0002). Es wird ein Geschwindigkeitssollwert am Eingang "VelocityCmd" vorgegeben.

Der Geschwindigkeitsistwert wird am Ausgang "VelocityAct" angezeigt.

Für die ablösende Regelung können am Eingang "TorquePosCmd" ein positiver Drehmoment- bzw. Kraft-Sollwert und am Eingang "TorqueNegCmd" ein negativer Drehmoment bzw. Kraft-Sollwert vorgegeben werden.

Der Drehmoment- bzw. Kraftistwert wird am Ausgang "TorqueAct" angezeigt.

Die Umrechnung dieser Gleitkommagrößen (REAL) in die zyklischen Festpunktwerte des Gerätes (DINT) erfolgt intern im Baustein über die Strukturen "InProc" und "OutProc".

Das Erreichen des Geschwindigkeitssollwertes wird am Ausgang "VelocityReached" (S-0-0330, Meldung $v_{\text{ist}} = v_{\text{soll}} \text{Bit } 0$ angezeigt).

Das Erreichen des Drehmoment- Kraftsollwertes wird am Ausgang "TorqueReached" (S-0-0824, Meldung Drehmoment-/Kraftsollwert erreichtBit 0) angezeigt.

Der wirksame Ventilsollwert (P-0-2914, Wirksamer Ventil-Sollwert) wird zu Diagnosezwecken am Ausgang "ValveAct" angezeigt.

Die gerätespezifische Umschaltung zwischen Ventildirektsteuerung und Drehmoment- bzw. Kraftregelung wird am Ausgang "ControlMode" angezeigt. Dazu wird

P-0-2855, Ablösende Regelung Statuswort ausgewertet.

Zusätzlich wird eine (vorhandene) Warnung bzw. Fehler am Baustein angezeigt.

Konfiguration der zyklischen Daten für Sercos:

Um die zyklische Kommunikation des Funktionsbausteins mit dem Antrieb aufzubauen, müssen Parameter auf die jeweiligen Ein- und Ausgangsvariablen kopiert werden. Die Konfiguration der zyklisch übertragenen Parameter erfolgt im Projektskript unterhalb des Sercos-Knotens. Dazu den Knoten "Drive" unterhalb des jeweiligen Gerätes anwählen.

Folgende Parameter müssen zyklisch konfiguriert werden:

Eingänge:

- S-0-0135, Antriebs-Status
- S-0-0144, Signal-Statuswort
- S-0-0040, Geschwindigkeitsistwert Geber 1
- S-0-0552, Drehmoment-/Kraft-Istwert
- P-0-2914, Wirksamer Ventil-Sollwert

Ausgänge:

- S-0-0134, Master-Steuerwort
- S-0-0145, Signal-Steuerwort
- S-0-0036, Geschwindigkeits-Sollwert
- S-0-0550, Drehmoment-/Kraft-Sollwert positiv
- S-0-0551, Drehmoment-/Kraft-Sollwert negativ

Es müssen zunächst im Reiter "Allgemeine Ein- und Ausgänge" die zyklischen Parameter konfiguriert werden.

Eingänge				Ausgänge			
Name	Datentyp	IDN.SLSE		Name	Datentyp	IDN.SLSE	
✓ Statuswort	WORD	S-0-0135.0.0		✓ Steuerwort	WORD	S-0-0134.0.0	
✓ Producer RTB word container	WORD	S-0-0144.0.0		✓ Consumer RTB word container	WORD	S-0-0145.0.0	
✓ TorqueFeedbackValue 2	DINT	S-0-0552.0.0		✓ TorqueCmdPositive	DINT	S-0-0550.0.0	
✓ Velocity feedback value 1	DINT	S-0-0040.0.0		✓ TorqueCmdNegative	DINT	S-0-0551.0.0	
✓ ValveCmdEffective	DINT	P-0-2914.0.0		✓ Velocity command value	DINT	S-0-0036.0.0	

Abb. 12-67: Konfiguration der zyklischen Parameter

Anschließend werden im Reiter "E/A-Abbild" die entsprechenden Ein- und Ausgangsvariablen mit Instanzvariablen des SPS-Programms verknüpft. Im unteren Bild ist dies eine globale Struktur HPC_DevMapping_gb mit den entsprechenden Strukturelementen. Diese wird dann im SPS-Programm auf die "InProc"- und "OutProc"-Strukturen des Funktionsbausteins kopiert.

Hier können aber auch direkt die "InProc"- und "OutProc"-Strukturen des Funktionsbausteins eingetragen werden.

Variable	Mapping	Kanal	Adresse	Typ	Standardwert	Einheit
Steuerwort		Steuerwort	%QW29	WORD		
Consumer RTB word container		Consumer RTB word container	%QW22	WORD		
TorqueCmdPositive		TorqueCmdPositive	%QD24	DINT		
TorqueCmdNegative		TorqueCmdNegative	%QD28	DINT		
Velocity command value		Velocity command value	%QD32	DINT		
Statuswort		Statuswort	%IW24	WORD		
Producer RTB word container		Producer RTB word container	%IW26	WORD		
TorqueFeedbackValue 2		TorqueFeedbackValue 2	%ID28	DINT		
Velocity feedback value 1		Velocity feedback value 1	%ID32	DINT		
ValveCmdEffective		ValveCmdEffective	%ID36	DINT		

Abb. 12-68: Mappen der Ein- und Ausgangsvariablen

Parametrierung des Gerätes:



Bei Rexroth-Geräten wird die Parametrierung mit "IndraWorks Ds" empfohlen. Dort sind gerätespezifische Inbetriebnahmedialoge realisiert.

Über den Eintrag "IndraWorks Ds hier" (unterhalb des Sercos-Knoten, Geräteknoten, rechte Maustaste) kann "IndraWorks Ds" innerhalb der Engineering-Suite IndraWorks gestartet werden (Routing von der Steuerung zum Gerät erfolgt automatisch).

Folgende Parameter müssen eingestellt werden:

- Hauptbetriebsart auf Ventildirektsteuerung einstellen
 - S-0-0032 " Hauptbetriebsart" Wert "0x0002"
- Parameter und Bits einstellen S-0-0144 " Signal-Statuswort"
 - ...
 - Bit 7: S-0-0824 " Meldung Drehmoment-/Kraftsollwert erreicht" Bit 0
 - Bit 8: P-0-2855 " Ablösende Regelung Statuswort" Bit 0
 - Bit 9: S-0-0330 " Meldung v_ist = v_soll" Bit 0

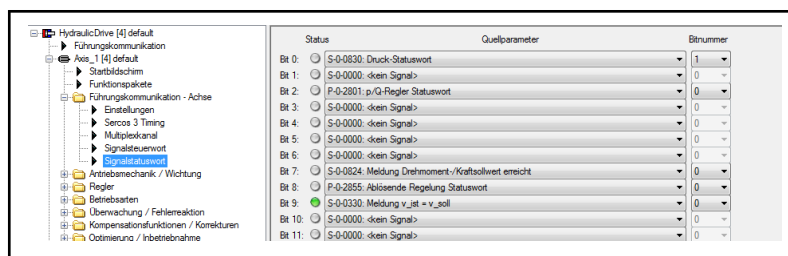


Abb. 12-69: Parametrierung Signalstatuswort S-0-0144

- Parameter und Bits einstellen S-0-0145 "Signal-Steuerwort"
 - Bit 0: P-0-2854 "Ablösende Regelung Steuerwort 2" Bit 3
 - Bit 1: P-0-2854 "Ablösende Regelung Steuerwort 2" Bit 4

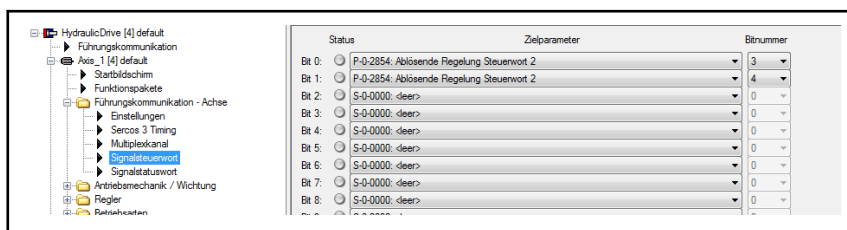


Abb. 12-70: Parametrierung Signalsteuerwort S-0-0145

- Parametrierung des Gerätes gemäß Applikation (Messsystem, Wichtung, Überwachung/Fehlerreaktion...)



Insbesondere überprüfen, dass die vom Funktionsbaustein vorgegebenen Sollwerte nicht durch eine gerätespezifische Parametrierung überschrieben werden,

z.B. durch Parametrierung der Analogen Eingänge des Gerätes mit internen Sollwerten.

Fehlerbehandlung

Zusätzlich zu den Fehlern der intern genutzten Funktionsbausteine MB_SII-IPlcDriveType01 und IL_SIIISvcRead sind folgende Fehler des Bausteines in der Fehlertabelle F_RELATED_TABLE (ERROR_TABLE = 16#0170) zu finden.

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#00000006	State	Fehler in der Zustandsmaschine
INPUT_RANGE_ERROR	16#000029B7	Wert des Attributes	Attribut der Parameter S-0-0040, S-0-0550, S-0-0551 oder P-0-2914 ist nicht 4 Byte dezimal mit Vorzeichen

Tab. 12-88: Fehlercodes MB_SIIIPlcDriveVelocityAltCtrl

12.5.10 MB_SIII_HYDR_CONTROL_MODE

Kurzbeschreibung

Der Aufzählungstyp MB_SIII_HYDR_CONTROL_MODE beschreibt die möglichen Betriebsarten für Rexroth HydraulicDrives.

Name	Wert	Beschreibung
NO_CONTROL_MODE	0	default
FLOW_CONTROL_MODE	1	Volumenstromregelung
PRESSURE_CONTROL_MODE	2	Druckregelung
SWIVEL_ANGLE_CONTROL_MODE	3	Schwenkwinkelregelung

Name	Wert	Beschreibung
VALVE_DIRECT_CONTROL_MODE	4	Ventildirektsteuerung
TORQUE_CONTROL_MODE	5	Drehmoment-/Kraftregelung
VELOCITY_CONTROL_MODE	6	Geschwindigkeitsregelung (mit ablösen-der Kraftregelung)
POSITION_CONTROL_MODE	7	Lageregelung

Tab. 12-89: Elemente des Aufzählungstyps MB_SIII_HYDR_CONTROL_MODE

12.5.11 MB_SIII_IN_PROC_FLOW

Kurzbeschreibung Die Datenstruktur MB_SIII_IN_PROC_FLOW enthält die Prozess-Eingangsdaten des Funktionsbausteins [MB_SIIIPlcDriveFlowControl auf Seite 498](#).

Strukturelement	Typ	Default-Wert	Beschreibung
Statusword	MB_SIII_STATUSWORD		Eingangsvariable %IW des Statuswortes S-0-0135, Antriebs-Status, am Element "wStatusword" anschließen. MB_SIII_STATUSWORD auf Seite 459
ConfStatus	WORD		Frei konfigurierbares Signalstatuswort S-0-0144, Signal-Statuswort Dort wird im Bit 1 der Parameter S-0-0849, Volumenstrom-Statuswort, mit dem Bit 1 konfiguriert. (Anzeige Volumenstrom erreicht)
Flow	DINT	16#0	Eingangsvariable %IW des aktuellen Volumenstromistwertes S-0-0842, Volumenstrom-Istwert
ValveEff	DINT	16#0	Eingangsvariable %IW des wirksamen Ventilsollwertes P-0-2914, Wirksamer Ventil-Sollwert

Tab. 12-90: Elemente der Struktur MB_SIII_IN_PROC_FLOW

12.5.12 MB_SIII_OUT_PROC_FLOW

Kurzbeschreibung Die Datenstruktur MB_SIII_OUT_PROC_FLOW enthält die Prozess-Ausgangsdaten des Funktionsbausteins [MB_SIIIPlcDriveFlowControl auf Seite 498](#).

Strukturelement	Typ	Default-Wert	Beschreibung
Controlword	MB_SIII_CONTROLWORD		Ausgangsvariable %QW des Steuerwortes S-0-0134, Master-Steuerwort am Element "wControlword" anschließen. MB_SIII_CONTROLWORD auf Seite 458
Flow	DINT	16#0	Ausgangsvariable %QW des Volumenstromsollwertes S-0-0840, Volumenstrom-Sollwert

Tab. 12-91: Elemente der Struktur MB_SIII_OUT_PROC_FLOW

12.5.13 MB_SIII_IN_PROC_PRESSURE_FLOW

Kurzbeschreibung Die Datenstruktur MB_SIII_IN_PROC_PRESSURE_FLOW enthält die Prozess-Eingangsdaten des Funktionsbausteins [MB_SIIIPlcDrivePressureFlowControl auf Seite 503](#).

Strukturelement	Typ	Default-Wert	Beschreibung
Statusword	MB_SIII_STATUSWORD		Eingangsvariable %IW des Statuswortes S-0-0135, Antriebs-Status, am Element "wStatusword" anschließen. MB_SIII_STATUSWORD auf Seite 459
ConfStatus	WORD		Frei konfigurierbares Signalstatuswort S-0-0144, Signal-Statuswort Dort wird im Bit 0 der Parameter S-0-0830, Druck-Statuswort, mit dem Bit 1 konfiguriert. (Anzeige Druck erreicht) Dort wird im Bit 1 der Parameter S-0-0849, Volumenstrom-Statuswort, mit dem Bit 1 konfiguriert. (Anzeige Volumenstrom erreicht) Funktionspaket Pumpenregelung: Dort wird im Bit 2 der Parameter P-0-2951, Pumpenregler Statuswort, mit dem Bit 0 konfiguriert. (Anzeige Druck oder Schwenkwinkelregelung aktiv) Funktionspaket P/Q-Regelung: Dort wird im Bit 2 der Parameter P-0-2801, p/Q-Regler Statuswort, mit dem Bit 0 konfiguriert. (Anzeige Druck oder Volumenstromregelung aktiv)
Pressure	DINT	16#0	Eingangsvariable %IW des aktuellen Druckistwertes S-0-0809, Druck-Istwert
Flow	DINT	16#0	Eingangsvariable %IW des aktuellen Volumenstromistwertes S-0-0842, Volumenstrom-Istwert
ValveEff	DINT	16#0	Eingangsvariable %IW des wirksamen Ventilsollwertes P-0-2914, Wirksamer Ventil-Sollwert

Tab. 12-92: Elemente der Struktur MB_SIII_IN_PROC_PRESSURE_FLOW

12.5.14 MB_SIII_OUT_PROC_PRESSURE_FLOW

Kurzbeschreibung Die Datenstruktur MB_SIII_OUT_PROC_PRESSURE_FLOW enthält die Prozess-Ausgangsdaten des Funktionsbausteins [MB_SIIIPlcDrivePressureFlow-Control auf Seite 503](#).

Strukturelement	Typ	Default-Wert	Beschreibung
Controlword	MB_SIII_CONTROLWORD		Ausgangsvariable %QW des Steuerwortes S-0-0134, Master-Steuerwort am Element "wControlword" anschließen. MB_SIII_CONTROLWORD auf Seite 458
Pressure	DINT	16#0	Ausgangsvariable %QW des Drucksollwertes S-0-0800, Druck-Sollwert
Flow	DINT	16#0	Ausgangsvariable %QW des Volumenstromsollwertes S-0-0840, Volumenstrom-Sollwert

Tab. 12-93: Elemente der Struktur MB_SIII_OUT_PROC_PRESSURE_FLOW

12.5.15 MB_SIII_IN_PROC_SWIVEL

Kurzbeschreibung Die Datenstruktur MB_SIII_IN_PROC_SWIVEL enthält die Prozess-Eingangsdaten des Funktionsbausteins [MB_SIIIPlcDrivePressureSwivelAngle-Control auf Seite 508](#).

Strukturelement	Typ	Default-Wert	Beschreibung
Statusword	MB_SIII_STATUSWORD		Eingangsvariable %IW des Statuswortes S-0-0135, Antriebs-Status, am Element "wStatusword" anschließen. MB_SIII_STATUSWORD auf Seite 459
ConfStatus	WORD		Frei konfigurierbares Signalstatuswort S-0-0144, Signal-Statuswort Dort wird im Bit 0 der Parameter S-0-0830, Druck-Statuswort, mit dem Bit 1 konfiguriert. (Anzeige Druck erreicht) Dort wird im Bit 3 der Parameter S-0-0887, Schwenkwinkel Statuswort, mit dem Bit 1 konfiguriert. (Anzeige Schwenkwinkel erreicht) Dort wird im Bit 4 der Parameter P-0-2951, Pumpenregler Statuswort, mit dem Bit 0 konfiguriert. (Anzeige Druck oder Schwenkwinkelregelung aktiv) Dort wird im Bit 5 der Parameter P-0-2951, Pumpenregler Statuswort, mit dem Bit 1 konfiguriert. (Anzeige Druck oder Schwenkwinkelregelung aktiv)
Pressure	DINT	16#0	Eingangsvariable %IW des aktuellen Druckistwertes S-0-0809, Druck-Istwert
SwivelAngle	DINT	16#0	Eingangsvariable %IW des aktuellen Schwenkwinkelwertes S-0-0882, Schwenkwinkel-Istwert
ValveEff	DINT	16#0	Eingangsvariable %IW des wirksamen Ventilsollwertes P-0-2914, Wirksamer Ventil-Sollwert

Tab. 12-94: Elemente der Struktur MB_SIII_IN_PROC_SWIVEL

12.5.16 MB_SIII_OUT_PROC_SWIVEL

Kurzbeschreibung Die Datenstruktur MB_SIII_OUT_PROC_SWIVEL enthält die Prozess-Ausgangsdaten des Funktionsbausteins [MB_SIIIPlcDrivePressureSwivelAngle-Control auf Seite 508](#).

Strukturelement	Typ	Default-Wert	Beschreibung
Controlword	MB_SIII_CONTROLWORD		Ausgangsvariable %QW des Steuerwortes S-0-0134, Master-Steuerwort am Element "wControlword" anschließen. MB_SIII_CONTROLWORD auf Seite 458
Pressure	DINT	16#0	Ausgangsvariable %QW des Drucksollwertes S-0-0800, Druck-Sollwert
SwivelAngle	DINT	16#0	Ausgangsvariable %QW des Schwenkwinkelsollwertes S-0-0880, Schwenkwinkel-Sollwert

Tab. 12-95: Elemente der Struktur MB_SIII_OUT_PROC_SWIVEL

12.5.17 MB_SIII_IN_PROC_SWIVEL_SLAVE

Kurzbeschreibung Die Datenstruktur MB_SIII_IN_PROC_SWIVEL_SLAVE enthält die Prozess-Eingangsdaten des Funktionsbausteins [MB_SIIIPlcDrivePressureSwivelAngleSlave auf Seite 513](#).

Strukturelement	Typ	Default-Wert	Beschreibung
Statusword	MB_SIII_STATUSWORD		Eingangsvariable %IW des Statuswortes S-0-0135, Antriebs-Status, am Element "wStatusword" anschließen. MB_SIII_STATUSWORD auf Seite 459
ConfStatus	WORD		Frei konfigurierbares Signalstatuswort S-0-0144, Signal-Statuswort Dort wird im Bit 3 der Parameter S-0-0887, Schwenkwinkel Statuswort, mit dem Bit 1 konfiguriert. (Anzeige Schwenkwinkel erreicht)
SwivelAngle	DINT	16#0	Eingangsvariable %IW des aktuellen Schwenkwinkelwertes S-0-0882, Schwenkwinkel-Istwert
ValveEff	DINT	16#0	Eingangsvariable %IW des wirksamen Ventilsollwertes P-0-2914, Wirksamer Ventilsollwert

Tab. 12-96: Elemente der Struktur MB_SIII_IN_PROC_SWIVEL

12.5.18 MB_SIII_OUT_PROC_SWIVEL_SLAVE

Kurzbeschreibung Die Datenstruktur MB_SIII_OUT_PROC_SWIVEL_SLAVE enthält die Prozess-Ausgangsdaten des Funktionsbausteins [MB_SIIIPlcDrivePressureSwivelAngleSlave auf Seite 513](#).

Strukturelement	Typ	Default-Wert	Beschreibung
Controlword	MB_SIII_CONTROLWORD		Ausgangsvariable %QW des Steuerwortes S-0-0134, Master-Steuerwort am Element "wControlword" anschließen. MB_SIII_CONTROLWORD auf Seite 458
Pressure	DINT	16#0	Ausgangsvariable %QW des Drucksollwertes S-0-0800, Druck-Sollwert
SwivelAngle	DINT	16#0	Ausgangsvariable %QW des Schwenkwinkelsollwertes S-0-0880, Schwenkwinkel-Sollwert

Tab. 12-97: Elemente der Struktur MB_SIII_OUT_PROC_SWIVEL_SLAVE

12.5.19 MB_SIII_IN_PROC_SPOOL

Kurzbeschreibung Die Datenstruktur MB_SIII_IN_PROC_SPOOL enthält die Prozess-Eingangsdaten des Funktionsbausteins [MB_SIIIPlcDriveDirectValveControl auf Seite 517](#).

Strukturelement	Typ	Default-Wert	Beschreibung
Statusword	MB_SIII_STATUSWORD		Eingangsvariable %IW des Statuswortes S-0-0135, Antriebs-Status, am Element "wStatusword" anschließen MB_SIII_STATUSWORD auf Seite 459
ConfStatus	WORD		Frei konfigurierbares Signalstatuswort S-0-0144, Signal-Statuswort. Es wird S-0-0867/Bit 0 (ValveSoll = Valvelst), konfiguriert
Spool	DINT	16#0	Eingangsvariable %ID des Ventilistwertes S-0-0862 Ventil-Istwert
ValveEff	DINT	16#0	Eingangsvariable %IW des wirksamen Ventil-sollwertes P-0-2914, Wirksamer Ventil-Sollwert

Tab. 12-98: Elemente der Struktur MB_SIII_IN_PROC_SPOOL

12.5.20 MB_SIII_OUT_PROC_SPOOL

Kurzbeschreibung Die Datenstruktur MB_SIII_OUT_PROC_SPOOL enthält die Prozess-Ausgangsdaten des Funktionsbausteins [MB_SIIIPlcDriveDirectValveControl auf Seite 517](#).

Strukturelement	Typ	Default-Wert	Beschreibung
Controlword	MB_SIII_CONTROLWORD		Ausgangsvariable %QW des Steuerwortes S-0-0134, Master-Steuerwort am Element "wControlword" anschließen MB_SIII_CONTROLWORD auf Seite 458
Spool	DINT	16#0	Ausgangsvariable %QD des Ventil-Sollwertes S-0-0860 Ventil-Sollwert

Tab. 12-99: Elemente der Struktur MB_SIII_OUT_PROC_SPOOL

12.5.21 MB_SIII_IN_PROC_TORQUE_SPOOL

Kurzbeschreibung Die Datenstruktur MB_SIII_IN_PROC_TORQUE_SPOOL enthält die Prozess-Eingangsdaten des Funktionsbausteins [MB_SIIIPlcDriveTorqueDirectValveControl](#) auf Seite 521.

Strukturelement	Typ	Default-Wert	Beschreibung
Statusword	MB_SIII_STATUSWORD		Eingangsvariable %IW des Statuswortes S-0-0135, Antriebs-Status, am Element "wStatusword" anschließen MB_SIII_STATUSWORD auf Seite 459
ConfStatus	WORD		Frei konfigurierbares Signalstatuswort S-0-0144, Signal-Statuswort . Es wird S-0-0824 Bit 0 (Torque) und S-0-0867/Bit 0 (Ventil _{Soll} = Ventil _{Ist}), konfiguriert
Torque	DINT	16#0	S-0-0552 Drehmoment-/Kraft-Istwert
Spool	DINT	16#0	S-0-0862 Ventil-Istwert
ValveEff	DINT	16#0	Eingangsvariable %IW des wirksamen Ventilsollwertes P-0-2914, Wirksamer Ventil-Sollwert

Tab. 12-100: Elemente der Struktur MB_SIII_IN_PROC_TORQUE_SPOOL

12.5.22 MB_SIII_OUT_PROC_TORQUE_SPOOL

Kurzbeschreibung Die Datenstruktur MB_SIII_OUT_PROC_TORQUE_SPOOL enthält die Prozess-Ausgangsdaten des Funktionsbausteins [MB_SIIIPlcDriveTorqueDirectValveControl](#) auf Seite 521.

Strukturelement	Typ	Default-Wert	Beschreibung
Controlword	MB_SIII_CONTROLWORD		Ausgangsvariable %QW des Steuerwortes S-0-0134, Master-Steuerwort am Element "wControlword" anschließen MB_SIII_CONTROLWORD auf Seite 458
ConfControl	WORD		S-0-0145 Signal-Steuerwort. Es wird im P-0-2854 Bit 3 und Bit 4 eingetragen
TorquePos	DINT		S-0-0550 Drehmoment-/Kraft-Sollwert positiv
TorqueNeg	DINT		S-0-0551 Drehmoment-/Kraft-Sollwert negativ
Spool	DINT		S-0-0860 Ventil-Sollwert

Tab. 12-101: Elemente der Struktur MB_SIII_OUT_PROC_TORQUE_SPOOL

12.5.23 MB_SIII_IN_PROC_TORQUE_FLOW

Kurzbeschreibung Die Datenstruktur MB_SIII_IN_PROC_TORQUE_FLOW enthält die Prozess-Eingangsdaten des Funktionsbausteins [MB_SIIIPlcDriveTorqueFlowControl](#) auf Seite 527.

Strukturelement	Typ	Default-Wert	Beschreibung
Statusword	MB_SIII_STATUSWORD		Eingangsvariable %IW des Statuswortes S-0-0135, Antriebs-Status, am Element "wStatusword" anschließen MB_SIII_STATUSWORD auf Seite 459
ConfStatus	WORD		Frei konfigurierbares Signalstatuswort S-0-0144, Signal-Statuswort . Es wird S-0-0824 Bit 0 (Torque) und S-0-0867/Bit 0 (Ventil _{Soll} = Ventil _{Ist}), konfiguriert
Torque	DINT	16#0	S-0-0552 Drehmoment-/Kraft-Istwert
Spool	DINT	16#0	S-0-0862 Ventil-Istwert
ValveEff	DINT	16#0	Eingangsvariable %IW des wirksamen Ventil-sollwertes P-0-2914, Wirksamer Ventil-Sollwert

Tab. 12-102: Elemente der Struktur MB_SIII_IN_PROC_TORQUE_FLOW

12.5.24 MB_SIII_OUT_PROC_TORQUE_FLOW

Kurzbeschreibung Die Datenstruktur MB_SIII_OUT_PROC_TORQUE_FLOW enthält die Prozess-Ausgangsdaten des Funktionsbausteins [MB_SIIIPlcDriveTorqueFlow-Control auf Seite 527](#).

Strukturelement	Typ	Default-Wert	Beschreibung
Controlword	MB_SIII_CONTROLWORD		Ausgangsvariable %QW des Steuerwortes S-0-0134, Master-Steuerwort am Element "wControlword" anschließen MB_SIII_CONTROLWORD auf Seite 458
ConfControl	WORD		S-0-0145 Signal-Steuerwort. Es wird in P-0-2854 Bit 3 und Bit 4 eingetragen
TorquePos	DINT	16#0	S-0-0550 Drehmoment-/Kraft-Sollwert positiv
TorqueNeg	DINT	16#0	S-0-0551 Drehmoment-/Kraft-Sollwert negativ
Flow	DINT	16#0	S-0-0840 Volumenstrom-Sollwert

Tab. 12-103: Elemente der Struktur MB_SIII_OUT_PROC_TORQUE_FLOW

12.5.25 MB_SIII_IN_PROC_PRESSURE_SPOOL

Kurzbeschreibung Die Datenstruktur MB_SIII_IN_PROC_PRESSURE_SPOOL enthält die Prozess-Eingangsdaten des Funktionsbausteins [MB_SIIIPlcDrivePressureDirect-ValveControl auf Seite 532](#).

Strukturelement	Typ	Default-Wert	Beschreibung
Statusword	MB_SIII_STATUSWORD		Eingangsvariable %IW des Statuswortes S-0-0135, Antriebs-Status, am Element "wStatusword" anschließen MB_SIII_STATUSWORD auf Seite 459
ConfStatus	WORD		Frei konfigurierbares Signalstatuswort S-0-0144, Signal-Statuswort . Es wird S-0-0867/Bit 0 (ValveSoll = ValveIst) konfiguriert

Strukturelement	Typ	Default-Wert	Beschreibung
Pressure	DINT	16#0	Eingangsvariable %ID des Druckistwertes S-0-0809 Druck-Istwert
Spool	DINT	16#0	Eingangsvariable %ID des Ventilistwertes S-0-0862 Ventil-Istwert
ValveEff	DINT	16#0	Eingangsvariable %IW des wirksamen Ventilsollwertes P-0-2914, Wirksamer Ventilsollwert

Tab. 12-104: Elemente der Struktur MB_SIII_IN_PROC_PRESSURE_SPOOL

12.5.26 MB_SIII_OUT_PROC_PRESSURE_SPOOL

Kurzbeschreibung Die Datenstruktur MB_SIII_OUT_PROC_PRESSURE_SPOOL enthält die Prozess-Ausgangsdaten des Funktionsbausteins [MB_SIIIPlcDrivePressure-DirectValveControl](#) auf Seite 532.

Strukturelement	Typ	Default-Wert	Beschreibung
Controlword	MB_SIII_CONTROLWORD		Ausgangsvariable %QW des Steuerwortes S-0-0134, Master-Steuerwort am Element "wControlword" anschließen MB_SIII_CONTROLWORD auf Seite 458
Pressure	DINT	16#0	Ausgangsvariable %QD des Druck-Sollwertes S-0-0800 Druck-Sollwert
Spool	DINT	16#0	Ausgangsvariable %QD des Ventil-Sollwertes S-0-0860 Ventil-Sollwert

Tab. 12-105: Elemente der Struktur MB_SIII_OUT_PROC_PRESSURE_SPOOL

12.5.27 MB_SIII_IN_PROC_VEL_ALT_CTRL

Kurzbeschreibung Die Datenstruktur MB_SIII_IN_PROC_VEL_ALT_CTRL enthält die Prozess-Eingangsdaten des Funktionsbausteins [MB_SIIIPlcDriveVelocityAltCtrl](#) auf Seite 537.

Strukturelement	Typ	Default-Wert	Beschreibung
Statusword	MB_SIII_STATUSWORD		Eingangsvariable %IW des Statuswortes S-0-0135, Antriebs-Status, am Element "wStatusword" anschließen MB_SIII_STATUSWORD auf Seite 459
ConfStatus	WORD		Frei konfigurierbares Signalstatuswort S-0-0144, Signal-Statuswort . Es wird S-0-0824 Bit 0 (Torque) und S-0-0867/Bit 0 (Ventil _{Soll} = Ventil _{Ist}), konfiguriert
Velocity	DINT	16#0	S-0-0040 Geschwindigkeitsistwert Geber 1
Torque	DINT	16#0	S-0-0552 Drehmoment-/Kraft-Istwert
ValveEff	DINT	16#0	Eingangsvariable %IW des wirksamen Ventilsollwertes P-0-2914, Wirksamer Ventilsollwert

Tab. 12-106: Elemente der Struktur MB_SIII_IN_PROC_VEL_ALT_CTRL

12.5.28 MB_SIII_OUT_PROC_VEL_ALT_CTRL

Kurzbeschreibung Die Datenstruktur MB_SIII_OUT_PROC_VEL_ALT_CTRL enthält die Prozess-Ausgangsdaten des Funktionsbausteins [MB_SIIIPlcDriveVelocityAltCtrl](#) auf Seite 537.

Strukturelement	Typ	Default-Wert	Beschreibung
Controlword	MB_SIII_CONTROLWORD		Ausgangsvariable %QW des Steuerwortes S-0-0134, Master-Steuerwort am Element "wControlword" anschließen MB_SIII_CONTROLWORD auf Seite 458
ConfControl	WORD		S-0-0145 Signal-Steuerwort. Es wird in P-0-2854 Bit 3 und Bit 4 eingetragen
TorquePos	DINT	16#0	S-0-0550 Drehmoment-/Kraft-Sollwert positiv
TorqueNeg	DINT	16#0	S-0-0551 Drehmoment-/Kraft-Sollwert negativ
VelocityCmd	DINT	16#0	S-0-0036 Geschwindigkeits-Sollwert

Tab. 12-107: Elemente der Struktur MB_SIII_OUT_PROC_VEL_ALT_CTRL

12.6 RexrothSytronix

12.6.1 MB_SIIIDRn5020

Kurzbeschreibung Der Funktionsbaustein MB_SIIIDRn5020 ermöglicht die Druck- und Schwenkwinkelsteuerung des auf EFC 5610 basierenden Sytronix DRn 5020 von Rexroth-Steuerungen über Sercos-Kommunikation. Mit diesem Funktionsbaustein können Druck-, Schwenkwinkel- und maximale Durchflussgrenzwerte eingestellt, die Steuerung aktiviert/gestoppt, Fehler des Antriebs zurückgesetzt werden. Der Funktionsbaustein beinhaltet auch die Überwachung verschiedener Antriebsfunktionen, die in den folgenden Abschnitten beschrieben werden.

Bibliothek:	RMB_SercosIII_Util
Gruppe:	"RexrothSytronix"
Zielsystem:	MTX 15VRS MLC 15VRS, Funktionspaket SPS MLD mit MPx20

Tab. 12-108: Referenztable MB_SIIIDRn5020

Schnittstellenbeschreibung

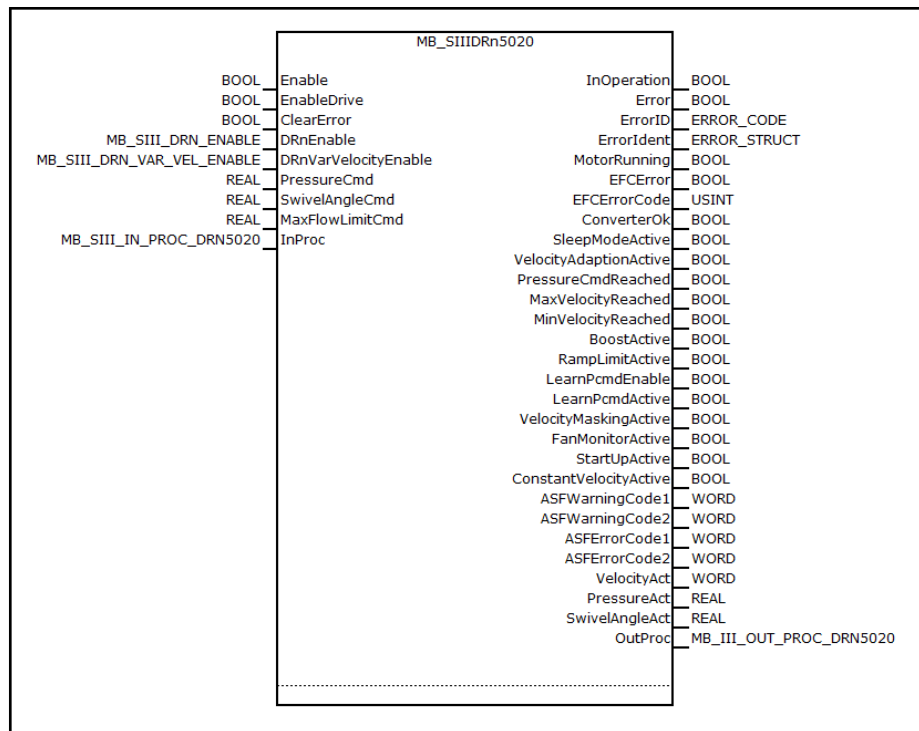


Abb. 12-71: Funktionsbaustein Schnittstelle MB_SIIDRn5020

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins
	EnableDrive	BOOL	TRUE - Antriebsfreigabe FALSE - Antrieb deaktivieren
	ClearError	BOOL	Antriebsfehler Reset-Befehl ausführen
	DRnEnable	MB_SIII_DRN_ENABLE	Freigabe DRn-Funktion MB_SIII_DRN_ENABLE auf Seite 561
	DRnVarVelocityEnable	MB_SIII_DRN_VAR_VEL_ENABLE	DRn variable Geschwindigkeitsauswahl MB_SIII_DRN_VAR_VEL_ENABLE auf Seite 562
	PressureCmd	REAL	Drucksollwert in bar
	SwivelAngleCmd	REAL	Schwenkwinkelsollwert in %
	MaxFlowLimitCmd	REAL	Maximaler Durchflussgrenzwert in U/min
	InProc	MB_SIII_IN_PROC_DRN5020	Eingangsstruktur für den Funktionsbaustein
VAR_OUTPUT	InOperation	BOOL	TRUE - Funktionsbaustein ist in Betrieb
	Error	BOOL	TRUE - Funktionsbausteinbetrieb gestoppt wegen Fehler
	ErrorID	ERROR_CODE	Fehlercode für den aktuellen Fehler
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerinformation
	MotorRunning	BOOL	Motorstatus TRUE - Motor dreht FALSE - Motor ist drehmomentfrei

E/A-Typ	Name	Datentyp	Beschreibung
	EFCError	BOOL	TRUE - Fehler aktiv am EFC FALSE - Kein Fehler
	EFCErrorCode	USINT	Fehlercode für EFC-Fehler
	ConverterOk	BOOL	TRUE - Konverter Ok FALSE - Konverter nicht Ok
	SleepModeActive	BOOL	TRUE - Ruhemodus aktiv FALSE - Inaktiv
	VelocityAdaptationActive	BOOL	Status der Geschwindigkeitsanpassung TRUE - Aktiv FALSE - Inaktiv
	PressureCmdReached	BOOL	TRUE - Aktueller Druck erreicht Sollwert und ist im erlaubten Bereich FALSE - Aktueller Druck ist nicht im erlaubten Bereich
	MaxVelocityReached	BOOL	Geschwindigkeitsstatus TRUE - Maximale Geschwindigkeit erreicht FALSE - Nicht erreicht
	MinVelocityReached	BOOL	Geschwindigkeitsstatus TRUE - Minimale Geschwindigkeit erreicht FALSE - Nicht erreicht
	BoostActive	BOOL	Status der Boost-Funktion TRUE - Boost aktiv FALSE - Boost inaktiv
	RampLimitActive	BOOL	Status der Rampenbegrenzung TRUE - Aktiv FALSE - Inaktiv
	LearnPcmdEnable	BOOL	TRUE - Lernen von p_cmd aktiviert FALSE - Keine Freigabe
	LearnPcmdActive	BOOL	TRUE - Lernen von p_cmd aktiv FALSE - Inaktiv
	VelocityMaskingActive	BOOL	TRUE - Geschwindigkeitsmaskierung aktiv FALSE - Inaktiv
	FanMonitorActive	BOOL	TRUE - Motorleistungsreduzierung (Lüfterüberwachung) aktiv FALSE - Inaktiv
	StartUpActive	BOOL	TRUE - Inbetriebnahme aktiv FALSE - Inaktiv
	ConstantVelocityActive	BOOL	TRUE - Konstante Drehzahl aktiv FALSE - Inaktiv

E/A-Typ	Name	Datentyp	Beschreibung
	ASFWarningCode1	WORD	Warncode 1
	ASFWarningCode2	WORD	Warncode 2
	ASFErrorCode1	WORD	Fehlercode 1
	ASFErrorCode2	WORD	Fehlercode 2
	VelocityAct	REAL	Aktuelle Geschwindigkeit in U/min
	PressureAct	REAL	Aktueller Druck in bar
	SwivelAngleAct	REAL	Aktueller Schwenkwinkel in %
	OutProc	MB_SI-II_OUT_PROC_DRN5020	Ausgangsstruktur für den Funktionsbaustein

Tab. 12-109: Schnittstelle MB_SIIIDRn5020

Min-/Max und Default-Werte der Eingänge

Name	Datentyp	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
StartCommand	BOOL			FALSE	Kontinuierlich
ClearError	BOOL			FALSE	Kontinuierlich
DRnEnable	MB_SIII_DRN_ENABLE			DRN_DISABLE	Kontinuierlich
DRnVarVelocityEnable	MB_SI-II_DRN_VAR_VEL_ENABLE			DRN_VAR_VEL_DISABLE	Kontinuierlich
PressureCmd	REAL	0,0	1000,0	0,0	Kontinuierlich
SwivelAngleCmd	REAL	0,00	100,00	0,00	Kontinuierlich
MaxFlowLimitCmd	REAL	0,0	10000,0	0,0	Kontinuierlich
InProc	MB_SIII_IN_PROC_DRn5020				Kontinuierlich

Tab. 12-110: Min-/Max und Default-Werte der Eingänge MB_SIIIDRn5020

Funktionsbaustein Eingänge/ Ausgänge - Parameter-Mapping

E/A-Typ	Name	Parameter
VAR_INPUT	EnableDrive	Parameter S-0-0134 Bit 13, 14, 15
	ClearError	S-0-0099 (unter Verwendung von IL_SIIICommand)
	DRnEnable	Parameter F0.20 - Bit 0
	DRnVarVelocityEnable	Parameter F0.20 - Bit 1
	PressureCmd	Parameter F0.21
	SwivelAngleCmd	Parameter F0.22
	MaxFlowLimitCmd	Parameter F0.23
VAR_OUTPUT	MotorRunning	Parameter H0.01 - Bit 1
	EFCErrror	Parameter H0.01 - Bit 7
	EFCErrrorCode	Parameter H0.01 - Bits 8-15

E/A-Typ	Name	Parameter
	ConverterOk	Parameter H0.02 – Bit 2
	SleepModeActive	Parameter H0.02 – Bit 1
	VelocityAdaptationActive	Parameter d0.80 - Bit 1
	PressureCmdReached	Parameter d0.80 - Bit 4
	MaxVelocityReached	Parameter d0.80 - Bit 5
	MinVelocityReached	Parameter d0.80 - Bit 6
	BoostActive	Parameter d0.80 - Bit 7
	RampLimitActive	Parameter d0.80 - Bit 8
	LearnPcmdEnable	Parameter d0.80 - Bit 9
	LearnPcmdActive	Parameter d0.80 - Bit 10
	VelocityMaskingActive	Parameter d0.80 - Bit 11
	FanMonitorActive	Parameter d0.80 - Bit 12
	StartUpActive	Parameter d0.80 - Bit 13
	ConstantVelocityActive	Parameter d0.80 - Bit 14
	ASFWarningCode1	Parameter d0.86
	ASFWarningCode2	Parameter d0.87
	ASFErrorCode1	Parameter d0.88
	ASFErrorCode2	Parameter d0.89
	VelocityAct	Parameter d0.83
	PressureAct	Parameter d0.84
	SwivelAngleAct	Parameter d0.85

Tab. 12-111: Funktionsbaustein Eingänge/Ausgänge - Parameter-Mapping

Funktionsbeschreibung

Der Funktionsbaustein MB_SIIIDRn5020 verwendet den zyklischen Lese-/Schreibkanal für den Datenaustausch zwischen der SPS und dem Antrieb. Alle erforderlichen Parameter werden im AT/MDT des Antriebs konfiguriert. Das Kommunikationsprotokoll zwischen SPS und Antrieb ist Sercos. EFC-Parameter müssen für die Kommunikation entsprechend eingestellt werden. Wenn der Freigabeeingang während des Betriebs zurückgesetzt wird, werden die an den Antrieb gesendeten Steuerwortdaten und Sollwerte nicht zurückgesetzt. Sie spiegeln ihre zuletzt aktualisierten Werte wider.

In diesem Funktionsbaustein stehen die folgenden Steuerungsfunktionen zur Verfügung:

- Start/Stopp-Befehl
- Löschen eines Antriebsfehlers
- Aktivieren/deaktivieren DRn
- Aktivieren/Deaktivieren der Geschwindigkeitsanpassung
- Einstellen des Drucksollwertes
- Einstellen des Schwenkwinkels
- Einstellen des maximalen Durchflussgrenzwertes

In diesem Funktionsbaustein stehen die folgenden Überwachungsfunktionen zur Verfügung:

- Motorstatus
- Fehler EFC
- Fehlercode EFC
- Status des Konverters
- Status des Ruhemodus
- Status der Geschwindigkeitsanpassung
- Aktueller Druckstatus gegen Solldruck
- Maximale Drehzahl erreicht Status
- Minimale Drehzahl erreicht Status
- Status der Boost-Funktion
- Status der Rampenbegrenzung
- Lernen von p_cmd aktiviert
- Lernen von p_cmd aktiv
- Status Geschwindigkeitsmaskierung
- Status der Motorleistungsreduzierung (Lüfterüberwachung)
- Status der Startsequenz
- Status konstante Drehzahl aktiv
- Tatsächliche Geschwindigkeit
- Tatsächlicher Druck
- Tatsächlicher Schwenkwinkel
- ASF Warncode und verschiedene Warnungen
- ASF Fehlercode und verschiedene Fehler

Weitere Informationen finden Sie auf der Webseite [Druckregelsysteme Sytronix DRn 5020](#).

Konfiguration der Echtzeit-Eingabe (AT) und Echtzeit-Ausgabe (MDT)

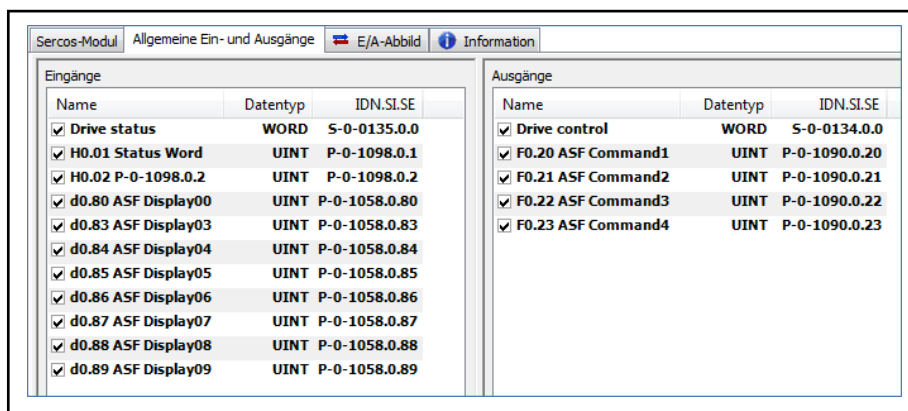


Abb. 12-72: AT- und MDT-Konfiguration für EFC

Fehlerbehandlung

Der Funktionsbaustein kann durch ungültige Eingaben einen Fehlerzustand erreichen. Während dieser Zeit meldet der Funktionsbaustein einen Fehler und stoppt den Betrieb. Die Steuerwortbits und die zuvor eingestellten Sollwerte **werden aber nicht** gemäß den Systemanforderungen für DRn5020 **zurückgesetzt**. Der Benutzer muss die Eingabewerte so korrigieren, dass sie gültig sind und dann den Freigabe-Eingang des Funktionsbausteins erneut

setzen (Standardfehler-Reset-Verfahren), damit der Funktionsbaustein den Betrieb wieder aufnimmt. Die Fehlercodes sind in der folgenden Tabelle aufgeführt.

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR	16#000029E0	16#00000001	Eingang DRnEnable ist ungültig
INPUT_INVALID_ERROR	16#000029E1	16#00000002	Eingang DRnVarVelocityEnable ist ungültig
INPUT_RANGE_ERROR	16#00000002	16#00000003	Eingang PressureCmd liegt außerhalb des zulässigen Bereichs
INPUT_RANGE_ERROR	16#00000002	16#00000004	Eingang SwivelAngleCmd liegt außerhalb des zulässigen Bereichs
INPUT_RANGE_ERROR	16#00000002	16#00000005	Eingang MaxFlowLimitCmd liegt außerhalb des zulässigen Bereichs

Tab. 12-112: Fehlercodes Funktionsbaustein MB_SIIIDRn5020

12.6.2 MB_SIIIFcP5020

Kurzbeschreibung

Der Funktionsbaustein MB_SIIIFcP5020 ermöglicht die Drucksteuerung und Volumenstromregelung von Rexroth-Steuerungen über Sercos III Kommunikation des EFC5610 basierend auf Sytronix FcP5020. Die Sollwerte von Druck und Volumenstrom können mit diesem Funktionsbaustein eingestellt werden, die Steuerung kann aktiviert bzw. gestoppt werden, der Antrieb kann als Master oder Slave eingesetzt werden, die Parametergruppe für die p/Q-PID-Steuerung kann zwischen Gruppe 0 und Gruppe 1 umgeschaltet werden, etc. Der Funktionsbaustein beinhaltet auch die Überwachung verschiedener Antriebsfunktionen, die in den folgenden Abschnitten beschrieben werden.

Bibliothek:	RMB_SercosIII_Util
Gruppe:	"RexrothSytronix"
Zielsystem:	MTX 15VRS MLC 15VRS, Funktionspaket SPS MLD mit MPx20

Tab. 12-113: Referenztable MB_SIIIFcP5020

Schnittstellenbeschreibung

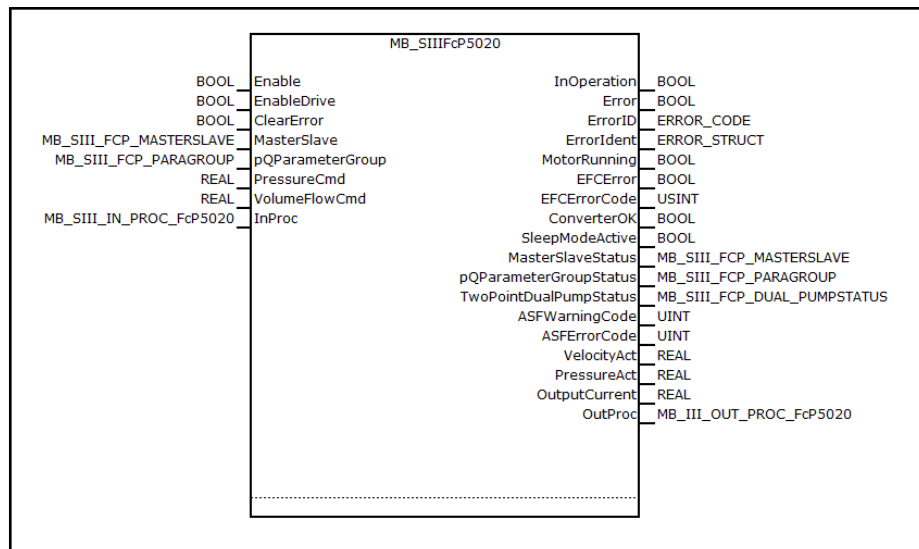


Abb. 12-73: Funktionsbaustein Schnittstelle MB_SIIIFcP5020

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins
	EnableDrive	BOOL	TRUE - Antriebsfreigabe FALSE - Antrieb deaktivieren
	ClearError	BOOL	Antriebsfehler-Reset-Befehl ausführen
	MasterSlave	MB_SIII_FCP_MASTERSLAVE	Auswahl zwischen Master- und Slave-Betrieb MB_SIII_FCP_MASTERSLAVE auf Seite 562
	pQParameter-Group	MB_SIII_FCP_PARAGROUP	Auswahl zwischen Parametergruppe 0 und 1 für p/Q PID-Steuerung MB_SIII_FCP_PARAGROUP auf Seite 562
	PressureCmd	REAL	Drucksollwert in bar
	VolumeFlowCmd	REAL	Volumenstromsollwert in U/min
VAR_OUTPUT	InProc	MB_SIII_IN_PROC_FcP5020	Eingangsstruktur für den Funktionsbaustein
	InOperation	BOOL	TRUE - Funktionsbaustein ist in Betrieb
	Error	BOOL	TRUE - Funktionsbausteinbetrieb aufgrund eines Fehlers gestoppt
	ErrorID	ERROR_CODE	Fehlercode für den aktuellen Fehler
	ErrorIdent	ERROR_STRUCT	Zusätzliche Fehlerinformation
	MotorRunning	BOOL	Motorstatus TRUE - Motor dreht FALSE - Motor steht
	EFCErr	BOOL	TRUE - Fehler am EFC FALSE - Kein Fehler
	EFCErrCode	USINT	Fehlercode für den EFC-Fehler
	ConverterOK	BOOL	TRUE - Konverter OK FALSE - Konverter nicht OK
	OutProc	MB_SIII_OUT_PROC_FcP5020	

E/A-Typ	Name	Datentyp	Beschreibung
	SleepModeActive	BOOL	TRUE - Ruhemodus aktiv FALSE - Inaktiv
	MasterSlaveStatus	MB_SIII_FCP_MASTERSLAVE	Master-/Slave-Status MB_SIII_FCP_MASTERSLAVE auf Seite 562
	pQParameter-GroupStatus	MB_SIII_FCP_PARAGROUP	Parametergruppe für p/Q PID-Steuerung MB_SIII_FCP_PARAGROUP auf Seite 562
	TwoPointDual-PumpStatus	MB_SIII_FCP_DUALPUMP-STATUS	Status Zwei-Punkt-/Doppelpumpensteuerung MB_SIII_FCP_DUALPUMPSTATUS auf Seite 562
	ASFWarningCode	UINT	ASF Warncode
	ASFErrorCode	UINT	ASF Fehlercode
	VelocityAct	REAL	Aktuelle Drehzahl in U/min
	PressureAct	REAL	Aktueller Druck in bar
	OutputCurrent	REAL	Ausgangsspannung in A
	OutProc	MB_SI-II_OUT_PROC_FcP5020	Ausgangsstruktur für den Funktionsbaustein

Tab. 12-114: Schnittstelle MB_SIIIFcP5020

Min-/Max und Default-Werte der Eingänge

Name	Datentyp	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
StartCommand	BOOL			FALSE	Kontinuierlich
ClearError	BOOL			FALSE	Kontinuierlich
MasterSlave	MB_SIII_FCP_MASTERSLAVE			FCP_MASTE R	Kontinuierlich
pQParameterGroup	MB_SIII_FCP_PARAGROUP			FCP_PARA- GROUP	Kontinuierlich
PressureCmd	REAL	0,0	1000,0	0,0	Kontinuierlich
VolumeFlowCmd	REAL	0	5000	0	Kontinuierlich
InProc	MB_SIII_IN_PROC_FcP5020				Kontinuierlich

Tab. 12-115: Min-/Max und Default-Werte der Eingänge MB_SIIIFcP5020

**Funktionsbaustein Eingänge/
Ausgänge - Parameter-Mapping**

E/A-Typ	Name	Parameter
VAR_INPUT	EnableDrive	Parameter S-0-0134 Bit 13, 14, 15
	ClearError	S-0-0099 (unter Verwendung von IL_SIIICommand)
	MasterSlave	Parameter F0.20 - Bit 2
	pQParameterGroup	Parameter F0.20 - Bit 3
	PressureCmd	Parameter F0.21
	VolumeFlowCmd	Parameter F0.22

E/A-Typ	Name	Parameter
VAR_OUTPUT	MotorRunning	Parameter H0.01 - Bit 1
	EFCErrror	Parameter H0.01 - Bit 7
	EFCErrrorCode	Parameter H0.01 - Bits 8-15
	ConverterOk	Parameter H0.02 – Bit 2
	SleepModeActive	Parameter H0.02 – Bit 1
	MasterSlaveStatus	Parameter d0.80 ASF Statuswort Bit 2
	pQParameterGroupStatus	Parameter d0.80 ASF Statuswort Bit 3
	TwoPointDualPumpStatus	Parameter d0.80 ASF Statuswort Bit 10
	ASFWarningCode	Parameter d0.88
	ASFErrorCode	Parameter d0.89
	VelocityAct	Parameter d0.01
	PressureAct	Parameter d0.82
	OutputCurrent	Parameter d0.11

Tab. 12-116: Funktionsbaustein Eingänge/Ausgänge - Parameter-Mapping

Funktionsbeschreibung

Der Funktionsbaustein MB_SIIIFcP5020 verwendet den zyklischen Lese-/Schreibkanal für den Datenaustausch zwischen der SPS und dem Antrieb. Alle erforderlichen Parameter werden im AT/MDT des Antriebs konfiguriert. Das Kommunikationsprotokoll zwischen SPS und Antrieb ist Sercos. EFC-Parameter müssen für die Kommunikation entsprechend eingestellt werden. Wenn der Freigabeeingang während des Betriebs zurückgesetzt wird, werden die an den Antrieb gesendeten Sollwerte nicht zurückgesetzt. Sie reflektieren ihre zuletzt aktualisierten Werte.

In diesem Funktionsbaustein stehen die folgenden Steuerungsfunktionen zur Verfügung:

- Start/Stopp-Befehl
- Antriebsfehler zurücksetzen
- Auswahl zwischen Master-/Slave-Betrieb
- Auswahl zwischen Parametergruppe 0 und 1 für p/Q PID-Steuerung
- Drucksollwert einstellen
- Einstellen des Volumenstromsollwert

In diesem Funktionsbaustein stehen die folgenden Überwachungsfunktionen zur Verfügung:

- Motorstatus
- Fehler EFC
- Fehlercode EFC
- Konverterstatus
- Status Ruhemodus
- Master-/Slave-Status

- Aktive Parametergruppe für p/Q PID-Steuerung
- Status Zwei-Punkt-/Doppelpumpensteuerung
- ASF Warncode
- ASF Fehlercode
- Tatsächliche Drehzahl
- Tatsächlicher Druck
- Ausgangsspannung

Weitere Informationen finden Sie auf der Webseite [Druckregelsysteme Sytronix FcP 5020](#).

Konfiguration der Echtzeit-Eingabe (AT) und Echtzeit-Ausgabe (MDT)

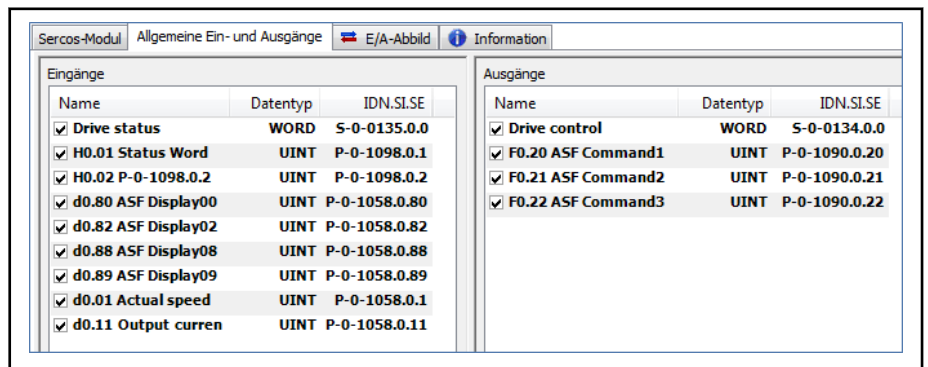


Abb. 12-74: AT- und MDT-Konfiguration für EFC

Fehlerbehandlung

Der Funktionsbaustein kann durch ungültige Eingaben einen Fehlerzustand erreichen. Während dieser Zeit meldet der Funktionsbaustein einen Fehler und stoppt den Betrieb. Die Steuerwortbits und die zuvor eingestellten Sollwerte **werden aber nicht** gemäß den Systemanforderungen für FcP5020 **zurückgesetzt**. Der Benutzer muss die Eingabewerte so korrigieren, dass sie gültig sind und dann den Freigabe-Eingang des Funktionsbausteins erneut setzen (Standardfehler-Reset-Verfahren), damit der Funktionsbaustein den Betrieb wieder aufnimmt. Die Fehlercodes sind in der folgenden Tabelle aufgeführt.

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR	16#000029E2	16#00000001	EingangMasterSlave ist ungültig
INPUT_INVALID_ERROR	16#000029E3	16#00000002	Eingang pQParameterGroup ist ungültig
INPUT_RANGE_ERROR	16#00000002	16#00000003	Eingang PressureCmd liegt außerhalb des zulässigen Bereichs
INPUT_RANGE_ERROR	16#00000002	16#00000004	Eingang VolumeFlowCmd liegt außerhalb des zulässigen Bereichs

Tab. 12-117: Fehlercodes Funktionsbaustein MB_SIIIFcP5020

12.6.3 MB_SIII_DRN_ENABLE

Diese Aufzählung wird vom Eingang DRnEnable verwendet.

Element	Beschreibung
DRN_DISABLE	DRn Freigabe (X3) - Inaktiv
DRN_ENABLE	DRn Freigabe (X3) - Aktiv

Tab. 12-118: Strukturbeschreibung MB_SIII_DRN_ENABLE

12.6.4 MB_SIII_DRN_VAR_VEL_ENABLE

Diese Aufzählung wird vom Eingang DRnVarVelocityEnable verwendet.

Element	Beschreibung
DRN_VAR_VEL_DISABLE	DRn variable Geschwindigkeitsfreigabe (X4) - Inaktiv
DRN_VAR_VEL_ENABLE	DRn variable Geschwindigkeitsfreigabe (X4) - Aktiv

Tab. 12-119: Strukturbeschreibung MB_SIII_DRN_VAR_VEL_ENABLE

12.6.5 MB_SIII_FCP_MASTERSLAVE

Diese Aufzählung wird vom Eingang MasterSlave verwendet.

Element	Beschreibung
FCP_MASTER	Master
FCP_SLAVE	Slave

Tab. 12-120: Strukturbeschreibung MB_SIII_FCP_MASTERSLAVE

12.6.6 MB_SIII_FCP_PARAGROUP

Diese Aufzählung wird vom Eingang pQParameterGroup und vom Ausgang pQParameterGroupStatus verwendet.

Element	Beschreibung
FCP_PARA_GROUP0	Parametergruppe 0
FCP_PARA_GROUP1	Parametergruppe 1

Tab. 12-121: Strukturbeschreibung MB_SIII_FCP_PARAGROUP

12.6.7 MB_SIII_FCP_DUALPUMPSTATUS

Diese Aufzählung wird vom Ausgang TwoPointDualPumpStatus verwendet.

Element	Beschreibung
FCP_DUALPUMP_VGMAX	$V_{g_{\max}}$
FCP_DUALPUMP_VGMIN	$V_{g_{\min}}$

Tab. 12-122: Strukturbeschreibung MB_SIII_FCP_DUALPUMPSTATUS

13 RIL_SercosIIIDeviceReplacement.library

13.1 Einführung und Übersicht

Die Bibliothek RIL_SercosIIIDeviceReplacement.library stellt Bausteine zur Verfügung, die einen PC-losen Gerätetausch ermöglichen. Damit ist es z.B. möglich, die aktuelle Gerätekonfiguration zu ermitteln, mit einer vorher abgespeicherten Konfiguration zu vergleichen und bei Änderungen die Parametrierung und Firmware der Geräte wiederherzustellen.

Zielsysteme Die Bibliothek kann auf folgenden Systemen eingesetzt werden:

Zielsystem	Bibliothek
MLC	RIL_SercosIIIDeviceReplacement

Überblick Die Funktionalität für den Sercos-Gerätetausch dient dazu, defekte Geräte im Sercos-Bus ohne Zuhilfenahme eines PCs zu tauschen. Dabei wird die Konfiguration einer Anlage zunächst gesichert und als zukünftige Referenz verwendet. Beim zukünftigen Neustart oder auf Anforderung wird diese Referenz mit der aktuellen Konfiguration verglichen. Bei Abweichung wird die Konfiguration der Referenz wiederhergestellt. Die Referenz der Konfiguration wird auch "Fingerabdruck" genannt, da diese Referenz der eindeutigen Identifizierung der Anlage dient. Die Funktionalität für den Sercos-Gerätetausch ist kein einzelner Baustein, der einfach verwendet werden kann und damit allen Anwendungen gerecht wird. Vielmehr ist es eine Ansammlung von Bausteinen, die durch den Anwender miteinander verschaltet werden müssen, um eine für die Anwendung passende Funktionalität zu erhalten.

Dabei wird in drei große Gruppen an Funktionalität unterschieden:

- Bausteine zum Lesen und Schreiben des Fingerabdrucks, dem Lesen der aktuellen Konfiguration und Vergleichen des Fingerabdrucks mit der aktuellen Konfiguration
- Sichern und Wiederherstellen der Firmware und Parameterdaten, sowie Rebooten von Geräten
- Hilfsfunktionen, wie z.B. Logging

Anwendungsfälle Für die Gerätetausch-Funktionalität gibt es mehrere Anwendungsfälle:

- Ein ausgefallenes Gerät soll ersetzt werden und muss anschließend mit definierter Firmware und Parametern beschrieben werden
- Mehrere ausgefallene Geräte sollen ersetzt werden
- Alle Geräte sollen ersetzt werden

Dies ist kein realer Fall an einer Anlage, kann aber bei der Serieninbetriebnahme genutzt werden. In diesem Fall würde eine Anlage mit allen Geräten aufgebaut werden. Anschließend kopiert man alle Dateien einer bestehenden Anlage auf die neue Anlage und startet die Gerätetauschfunktionalität. Diese Funktionalität wird dann erkennen, dass sich alle Geräte geändert haben und entsprechend auch alle Geräte aktualisieren.

Abgrenzung - nicht unterstützte Fälle

Mit dem Begriff Gerätetausch können sehr viele Funktionalitäten in Verbindung gebracht werden. Im vorhergehenden Abschnitt wurden daher die unterstützten Anwendungsfälle aufgelistet. Zur besseren Abgrenzung

werden hier aber auch Fälle aufgelistet, die explizit nicht unterstützt werden. Diese sind:

- Wiederherstellen von sicherheitsgerichteten Parametern. Diese sind für Bosch Rexroth Antriebe zwar in einer Parametersicherung enthalten und werden somit beim Wiederherstellen auch in die Geräte eingespielt, es sind aber noch weitere Schritte erforderlich, damit auch sicherheitsgerichtete Parameter aktiv werden. Dies wird im Rahmen des Gerätetausches nicht durchgeführt
- Geänderte Topologie:
Die Gerätetauschfunktionalität basiert auf dem Konzept, dass sich die Geräte an der topologisch richtigen Adresse befinden. Wird die Topologie geändert, so nimmt die Gerätetauschfunktionalität an, dass sich die Geräte entsprechend geändert haben und wird versuchen diese wiederherzustellen. Z.B. wird beim Vertauschen der Stecker an den beiden Sercos-Ports der Steuerung die komplette Topologie invertiert. Die Folge wäre, dass die Gerätetauschfunktionalität versuchen würde, alle Geräte wiederherzustellen.
- Kompatible Hardware:
Bei einem Gerätetausch kann es passieren, dass neuere, kompatible Hardware verwendet wird. Diese kann die alte problemlos ersetzen, aber die Gerätetauschfunktionalität wird die neue Hardware als unterschiedlich erkennen und mit Fehler abbrechen. In diesem Fall müsste der Gerätetausch per Hand erfolgen und anschließend eine neue Referenzkonfiguration erstellt werden.

Referenzkonfiguration/Fingerabdruck und aktuelle Konfiguration

Die Referenzkonfiguration bzw. der Fingerabdruck und auch die aktuelle Konfiguration sind zentral für die Funktionalität. Der Fingerabdruck muss remanent gespeichert werden und wird deshalb auf der Compact-Flash-Karte der Steuerung als Datei abgelegt. Pfad und Name der Datei kann der Anwender dabei selbst festlegen. Die Datei enthält neben den eigentlichen Gerätedaten auch noch Steuer- und Typdaten. Für Fehlersuche und Überprüfung wird die Datei als Textdatei im CSV-Format abgelegt.



WARNUNG

Datenverlust

Das CSV-Format ist ein rein textbasiertes Format, das Werte durch Kommas trennt. Dieses Format wird auch von Microsoft Excel verwendet und kann deshalb auch mit Microsoft Excel geöffnet werden. Microsoft Excel konvertiert einige Daten. Dadurch können Daten verfälscht werden. Werden diese Daten dann abgespeichert, kann dies zu Datenverlust bzw. Verlust der Gerätetauschfunktionalität führen. Es wird daher empfohlen, die Datei nur mit einem einfachen Texteditor zu öffnen.

Die aktuelle Konfiguration muss eigentlich nicht abgespeichert werden, da die Konfiguration direkt ausgelesen werden kann. Für Vergleiche ist es aber sinnvoll auch diese Konfiguration remanent abzuspeichern. Pfad- und Dateinamen können auch hier frei gewählt werden.

Datenbank

Sowohl für den Fingerabdruck, als auch für die aktuelle Konfiguration wird eine "Datenbank" verwendet. Die Datenbank ist eine interne Verwaltung der Daten, die es ermöglicht Geräte hinzuzufügen oder zu entfernen und verschiedene Sortierungen zu ermöglichen. Dies erfolgt ohne Einwirkung des Anwenders im Hintergrund. Dazu ist es notwendig, das Programm "IL_DB_ControlAsync" aus dem Verzeichnis "400_Database/PRG" in einer

niederprioren Task aufzurufen. Es wird empfohlen, diese Task als freilaufenden Typ anzulegen.

Für den Betrieb ist keine weitere Aktion notwendig. Lediglich der Pfad und Name der Datenbank muss angegeben werden. Bei Funktionsbausteinen, die auf die Datenbank zugreifen müssen, muss außerdem noch eine Variable des Typs IL_DB_TYPE angelegt werden, über die die Datenbank verwaltet wird.

Zeitverhalten

Die Gerätetauschfunktion ist eine Funktionalität, die im laufenden Betrieb einer Anlage nicht verwendet wird, sondern nur beim Starten oder auf Anforderung. Außerdem werden Daten als Datei abgelegt, Parameter geschrieben und gelesen, sowie Firmware ggf. ausgetauscht. Diese Aktionen können sehr zeitintensiv sein. Deshalb sollte die Gerätetauschfunktionalität in einer niederprioren Task erfolgen. Dies kann auch die gleiche Task sein, in der das Programm für die Datenbank aufgerufen wird.

VORSICHT

Automatisches Schalten der Sercos-Phase

Die Gerätetauschfunktionalität muss sich für die meisten Aktionen in Sercos-Phase null oder zwei befinden. Deshalb wird diese Phase durch die Bausteine automatisch gewechselt. Außerdem werden beim Firmwaretausch Geräte neu gebootet und sind damit kurzzeitig nicht am Sercos-Bus verfügbar. Daher ist es notwendig, vor dem Start der Gerätetauschfunktion sicherzustellen, dass sich die Anlage in einem Zustand befindet, in dem dies ohne Schaden möglich ist (z.B. alle Achsen befinden sich im Stillstand).

Beispiel für den automatisierten Gerätetausch

Die Gerätetauschfunktionalität hat verschiedene Anwendungsmöglichkeiten. Ein Beispiel ist in der Bibliothek selbst enthalten. Im Bibliotheksverwalter kann der Code im Verzeichnis "_Example" betrachtet und auch für eigene Zwecke kopiert werden. Ebenso enthalten ist eine kleine Visualisierung, in der die einzelnen Schritte noch einmal visualisiert werden. Im nächsten Abschnitt wird auf dieses Beispiel näher eingegangen.

Vorbereitungen

Um die Gerätetauschfunktionalität zu nutzen, müssen einige Voraussetzungen vorliegen.

Aufruf der Programme

In der Gerätetauschfunktionalität wird eine Datenbank verwendet, um die Gerätedaten zu verwalten. Dies erfordert, dass ein Hintergrundprogramm für die Datenbank aufgerufen wird. Das Hintergrundprogramm liegt in der Bibliothek im Verzeichnis "400_Database/PRG" und ist das Programm "IL_DB_ControlAsync". Es wird empfohlen, diese Task als freilaufenden Typ mit niedriger Priorität anzulegen.

Ebenso ist es notwendig die Gerätetauschfunktionalität in einer Task zu implementieren. Sinnvollerweise erfolgt dies auch in einem eigenen Programm. Dieses Programm kann man dann auch in der gleichen Task wie das Hintergrundprogramm für die Datenbank aufrufen.

Dynamischer Speicher

Für die Ausführung benötigt die Gerätetauschfunktionalität dynamischen Speicher, der für das dynamische Anlegen von Funktionsbausteinen verwendet wird. Um den dynamischen Speicher anzulegen wählt man über das Kontextmenü auf dem Applikationsknoten die Eigenschaften. Im sich öffnenden Dialog wechselt man auf den Reiter "Optionen Applikationserzeugung". Hier kann man die dynamische Speicherallokierung aktivieren und auch eine maximale Größe des Speichers angeben:

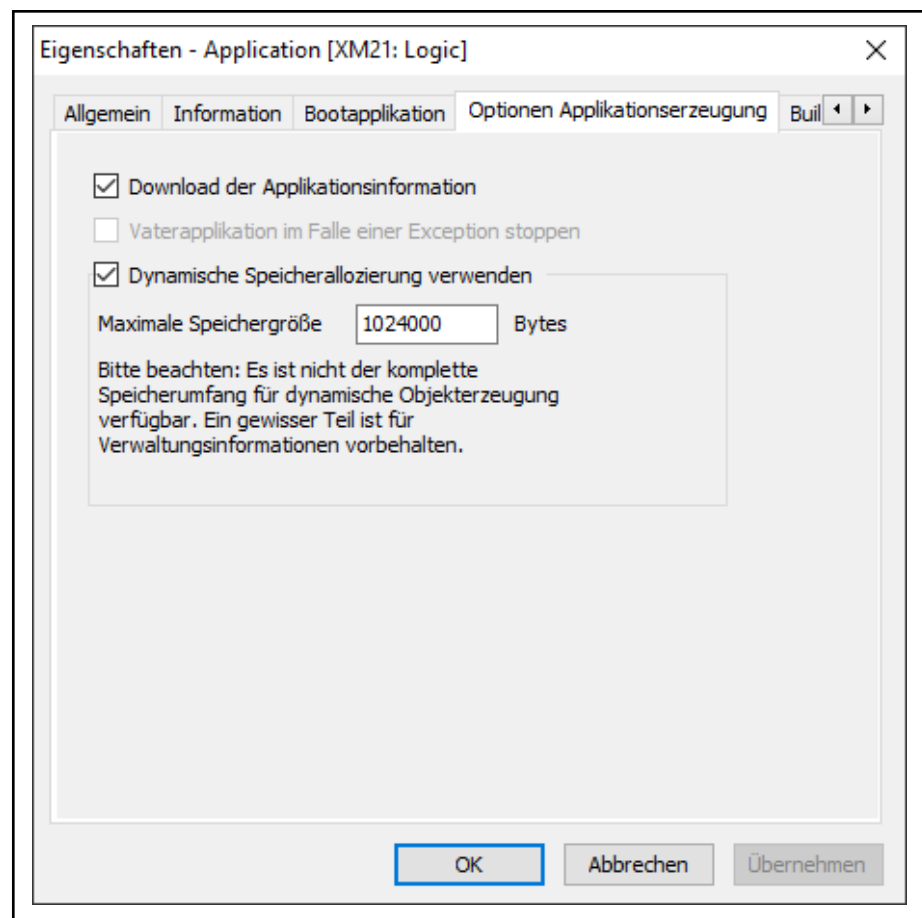


Abb. 13-1: Dialog zur Eingabe der Größe des dynamischen Speichers

Die Größe des dynamischen Speichers hängt von der Anzahl der zu tauschenden Geräte ab und wie schnell dieser Tausch erfolgen soll. Minimal sollten 130.000 Bytes zur Verfügung gestellt werden. Dabei kann nur ein Gerät bearbeitet werden (Parameter Backup/Restore oder Firmware Backup/Restore). Bei 1.000.000 Bytes können bis zu 9 Geräte parallel bearbeitet werden. Je mehr Speicher zur Verfügung steht, um so mehr Geräte können parallel bearbeitet werden und umso schneller erfolgt die Abarbeitung. Die maximale Anzahl liegt bei 32 Geräten.

Die Empfehlung lautet daher für den Anwendungsfall, dass ein defektes Gerät getauscht werden soll, 130.000 Bytes zur Verfügung zu stellen.

Für den Anwendungsfall, dass mehrere Geräte gleichzeitig gewechselt werden, hat sich ein Speicher von 1.000.000 als sinnvoller Kompromiss zwischen Speicherverbrauch und paralleler Abarbeitung ergeben.

Beispielprogramm Deklaration

Das Beispielprogramm ist nicht in der Bibliothek enthalten, kann aber als GAT-Plugin installiert werden. Hier soll das Programm analysiert werden, um die Funktion besser zu verstehen.

Deklaration der Variablen

Die Eingänge des Programms dienen dazu, dessen Funktion zu steuern:

- "Check":

Bei gesetzter Variable wird die Referenzkonfiguration geladen, dann die aktuelle Konfiguration gebildet und anschließend werden beide Konfigurationen verglichen. Das Ergebnis des Vergleichs kann an den

Ausgängen NofSercosAddrDifferent, NofParametersDifferent und NofFirmwareDifferent abgelesen werden.

- "Init":

Bei gesetzter Variable wird auch wieder die Referenz und aktuelle Konfiguration geladen und bei Differenzen wird versucht die Referenzkonfiguration wiederherzustellen. Dabei werden die Sercos-Adressen am Bus wiederhergestellt, und bei Bedarf die Firmware und die Parameterdateien wieder in die Geräte eingespielt.

- "Store":

Bei gesetzter Variable wird die aktuelle Konfiguration als neue Referenzkonfiguration abgespeichert.

Mit der Konstanten "CHECK_FIRMWARE" kann gesteuert werden, ob die Firmware auch gesichert und bei Bedarf wieder eingespielt werden soll (= TRUE) oder nicht. Die Parameter werden immer gesichert und wieder eingespielt.

Beispielprogramm - Implementierung

Zu Beginn des Programms werden zuerst nur die boolschen Variablen für die Anzeige in der Visualisierung initialisiert und der Startzustand der Zustandsmaschine in Abhängigkeit der gewählten Funktion gesetzt. Anschließend beginnt schon die Zustandsmaschine. Im Zustand STATE_BOOTING wird zuerst gewartet, dass sich die Steuerung nicht mehr im Bootzustand befindet. Anschließend wird im Zustand STATE_SWITCH_P2 in Sercos-Phase 2 geschaltet. Hier ist sichergestellt, dass keine Bewegung stattfindet. Danach wird im Zustand STATE_LOAD_REF mit der Gerätetauschfunktionalität begonnen. Dazu wird die Referenzkonfiguration von der Compact-Flash-Karte gelesen. Das Ergebnis wird in SercosRefList abgelegt.

Eigentlich würde man jetzt die aktuelle Konfiguration lesen. Sollte aber ein Gerät bereits getauscht worden sein, dann ist es sehr wahrscheinlich, dass die Sercos-Adresse dieses Geräts nicht korrekt eingestellt ist. Deshalb wird zunächst im Zustand STATE_RESTORE_BUS versucht, alle Sercos-Adressen anhand der Referenzkonfiguration einzustellen. Der Zeitbedarf dafür ist gering. Schließlich wird im Zustand STATE_CREATE_ACT die aktuelle Buskonfiguration gelesen, die jetzt auch schon korrigierte Sercos-Adressen enthält.

Daran anschließend wird im Zustand STATE_GET_ACT die gerade gelesene aktuelle Konfiguration in ein Array von Strukturen gelesen. Diese Aktion ist für den Gerätetausch nicht notwendig, denn es werden lediglich die Daten, die in der Datenbank "SercosActList" enthalten sind, in ein Array von Strukturen kopiert. Damit ist es möglich auf Anwenderseite die Daten zu verwenden. Die Gerätetauschfunktionalität verwendet dagegen nur die Datenbank.

Im nächsten Zustand (STATE_COMPARE) erfolgt der Vergleich zwischen Referenz und aktueller Konfiguration. Dabei wird eine Liste zurückgegeben. Darin sind alle Informationen enthalten, um einen Gerätetausch vorzunehmen. Im Index IL_SDR_CONST.ALL (= -1) werden die Unterschiede gesammelt für alle Geräte angegeben. Für einen ersten Vergleich ist deshalb lediglich dieser Index zu prüfen. Im Index 0 sind die Unterschiede für die Steuerung enthalten und in den Indizes 1-x sind die Unterschiede jedes einzelnen Gerätes in topologischer Reihenfolge enthalten. Der Funktionsbaustein für den Vergleich der Konfigurationen besitzt außerdem den binären Ausgang "Difference". Ist dieser Ausgang gesetzt, so gibt es Unterschiede. Ist dieser Ausgang nicht gesetzt, gibt es keine Unterschiede und es sind keine weiteren Aktionen notwendig.

Die Bedeutung der einzelnen Strukturelemente in der Liste sind im folgenden aufgelistet.

Inhalt und Bedeutung der Vergleichsliste

Strukturelement	Verwendete Parameter	Anmerkung
RestoreHW	S-0-1300.0.1, S-0-1300.0.4, S-0-1300.0.5, S-0-1026, S-0-1300.0.11, S-0-1300.0.3	Dies ist eine Verknüpfung mehrerer Parameter, die sich bei gleicher Hardware nicht ändern. Dabei müssen alle Parameter identisch sein. Ist hier der Wert 65535 (= 16#FFFF) eingetragen, so zeigt dies an, dass sich die Anzahl der Geräte am Sercos-Ring unterscheiden. Eine Wiederherstellung der Buskonfiguration ist dann nicht möglich
RestoreFW	S-0-1300.0.9 und TFTP=TRUE	Dies ist eine Verknüpfung aus dem Firmware-String und der TFTP-Unterstützung, da zum Firmware-Update TFTP zwingend notwendig ist
RestorePAR	Dies ist eine Kombination aus geändertem S-0-0531 und RestoreFW	Parameter Restore ist notwendig, sofern sich die Parameter-Checksumme ändert, oder auch nach einem Firmware-Update
ComponentName	S-0-1300.0.1	
VendorCode	S-0-1300.0.3	
DevDesc	S-0-1300.0.4	
Deviceld	S-0-1300.0.5	
HW_String	S-0-1300.0.8	
FW_String	S-0-1300.0.9	
OrderInfo	S-0-1300.0.11	
ManufDate	S-0-1300.0.13	
HW_Ident	S-0-1026	
FW_String	S-0-1300.0.9 alternativ S-0-0030	Es wird der Parameter S-0-1300.0.9 gelesen, falls dies fehlschlägt, wird alternativ S-0-0030 gelesen
TFTP	S-0-1000.0.0 enthält 16#0E01=TFTP	Wert enthält entweder TRUE (wenn 16#0E01 in der Parameterliste gefunden wird) oder FALSE, sofern nicht
ParChkSum	S-0-0531	

Im Anschluss wird dann entweder in den Zustand STATE_DONE gewechselt, sofern keine Unterschiede vorhanden sind. Bei Unterschieden wird entweder in den Zustand STATE_RESTORE_FW (bei geänderter Firmware) oder

STATE_RESTORE_PAR (bei geänderten Parametern) gewechselt. Nach einer Firmwareänderung erfolgt immer ein Reboot (Zustand STATE_REBOOT) und die Parameter werden neu geladen. Damit ist ein Gerätetausch abgeschlossen. Die in den Restore-Zuständen verwendeten Bausteine benötigen Informationen, um die Geräte entsprechend zu aktualisieren. Dazu dient die bereits bekannte Liste von Unterschieden, die beim Vergleich erzeugt wurde. Der Anwender muss lediglich die Liste der Unterschiede übergeben. Soll ein individueller Vergleich erfolgen (z.B. ab einer bestimmten Seriennummer soll eine andere Firmware verwendet werden), so muss der Anwender den Firmware-/Parameter-Update selbst vornehmen.

Beim Update der Firmware und der Parameter können Fehler bei einzelnen Geräten auftreten. Dazu werden den Bausteinen Listen übergeben, die dieser Baustein im Fehlerfall füllt. Darin ist sowohl die Sercos-Adresse enthalten, als auch der aufgetretene Fehler.

In der Zustandsmaschine sind noch drei weitere Zustände (STATE_CREATE_REF, STATE_BACKUP_FW, STATE_BACKUP_PAR) enthalten, die bei einem Speichern einer neuen Referenzkonfiguration verwendet werden. Die Namen sind hierbei entsprechend bezeichnend, zunächst wird die Referenzkonfiguration erzeugt und als Datei abgelegt, dann werden von allen Geräten die Firmware und die Parameter gesichert.

Fehler, die im Beispielcode auftreten, werden nicht detailliert ausgewertet. Sie führen lediglich dazu, dass in den jeweiligen Fehlerzustand gesprungen wird. Hier werden lediglich die Statusmeldung generiert und die Programmausgänge gesetzt. In einer realen Applikation könnten dann weitere Aktionen erfolgen.

Weitere Details sind auch in der Beschreibungsdatei für das GAT-Plugin enthalten. Bei weiteren Optimierungen im Code des GAT-Plugins werden diese dort auch beschreiben.

Individueller Vergleich

Im Beispielcode erfolgt der Gerätetausch aufgrund definierter Vergleiche. So wird bei einer geänderten Firmwareversion z. B. die Firmware getauscht. Soll aber ein anwenderspezifischer Vergleich erfolgen, so muss dies, und auch die anschließende individuelle Aktion (Firmware-/Parameter-Update), durch den Anwender erfolgen.

Vergleich

Der Baustein zum Vergleichen der Konfigurationen liefert eine Liste zurück. Diese Liste wird jetzt nicht im Index IL_SDR_CONST.ALL (= -1) ausgewertet, sondern muss von Index 1 bis MB_MAX_SERCOS_ADDRESS (bzw. die Gesamtlänge der Liste) durchlaufen werden. Für jedes Gerät (jeder Index) kann dann individuell geprüft und ausgewertet werden, ob eine Aktion erfolgen soll. In jedem Index ist dabei für alle Elemente nur eine 1 (es existiert ein Unterschied) oder eine 0 (es existiert kein Unterschied) vorhanden.

Soll bei einer Änderung ein Firmware-Update oder ein Parameter-Update erfolgen, so muss dies auch individuell erfolgen. Im Beispielcode werden die Bausteine IL_SDR_RestoreFwGroup bzw. IL_SDR_RestoreParGroup verwendet. Diese Bausteine sind aber nur zu verwenden, wenn die gesamte Liste aus dem Vergleich verwendet wird. Soll ein Update nur bei einzelnen Geräten erfolgen, so müssen die Bausteine IL_SDR_RestoreFwDevice bzw. IL_SDR_RestoreParDevice verwendet werden. Dazu muss die Struktur IL_SDR_DEV_INFO_TYPE übergeben werden. Diese Struktur muss durch den Anwender selbst gefüllt werden.

Inhalt der Bibliothek RIL_SercosIII-DeviceReplacement

Die Bibliothek RIL_SercosIIIDeviceReplacement enthält viele Bausteine. Diese sind durch eine Ordnerstruktur in sinnvolle Abschnitte unterteilt. Eine genaue Beschreibung der Bausteine erfolgt in diesem Dokument nicht. Stattdessen ist die Dokumentation bereits in der Bibliothek selbst enthalten und kann über den Bibliotheksverwalter betrachtet werden.

Funktionsbausteine von RIL_SercosIIIDeviceReplacement

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine / Datentypen	Beschreibung
IL_SDR_CONFIG	Konfigurationsdaten für die Gerätetauschfunktionalität. Die Pfad- und Dateinamen können durch den Anwender angepasst werden. Die anderen Werte sollten nur nach Aufforderung durch Bosch Rexroth Mitarbeiter geändert werden
Datentypen (100_Type)	
IL_SDR_DEVICE_REF	Struktur zur eindeutigen Adressierung eines Sercos-Teilnehmers
IL_SDR_DEV_DIFF_TYPE	Struktur zum Auflisten von Unterschieden zwischen zwei Konfigurationen
IL_SDR_DEV_INFO_TYPE	Datenstruktur die ein Sercos-Gerät beschreibt. Diese Datenstruktur kann auch abgespeichert und mit anderen Konfigurationen verglichen werden
IL_SDR_MSG_LOG_TYPE	Struktur zur Definition von Logging-Nachrichten
Globale Variablen (200_Var)	
IL_SDR_DATA	Globale Datennachrichten-Logger und genutzter dynamischer Speicher
IL_SDR_ERR_CODE	Globale konstante Daten zum Definieren von Fehlercodes
Administrations-Bausteine (300_POU/310_Administration)	
IL_SDR_CompareCofig	Baustein zum Vergleichen zweier Konfigurationen
IL_SDR_CreateDeviceList	Baustein zum Erstellen der aktuellen Konfiguration am Sercos-Bus
IL_SDR_CreateReferenceList	Baustein zum Erstellen einer Referenzkonfiguration
IL_SDR_CreateUserList	Baustein zum Überschreiben einer Konfiguration mit modifizierten Daten
IL_SDR_GetDeviceList	Liest eine Konfiguration und gibt sie als Array von Strukturen aus. Damit ist ein einfacher Zugriff auf die Gerätedaten möglich
IL_SDR_OpenReferenceList	Baustein zum Öffnen einer Referenzkonfiguration vom Speichermedium
Backup/Restore Bausteine für einzelne Geräte (300_POU/320_Single Device)	
IL_SDR_BackupFwDevice	Sichern einer Firmware eines Gerätes auf der Steuerung
IL_SDR_BackupParDevice	Sichern eines Parametersatzes eines Geräts auf der Steuerung
IL_SDR_RestoreFwDevice	Wiederherstellen einer Firmware eines Geräts von der Steuerung
IL_SDR_RestoreParDevice	Wiederherstellen eines Parametersatzes eines Geräts von der Steuerung
Backup/Restore Bausteine für Gruppen von Geräten (300_POU/330_Device Group)	
IL_SDR_BackupFwGroup	Sichern von Firmware einer Gruppe von Geräten auf der Steuerung

IL_SDR_BackupParGroup	Sichern von Parametersätzen von Geräten auf der Steuerung
IL_SDR_RebootGroup	Bootet alle Geräte einer Gruppe neu
IL_SDR_RestoreFwGroup	Wiederherstellen von Firmware einer Gruppe von Geräten von der Steuerung
IL_SDR_RestoreParGroup	Wiederherstellen von Parametersätzen von Geräten von der Steuerung
Target abhängige Bausteine (300_POU/380_TargetSpecific)	
IL_SDR_ResetBusConfig_MLC	Schreibt die Sercos-Adresse in alle Geräte am Sercos-Bus (Adressvergabe). Nur für Steuerung MLC!
IL_SDR_ResetBusConfig_MTX	Schreibt die Sercos-Adresse in alle Geräte am Sercos-Bus (Adressvergabe). Nur für Steuerung MTX (noch nicht implementiert)!
Sercos Phase (300_POU/390_Tools/391_SIII Phase)	
IL_SDR_GetPhase	Liefert die aktuelle Sercos-Phase
IL_SDR_SetPhase	Setzt die Sercos-Phase
Sercos Phase (300_POU/390_Tools/394_Logger)	
IL_SDR_ErrorText	Übersetzung von Fehlernummern in Text
IL_SDR_MessageLog	Funktion zum Aufzeichnen von Textnachrichten
Datenbank (400_Database)	
IL_DB_CONFIG	Konfigurationsdaten für die Datenbankfunktionalität. Werte sollten nur nach Aufforderung durch Bosch Rexroth Mitarbeiter geändert werden
Datenbank (400_Database/PRG)	
IL_DB_ControlAsync	Programm zum Abarbeiten von Datenbankanfragen. Muss in einer niederprioreren Task aufgerufen werden
Datenbank Datentypen (400_Database/TYPE)	
IL_DB_DESCRIPTION_TYPE	Datentyp zur Beschreibung von Datenbankeinträgen
IL_DB_TYPE	Datentyp zur Beschreibung einer Datenbank
Datenbank Globale Daten (400_Database/VAR)	
IL_DB_ERR_CODE	Definitionen für Fehlernummern

Tab. 13-1: RIL_SercosIIIDeviceReplacement-Bibliothek

14 RIL_InlineIOModules.library

14.1 Einführung und Überblick

Diese Bibliothek beschreibt die Zugriffsmöglichkeiten des IEC-Programms auf InlineIO-Module.

Bei der Realisierung der Bibliothek wird auf die RIL_Uutilities-Bibliothek aufgesetzt.

Es werden die Basisfunktionsbausteine (Basisklassen) IL_EnableDoneBase, IL_EnableInOpBase, IL_ExecuteDoneBase und IL_ExecuteInOpBase verwendet und deren Methoden erweitert oder überschrieben.

Besonders bei komplexen Klemmen ist es wichtig, eine Unterstützung durch abgestimmte Funktionsbausteine für IndraLogic (2G) zu haben.

Mit Hilfe dieser Funktionsbausteine ist es dem Anwender möglich in wenigen Schritten die Klemmen in Betrieb zu nehmen.

Die Bibliothek ist nutzbar für folgende IO-Module:

Analoge Eingangsklemmen	Materialnummer	Funktionsbaustein
R-IB IL AI 2/SF-PAC Rexroth Inline-Klemme mit zwei analogen Eingangskanälen	Datenblatt R911170605	IL_AI2Sf auf Seite 576
	Klemme R911170784	
R-IB IL AI 2/SF-230-PAC Rexroth Inline-Klemme mit zwei analogen Eingangskanälen	Datenblatt R911170521	IL_AI2Sf auf Seite 576
	Klemme R911170425	
R-IB IL AI 4/EF-PAC Rexroth Inline-Klemme mit vier analogen Differenz-Eingangskanälen	Datenblatt R911170541	IL_AI4Ef auf Seite 580
	Klemme R911170426	
R-IB IL AI 4/I-PAC Rexroth Inline-Klemme mit 4 Eingängen zum Anschluss von Stromsignalen	Datenblatt R911336654	IL_AI4IU auf Seite 585
	Klemme R911172894	
R-IB IL AI 4/U-PAC Rexroth Inline-Klemme mit 4 Eingängen zum Anschluss von Spannungssignalen	Datenblatt R911336656	IL_AI4IU auf Seite 585
	Klemme R911172895	

Analoge Eingangsklemmen	Materialnummer	Funktionsbaustein
R-IB IL AI 8/SF-PAC Rexroth Inline-Klemme mit acht analogen Eingangskanälen	Datenblatt R911170483	IL_AI8SfIs auf Seite 590
	Klemme R911308493	
R-IB IL AI 8/IS-PAC Rexroth Inline-Klemme mit acht analogen Eingangskanälen	Datenblatt R911170485	IL_AI8SfIs auf Seite 590
	Klemme R911308494	
R-IB IL SGI 2/F-PAC Rexroth Inline-Klemme mit zwei analogen Eingängen für Dehnungsmessstreifen	Datenblatt R911170581	IL_Sgi2f auf Seite 594
	Klemme R911170432	IL_Sgi2fOffset auf Seite 600
R-IB IL TEMP 2 RTD-PAC Rexroth Inline-Klemme mit zwei Eingängen für Temperatursensoren	Datenblatt R911170603	IL_Temp2Rtd auf Seite 605
	Klemme R911170785	
R-IB IL TEMP 2 UTH-PAC Rexroth Inline-Klemme mit zwei Eingängen für Thermoelemente	Datenblatt R911170517	IL_Temp2Uth auf Seite 600
	Klemme R911170431	
R-IB IL TEMP 4/8 RTD-PAC Rexroth Inline-Klemme mit acht resistiven Temperatursensor-Eingängen	Datenblatt R911170513	IL_Temp4_8Rtd auf Seite 611
	Klemme R911170428	
R-IB IL TEMP 4/8 RTD-EF-PAC Rexroth Inline-Klemme mit acht resistiven Temperatursensor-Eingängen, unterstützt gegenüber der R-IB IL TEMP 4/8 RTD-PAC zusätzlich 4-Draht Sensoren	Datenblatt R911338006	IL_Temp4_8RtdEf auf Seite 617
	Klemme R911173029	

Tab. 14-1: Analoge Eingangsklemmen und ihre Funktionsbausteine

Analoge Ausgangsklemmen	Materialnummer	Funktionsbaustein
R-IB IL AO 1/SF-PAC Rexroth Inline-Klemme mit einem analogen Ausgang	Datenblatt R911170601	IL_AO1Sf auf Seite 623
	Klemme R911170787	
R-IB IL AO 2/SF-PAC Rexroth Inline-Klemme mit zwei analogen Ausgängen	Datenblatt R911170519	IL_AO2SfBp auf Seite 626
	Klemme R911170436	

Analoge Ausgangsklemmen	Materialnummer	Funktionsbaustein
R-IB IL AO 2/U/BP-PAC	Datenblatt	IL_AO2SfBp auf Seite 626
	Klemme R911170786	
R-IB IL AO 2/UI-PAC Rexroth Inline-Klemme mit zwei analogen Ausgängen	Datenblatt R911338448	IL_AO2UI auf Seite 630
	Klemme R911173634	

Tab. 14-2: Analoge Ausgangsklemmen und ihre Funktionsbausteine

Funktionsklemmen	Materialnummer	Funktionsbaustein
R-IB IL CNT-PAC Rexroth Inline Zählerklemme	Datenblatt R911170545	IL_CntCount auf Seite 634 IL_CntFrequency auf Seite 641
	Klemme R911170788	IL_CntPulse auf Seite 647 IL_CntTime auf Seite 650
R-IB IL PWM/2-PAC Rexroth Inline-Klemme zur Pulsweiten- und Frequenzmodulation IL PWM 2/CC-3A	Datenblatt R911170575	IL_Pwm2Frequency auf Seite 656 IL_Pwm2Pwm auf Seite 659
	Klemme R911170444	IL_Pwm2PulseDirection auf Seite 663 IL_Pwm2SinglePulse auf Seite 666
R-IB IL SSI-PAC Rexroth Positionierklemme für Absolutwertgeber	Datenblatt R911318640	IL_Ssi auf Seite 669
	Klemme R911308594	
R-IB IL SSI-IN-PAC Rexroth Inline-Klemme für Absolutwertgeber mit SSI-Schnittstelle	Datenblatt R911170583	IL_Ssiln auf Seite 684
	Klemme R911171514	
R-IB IL INC-IN-PAC Rexroth Inline-Klemme für Inkrementalwertgeber	Datenblatt R911170487	IL_IncIn auf Seite 689
	Klemme R911308491	

Tab. 14-3: Funktionsmodule und ihre Funktionsbausteine

Kommunikationsmodule		Funktionsbaustein
R-IB IL RS232-PRO-PAC Rexroth Inline-Klemme zur seriellen Datenübertragung	Datenblatt R911170479	IL_RsCom auf Seite 694
	Klemme R911170440	

Kommunikationsmodule		Funktionsbaustein
R-IB IL RS485/422-PRO-PAC Rexroth Inline-Klemme zur seriellen Datenübertragung	Datenblatt R911170481	IL_RsCom auf Seite 694
	Klemme R911170442	
R-IB IL RS UNI-PAC Rexroth Inline-Klemme zur seriellen Datenübertragung 1 serieller Ein- und Ausgabekanal in RS-485/422- oder RS-232-Ausführung	Datenblatt R911338450	IL_RsCom auf Seite 694
	Klemme R911173635	

Tab. 14-4: Kommunikationsmodule und ihre Funktionsbausteine

INLINEIO_TABLE 16#0190 Die Funktionsbausteine nutzen die Fehlertabelle INLINEIO_TABLE, 16#0190.

14.2 Analoge Eingangsklemmen

14.2.1 IL_AI2Sf

Kurzbeschreibung Der Funktionsbaustein **IL_AI2Sf** konfiguriert die Inline-Klemme **R-IB IL AI 2/SF** mit zwei analogen Eingangskanälen und normalisiert die von der Klemme gelesenen Daten bezüglich vom Anwender definierbarer Grenzwerte.

Für jede Inline-EA-Klemme wird eine Funktionsbausteininstanz benötigt.

Schnittstellenbeschreibung

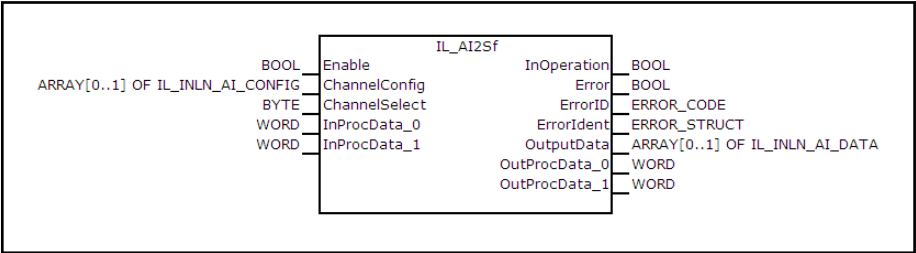


Abb. 14-1: Funktionsbaustein IL_AI2Sf

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Kontinuierliche Bearbeitung des FBs, solange "Enable" TRUE ist
	ChannelConfig	ARRAY[0..1] OF IL_INLN_AI_CONFIG	Enthält die Konfigurationsdaten für jeden einzelnen Kanal, siehe " IL_INLN_AI_CONFIG " auf Seite 577
	ChannelSelect	BYTE	Kanalauswahl, BYTE#2#000000yx; x - Kanal 1 y - Kanal 2
	InProcData_0	WORD	Prozesseingangswort 0
	InProcData_1	WORD	Prozesseingangswort 1

	Name	Typ	Beschreibung
VAR OUTPUT	InOperation	BOOL	Wenn TRUE, arbeitet der Baustein in seiner aktuellen Task
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb der FB-Abarbeitung eingetreten ist
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Detailinformationen zum Fehler, siehe "Fehlerbehandlung" auf Seite 580
	OutputData	ARRAY[0..1] OF IL_INLN_AI_DATA	Enthält für jeden Kanal den normalisierten Wert und den Diagnosecode, siehe "IL_INLN_AI_DATA" auf Seite 578
	OutProcData_0	WORD	Prozessausgangswort 0
	OutProcData_1	WORD	Prozessausgangswort 1

Tab. 14-5: Schnittstelle IL_AI2Sf

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	kontinuierlich
ChannelSelect	BYTE	0	3	3	Positive Flanke an "Enable"

Tab. 14-6: Eingangsverhalten IL_AI2Sf

Funktionsbeschreibung

Aktivieren Sie die FB-Instanz erst nach dem ersten SPS-Zyklus durch eine positive Flanke am **Enable**-Eingang.

Um den jeweiligen Kanal zu parametrieren, muss die entsprechende Struktur **IL_INLN_AI_CONFIG** mit den erforderlichen Werten belegt sein. Mit der positiven Flanke am **Enable**-Eingang wird der Kanal konfiguriert. Der erfolgreiche Abschluss der Konfiguration wird durch **InOperation** = TRUE angezeigt.

Eine Rekonfiguration kann nur durch eine erneute positive Flanke am **Enable**-Eingang ausgelöst werden.

Bei ungültigen Eingangsparametern generiert der FB einen Fehler und die entsprechenden Informationen gelangen an **Error**, **ErrorID** und **ErrorIdent**.

Die Initialwerte des Funktionsbausteins sind abgestimmt mit den Initialwerten der Analogklemme.

Um Messwerte oder Diagnoseinformationen der Kanäle 1 und 2 der Klemme zu erhalten, muss **ChannelSelect** entsprechend eingestellt sein.

Voreingestellt sind beide Kanäle ausgewählt. Die aktivierten Bits des **ChannelSelect** erlauben das Lesen der Kanäle. Die Bits 0 und 1 korrespondieren mit den Kanälen 1 und 2.

Die gemessenen Werte sind normalisiert bezüglich der vom Anwender vorgegebenen Bereichsgrenzen und werden in **NormalizedValue** angezeigt. Falls ein Fehler dabei auftreten sollte, wird dieser für den entsprechenden Kanal in **DiagCode** angezeigt.

IL_INLN_AI_CONFIG

Die Struktur enthält die Informationen zur Konfiguration eines Kanals. Der Kanal ist durch **HighLimit**, **LowLimit**, **NoOfSamples** und **InputRange** definiert.

Die Konfiguration kann wie folgt erfolgen:

ParameterName [Channel Number].[Function].

z. B. zum Konfigurieren von **InputRange** für den ersten Aufruf des Kanals 1:

ChannelConfig [0].InputRange.

Name	Typ	Beschreibung	Min-Wert	Max-Wert	Default-Wert	Übernahme
HighLimit	REAL	Obere Grenze des normalisierten Wertes	-65535.0	65535.0	10.0	kontinuierlich mit "Enable"
LowLimit	REAL	Untere Grenze des normalisierten Wertes	-65535.0	65535.0	0.0	kontinuierlich mit "Enable"
NoOfSamples	BYTE	0 – keine Mittelwertbildung 16 – Mittelwertbildung über 16 Messwerte	0	16	16	Positive Flanke an "Enable"
InputRange	IL_INLN_AI_RANGE	Messbereich, siehe "IL_INLN_AI_RANGE" auf Seite 579	UNIPO-LAR10V	UNIPO-LAR4TO 20MA	UNIPOLAR10V	Positive Flanke an "Enable"

Tab. 14-7: Zulässige Werte von IL_INLN_AI_CONFIG für IL_AI2Sf

IL_INLN_AI_DATA

Die Struktur enthält die Ausgabeparameter eines Kanals.

Die Parameter sind für jeden einzelnen Kanal wie folgt zu ermitteln:

ParameterName[Channel_Number].[Function]

z.B. OutputData[0].NormalizedValue.

Name	Typ	Beschreibung
NormalizedValue	REAL	Normalisierter Wert bezogen auf oberen und unteren Grenzwert
DiagCode	IL_INLN_DIAG_CODE	Diagnosecode, siehe "IL_INLN_DIAG_CODE" auf Seite 578

Tab. 14-8: Strukturelemente IL_INLN_AI_DATA

IL_INLN_DIAG_CODE

Die Tabelle enthält kanalbezogene Fehler und Diagnosen, die von den Funktionsbausteinen der Bibliothek ausgegeben werden.

DiagCode	ENUM	Beschreibung
16#8001	OVERRANGE	Bereichsüberschreitung
16#8002	OPEN_CIRCUIT	Offener Stromkreis, der Ausgang ist als Stromquelle konfiguriert und der vorgegebene Strom kann nicht fließen
16#8003	OVER_LOAD	Kurzschluß/Überlast, der Ausgang ist als Spannungsquelle konfiguriert und überlastet
16#8004	MEASURED_VALUE_INVALID	Ungültiger Messwert / Messwert nicht verfügbar
16#8008	COLD_JUNCTION_DEFECTIVE	Vergleichsstelle defekt
16#8010	INVALID_CONFIGURATION	Ungültige Konfiguration
16#8020	ERROR_IN_IO_SUPPLY_VOLTAGE	Fehler in der Spannungsversorgung
16#8040	MODULE_FAULTY	Modulfehler
16#8080	UNDERRANGE	Bereichsunterschreitung

Tab. 14-9: Enum IL_INLN_DIAG_CODE

IL_INLN_AI_RANGE Die Aufzählung enthält die zu Konfiguration eines Kanals möglichen Messbereiche.

ENUM	Beschreibung
UNIPOLAR10V	Spannungsmessung 0 ...10 V
BIPOLAR10V	Spannungsmessung -10V ... +10 V
UNIPOLAR20MA	Strommessung 0 ... 20mA
BIPOLAR20MA	Strommessung -20mA +20mA
UNIPOLAR4TO20MA	Strommessung 4 ... 20mA

Tab. 14-10: Zulässige Werte von IL_INLN_AI_RANGE für IL_AI2Sf

Beispiel In nachstehendem Beispiel sind Kanal 1 konfiguriert für einen Bereich von 0 ...10V und Kanal 2 für einen Bereich von 4 ... 20mA.

Deklaration zum Beispiel

```
PROGRAM PrgIL_AI2Sf
VAR
  fbIL_AI2Sf          : IL_AI2Sf;
  rNormValCh1         : REAL;
  rNormValCh2         : REAL;
  enDiagCodeCh1       : IL_INLN_DIAG_CODE;
  enDiagCodeCh2       : IL_INLN_DIAG_CODE;
  wInProcessData_0    : WORD;
  wInProcessData_1    : WORD;
  wOutProcessData_0   : WORD;
  wOutProcessData_1   : WORD;
END_VAR
```

Implementation zum Beispiel

```
// Configuration
fbIL_AI2Sf.ChannelConfig[0].HighLimit := REAL#10.0 ;
fbIL_AI2Sf.ChannelConfig[0].LowLimit := REAL#0.0 ;
fbIL_AI2Sf.ChannelConfig[0].InputRange := IL_INLN_AI_RANGE.UNIPOLAR10V ;
fbIL_AI2Sf.ChannelConfig[0].NoOfSamples := BYTE#16 ;
fbIL_AI2Sf.ChannelConfig[1].HighLimit := REAL#20.0 ;
fbIL_AI2Sf.ChannelConfig[1].LowLimit := REAL#4.0 ;
fbIL_AI2Sf.ChannelConfig[1].InputRange := IL_INLN_AI_RANGE.UNIPOLAR4TO20MA ;
fbIL_AI2Sf.ChannelConfig[1].NoOfSamples := BYTE#16 ;
fbIL_AI2Sf.InProcData_0 := wInProcessData_0 ;
fbIL_AI2Sf.InProcData_1 := wInProcessData_1 ;
wOutProcessData_0 := fbIL_AI2Sf.OutProcData_0 ;
wOutProcessData_1 := fbIL_AI2Sf.OutProcData_1 ;
fbIL_AI2Sf.ChannelSelect := BYTE#2#00000011 ;

// FB call
fbIL_AI2Sf(Enable:=TRUE);

// Read data
IF fbIL_AI2Sf.InOperation = TRUE THEN
  rNormValCh1 := fbIL_AI2Sf.OutputData[0].NormalizedValue ;
  rNormValCh2 := fbIL_AI2Sf.OutputData[1].NormalizedValue ;
  enDiagCodeCh1 := fbIL_AI2Sf.OutputData[0].DiagCode ;
  enDiagCodeCh2 := fbIL_AI2Sf.OutputData[1].DiagCode ;
END_IF
```

Prozessdaten-Interface zum Beispiel

Die Prozesseingangs- und -ausgangsworte des Funktionsbausteines sind auf die Prozessdatenworte des einzelnen Feldbussystems zu mappen.

Die Variablen wInProcessData_x und wOutProcessData_x sind im Programm prgIL_AI2Sf deklariert.

Inline-Module I/O Mapping				
Channels				
Variable	Mapping	Channel	Address	Type
		Input0	%IW2	
Application.PrgIL_AI2Sf.wInProcessData_0		Word0	%IW2	WORD
Application.PrgIL_AI2Sf.wInProcessData_1		Word1	%IW4	WORD
		Output0	%QW2	
Application.PrgIL_AI2Sf.wOutProcessData_0		Word0	%QW2	WORD
Application.PrgIL_AI2Sf.wOutProcessData_1		Word1	%QW4	WORD

Abb. 14-2: Prozessdaten-Interface

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle INLINEIO_TABLE, 16#0190. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00003003	16#00000000	Ungültiger 'ChannelSelect'-Eingang
INPUT_INVALID_ERROR (16#0001)	16#00003004	Kanalnummer	'InputRange'-Eingang eines Kanals ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003005	Kanalnummer	'NoOfSamples'-Eingang eines Kanals ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003009	Kanalnummer	Ungültige Bereichsgrenzen HighLimit <= LowLimit.
DEVICE_ERROR (16#0008)	16#00003020	16#000x8010	Ungültige Konfiguration, Additional2 'x' zeigt die Kanalnummer an

Tab. 14-11: IL_AI2Sf Fehlercodes

14.2.2 IL_AI4Ef

Kurzbeschreibung Der Funktionsbaustein **IL_AI4Ef** konfiguriert die Inline-Klemme **R-IB IL AI 4/EF** mit vier analogen Eingangskanälen und normalisiert die von der Klemme gelesenen Daten bezüglich vom Anwender definierbarer Grenzwerte. Für jede Inline-EA-Klemme wird eine Funktionsbausteininstanz benötigt.

Kompatibilität Der Funktionsbaustein ist verwendbar für Inline- und Profibus.

Schnittstellenbeschreibung

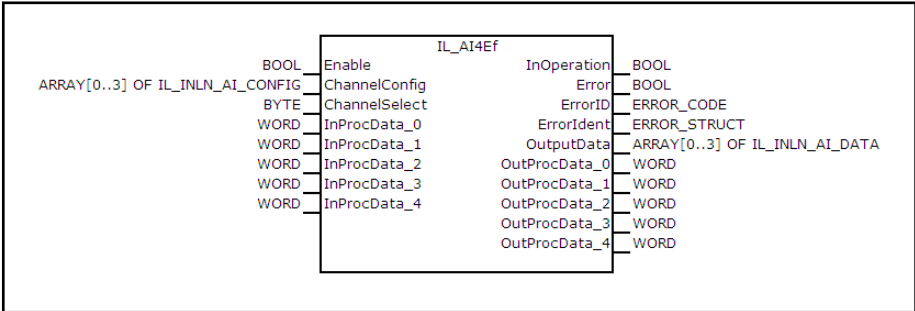


Abb. 14-3: Funktionsbaustein IL_AI4Ef

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Kontinuierliche Bearbeitung des FBs, solange "Enable" TRUE ist
	ChannelConfig	ARRAY[0..3] OF IL_INLN_AI_CONFIG	Enthält die Konfigurationsdaten für jeden einzelnen Kanal, siehe "IL_INLN_AI_CONFIG" auf Seite 582
	ChannelSelect	BYTE	Kanalauswahl, Byte#2#00001111; Individuelle Aktivierung des automatischen Lesens der Kanäle (Bit 0...Bit 3 korrespondieren mit Kanal 1...4)
	InProcData_0	WORD	Prozesseingangswort 0
	InProcData_1	WORD	Prozesseingangswort 1
	InProcData_2	WORD	Prozesseingangswort 2
	InProcData_3	WORD	Prozesseingangswort 3
	InProcData_4	WORD	Prozesseingangswort 4
VAR OUTPUT	InOperation	BOOL	Wenn TRUE, arbeitet der Baustein in seiner aktuellen Task
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb der FB-Abarbeitung eingetreten ist
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Detaillinformationen zum Fehler, siehe "Fehlerbehandlung" auf Seite 585
	OutputData	ARRAY[0..3] OF IL_INLN_AI_DATA	Enthält für jeden Kanal den normalisierten Wert und den Diagnosecode, siehe "IL_INLN_AI_DATA" auf Seite 582
	OutProcData_0	WORD	Prozessausgangswort 0
	OutProcData_1	WORD	Prozessausgangswort 1
	OutProcData_2	WORD	Prozessausgangswort 2
	OutProcData_3	WORD	Prozessausgangswort 3
	OutProcData_4	WORD	Prozessausgangswort 4

Tab. 14-12: Schnittstelle IL_AI4Ef

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	kontinuierlich
ChannelSelect	BYTE	0	15	15	Positive Flanke an "Enable"

Tab. 14-13: Eingangsverhalten IL_AI4Ef

Funktionsbeschreibung

Aktivieren Sie die FB-Instanz erst nach dem ersten SPS-Zyklus durch eine positive Flanke am **Enable**-Eingang.

Um den jeweiligen Kanal zu parametrieren, muss die entsprechende Struktur **IL_INLN_AI_CONFIG** mit den erforderlichen Werten belegt sein. Mit der positiven Flanke am **Enable**-Eingang wird der Kanal konfiguriert. Der erfolgreiche Abschluss der Konfiguration wird durch **InOperation** = TRUE angezeigt.

Eine Rekonfiguration kann nur durch eine erneute positive Flanke am **Enable**-Eingang ausgelöst werden.

Bei ungültigen Eingangsparametern generiert der FB einen Fehler und die entsprechenden Informationen gelangen an **Error**, **ErrorID** und **ErrorIdent**.

Die Initialwerte des Funktionsbausteins sind abgestimmt mit den Initialwerten der Analogklemme.

Um Messwerte oder Diagnoseinformationen der Kanäle 1 .. 4 der Klemme zu erhalten, muss **ChannelSelect** entsprechend eingestellt sein.

Die aktivierten Bits des **ChannelSelect** erlauben das Lesen der Kanäle. Bit 0 korrespondiert mit Kanal 1 bis zu Bit 3 mit Kanal 4.

Voreingestellt sind alle Kanäle ausgewählt.

Die gemessenen Werte sind normalisiert bezüglich der vom Anwender vorgegebenen Bereichsgrenzen und werden in **NormalizedValue** angezeigt. Falls ein Fehler dabei auftreten sollte, wird dieser für den entsprechenden Kanal in **DiagCode** angezeigt.

IL_INLN_AI_CONFIG

Die Struktur enthält die Informationen zur Konfiguration eines Kanals. Der Kanal ist durch **HighLimit**, **LowLimit**, **NoOfSamples** und **InputRange** definiert.

Die Konfiguration kann wie folgt erfolgen:

```
ParameterName[Channel_Number].[Function].
```

z. B. zum Konfigurieren von **InputRange** für den ersten Aufruf des Kanals 1:

```
ChannelConfig[0].InputRange.
```

Name	Typ	Beschreibung	Min-Wert	Max-Wert	Default-Wert	Übernahme
HighLimit	REAL	Obere Grenze des normalisierten Wertes	-65535.0	65535.0	10.0	kontinuierlich mit "Enable"
LowLimit	REAL	Untere Grenze des normalisierten Wertes	-65535.0	65535.0	0.0	kontinuierlich mit "Enable"
NoOfSamples	BYTE	0 – keine Mittelwertbildung 4 – Mittelwertbildung über 4 Messwerte 16 – Mittelwertbildung über 16 Messwerte 32 – Mittelwertbildung über 32 Messwerte	0	32	16	Positive Flanke an "Enable"
InputRange	IL_INLN_AI_RANGE	Messbereich, siehe "IL_INLN_AI_RANGE" auf Seite 583	UNIPO-LAR10V	UNIPO-LAR4TO 20MA	UNIPOLAR10V	Positive Flanke an "Enable"

Tab. 14-14: Zulässige Werte von IL_INLN_AI_CONFIG für IL_AI4Ef

IL_INLN_AI_DATA

Die Struktur enthält die Ausgabeparameter eines Kanals.

Die Parameter sind für jeden einzelnen Kanal wie folgt zu ermitteln:

```
ParameterName[Channel_Number].[Function]
```

z.B. `OutputData[0].NormalizedValue`.

Name	Typ	Beschreibung
NormalizedValue	REAL	Normalisierter Wert zwischen dem oberen und dem unteren Grenzwert
DiagCode	IL_INLN_DIAG_CODE	Diagnosecode, siehe "IL_INLN_DIAG_CODE" auf Seite 583

Tab. 14-15: Strukturelemente IL_INLN_AI_DATA

IL_INLN_DIAG_CODE Die Tabelle enthält kanalbezogene Fehler und Diagnosen, die von den Funktionsbausteinen der Bibliothek ausgegeben werden.

DiagCode	ENUM	Beschreibung
16#8001	OVERRANGE	Bereichsüberschreitung
16#8002	OPEN_CIRCUIT	Offener Stromkreis, der Ausgang ist als Stromquelle konfiguriert und der vorgegebene Strom kann nicht fließen
16#8003	OVER_LOAD	Kurzschluß/Überlast, der Ausgang ist als Spannungsquelle konfiguriert und überlastet
16#8004	MEASURED_VALUE_INVALID	Ungültiger Messwert / Messwert nicht verfügbar
16#8008	COLD_JUNCTION_DEFECTIVE	Vergleichsstelle defekt
16#8010	INVALID_CONFIGURATION	Ungültige Konfiguration
16#8020	ERROR_IN_IO_SUPPLY_VOLTAGE	Fehler in der Spannungsversorgung
16#8040	MODULE_FAULTY	Modulfehler
16#8080	UNDERRANGE	Bereichsunterschreitung

Tab. 14-16: Enum IL_INLN_DIAG_CODE

IL_INLN_AI_RANGE Die Aufzählung enthält die zu Konfiguration eines Kanals möglichen Messbereiche.

ENUM	Beschreibung
UNIPOLAR10V	Spannungsmessung 0 ... 10 V
BIPOLAR10V	Spannungsmessung -10V ... +10 V
UNIPOLAR5V	Spannungsmessung 0 ... 5V
BIPOLAR5V	Spannungsmessung -5 ... +5V
UNIPOLAR20MA	Strommessung 0 ... 20mA
BIPOLAR20MA	Strommessung -20mA ... +20mA
UNIPOLAR4TO20MA	Strommessung 4 ... 20mA

Tab. 14-17: Zulässige Werte von IL_INLN_AI_RANGE für IL_AI4Ef

Beispiel In nachstehendem Beispiel sind Kanal 1 konfiguriert für einen Bereich von +/- 10V und Kanal 2 für einen Bereich von 0 .. 20mA. Die Daten sind eingeordnet zwischen der oberen und der unteren Grenze und als normalisierte Variablenwerte verfügbar.

Deklaration zum Beispiel

```
PROGRAM PrgIL_AI4Ef
VAR
    fbIL_AI4Ef: IL_AI4Ef;
    wInProcessData_0: WORD;
```

RIL_InlineIOModules.library

```

wInProcessData_1: WORD;
wInProcessData_2: WORD;
wInProcessData_3: WORD;
wInProcessData_4: WORD;
wOutProcessData_0: WORD;
wOutProcessData_1: WORD;
wOutProcessData_2: WORD;
wOutProcessData_3: WORD;
wOutProcessData_4: WORD;
rNormalizedValue_0: REAL;
rNormalizedValue_1: REAL;
enDiagCode_0: IL_INLN_DIAG_CODE;
enDiagCode_1: IL_INLN_DIAG_CODE;
END_VAR

```

Implementation zum Beispiel

```

// Configuration of the channels 1 and 2
fbIL_AI4Ef.ChannelConfig[0].InputRange := IL_INLN_AI_RANGE.BIPOLAR10V;
fbIL_AI4Ef.ChannelConfig[0].NoOfSamples := BYTE#32;
fbIL_AI4Ef.ChannelConfig[0].HighLimit := REAL#10.0;
fbIL_AI4Ef.ChannelConfig[0].LowLimit := REAL#-10.0;
fbIL_AI4Ef.ChannelConfig[1].InputRange := IL_INLN_AI_RANGE.UNIPOLAR20MA;
fbIL_AI4Ef.ChannelConfig[1].NoOfSamples := BYTE#16;
fbIL_AI4Ef.ChannelConfig[1].HighLimit := REAL#20.0;
fbIL_AI4Ef.ChannelConfig[1].LowLimit := REAL#0.0;
fbIL_AI4Ef.ChannelSelect := BYTE#2#00000011;
fbIL_AI4Ef.InProcData_0 := wInProcessData_0;
fbIL_AI4Ef.InProcData_1 := wInProcessData_1;
fbIL_AI4Ef.InProcData_2 := wInProcessData_2;
fbIL_AI4Ef.InProcData_3 := wInProcessData_3;
fbIL_AI4Ef.InProcData_4 := wInProcessData_4;
wOutProcessData_0 := fbIL_AI4Ef.OutProcData_0;
wOutProcessData_1 := fbIL_AI4Ef.OutProcData_1;
wOutProcessData_2 := fbIL_AI4Ef.OutProcData_2;
wOutProcessData_3 := fbIL_AI4Ef.OutProcData_3;
wOutProcessData_4 := fbIL_AI4Ef.OutProcData_4;

// FB call
fbIL_AI4Ef(Enable:= TRUE);

// Read data
IF fbIL_AI4Ef.InOperation = TRUE THEN
  rNormalizedValue_0 := fbIL_AI4Ef.OutputData[0].NormalizedValue;
  rNormalizedValue_1 := fbIL_AI4Ef.OutputData[1].NormalizedValue;
  enDiagCode_0 := fbIL_AI4Ef.OutputData[0].DiagCode;
  enDiagCode_1 := fbIL_AI4Ef.OutputData[1].DiagCode;
END_IF

```

Prozessdaten-Interface zum Beispiel

Die Prozesseingangs- und -ausgangsworte des Funktionsbausteines sind auf die Prozessdatenworte des einzelnen Feldbussystems zu mappen.

Die Variablen wInProcessData_x und wOutProcessData_x sind im Programm prgIL_AI4Ef deklariert.

Variable	Mapping	Channel	Address	Type
Application.PrgIL_AI4Ef.wInProcessData_0		Input0	%IW10	
Application.PrgIL_AI4Ef.wInProcessData_1		Word0	%IW12	WORD
Application.PrgIL_AI4Ef.wInProcessData_2		Word1	%IW14	WORD
Application.PrgIL_AI4Ef.wInProcessData_3		Word2	%IW16	WORD
Application.PrgIL_AI4Ef.wInProcessData_4		Word3	%IW18	WORD
Application.PrgIL_AI4Ef.wOutProcessData_0		Output0	%QW10	
Application.PrgIL_AI4Ef.wOutProcessData_1		Word0	%QW12	WORD
Application.PrgIL_AI4Ef.wOutProcessData_2		Word1	%QW14	WORD
Application.PrgIL_AI4Ef.wOutProcessData_3		Word2	%QW16	WORD
Application.PrgIL_AI4Ef.wOutProcessData_4		Word3	%QW18	WORD

Abb. 14-4: Prozessdaten-Interface

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle INLINEIO_TABLE, 16#0190. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00003003	16#00000000	Ungültiger 'ChannelSelect'-Eingang
INPUT_INVALID_ERROR (16#0001)	16#00003004	Kanalnummer	'InputRange'-Eingang eines Kanals ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003005	Kanalnummer	'NoOfSamples'-Eingang eines Kanals ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003009	Kanalnummer	Ungültige Bereichsgrenzen HighLimit <= LowLimit
DEVICE_ERROR (16#0008)	16#00003020	16#00000000	Zeitüberschreitung, Gerät antwortet nicht
DEVICE_ERROR (16#0008)	16#00003020	16#00008010	Ungültige Konfiguration

Tab. 14-18: IL_AI4Ef Fehlercodes

14.2.3 IL_AI4IU

Kurzbeschreibung Der Funktionsbaustein **IL_AI4IU** konfiguriert die Inline-Klemmen **R-IB IL AI 4/I** und **R-IB IL AI 4/U** mit vier analogen Eingangskanälen und normalisiert die von den Klemmen gelesenen Daten bezüglich vom Anwender definierbarer Grenzwerte. .

Für jede Inline-EA-Klemme wird eine Funktionsbausteininstanz benötigt.

Kompatibilität Der Funktionsbaustein ist verwendbar für Inlinebus, Profibus und Profinet IO-Feldbus.

Schnittstellenbeschreibung

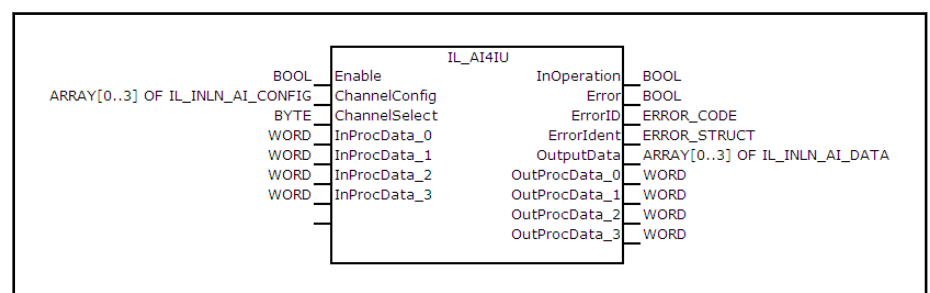


Abb. 14-5: Funktionsbaustein IL_AI4IU

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Kontinuierliche Bearbeitung des FBs, solange "Enable" TRUE ist
	ChannelConfig	ARRAY[0..3] OF IL_INLN_AI_CONFIG	Enthält die Konfigurationsdaten für jeden einzelnen Kanal, siehe " IL_INLN_AI_CONFIG " auf Seite 587
	ChannelSelect	BYTE	Byte#2#0000xxxx; Individuelle Aktivierung des automatischen Lesens der Kanäle (Bit 0...Bit 3 korrespondieren mit Kanal 1...4)
	InProcData_0	WORD	Prozesseingangswort 0
	InProcData_1	WORD	Prozesseingangswort 1
	InProcData_2	WORD	Prozesseingangswort 2
	InProcData_3	WORD	Prozesseingangswort 3
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, arbeitet der Baustein in seiner aktuellen Task
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb der FB-Abarbeitung eingetreten ist
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Detailinformationen zum Fehler, siehe " Fehlerbehandlung " auf Seite 589
	OutputData	ARRAY[0..3] OF IL_INLN_AI_DATA	Enthält für jeden Kanal den normalisierten Wert und den Diagnosecode, siehe " IL_INLN_AI_DATA " auf Seite 587
	OutProcData_0	WORD	Prozessausgangswort 0
	OutProcData_1	WORD	Prozessausgangswort 1
	OutProcData_2	WORD	Prozessausgangswort 2
	OutProcData_3	WORD	Prozessausgangswort 3

Tab. 14-19: Schnittstelle IL_AI4IU

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	kontinuierlich
ChannelSelect	BYTE	1	15	15	Positive Flanke an "Enable"

Tab. 14-20: Eingangsverhalten IL_AI4IU

Funktionsbeschreibung

Aktivieren Sie die FB-Instanz erst nach dem ersten SPS-Zyklus durch eine positive Flanke am **Enable**-Eingang.

Um den jeweiligen Kanal zu parametrieren, muss die entsprechende Struktur **IL_INLN_AI_CONFIG** mit den erforderlichen Werten belegt sein. Mit der positiven Flanke am **Enable**-Eingang wird der Kanal konfiguriert. Der erfolgreiche Abschluss der Konfiguration wird durch **InOperation** = TRUE angezeigt.

Eine Rekonfiguration kann nur durch eine erneute positive Flanke am **Enable**-Eingang ausgelöst werden.

Bei ungültigen Eingangsparametern generiert der FB einen Fehler und die entsprechenden Informationen gelangen an **Error**, **ErrorID** und **ErrorIdent**.

Die Initialwerte des Funktionsbausteins sind abgestimmt mit den Initialwerten der Analogklemme.

Um Messwerte oder Diagnoseinformationen der Kanäle 1 .. 4 der Klemme zu erhalten, muss **ChannelSelect** entsprechend eingestellt sein.

Die aktivierten Bits des **ChannelSelect** erlauben das Lesen der Kanäle. Bit 0 korrespondiert mit Kanal 1 bis zu Bit 3 mit Kanal 4.

Voreingestellt sind alle Kanäle ausgewählt.

Die gemessenen Werte sind normalisiert bezüglich der vom Anwender vorgegebenen Bereichsgrenzen und werden in **NormalizedValue** angezeigt. Falls ein Fehler dabei auftreten sollte, wird dieser für den entsprechenden Kanal in **DiagCode** angezeigt.

IL_INLN_AI_CONFIG Die Struktur enthält die Informationen zur Konfiguration eines Kanals. Der Kanal ist durch **HighLimit**, **LowLimit**, **NoOfSamples** und **InputRange** definiert.

Die Konfiguration kann wie folgt erfolgen:

`ParameterName[Channel_Number].[Function].`

z. B. zum Konfigurieren von **InputRange** für den ersten Aufruf des Kanals 1:

`ChannelConfig[0].InputRange.`

Name	Typ	Beschreibung	Min-Wert	Max-Wert	Default-Wert	Übernahme
HighLimit	REAL	Obere Grenze des normalisierten Wertes	-65535.0	65535.0	10.0	kontinuierlich mit "Enable"
LowLimit	REAL	Untere Grenze des normalisierten Wertes	-65535.0	65535.0	0.0	kontinuierlich mit "Enable"
NoOfSamples	BYTE	0 – keine Mittelwertbildung 4 – Mittelwertbildung über 4 Messwerte 16 – Mittelwertbildung über 16 Messwerte 32 – Mittelwertbildung über 32 Messwerte	0	32	16	Positive Flanke an "Enable"
InputRange	IL_INLN_AI_RANGE	Messbereich, siehe "IL_INLN_AI_RANGE" auf Seite 588	UNIPO-LAR10V	UNIPO-LAR4TO20MA	UNIPO-LAR10V	Positive Flanke an "Enable"

Tab. 14-21: Zulässige Werte von **IL_INLN_AI_CONFIG** für **IL_AI4IU**

IL_INLN_AI_DATA Die Struktur enthält die Ausgabeparameter eines Kanals.

Die Parameter sind für jeden einzelnen Kanal wie folgt zu ermitteln:

`ParameterName[Channel_Number].[Function]`

z.B. `OutputData[0].NormalizedValue.`

Name	Typ	Beschreibung
NormalizedValue	REAL	Normalisierter Wert zwischen dem oberen und dem unteren Grenzwert
DiagCode	IL_INLN_DIAG_CODE	Diagnosecode, siehe "IL_INLN_DIAG_CODE" auf Seite 588

Tab. 14-22: Strukturelemente **IL_INLN_AI_DATA**

IL_INLN_DIAG_CODE Die Tabelle enthält kanalbezogene Fehler und Diagnosen, die von den Funktionsbausteinen der Bibliothek ausgegeben werden.

DiagCode	ENUM	Beschreibung
16#8001	OVERRANGE	Bereichsüberschreitung
16#8002	OPEN_CIRCUIT	Offener Stromkreis, der Ausgang ist als Stromquelle konfiguriert und der vorgegebene Strom kann nicht fließen
16#8003	OVER_LOAD	Kurzschluß/Überlast, der Ausgang ist als Spannungsquelle konfiguriert und überlastet
16#8004	MEASURED_VALUE_INVALID	Ungültiger Messwert / Messwert nicht verfügbar
16#8008	COLD_JUNCTION_DEFECTIVE	Vergleichsstelle defekt
16#8010	INVALID_CONFIGURATION	Ungültige Konfiguration
16#8020	ERROR_IN_IO_SUPPLY_VOLTAGE	Fehler in der Spannungsversorgung
16#8040	MODULE_FAULTY	Modulfehler
16#8080	UNDERRANGE	Bereichsunterschreitung

Tab. 14-23: Enum IL_INLN_DIAG_CODE

IL_INLN_AI_RANGE Die Aufzählung enthält die zu Konfiguration eines Kanals möglichen Messbereiche.

ENUM	Beschreibung
UNIPOLAR10V	Spannungsmessung 0 ... 10 V
BIPOLAR10V	Spannungsmessung -10V ... +10 V
UNIPOLAR20MA	Strommessung 0 ... 20mA
UNIPOLAR4TO20MA	Strommessung 4 ... 20mA

Tab. 14-24: Zulässige Werte von IL_INLN_AI_RANGE für IL_AI4IU

Beispiel In nachstehendem Beispiel sind Kanal 1 konfiguriert für einen Bereich von 0 ... +10V und Kanal 2 für einen Bereich von +/- 10V. Die Daten sind eingeordnet zwischen der oberen und der unteren Grenze und als normalisierte Variablenwerte verfügbar.

Deklaration zum Beispiel

```
PROGRAM PrgIL_AI4IU
VAR
    fbIL_AI4IU: IL_AI4IU;
    wInProcessData_0: WORD;
    wInProcessData_1: WORD;
    wInProcessData_2: WORD;
    wInProcessData_3: WORD;
    wOutProcessData_0: WORD;
    wOutProcessData_1: WORD;
    wOutProcessData_2: WORD;
    wOutProcessData_3: WORD;
    rNormalizedValue_0: REAL;
    rNormalizedValue_1: REAL;
    enDiagCode_0: IL_INLN_DIAG_CODE;
    enDiagCode_1: IL_INLN_DIAG_CODE;
END_VAR
```

Implementation zum Beispiel

```
// Configuration
fbIL_AI4IU.ChannelConfig[0].InputRange := IL_INLN_AI_RANGE.UNIPOLAR10V;
fbIL_AI4IU.ChannelConfig[0].NoOfSamples := BYTE#16;
```



```

fbIL_AI4IU.ChannelConfig[0].HighLimit := REAL#10.0;
fbIL_AI4IU.ChannelConfig[0].LowLimit := REAL#0.0;
fbIL_AI4IU.ChannelConfig[1].InputRange := IL_INLN_AI_RANGE.BIPOLAR10V;
fbIL_AI4IU.ChannelConfig[1].NoOfSamples := BYTE#16;
fbIL_AI4IU.ChannelConfig[1].HighLimit := REAL#10.0;
fbIL_AI4IU.ChannelConfig[1].LowLimit := REAL#-10.0;
fbIL_AI4IU.ChannelSelect := BYTE#2#00000011;
fbIL_AI4IU.InProcData_0 := wInProcessData_0;
fbIL_AI4IU.InProcData_1 := wInProcessData_1;
fbIL_AI4IU.InProcData_2 := wInProcessData_2;
fbIL_AI4IU.InProcData_3 := wInProcessData_3;
wOutProcessData_0 := fbIL_AI4IU.OutProcData_0;
wOutProcessData_1 := fbIL_AI4IU.OutProcData_1;
wOutProcessData_2 := fbIL_AI4IU.OutProcData_2;
wOutProcessData_3 := fbIL_AI4IU.OutProcData_3;

// FB call
fbIL_AI4IU(Enable:=TRUE);

// Read data
IF fbIL_AI4IU.InOperation = TRUE THEN
  rNormalizedValue_0 := fbIL_AI4IU.OutputData[0].NormalizedValue;
  rNormalizedValue_1 := fbIL_AI4IU.OutputData[1].NormalizedValue;
  enDiagCode_0 := fbIL_AI4IU.OutputData[0].DiagCode;
  enDiagCode_1 := fbIL_AI4IU.OutputData[1].DiagCode;
END_IF

```

Prozessdaten-Interface zum Beispiel

Die Prozesseingangs- und -ausgangswerte des Funktionsbausteines sind auf die Prozessdatenworte des einzelnen Feldbussystems zu mappen.

Die Variablen wInProcessData_x und wOutProcessData_x sind im Programm prgIL_AI4IU deklariert.

Variable	Mapping	Channel	Address	Type
Application.PrgIL_AI4IU.wInProcessData_0		Input0	%IW64	WORD
Application.PrgIL_AI4IU.wInProcessData_1		Word0	%IW64	WORD
Application.PrgIL_AI4IU.wInProcessData_2		Word1	%IW66	WORD
Application.PrgIL_AI4IU.wInProcessData_3		Word2	%IW68	WORD
Application.PrgIL_AI4IU.wInProcessData_3		Word3	%IW70	WORD
Application.PrgIL_AI4IU.wOutProcessData_0		Output0	%QW66	WORD
Application.PrgIL_AI4IU.wOutProcessData_1		Word0	%QW66	WORD
Application.PrgIL_AI4IU.wOutProcessData_2		Word1	%QW68	WORD
Application.PrgIL_AI4IU.wOutProcessData_3		Word2	%QW70	WORD
Application.PrgIL_AI4IU.wOutProcessData_3		Word3	%QW72	WORD

Abb. 14-6: Prozessdaten-Interface

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle INLINEIO_TABLE, 16#0190. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00003003	16#00000000	Ungültiger 'ChannelSelect'-Eingang
INPUT_INVALID_ERROR (16#0001)	16#00003004	Kanalnummer	'InputRange'-Eingang eines Kanals ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003005	Kanalnummer	'NoOfSamples'-Eingang eines Kanals ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003009	Kanalnummer	Ungültige Bereichsgrenzen HighLimit <= LowLimit
DEVICE_ERROR (16#0008)	16#00003020	16#000x8010	Ungültige Konfiguration

Tab. 14-25: IL_AI4IU Fehlercodes

14.2.4 IL_AI8Sfls

Kurzbeschreibung Der Funktionsbaustein **IL_AI8Sfls** konfiguriert die Inline-Klemmen **R-IB IL AI8/SF** und **R-IB IL AI8/IS** mit acht analogen Eingangskanälen und normalisiert die von der Klemme gelesenen Daten bezüglich vom Anwender definierbarer Grenzwerte.

Für jede Inline-EA-Klemme wird eine Funktionsbausteininstanz benötigt.

Kompatibilität Der Funktionsbaustein ist verwendbar für Inlinebus, Profibus und Profinet IO-Feldbus.

Schnittstellenbeschreibung

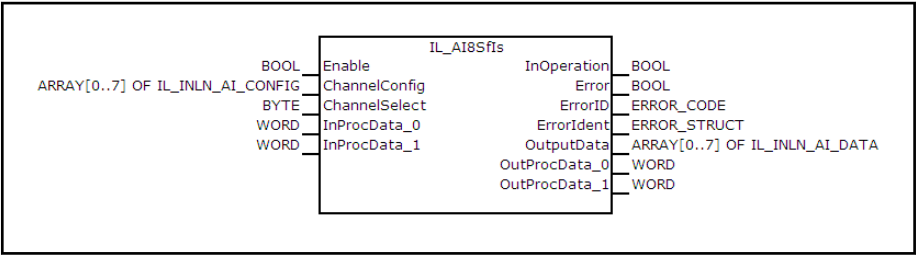


Abb. 14-7: Funktionsbaustein IL_AI8Sfls

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Kontinuierliche Bearbeitung des FBs, solange "Enable" TRUE ist
	ChannelConfig	ARRAY[0..7] OF IL_INLN_AI_CONFIG	Enthält die Konfigurationsdaten für jeden einzelnen Kanal, siehe " IL_INLN_AI_CONFIG " auf Seite 591
	ChannelSelect	BYTE	Kanalauswahl, Byte#2#11111111; Individuelle Aktivierung des automatischen Lesens der Kanäle (Bit 0...Bit 7 korrespondieren mit Kanal 1...8)
	InProcData_0	WORD	Prozesseingangswort 0
	InProcData_1	WORD	Prozesseingangswort 1
VAR OUTPUT	InOperation	BOOL	Wenn TRUE, arbeitet der Baustein in seiner aktuellen Task
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb der FB-Abarbeitung eingetreten ist
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Detailinformationen zum Fehler, siehe " Fehlerbehandlung " auf Seite 594
	OutputData	ARRAY[0..7] OF IL_INLN_AI_DATA	Enthält für jeden Kanal den normalisierten Wert und den Diagnosecode, siehe " IL_INLN_AI_DATA " auf Seite 592
	OutProcData_0	WORD	Prozessausgangswort 0
	OutProcData_1	WORD	Prozessausgangswort 1

Tab. 14-26: Schnittstelle IL_AI8Sfls

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	kontinuierlich
ChannelSelect	BYTE	1	255	255	Positive Flanke an "Enable"

Tab. 14-27: Eingangsverhalten IL_AI8Sfls

Funktionsbeschreibung

Aktivieren Sie die FB-Instanz erst nach dem ersten SPS-Zyklus durch eine positive Flanke am **Enable**-Eingang.

Um den jeweiligen Kanal zu parametrieren, muss die entsprechende Struktur **IL_INLN_AI_CONFIG** mit den erforderlichen Werten belegt sein. Mit der positiven Flanke am **Enable**-Eingang wird der Kanal konfiguriert. Der erfolgreiche Abschluss der Konfiguration wird durch **InOperation** = TRUE angezeigt.

Eine Rekonfiguration kann nur durch eine erneute positive Flanke am **Enable**-Eingang ausgelöst werden.

Bei ungültigen Eingangsparametern generiert der FB einen Fehler und die entsprechenden Informationen gelangen an **Error**, **ErrorID** und **ErrorIdent**.

Die Initialwerte des Funktionsbausteins sind abgestimmt mit den Initialwerten der Analogklemme.

Um Messwerte oder Diagnoseinformationen der Kanäle 1 .. 8 der Klemme zu erhalten, muss **ChannelSelect** entsprechend eingestellt sein.

Die aktivierten Bits des **ChannelSelect** erlauben das Lesen der Kanäle. Bit 0 korrespondiert mit Kanal 1 bis zu Bit 7 mit Kanal 8.

Voreingestellt sind alle Kanäle ausgewählt.

Die gemessenen Werte sind normalisiert bezüglich der vom Anwender vorgegebenen Bereichsgrenzen und werden in **NormalizedValue** angezeigt. Falls ein Fehler dabei auftreten sollte, wird dieser für den entsprechenden Kanal in **DiagCode** angezeigt.



R-IB IL AI8/SF und R-IB IL AI8/IS arbeiten multiplex, d.h. die Daten werden multiplex gelesen. Das Update der Daten aller Kanäle benötigt daher mehr als einen SPS-Zyklus.

IL_INLN_AI_CONFIG

Die Struktur enthält die Informationen zur Konfiguration eines Kanals. Der Kanal ist durch **HighLimit**, **LowLimit**, **NoOfSamples** und **InputRange** definiert.

Die Konfiguration kann wie folgt erfolgen:

`ParameterName[Channel Number].[Function].`

z. B. zum Konfigurieren von **InputRange** für den ersten Aufruf des Kanals 1:

`ChannelConfig[0].InputRange.`

Name	Typ	Beschreibung	Min-Wert	Max-Wert	Default-Wert	Übernahme
HighLimit	REAL	Obere Grenze des normalisierten Wertes	-65535.0	65535.0	10.0	kontinuierlich mit "Enable"
LowLimit	REAL	Untere Grenze des normalisierten Wertes	-65535.0	65535.0	0.0	kontinuierlich mit "Enable"

Name	Typ	Beschreibung	Min-Wert	Max-Wert	Default-Wert	Übernahme
NoOfSamples	BYTE	0 – keine Mittelwertbildung 4 – Mittelwertbildung über 4 Messwerte 16 – Mittelwertbildung über 16 Messwerte 32 – Mittelwertbildung über 32 Messwerte	0	32	16	Positive Flanke an "Enable"
InputRange	IL_INLN_AI_RANGE	Messbereich, siehe "IL_INLN_AI_RANGE" auf Seite 592	UNIPO-LAR10V	BIPO-LAR40MA	UNIPO-LAR10V	Positive Flanke an "Enable"

Tab. 14-28: Zulässige Werte von IL_INLN_AI_CONFIG für IL_AI8Sfls

IL_INLN_AI_DATA

Die Struktur enthält die Ausgabeparameter eines Kanals.

Die Parameter sind für jeden einzelnen Kanal wie folgt zu ermitteln:

ParameterName[Channel_Number].[Function]

z.B. OutputData[0].NormalizedValue.

Name	Typ	Beschreibung
NormalizedValue	REAL	Normalisierter Wert zwischen dem oberen und dem unteren Grenzwert
DiagCode	IL_INLN_DIAG_CODE	Diagnosecode, siehe "IL_INLN_DIAG_CODE" auf Seite 592

Tab. 14-29: Strukturelemente IL_INLN_AI_DATA

IL_INLN_DIAG_CODE

Die Tabelle enthält kanalbezogene Fehler und Diagnosen, die von den Funktionsbausteinen der Bibliothek ausgegeben werden.

DiagCode	ENUM	Beschreibung
16#8001	OVERRANGE	Bereichsüberschreitung
16#8002	OPEN_CIRCUIT	Offener Stromkreis, der Ausgang ist als Stromquelle konfiguriert und der vorgegebene Strom kann nicht fließen
16#8003	OVER_LOAD	Kurzschluß/Überlast, der Ausgang ist als Spannungsquelle konfiguriert und überlastet
16#8004	MEASURED_VALUE_INVALID	Ungültiger Messwert / Messwert nicht verfügbar
16#8008	COLD_JUNCTION_DEFECTIVE	Vergleichsstelle defekt
16#8010	INVALID_CONFIGURATION	Ungültige Konfiguration
16#8020	ERROR_IN_IO_SUPPLY_VOLTAGE	Fehler in der Spannungsversorgung
16#8040	MODULE_FAULTY	Modulfehler
16#8080	UNDERRANGE	Bereichsunterschreitung

Tab. 14-30: Enum IL_INLN_DIAG_CODE

IL_INLN_AI_RANGE

Die Aufzählung enthält die zu Konfiguration eines Kanals möglichen Messbereiche.

ENUM	Beschreibung
UNIPOLAR10V	Spannungsmessung 0 ... 10V
BIPOLAR10V	Spannungsmessung -10V ... +10V
UNIPOLAR5V	Spannungsmessung 0 ... 5V
BIPOLAR5V	Spannungsmessung -5 ... +5V
UNIPOLAR25V	Spannungsmessung 0 ... 25V
BIPOLAR25V	Spannungsmessung -25 ... +25V
UNIPOLAR50V	Spannungsmessung 0 ... 50V
UNIPOLAR20mA	Strommessung 0 ... 20mA
BIPOLAR20mA	Strommessung -20mA ... +20mA
UNIPOLAR4TO20mA	Strommessung 4 ... 20mA
UNIPOLAR40mA	Strommessung 0 ... 40 mA
BIPOLAR40mA	Strommessung -40 ... +40mA

Tab. 14-31: Zulässige Werte von IL_INLN_AI_RANGE für IL_AI8SfIs

Beispiel In nachstehendem Beispiel sind Kanal 1 konfiguriert für einen Bereich von 0 ... + 10V und Kanal 2 für einen Bereich von 0... +5V. Die Daten sind eingeordnet zwischen der oberen und der unteren Grenze und als normalisierte Variablenwerte verfügbar.

Deklaration zum Beispiel

```
PROGRAM PrgIL_AI8SfIs
VAR
    fbIL_AI8SfIs: IL_AI8SfIs;
    wInProcessData_0: WORD;
    wInProcessData_1: WORD;
    wOutProcessData_1: WORD;
    wOutProcessData_0: WORD;
    rNormValCH1: REAL;
    rNormValCH2: REAL;
    enDiagCodeCH1: IL_INLN_DIAG_CODE;
    enDiagCodeCH2: IL_INLN_DIAG_CODE;
END_VAR
```

Implementation zum Beispiel

```
// Configuration
fbIL_AI8SfIs.ChannelConfig[0].HighLimit := REAL#10.0;
fbIL_AI8SfIs.ChannelConfig[0].InputRange := IL_INLN_AI_RANGE.UNIPOLAR10V;
fbIL_AI8SfIs.ChannelConfig[0].LowLimit := REAL#0.0;
fbIL_AI8SfIs.ChannelConfig[0].NoOfSamples := BYTE#16;
fbIL_AI8SfIs.ChannelConfig[1].HighLimit := REAL#5.0;
fbIL_AI8SfIs.ChannelConfig[1].InputRange := IL_INLN_AI_RANGE.UNIPOLAR5V;
fbIL_AI8SfIs.ChannelConfig[1].LowLimit := REAL#0.0;
fbIL_AI8SfIs.ChannelConfig[1].NoOfSamples := BYTE#32;
fbIL_AI8SfIs.ChannelSelect := BYTE#2#00000011;
fbIL_AI8SfIs.InProcData_0 := wInProcessData_0;
fbIL_AI8SfIs.InProcData_1 := wInProcessData_1;
wOutProcessData_0 := fbIL_AI8SfIs.OutProcData_0;
wOutProcessData_1 := fbIL_AI8SfIs.OutProcData_1;

// FB call
fbIL_AI8SfIs(Enable :=TRUE);

// Read data
IF fbIL_AI8SfIs.InOperation = TRUE THEN
    rNormValCH1 := fbIL_AI8SfIs.OutputData[0].NormalizedValue;
    rNormValCH2 := fbIL_AI8SfIs.OutputData[1].NormalizedValue;
    enDiagCodeCH1 := fbIL_AI8SfIs.OutputData[0].DiagCode;
    enDiagCodeCH2 := fbIL_AI8SfIs.OutputData[1].DiagCode;
END_IF
```

Prozessdaten-Interface zum Beispiel

Die Prozesseingangs- und -ausgangswerte des Funktionsbausteines sind auf die Prozessdatenworte des einzelnen Feldbussystems zu mappen.

Die Variablen wInProcessData_x und wOutProcessData_x sind im Programm prgIL_AI8SfIs deklariert.

Variable	Mapping	Channel	Address	Type
Application.PrgIL_AI8SfIs.wInProcessData_0		Input0	%IW34	WORD
Application.PrgIL_AI8SfIs.wInProcessData_1		Word0	%IW34	WORD
Application.PrgIL_AI8SfIs.wOutProcessData_0		Output0	%QW34	WORD
Application.PrgIL_AI8SfIs.wOutProcessData_1		Word0	%QW34	WORD

Abb. 14-8: Prozessdaten-Interface

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle INLINEIO_TABLE, 16#0190. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00003003	16#00000000	Ungültiger 'ChannelSelect'-Eingang
INPUT_INVALID_ERROR (16#0001)	16#00003004	Kanalnummer	'InputRange'-Eingang eines Kanals ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003005	Kanalnummer	'NoOfSamples'-Eingang eines Kanals ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003009	Kanalnummer	Ungültige Bereichsgrenzen HighLimit <= LowLimit
DEVICE_ERROR (16#0008)	16#00003020	16#00000000	Zeitüberschreitung, Gerät antwortet nicht
DEVICE_ERROR (16#0008)	16#00003020	16#000x_8010	Ungültige Konfiguration

Tab. 14-32: IL_AI8SfIs Fehlercodes

14.2.5 IL_Sgi2f

Kurzbeschreibung

Der Funktionsbaustein **IL_Sgi2f** initialisiert und konfiguriert die Inline-Klemme **R-IB IL SGI 2/F-PAC** mit zwei analogen Eingängen für Dehnungsmessstreifen und normalisiert die Daten bezüglich der vom Anwender definierte Nennlast.

Für jede Inline-EA-Klemme wird eine Funktionsbausteininstanz benötigt.

Kompatibilität

Der Funktionsbaustein ist verwendbar für Inlinebus, Profibus und Profinet IO-Feldbus.

Schnittstellenbeschreibung

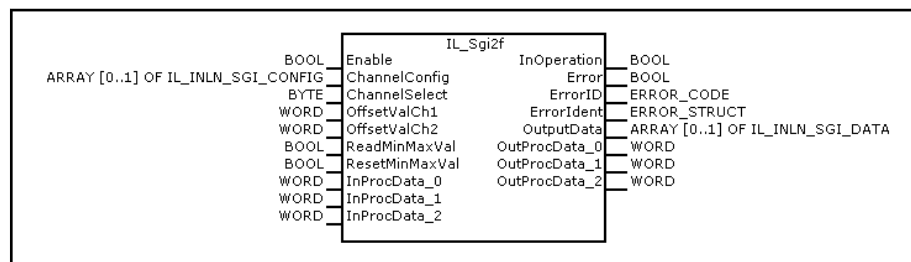


Abb. 14-9: Funktionsbaustein IL_Sgi2f

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Kontinuierliche Bearbeitung des FBs, solange "Enable" TRUE ist
	ChannelConfig	ARRAY[0..1] OF IL_INLN_SGI_CONFIG	Enthält die Konfigurationsdaten für jeden einzelnen Kanal, siehe "IL_INLN_SGI_CONFIG" auf Seite 596
	ChannelSelect	BYTE	Byte#2#0000_0011; Kanalauswahl Bit0- Kanal 1, Bit1- Kanal 2
	OffsetValCh1	WORD	Offset für Kanal 1
	OffsetValCh2	WORD	Offset für Kanal 2
	ReadMinMaxVal	BOOL	Erlaubt das Lesen der Min/Max-Werte für die ausgewählten Kanäle
	ResetMinMaxVal	BOOL	Erlaubt das Rücksetzen der Min/Max-Werte für die ausgewählten Kanäle
	InProcData_0	WORD	Prozesseingangswort 0
	InProcData_1	WORD	Prozesseingangswort 1
	InProcData_2	WORD	Prozesseingangswort 2
VAR OUTPUT	InOperation	BOOL	Wenn TRUE, arbeitet der Baustein in seiner aktuellen Task
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb der FB-Abarbeitung eingetreten ist
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Detailinformationen zum Fehler, siehe "Fehlerbehandlung" auf Seite 599
	OutputData	ARRAY[0..1] OF IL_INLN_SGI_DATA	Enthält für jeden Kanal den Minimal-, Maximal und normalisierten Wert sowie den Diagnosecode, siehe "IL_INLN_SGI_DATA" auf Seite 597
	OutProcData_0	WORD	Prozessausgangswort 0
	OutProcData_1	WORD	Prozessausgangswort 1
	OutProcData_2	WORD	Prozessausgangswort 2

Tab. 14-33: Schnittstelle IL_Sgi2f

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	kontinuierlich
ChannelSelect	BYTE	1	3	3	Positive Flanke an "Enable"
ReadMinMaxVal	BOOL	FALSE	TRUE	FALSE	kontinuierlich mit "Enable"
ResetMinMaxVal	BOOL	FALSE	TRUE	FALSE	kontinuierlich mit "Enable"

Tab. 14-34: Eingangsverhalten IL_Sgi2f

Funktionsbeschreibung Aktivieren Sie die FB-Instanz erst nach dem ersten SPS-Zyklus durch eine positive Flanke am **Enable**-Eingang.

Um den jeweiligen Kanal zu parametrieren, muss die entsprechende Struktur **IL_INLN_SGI_CONFIG** mit den erforderlichen Werten belegt sein. Mit der positiven Flanke am **Enable**-Eingang wird der Kanal konfiguriert. Der erfolgreiche Abschluss der Konfiguration wird durch **InOperation** = TRUE angezeigt.

Eine Rekonfiguration kann nur durch eine erneute positive Flanke am **Enable**-Eingang ausgelöst werden.

Bei ungültigen Eingangsparametern generiert der FB einen Fehler und die entsprechenden Informationen gelangen an **Error**, **ErrorID** und **ErrorIdent**.

Die Initialwerte des Funktionsbausteins sind abgestimmt mit den Initialwerten der Analogklemme.

Um Messwerte oder Diagnoseinformationen der Kanäle 1 und 2 der Klemme zu erhalten, muss **ChannelSelect** entsprechend eingestellt sein.

Die aktivierten Bits des **ChannelSelect** erlauben das Lesen der Kanäle. Bit 0 korrespondiert mit Kanal 1 und Bit 1 mit Kanal 2.

Voreingestellt sind alle Kanäle ausgewählt.

Die gemessenen Werte sind normalisiert bezüglich der vom Anwender vorgegebenen Nennlast und werden im **NormalizedValue** angezeigt.

Falls ein Fehler dabei auftreten sollte, wird dieser für den entsprechenden Kanal in **DiagCode** angezeigt.

Die Minimum- / Maximum-Werte eines Kanals werden nach Triggern von **ReadMinMaxVal** angezeigt. **ResetMinMaxVal** setzt die Min/Max-Werte auf Null.



1. Wenn **ReadMinMaxVal** oder **ResetMinMaxVal** getriggert wurden, beträgt die Zeit zur Berechnung eines **NormalizedValue** mehr als einen SPS-Zyklus.
2. Tritt eine Bereichsüberschreitung beim Lesen der 'sgi'-Daten auf, wird der Minimalwert des 'sgi' anstelle des Wertes mit Bereichsüberschreitung verwendet. Dieser Wert bleibt bis zu einem manuellen Reset des Minimalwerts durch den Anwender erhalten.

IL_INLN_SGI_CONFIG Die Struktur unterstützt die Konfiguration eines Kanals.

Der Kanal kann konfiguriert werden bezüglich **Filter**, **InputRange**, **NominalLoad** und **Offset**.

Die Konfiguration kann wie folgt erfolgen:

`ParameterName[Channel_Number].[Function].`

z. B. zum Konfigurieren von **InputRange** für den ersten Aufruf des Kanals 1:

`ChannelConfig[0].InputRange.`

Name	Typ	Beschreibung	Min-Wert	Max-Wert	Default-Wert	Übernahme
Filter	BOOL	Filterauswahl TRUE- 16fach Mittelwertbildung FALSE - keine Mittelwertbildung	FALSE	TRUE	TRUE	Positive Flanke an "Enable"
InputRange	IL_INLN_SGI_RANGE	Eingangsbereich	UNIPO-LAR1MV	BIPO-LAR4MV	UNIPO-LAR2MV	Positive Flanke an "Enable"

Name	Typ	Beschreibung	Min-Wert	Max-Wert	Default-Wert	Übernahme
NominalLoad	REAL	Nennlast (Sollte >0 sein)	----	----	0	Positive Flanke an "Enable"
Offset	WORD	Offset, Dehnungsmessstreifen	----	----	0	kontinuierlich

Eingangsbereich, siehe "[IL_INLN_SGI_RANGE](#)" auf Seite 597

Die Offset-Werte können unter Nutzung des Funktionsbausteins [IL_Sgi2fOffset](#), Seite 600, bezogen auf das Gerät für den jeweiligen Kanal ermittelt werden.

IL_INLN_SGI_DATA

Die Struktur enthält die Ausgabeparameter eines Kanals.

Die Parameter sind für jeden einzelnen Kanal wie folgt zu ermitteln:

`ParameterName[Channel_Number].[Function]`

z.B. `OutputData[0].NormalizedValue`.

Name	Typ	Beschreibung
NormalizedValue	REAL	Normalisierter Wert entsprechend der Nennlast
DiagCode	IL_INLN_DIAG_CODE	Diagnosecode
MinValue	REAL	Minimalwert vom Dehnmessstreifen gelesen
MaxValue	REAL	Maximalwert vom Dehnmessstreifen gelesen

Diagnosecode, siehe "[IL_INLN_DIAG_CODE](#)" auf Seite 597

IL_INLN_DIAG_CODE

Die Tabelle enthält kanalbezogene Fehler und Diagnosen, die von den Funktionsbausteinen der Bibliothek ausgegeben werden.

DiagCode	ENUM	Beschreibung
16#8001	OVERRANGE	Bereichsüberschreitung
16#8002	OPEN_CIRCUIT	Offener Stromkreis, der Ausgang ist als Stromquelle konfiguriert und der vorgegebene Strom kann nicht fließen
16#8003	OVER_LOAD	Kurzschluß/Überlast, der Ausgang ist als Spannungsquelle konfiguriert und überlastet
16#8004	MEASURED_VALUE_INVALID	Ungültiger Messwert / Messwert nicht verfügbar
16#8008	COLD_JUNCTION_DEFECTIVE	Vergleichsstelle defekt
16#8010	INVALID_CONFIGURATION	Ungültige Konfiguration
16#8020	ERROR_IN_IO_SUPPLY_VOLTAGE	Fehler in der Spannungsversorgung
16#8040	MODULE_FAULTY	Modulfehler
16#8080	UNDERRANGE	Bereichsunterschreitung

Tab. 14-35: Enum [IL_INLN_DIAG_CODE](#)

IL_INLN_SGI_RANGE

Die Aufzählung enthält die zu Konfiguration eines Kanals möglichen Messbereiche.

ENUM	Beschreibung
UNIPOLAR1MV	+1 mV/V
BIPOLAR1MV	±1 mV/V

ENUM	Beschreibung
UNIPOLAR2MV	+2 mV/V
BIPOLAR2MV	±2 mV/V
UNIPOLAR3MV	+3 mV/V
BIPOLAR3MV	±3 mV/V
UNIPOLAR4MV	+4 mV/V
BIPOLAR4MV	±4 mV/V

Beispiel Der Funktionsbaustein wird verwendet, um eine Inline-Klemme mit zwei analogen Eingängen für Dehnungsmessstreifen zu konfigurieren, die zum Anschluss von Wägezellen, Kraftaufnehmern, Massedruckaufnehmern und Vergleichbarem auf der Basis von Dehnungsmessstreifen (DMS) geeignet ist.

Konfiguriert werden Kanalauswahl, Offset-Werte und Eingabebereiche; ferner erfolgt das Mapping der Prozessdaten.

In diesem Beispiel ist der Funktionsbaustein **Sgi2f** für Kanal 1 auf +2mV/V und für Kanal 2 auf +1mV/V voreingestellt.

Die gelesenen Daten werden bezüglich der Nennlast skaliert und sind in **rNormValCH1** und **rNormValCH2** für den jeweiligen Kanal verfügbar.

Die Offset-Werte können unter Nutzung des Funktionsbausteins **IL_Sgi2fOffset**, Seite 600, bezogen auf das Gerät für den jeweiligen Kanal ermittelt werden.

Deklaration zum Beispiel

```
PROGRAM PrgIL_Sgi2f
VAR
  fbIL_Sgi2f: IL_Sgi2f;
  wInProcessData_0: WORD;
  wInProcessData_1: WORD;
  wInProcessData_2: WORD;
  wOutProcessData_0: WORD;
  wOutProcessData_1: WORD;
  wOutProcessData_2: WORD;
  fbIL_Sgi2fOffsetCH1: IL_Sgi2fOffset;
  fbIL_Sgi2fOffsetCH2: IL_Sgi2fOffset;
  rNormValCH1: REAL;
  rNormValCH2: REAL;
  enDiagCodeCH1: IL_INLN_DIAG_CODE;
  enDiagCodeCH2: IL_INLN_DIAG_CODE;
  rMinValCH1: REAL;
  rMinValCH2: REAL;
  rMaxValCH1: REAL;
  rMaxValCH2: REAL;
END_VAR
```

Implementation zum Beispiel

```
// Offset configuration
fbIL_Sgi2fOffsetCH1.InputData := wInProcessData_1;
fbIL_Sgi2fOffsetCH1.NoOfSamples := INT#16;
fbIL_Sgi2fOffsetCH1(Calibrate := TRUE);
fbIL_Sgi2fOffsetCH2.InputData := wInProcessData_2;
fbIL_Sgi2fOffsetCH2.NoOfSamples := INT#16;
fbIL_Sgi2fOffsetCH2(Calibrate := TRUE);
// Configuration
fbIL_Sgi2f.ChannelConfig[0].Filter := TRUE;
fbIL_Sgi2f.ChannelConfig[0].InputRange := IL_INLN_SGI_RANGE.UNIPOLAR2MV;
fbIL_Sgi2f.ChannelConfig[0].NominalLoad := REAL#60;
fbIL_Sgi2f.ChannelConfig[1].Filter := FALSE;
fbIL_Sgi2f.ChannelConfig[1].InputRange := IL_INLN_SGI_RANGE.UNIPOLAR1MV;
fbIL_Sgi2f.ChannelConfig[1].NominalLoad := REAL#100;
fbIL_Sgi2f.ReadMinMaxVal := FALSE;
fbIL_Sgi2f.ResetMinMaxVal := FALSE;
fbIL_Sgi2f.InProcData_0 := wInProcessData_0;
```

```

fbIL_Sgi2f.InProcData_1 := wInProcessData_1;
fbIL_Sgi2f.InProcData_2 := wInProcessData_2;
wOutProcessData_0 := fbIL_Sgi2f.OutProcData_0;
wOutProcessData_1 := fbIL_Sgi2f.OutProcData_1;
wOutProcessData_2 := fbIL_Sgi2f.OutProcData_2;
fbIL_Sgi2f.ChannelSelect := BYTE#2#00000011;
fbIL_Sgi2f.OffsetValCh1 := fbIL_Sgi2f.OffsetCH1.OffsetVal;
fbIL_Sgi2f.OffsetValCh2 := fbIL_Sgi2f.OffsetCH2.OffsetVal;

// FB call
fbIL_Sgi2f(Enable := TRUE);

// Read data
IF fbIL_Sgi2f.InOperation = TRUE THEN
  fbIL_Sgi2f.ReadMinMaxVal := FALSE;
  fbIL_Sgi2f.ResetMinMaxVal := FALSE;
  rNormValCH1 := fbIL_Sgi2f.OutputData[0].NormalizedValue;
  rNormValCH2 := fbIL_Sgi2f.OutputData[1].NormalizedValue;
  enDiagCodeCH1 := fbIL_Sgi2f.OutputData[0].DiagCode;
  enDiagCodeCH2 := fbIL_Sgi2f.OutputData[1].DiagCode;
END_IF
// Read min. and max. values
IF fbIL_Sgi2f.InOperation = TRUE THEN
  fbIL_Sgi2f.ReadMinMaxVal := TRUE;
  fbIL_Sgi2f.ResetMinMaxVal := FALSE;
  rMinValCH1 := fbIL_Sgi2f.OutputData[0].MinimumValue;
  rMinValCH2 := fbIL_Sgi2f.OutputData[1].MinimumValue;
  rMaxValCH1 := fbIL_Sgi2f.OutputData[0].MaximumValue;
  rMaxValCH2 := fbIL_Sgi2f.OutputData[1].MaximumValue;
END_IF
// Reset min. and max. values
IF fbIL_Sgi2f.InOperation = TRUE THEN
  fbIL_Sgi2f.ReadMinMaxVal := FALSE;
  fbIL_Sgi2f.ResetMinMaxVal := TRUE;
  rMinValCH1 := fbIL_Sgi2f.OutputData[0].MinimumValue;
  rMinValCH2 := fbIL_Sgi2f.OutputData[1].MinimumValue;
  rMaxValCH1 := fbIL_Sgi2f.OutputData[0].MaximumValue;
  rMaxValCH2 := fbIL_Sgi2f.OutputData[1].MaximumValue;
END_IF

```

Prozessdaten-Interface zum Beispiel

Die Prozesseingangs- und -ausgangswerte des Funktionsbausteines sind auf die Prozessdatenworte des einzelnen Feldbussystems zu mappen.

Die Variablen wInProcessData_x und wOutProcessData_x sind im Programm prgIL_Sgi2f deklariert.

Variable	Mapping	Channel	Address	Type
Application.PrgIL_Sgi2f.wInProcessData_0		Input0	%IW24	WORD
Application.PrgIL_Sgi2f.wInProcessData_1		Word0	%IW24	WORD
Application.PrgIL_Sgi2f.wInProcessData_2		Word1	%IW26	WORD
		Word2	%IW28	WORD
Application.PrgIL_Sgi2f.wOutProcessData_0		Output0	%QW24	WORD
Application.PrgIL_Sgi2f.wOutProcessData_1		Word0	%QW24	WORD
Application.PrgIL_Sgi2f.wOutProcessData_2		Word1	%QW26	WORD
		Word2	%QW28	WORD

Abb. 14-10: Prozessdaten-Interface

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle INLINEIO_TABLE, 16#0190. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00003003	16#00000000	'ChannelSelect'-Eingang ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003004	Kanalnummer	'InputRange'-Eingang eines Kanals ist ungültig

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00003015	Kanalnummer	'NominalLoad'-Eingang eines Kanals ist ungültig
DEVICE_ERROR (16#0008)	16#00003020	16#00000000	Zeitüberschreitung, Gerät antwortet nicht
DEVICE_ERROR (16#0008)	16#00003020	16#000x8010	Ungültige Konfiguration <i>Mögliche Ursachen:</i> 1. Die Brückenspannung ist weder 3.3V±0.5V noch 5V±0.5 V 2. Der Kennwert ist ±3 mV/V oder ±4 mV/V und die Brückenspannung ist 5 V ±0.5 V. Ändere die Brückenspannung auf 3.3V ±0.5 V

Tab. 14-36: IL_Sgi2f Fehlercodes

14.2.6 IL_Sgi2fOffset

Kurzbeschreibung

Der Funktionsbaustein **IL_Sgi2fOffset** kann zum Trieren der Inline-Klemme **R-IB IL SGI 2/F-PAC** (Inline-Klemme mit zwei analogen Eingängen für Dehnungsmessstreifen) verwendet werden.

Nach einer positiven Flanke am **Calibrate**-Eingang berechnet der Funktionsbaustein den Mittelwert in Abhängigkeit von der vom Anwender vorgegebenen Probemanzahl.

Schnittstellenbeschreibung

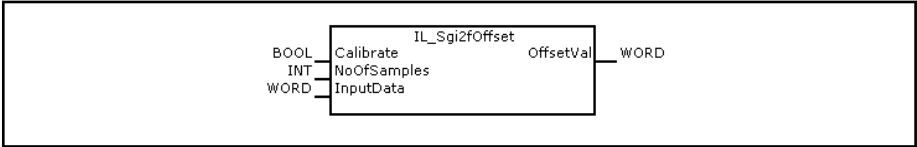


Abb. 14-11: Funktionsbaustein IL_Sgi2fOffset

	Name	Typ	Beschreibung
VAR_INPUT	Calibrate	BOOL	Positiven Flanke startet die Berechnung
	NoOfSamples	INT	Anzahl der zu lesenden Proben (1...32)
	InputData	WORD	Eingangsdaten
VAR OUTPUT	OffsetVal	WORD	Offset

Tab. 14-37: Schnittstelle IL_Sgi2fOffset



Ein Anwendungsbeispiel ist in der Beschreibung des Funktionsbausteines [IL_Sgi2f](#), Seite 594, enthalten.

14.2.7 IL_Temp2Uth

Kurzbeschreibung

Der Funktionsbaustein **IL_Temp2Uth** konfiguriert die Inline-Klemme **R-IB IL TEMP 2 UTH-PAC** mit zwei Eingängen für Thermoelemente und normalisiert die Daten entsprechend der vom Anwender vorgegebenen Auflösung.

Kompatibilität

Der Funktionsbaustein ist verwendbar für Inlinebus und Profinet IO-Feldbus.

Schnittstellenbeschreibung

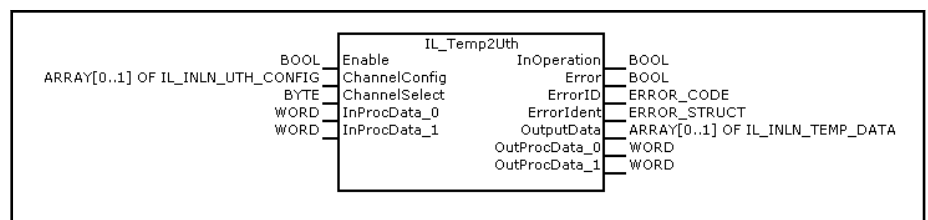


Abb. 14-12: Funktionsbaustein IL_Temp2Uth

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Kontinuierliche Bearbeitung des FBs, solange "Enable" TRUE ist
	ChannelConfig	ARRAY[0..1] OF IL_INLN_UTH_CONFIG	Enthält die Konfigurationsdaten für die Kanäle 1...2, siehe "IL_INLN_UTH_CONFIG" auf Seite 602
	ChannelSelect	BYTE	Byte#2#0000_0011; Individuelle Aktivierung der Kanäle (Bit 0 - Kanal 1, Bit 1 - Kanal 2)
	InProcData_0	WORD	Prozesseingangswort 0
	InProcData_1	WORD	Prozesseingangswort 1
VAR OUTPUT	InOperation	BOOL	Wenn TRUE, arbeitet der Baustein in seiner aktuellen Task
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb der FB-Abarbeitung eingetreten ist
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Detailinformationen zum Fehler, siehe "Fehlerbehandlung" auf Seite 605
	OutputData	ARRAY[0..1] OF IL_INLN_TEMP_DATA	Enthält für jeden Kanal den normalisierten Wert und den Diagnosecode, siehe "IL_INLN_TEMP_DATA" auf Seite 602
	OutProcData_0	WORD	Prozessausgangswort 0
	OutProcData_1	WORD	Prozessausgangswort 1

Tab. 14-38: Schnittstelle IL_Temp2Uth

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	kontinuierlich
ChannelSelect	BYTE	0	3	3	Positive Flanke an "Enable"

Tab. 14-39: Eingangsverhalten IL_Temp2Uth

Funktionsbeschreibung

Aktivieren Sie die FB-Instanz erst nach dem ersten SPS-Zyklus durch eine positive Flanke am **Enable**-Eingang.

Um den jeweiligen Kanal zu parametrieren, muss die entsprechende Struktur **IL_INLN_UTH_CONFIG** mit den erforderlichen Werten belegt sein. Mit der positiven Flanke am **Enable**-Eingang wird der Kanal konfiguriert. Der erfolgreiche Abschluss der Konfiguration wird durch **InOperation** = TRUE angezeigt.

Eine Rekonfiguration kann nur durch eine erneute positive Flanke am **Enable**-Eingang ausgelöst werden.

Bei ungültigen Eingangsparametern generiert der FB einen Fehler und die entsprechenden Informationen gelangen an **Error**, **ErrorID** und **ErrorIdent**.

Die Initialwerte des Funktionsbausteins sind abgestimmt mit den Initialwerten der Analogklemme.

Um Messwerte oder Diagnoseinformationen der Kanäle 1 .. 2 der Klemme zu erhalten, muss **ChannelSelect** entsprechend eingestellt sein.

Die aktivierten Bits des **ChannelSelect** erlauben das Lesen der Kanäle. Bit 0 korrespondiert mit Kanal 1 und Bit 1 mit Kanal 2.

Voreingestellt sind alle Kanäle ausgewählt.

Die gemessenen Werte sind normalisiert bezüglich der vom Anwender vorgegebenen Auflösung und werden in **NormalizedValue** angezeigt. Falls ein Fehler dabei auftreten sollte, wird dieser für den entsprechenden Kanal in **DiagCode** angezeigt.

IL_INLN_UTH_CONFIG

Die Struktur enthält die Informationen zur Konfiguration eines Kanals. Der Kanal ist durch **Resolution**, **SensorType** und **CJCompensation** definiert.

Die Konfiguration kann wie folgt erfolgen:

`ParameterName[Channel_Number].[Function].`

z. B. zum Konfigurieren von **SensorType** für den ersten Aufruf des Kanals 1:

`ChannelConfig[0].SensorType.`

Name	Typ	Beschreibung	Min-Wert	Max-Wert	Default-Wert	Übernahme
Resolution	WORD	Auslösung: 0 : 0.1°C 1 : 1°C 2 : 0.1°F 3 : 1°F Siehe Tabelle "Auflösung" unterhalb	0	3	0	Positive Flanke an "Enable"
SensorType	IL_INLN_UTH_SENSOR	Auswahl des Sensortyps, siehe "IL_INLN_UTH_SENSOR" auf Seite 603 Bsp.: ETYPE	KTYPE	VOLT	KTYPE	Positive Flanke an "Enable"
CJCompensation	IL_INLN_UTH_CJCOMP	Auswahl der Vergleichsstellenkompensation siehe "IL_INLN_UTH_CJCOMP" auf Seite 603	INTERNALCJ_ACTIVE	EXTERNALCJ_NI1000	INTERNALCJ_ACTIVE	Positive Flanke an "Enable"

Auflösung

dezimal	binär	Beschreibung
0	00	Auflösung: 0.1°C (1 µV)
1	01	Auflösung: 1°C (10 µV)
2	10	Auflösung: 0.1°F
3	11	Auflösung: 1°F

IL_INLN_TEMP_DATA

Die Struktur enthält die Ausgabeparameter eines Kanals.

Die Parameter sind für jeden einzelnen Kanal wie folgt zu ermitteln:

`ParameterName[Channel_Number].[Function]`

z.B. `OutputData[0].NormalizedValue`.

Name	Typ	Beschreibung
NormalizedValue	REAL	Normalisierter Wert in korrespondierender Auflösung
DiagCode	IL_INLN_DIAG_CODE	Diagnosecode, siehe "IL_INLN_DIAG_CODE" auf Seite 603

Tab. 14-40: Strukturelemente IL_INLN_TEMP_DATA

IL_INLN_DIAG_CODE Die Tabelle enthält kanalbezogene Fehler und Diagnosen, die von den Funktionsbausteinen der Bibliothek ausgegeben werden.

DiagCode	ENUM	Beschreibung
16#8001	OVERRANGE	Bereichsüberschreitung
16#8002	OPEN_CIRCUIT	Offener Stromkreis, der Ausgang ist als Stromquelle konfiguriert und der vorgegebene Strom kann nicht fließen
16#8003	OVER_LOAD	Kurzschluß/Überlast, der Ausgang ist als Spannungsquelle konfiguriert und überlastet
16#8004	MEASURED_VALUE_INVALID	Ungültiger Messwert / Messwert nicht verfügbar
16#8008	COLD_JUNCTION_DEFECTIVE	Vergleichsstelle defekt
16#8010	INVALID_CONFIGURATION	Ungültige Konfiguration
16#8020	ERROR_IN_IO_SUPPLY_VOLTAGE	Fehler in der Spannungsversorgung
16#8040	MODULE_FAULTY	Modulfehler
16#8080	UNDERRANGE	Bereichsunterschreitung

Tab. 14-41: Enum IL_INLN_DIAG_CODE

IL_INLN_UTH_CJCOMP Die Aufzählung dient der Auswahl der Vergleichsstellenkompensation der Kanäle.

ENUM	Beschreibung
INTERNALCJ_ACTIVE	Interne Vergleichsstelle aktiv; TC-Messung mit interner Vergleichsstellenkompensation
INTERNALCJ_INACTIVE	interne Vergleichsstelle inaktiv; TC-Differenzmessung ohne Vergleichsstellenkompensation
EXTERNALCJ_PT1000	Externe Vergleichsstelle PT1000; TC-Messung mit externer Vergleichsstellenkompensation an einem Isothermalblock (nur für Laboranwendungen)
EXTERNALCJ_NI1000	Externe Vergleichsstelle Ni1000; TC-Messung mit externer Vergleichsstellenkompensation an einem Isothermalblock (nur für Laboranwendungen)

IL_INLN_UTH_SENSOR Der Aufzählungstyp dient der Sensortyp-Auswahl für die einzelnen Kanäle.

ENUM	Beschreibung
KTYPE	ThermocoupleTyp - K Sensor
JTYPE	ThermocoupleTyp - J Sensor
ETYPE	ThermocoupleTyp - E Sensor
RTYPE	ThermocoupleTyp - R Sensor

ENUM	Beschreibung
STYPE	ThermocoupleTyp - S Sensor
TTYPE	ThermocoupleTyp - T Sensor
BTYPE	ThermocoupleTyp - B Sensor
NTYPE	ThermocoupleTyp - N Sensor
UTYPE	ThermocoupleTyp - U Sensor
LTYPE	ThermocoupleTyp - L Sensor
CTYPE	ThermocoupleTyp - C Sensor
WTYPE	ThermocoupleTyp - W Sensor
HKTYPE	ThermocoupleTyp - HK Sensor
COLDJUNCTION	Vergleichsstelle (Cold junction; CJ)
VOLT	U: Spannung (-15mV ... +85mV)

Beispiel Der Funktionsbaustein konfiguriert eine UTH-Temperatur-Klemme zur Temperaturmessung.

In der Konfigurationsvoreinstellung sind vorzunehmen: Konfiguration der Eingänge, Kanalauswahl und Prozessdaten-Mapping zum Gerät.

In diesem Beispiel ist Temp2Uth wie folgt konfiguriert:

Kanal 1: 'KTYPE' und Kanal 2: 'JTYPE'.

Zum Datenlesen steht für jeden Kanal der 'NormalizedValue' der Ausgabe-Struktur zur Verfügung.

Deklaration zum Beispiel

```
PROGRAM PrgIL_Temp2Uth
VAR
    fbIL_Temp2Uth: IL_Temp2Uth;
    wInProcessData_0: WORD;
    wInProcessData_1: WORD;
    wOutProcessData_0: WORD;
    wOutProcessData_1: WORD;
    rNormValCH1: REAL;
    rNormValCH2: REAL;
    enDiagCodeCH1: IL_INLN_DIAG_CODE;
    enDiagCodeCH2: IL_INLN_DIAG_CODE;
END_VAR
```

Implementation zum Beispiel

```
// Configuration
fbIL_Temp2Uth.ChannelConfig[0].CJCompensation := IL_INLN_UTH_CJCOMP.INTERNALCJ_ACTIVE;
fbIL_Temp2Uth.ChannelConfig[0].Resolution    := WORD#0;
fbIL_Temp2Uth.ChannelConfig[0].SensorType    := IL_INLN_UTH_SENSOR.KTYPE;
fbIL_Temp2Uth.ChannelConfig[1].CJCompensation := IL_INLN_UTH_CJCOMP.INTERNALCJ_INACTIVE;
fbIL_Temp2Uth.ChannelConfig[1].Resolution    := WORD#1;
fbIL_Temp2Uth.ChannelConfig[1].SensorType    := IL_INLN_UTH_SENSOR.JTYPE;
fbIL_Temp2Uth.ChannelSelect                  := BYTE#2#00000011;
fbIL_Temp2Uth.InProcData_0                   := wInProcessData_0;
fbIL_Temp2Uth.InProcData_1                   := wInProcessData_1;
wOutProcessData_0                           := fbIL_Temp2Uth.OutProcData_0;
wOutProcessData_1                           := fbIL_Temp2Uth.OutProcData_1;

// FB call
fbIL_Temp2Uth(Enable := TRUE);

// Data read
IF fbIL_Temp2Uth.InOperation = TRUE THEN
    rNormValCH1 := fbIL_Temp2Uth.OutputData[0].NormalizedValue;
    rNormValCH2 := fbIL_Temp2Uth.OutputData[1].NormalizedValue;
    enDiagCodeCH1 := fbIL_Temp2Uth.OutputData[0].DiagCode;
```



```

enDiagCodeCH2      := fbIL_Temp2Uth.OutputData[1].DiagCode;
END_IF

```

Prozessdaten-Interface zum Beispiel

Die Prozesseingangs- und -ausgangswerte des Funktionsbausteines sind auf die Prozessdatenworte des einzelnen Feldbussystems zu mappen.

Die Variablen wInProcessData_x und wOutProcessData_x sind im Programm prgIL_Temp2Uth deklariert.

Variable	Mapping	Channel	Address	Type
Application.PrgIL_Temp2Uth.wInProcessData_0	Word0	Input0	%IW16	WORD
Application.PrgIL_Temp2Uth.wInProcessData_1	Word1	Input0	%IW18	WORD
Application.PrgIL_Temp2Uth.wOutProcessData_0	Word0	Output0	%QW16	WORD
Application.PrgIL_Temp2Uth.wOutProcessData_1	Word1	Output0	%QW18	WORD

Abb. 14-13: Prozessdaten-Interface

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle INLINEIO_TABLE, 16#0190. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00003003	16#00000000	Ungültiger 'ChannelSelect'-Eingang
INPUT_INVALID_ERROR (16#0001)	16#00003007	Kanalnummer	'Resolution'-Eingang eines Kanals ist ungültig oder die Auflösung passt nicht zum Sensortyp
INPUT_INVALID_ERROR (16#0001)	16#00003008	Kanalnummer	'SensorType'-Eingang eines Kanals ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003016	Kanalnummer	Vergleichsstellenkompensation eines Kanals ist ungültig

Tab. 14-42: IL_Temp2Uth Fehlercodes

14.2.8 IL_Temp2Rtd

Kurzbeschreibung

Der Funktionsbaustein **IL_Temp2Rtd** initialisiert und konfiguriert die Inline-Klemme **R-IB IL TEMP2 RTD-PAC** mit zwei Eingängen für Temperatursensoren und normalisiert die Daten entsprechend der vom Anwender vorgegebenen Auflösung.

Kompatibilität

Der Funktionsbaustein ist verwendbar für Inlinebus und Profinet IO-Feldbus.

Schnittstellenbeschreibung

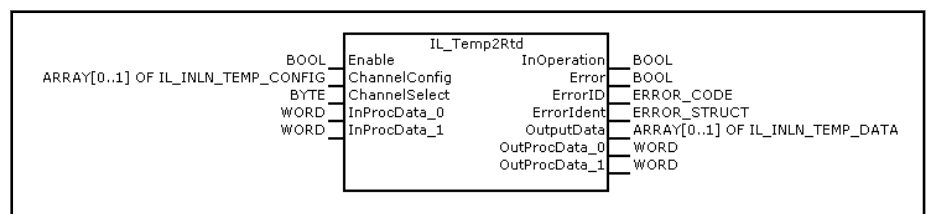


Abb. 14-14: Funktionsbaustein IL_Temp2Rtd

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Kontinuierliche Bearbeitung des FBs, solange "Enable" TRUE ist
	ChannelConfig	ARRAY[0..1] OF IL_INLN_TEMP_CONFIG	Enthält die Konfigurationsdaten für die Kanäle 1...2, siehe " IL_INLN_TEMP_CONFIG " auf Seite 607
	ChannelSelect	BYTE	Byte#2#0000_0011; Individuelle Aktivierung der Kanäle (Bit 0 - Kanal 1, Bit 1 - Kanal 2)
	InProcData_0	WORD	Prozesseingangswort 0
	InProcData_1	WORD	Prozesseingangswort 1
VAR OUTPUT	InOperation	BOOL	Wenn TRUE, arbeitet der Baustein in seiner aktuellen Task
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb der FB-Abarbeitung eingetreten ist
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Detailinformationen zum Fehler, siehe " Fehlerbehandlung " auf Seite 610
	OutputData	ARRAY[0..1] OF IL_INLN_TEMP_DATA	Enthält für jeden Kanal den normalisierten Wert und den Diagnosecode, siehe " IL_INLN_TEMP_DATA " auf Seite 608
	OutProcData_0	WORD	Prozessausgangswort 0
	OutProcData_1	WORD	Prozessausgangswort 1

Tab. 14-43: Schnittstelle IL_Temp2Rtd

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	kontinuierlich
ChannelSelect	BYTE	1	3	3	Positive Flanke an "Enable"

Tab. 14-44: Eingangsverhalten IL_Temp2Rtd

Funktionsbeschreibung

Aktivieren Sie die FB-Instanz erst nach dem ersten SPS-Zyklus durch eine positive Flanke am **Enable**-Eingang.

Um den jeweiligen Kanal zu parametrieren, muss die entsprechende Struktur **IL_INLN_AI_CONFIG** mit den erforderlichen Werten belegt sein. Mit der positiven Flanke am **Enable**-Eingang wird der Kanal konfiguriert. Der erfolgreiche Abschluss der Konfiguration wird durch **InOperation** = TRUE angezeigt.

Eine Rekonfiguration kann nur durch eine erneute positive Flanke am **Enable**-Eingang ausgelöst werden.

Bei ungültigen Eingangsparametern generiert der FB einen Fehler und die entsprechenden Informationen gelangen an **Error**, **ErrorID** und **ErrorIdent**.

Die Initialwerte des Funktionsbausteins sind abgestimmt mit den Initialwerten der Analogklemme.

Um Messwerte oder Diagnoseinformationen der Kanäle 1 .. 2 der Klemme zu erhalten, muss **ChannelSelect** entsprechend eingestellt sein.

Die aktivierten Bits des **ChannelSelect** erlauben das Lesen der Kanäle. Bit 0 korrespondiert mit Kanal 1 und Bit 1 mit Kanal 2.

Voreingestellt sind alle Kanäle ausgewählt.

Die gemessenen Werte sind normalisiert bezüglich der vom Anwender vorgegebenen Auflösung und werden in **NormalizedValue** angezeigt. Falls ein Fehler dabei auftreten sollte, wird dieser für den entsprechenden Kanal in **DiagCode** angezeigt.

IL_INLN_TEMP_CONFIG

Die Struktur enthält die Informationen zur Konfiguration eines Kanals. Der Kanal ist durch **SensorType**, **ConnectionType**, **NoOfSamples**, **Resolution** und **Resistance** definiert.

Die Konfiguration kann wie folgt erfolgen:

`ParameterName[Channel_Number].[Function].`

z. B. zum Konfigurieren von **SensorType** für den ersten Aufruf des Kanals 1:

`ChannelConfig[0].SensorType.`

Name	Typ	Beschreibung	Min-Wert	Max-Wert	Default-Wert	Übernahme
NoOfSamples	BYTE	Verwendung nur bei IL_Temp4_8Rtd				
SensorType	IL_INLN_TEMP_SENSOR	Auswahl des Sensortyps, siehe "IL_INLN_TEMP_SENSOR" auf Seite 609 Bsp.: PTSAMA	PTDIN	LINE-AR4000	PTDIN	Positive Flanke an "Enable"
Connection-Type	IL_INLN_TEMP_CONNECTIONTYPE	Sensor-Verbindung: THREEWIRE := 3-Leiter TWO WIRE := 2-Leiter FOURWIRE := 4-Leiter siehe "IL_INLN_TEMP_CONNECTIONTYPE" auf Seite 609	THREEWIRE	FOURWIRE	THREEWIRE	Positive Flanke an "Enable"
Resolution	WORD	Resolution: 0 : 0.1°C/1%/0.1 Ω 1 : 0.01°C/0.1%/0.01Ω 2 : 0.1°F/ Reserviert 3 : 0.01°F/ Reserviert Siehe Auflösungstabelle unten	0	3	0	Positive Flanke an "Enable"
Resistance	IL_INLN_TEMP_R0	Widerstands-Referenzwert R0 [Ω], siehe "IL_INLN_TEMP_R0" auf Seite 608	R10	R3000	R100	Positive Flanke an "Enable"

Tab. 14-45: Zulässige Werte von IL_INLN_AI_CONFIG für IL_Temp2Rtd

Auflösungstabelle

Code		Auflösung für Sensortyp			
dez.	bin.	0 ...11	13 (pot)%	14 (Linear400)	16(Linear4000)
0	00	0.1°C	1%	0.1 Ω	1 Ω
1	01	0.01°C	0.1%	0.01 Ω	0.1 Ω
2	10	0.1°F	Reserviert	Reserviert	Reserviert
3	11	0.01°F	Reserviert	Reserviert	Reserviert

IL_INLN_TEMP_DATA Die Struktur enthält die Ausgabeparameter eines Kanals.
 Die Parameter sind für jeden einzelnen Kanal wie folgt zu ermitteln:
`ParameterName[Channel_Number].[Function]`
 z.B. `OutputData[0].NormalizedValue`.

Name	Typ	Beschreibung
NormalizedValue	REAL	Normalisierter Wert in korrespondierender Auflösung
DiagCode	IL_INLN_DIAG_CODE	Diagnosecode, siehe "IL_INLN_DIAG_CODE" auf Seite 608

Tab. 14-46: Strukturelemente IL_INLN_TEMP_DATA

IL_INLN_DIAG_CODE Die Tabelle enthält kanalbezogene Diagnosen, die von den Funktionsbausteinen der Bibliothek ausgegeben werden.

DiagCode	ENUM	Beschreibung
16#8001	OVERRANGE	Bereichsüberschreitung
16#8002	OPEN_CIRCUIT	Offener Stromkreis, der Ausgang ist als Stromquelle konfiguriert und der vorgegebene Strom kann nicht fließen
16#8003	OVER_LOAD	Kurzschluß/Überlast, der Ausgang ist als Spannungsquelle konfiguriert und überlastet
16#8004	MEASURED_VALUE_INVALID	Ungültiger Messwert / Messwert nicht verfügbar
16#8008	COLD_JUNCTION_DEFECTIVE	Vergleichsstelle defekt
16#8010	INVALID_CONFIGURATION	Ungültige Konfiguration
16#8020	ERROR_IN_IO_SUPPLY_VOLTAGE	Fehler in der Spannungsversorgung
16#8040	MODULE_FAULTY	Modulfehler
16#8080	UNDERRANGE	Bereichsunterschreitung

Tab. 14-47: Enum IL_INLN_DIAG_CODE

IL_INLN_TEMP_R0 Dieser Aufzählungstyp gestattet die Auswahl des Widerstands-Referenzwertes zur Konfiguration des Kanals.

Enum	Beschreibung
R10	Widerstands-Referenzwert R0: 10Ω
R20	Widerstands-Referenzwert R0: 20Ω
R30	Widerstands-Referenzwert R0: 30Ω
R50	Widerstands-Referenzwert R0: 50Ω
R100	Widerstands-Referenzwert R0: 100Ω
R120	Widerstands-Referenzwert R0: 120Ω
R150	Widerstands-Referenzwert R0: 150Ω
R200	Widerstands-Referenzwert R0: 200Ω
R240	Widerstands-Referenzwert R0: 240Ω
R300	Widerstands-Referenzwert R0: 300Ω
R400	Widerstands-Referenzwert R0: 400Ω

Enum	Beschreibung
R500	Widerstands-Referenzwert R0: 500Ω
R1000	Widerstands-Referenzwert R0: 1000Ω
R1500	Widerstands-Referenzwert R0: 1500Ω
R2000	Widerstands-Referenzwert R0: 2000Ω
R10000	Nur für FB IL_Temp4_8Rtd
R3000	Widerstands-Referenzwert R0: 3000Ω

Tab. 14-48: Enum IL_INLN_TEMP_R0 für IL_Temp2Rtd

IL_INLN_TEMP_SENSOR

Dieser Aufzählungstyp gestattet die Auswahl des Sensortyps zur Konfiguration eines Kanals.

Enum	Sensortyp
PTDIN	PT DIN
PTSAMA	PT SAMA
NIDIN	NI DIN
NISAMA	NI SAMA
CU10	CU10
CU50	CU50
CU53	CU53
NI1000	NI1000 (Landis & Gyr)
NI500	NI500 (Viesmann)
KTY81110	KTY81-100
KTY84	KTY84
KTY81210	KTY81210
POT	POTENTIOMETER (%)
LINEAR400	Linear R: 0 ... 400 Ω
LINEAR4000	Linear R:0 ... 4000 Ω

Tab. 14-49: Enum IL_INLN_TEMP_SENSOR für FB IL_Temp2Rtd

**IL_INLN_TEMP_CONNECTION-
TYPE**

Dieser Aufzählungstyp gestattet die Auswahl des Verbindungstyps des Sensors.

Enum	Beschreibung
THREEWIRE	Verbindungstyp: 3-Leiter-Sensor
TWOWIRE	Verbindungstyp: 2-Leiter-Sensor
FOURWIRE	Verbindungstyp: 4-Leiter-Sensor

Tab. 14-50: Enum IL_INLN_TEMP_CONNECTIONTYPE für IL_Temp2Rtd

Beispiel

Der Funktionsbaustein konfiguriert eine RTD-Temperatur-Klemme zur Temperaturmessung.

In der Konfigurationsvoreinstellung sind vorzunehmen: Konfiguration der Eingänge, Kanalauswahl und Prozessdaten-Mapping zum Gerät.

In diesem Beispiel ist Temp2Rtd wie folgt konfiguriert:
Kanal 1: 'PTDIN' und Kanal 2: 'POT'.
Zum Datenlesen steht für jeden Kanal der 'NormalizedValue' der Ausgabe-Struktur zur Verfügung.
Deklaration zum Beispiel

```
PROGRAM PrgIL_Temp2Rtd
VAR
  fbIL_Temp2Rtd: IL_Temp2Rtd;
  wInProcessData_0: WORD;
  wInProcessData_1: WORD;
  wOutProcessData_0: WORD;
  wOutProcessData_1: WORD;
  rNormValCH1: REAL;
  rNormValCH2: REAL;
  enDiagCodeCH1: IL_INLN_DIAG_CODE;
  enDiagCodeCH2: IL_INLN_DIAG_CODE;
END_VAR
```

Implementation zum Beispiel

```
// Configuration
fbIL_Temp2Rtd.ChannelConfig[0].ConnectionType := IL_INLN_TEMP_CONNECTIONTYPE.THREEWIRE;
fbIL_Temp2Rtd.ChannelConfig[0].NoOfSamples      := BYTE#16;
fbIL_Temp2Rtd.ChannelConfig[0].Resistance      := IL_INLN_TEMP_R0.R100;
fbIL_Temp2Rtd.ChannelConfig[0].Resolution      := WORD#1;
fbIL_Temp2Rtd.ChannelConfig[0].SensorType      := IL_INLN_TEMP_SENSOR.PTDIN;
fbIL_Temp2Rtd.ChannelConfig[1].ConnectionType := IL_INLN_TEMP_CONNECTIONTYPE.TWOWIRE;
fbIL_Temp2Rtd.ChannelConfig[1].NoOfSamples      := BYTE#32;
fbIL_Temp2Rtd.ChannelConfig[1].Resistance      := IL_INLN_TEMP_R0.R1000;
fbIL_Temp2Rtd.ChannelConfig[1].Resolution      := WORD#0;
fbIL_Temp2Rtd.ChannelConfig[1].SensorType      := IL_INLN_TEMP_SENSOR.POT;
fbIL_Temp2Rtd.ChannelSelect                    := BYTE#2#00000011;
fbIL_Temp2Rtd.InProcData_0                    := wInProcessData_0;
fbIL_Temp2Rtd.InProcData_1                    := wInProcessData_1;
wOutProcessData_0                             := fbIL_Temp2Rtd.OutProcData_0;
wOutProcessData_1                             := fbIL_Temp2Rtd.OutProcData_1;

// FB call
fbIL_Temp2Rtd(Enable:= TRUE );

// Data read
IF fbIL_Temp2Rtd.InOperation = TRUE THEN
  rNormValCH1 := fbIL_Temp2Rtd.OutputData[0].NormalizedValue;
  rNormValCH2 := fbIL_Temp2Rtd.OutputData[1].NormalizedValue;
  enDiagCodeCH1 := fbIL_Temp2Rtd.OutputData[0].DiagCode;
  enDiagCodeCH2 := fbIL_Temp2Rtd.OutputData[1].DiagCode;
END_IF
```

Prozessdaten-Interface zum Beispiel

Die Prozesseingangs- und -ausgangsworte des Funktionsbausteines sind auf die Prozessdatenworte des einzelnen Feldbussystems zu mappen.
Die Variablen wInProcessData_x und wOutProcessData_x sind im Programm prgIL_Temp2Rtd deklariert.

Inline-Module I/O Mapping				
Channels				
Variable	Mapping	Channel	Address	Type
Application.PrgIL_Temp2Rtd.wInProcessData_0		Input0	%IW12	
Application.PrgIL_Temp2Rtd.wInProcessData_1		Word0	%IW12	WORD
		Word1	%IW14	WORD
Application.PrgIL_Temp2Rtd.wOutProcessData_0		Output0	%QW12	
Application.PrgIL_Temp2Rtd.wOutProcessData_1		Word0	%QW12	WORD
		Word1	%QW14	WORD

Abb. 14-15: Prozessdaten-Interface

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle INLINEIO_TABLE, 16#0190. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00003003	16#00000000	Ungültiger 'ChannelSelect'-Eingang
INPUT_INVALID_ERROR (16#0001)	16#00003006	Kanalnummer	'Resistance'-Eingang ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003007	Kanalnummer	'Resolution'-Eingang eines Kanals ist ungültig oder die Auflösung passt nicht zum Sensortyp
INPUT_INVALID_ERROR (16#0001)	16#00003008	Kanalnummer	'SensorType'-Eingang eines Kanals ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003014	Kanalnummer	Der 'ConnectionType'-Eingang eines Kanals ist ungültig oder 'FOURWIRE' passt nicht zum Kanal 2.

Tab. 14-51: IL_Temp2Rtd Fehlercodes

14.2.9 IL_Temp4_8Rtd

Kurzbeschreibung

Der Funktionsbaustein **IL_Temp4_8Rtd** initialisiert und konfiguriert die Inline-Klemme **R-IB IL TEMP 4/8 RTD-PAC** mit acht resistiven Temperatursensor-Eingängen und normalisiert die Daten entsprechend der vom Anwender vorgegebenen Auflösung.

Kompatibilität

Der Funktionsbaustein ist verwendbar für Inlinebus, Profibus und Profinet IO-Feldbus.

Schnittstellenbeschreibung

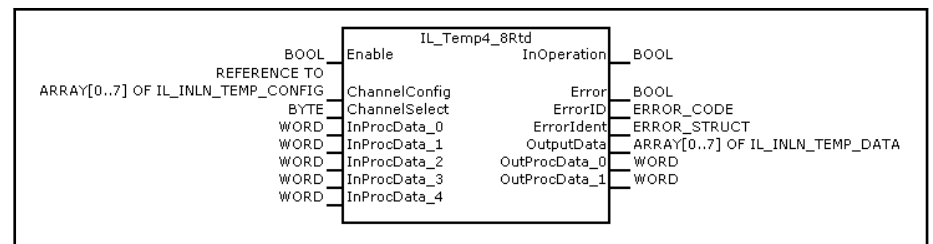


Abb. 14-16: Funktionsbaustein IL_Temp4_8Rtd

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Kontinuierliche Bearbeitung des FBs, solange "Enable" TRUE ist
	ChannelConfig	REFERENCE TO ARRAY[0..7] OF IL_INLN_TEMP_CONFIG	Enthält die Konfigurationsdaten für die Kanäle 1...8, siehe "IL_INLN_TEMP_CONFIG" auf Seite 613
	ChannelSelect	BYTE	Byte#2#11111111; Individuelle Aktivierung der Kanäle (Bit 0...Bit 7 korrespondieren mit Kanal 1...8)
	InProcData_0	WORD	Prozesseingangswort 0
	InProcData_1	WORD	Prozesseingangswort 1
	InProcData_2	WORD	Prozesseingangswort 2
	InProcData_3	WORD	Prozesseingangswort 3
	InProcData_4	WORD	Prozesseingangswort 4

	Name	Typ	Beschreibung
VAR OUTPUT	InOperation	BOOL	Wenn TRUE, arbeitet der Baustein in seiner aktuellen Task
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb der FB-Abarbeitung eingetreten ist
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Detailinformationen zum Fehler, siehe "Fehlerbehandlung" auf Seite 617
	OutputData	ARRAY[0..7] OF IL_INLN_TEMP_DATA	Enthält für jeden Kanal den normalisierten Wert und den Diagnosecode, siehe "IL_INLN_TEMP_DATA" auf Seite 614
	OutProcData_0	WORD	Prozessausgangswort 0
	OutProcData_1	WORD	Prozessausgangswort 1

Tab. 14-52: Schnittstelle IL_Temp4_8Rtd

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	kontinuierlich
ChannelSelect	BYTE	1	255	255	Positive Flanke an "Enable"

Tab. 14-53: Eingangsverhalten IL_Temp4_8Rtd

Funktionsbeschreibung

Aktivieren Sie die FB-Instanz erst nach dem ersten SPS-Zyklus durch eine positive Flanke am **Enable**-Eingang.

Um den jeweiligen Kanal zu parametrieren, muss die entsprechende Struktur **IL_INLN_TEMP_CONFIG** mit den erforderlichen Werten belegt sein. Mit der positiven Flanke am **Enable**-Eingang wird der Kanal konfiguriert. Der erfolgreiche Abschluss der Konfiguration wird durch **InOperation** = TRUE angezeigt.

Eine Rekonfiguration kann nur durch eine erneute positive Flanke am **Enable**-Eingang ausgelöst werden.

Bei ungültigen Eingangsparametern generiert der FB einen Fehler und die entsprechenden Informationen gelangen an **Error**, **ErrorID** und **ErrorIdent**.

Die Initialwerte des Funktionsbausteins sind abgestimmt mit den Initialwerten der Analogklemme.

Um Messwerte oder Diagnoseinformationen der Kanäle 1 .. 8 der Klemme zu erhalten, muss **ChannelSelect** entsprechend eingestellt sein.

Die aktivierten Bits des **ChannelSelect** erlauben das Lesen der Kanäle. Bit 0 korrespondiert mit Kanal 1 bis zu Bit 7 mit Kanal 8.

Voreingestellt sind alle Kanäle ausgewählt.

Die gemessenen Werte sind normalisiert bezüglich der vom Anwender vorgegebenen Auflösung und werden in **NormalizedValue** angezeigt. Falls ein Fehler dabei auftreten sollte, wird dieser für den entsprechenden Kanal in **DiagCode** angezeigt.



R-IB IL TEMP4_8RTD-PAC arbeitet im Multiplex-Betrieb, entsprechend werden die Daten auch multiplex gelesen. Das Update der Daten für jeden einzelnen Zyklus benötigt damit mehr als einen SPS-Zyklus.

IL_INLN_TEMP_CONFIG Die Struktur enthält die Informationen zur Konfiguration eines Kanals. Der Kanal ist durch **SensorType**, **ConnectionType**, **NoOfSamples**, **Resolution** und **Resistance** definiert.

Die Konfiguration kann wie folgt erfolgen:

`ParameterName[Channel_Number].[Function].`

z. B. zum Konfigurieren von **SensorType** für den ersten Aufruf des Kanals 1:

`ChannelConfig[0].SensorType.`

Name	Typ	Beschreibung	Min-Wert	Max-Wert	Default-Wert	Übernahme
NoOfSamples	BYTE	0 – keine Mittelwertbildung 4 – Mittelwertbildung über 4 Messwerte 16 – Mittelwertbildung über 16 Messwerte 32 – Mittelwertbildung über 32 Messwerte	0	32	16	Positive Flanke an "Enable"
SensorType	IL_INLN_TEMP_SENSOR	Auswahl des Sensortyps, siehe "IL_INLN_TEMP_SENSOR" auf Seite 615 Bsp.: PTSAMA	PTDIN	LINE-AR20000	PTDIN	Positive Flanke an "Enable"
Connection-Type	IL_INLN_TEMP_CONNECTIONTYPE	Sensor-Verbindung: THREEWIRE := 3-Leiter TWOIRE := 2-Leiter siehe "IL_INLN_TEMP_CONNECTIONTYPE" auf Seite 615	THREEWIRE	TWOIRE	THREEWIRE	Positive Flanke an "Enable"
Resolution	WORD	Auflösung: 0 : 0.1°C/1%/0.1 Ω 1 : 0.01°C/0.1%/0.01Ω 2 : 0.1°F/ Reserviert 3 : 0.01°F/ Reserviert Siehe Auflösungstabelle unten	0	3	0	Positive Flanke an "Enable"
Resistance	IL_INLN_TEMP_R0	Widerstands-Referenzwert R0 [Ω], siehe "IL_INLN_TEMP_R0" auf Seite 614	R10	R10000	R100	Positive Flanke an "Enable"

Tab. 14-54: Zulässige Werte von `IL_INLN_AI_CONFIG` für `IL_Temp4_8Rtd`

Auflösungstabelle

Code		Auflösung für Sensortyp			
dez.	bin.	0 ... 11	13 (pot)%	14 (Linear400)	15 (Linear20000)
0	00	0.1°C	1%	0.1 Ω	1 Ω
1	01	0.01°C	0.1%	0.01 Ω	0.1 Ω
2	10	0.1°F	Reserviert	Reserviert	Reserviert
3	11	0.01°F	Reserviert	Reserviert	Reserviert

IL_INLN_TEMP_DATA Die Struktur enthält die Ausgabeparameter eines Kanals.
 Die Parameter sind für jeden einzelnen Kanal wie folgt zu ermitteln:
`ParameterName[Channel_Number].[Function]`
 z.B. `OutputData[0].NormalizedValue`.

Name	Typ	Beschreibung
NormalizedValue	REAL	Normalisierter Wert in korrespondierender Auflösung
DiagCode	IL_INLN_DIAG_CODE	Diagnosecode, siehe "IL_INLN_DIAG_CODE" auf Seite 614

Tab. 14-55: Strukturelemente IL_INLN_TEMP_DATA

IL_INLN_DIAG_CODE Die Tabelle enthält kanalbezogene Fehler und Diagnosen, die von den Funktionsbausteinen der Bibliothek ausgegeben werden.

DiagCode	ENUM	Beschreibung
16#8001	OVERRANGE	Bereichsüberschreitung
16#8002	OPEN_CIRCUIT	Offener Stromkreis, der Ausgang ist als Stromquelle konfiguriert und der vorgegebene Strom kann nicht fließen
16#8003	OVER_LOAD	Kurzschluß/Überlast, der Ausgang ist als Spannungsquelle konfiguriert und überlastet
16#8004	MEASURED_VALUE_INVALID	Ungültiger Messwert / Messwert nicht verfügbar
16#8008	COLD_JUNCTION_DEFECTIVE	Vergleichsstelle defekt
16#8010	INVALID_CONFIGURATION	Ungültige Konfiguration
16#8020	ERROR_IN_IO_SUPPLY_VOLTAGE	Fehler in der Spannungsversorgung
16#8040	MODULE_FAULTY	Modulfehler
16#8080	UNDERRANGE	Bereichsunterschreitung

Tab. 14-56: Enum IL_INLN_DIAG_CODE

IL_INLN_TEMP_R0 Dieser Aufzählungstyp gestattet die Auswahl des Widerstands-Referenzwertes zur Konfiguration des Kanals.

Enum	Beschreibung
R10	Widerstands-Referenzwert R0: 10Ω
R20	Widerstands-Referenzwert R0: 20Ω
R30	Widerstands-Referenzwert R0: 30Ω
R50	Widerstands-Referenzwert R0: 50Ω
R100	Widerstands-Referenzwert R0: 100Ω
R120	Widerstands-Referenzwert R0: 120Ω
R150	Widerstands-Referenzwert R0: 150Ω
R200	Widerstands-Referenzwert R0: 200Ω
R240	Widerstands-Referenzwert R0: 240Ω
R300	Widerstands-Referenzwert R0: 300Ω
R400	Widerstands-Referenzwert R0: 400Ω

Enum	Beschreibung
R500	Widerstands-Referenzwert R0: 500Ω
R1000	Widerstands-Referenzwert R0: 1000Ω
R1500	Widerstands-Referenzwert R0: 1500Ω
R2000	Widerstands-Referenzwert R0: 2000 Ω
R10000	Widerstands-Referenzwert R0: 10000Ω
R3000	Nur für FB IL_Temp2Rtd

Tab. 14-57: Enum IL_INLN_TEMP_R0 für IL_AlsfIs

IL_INLN_TEMP_SENSOR

Dieser Aufzählungstyp gestattet die Auswahl des Sensortyps zur Konfiguration eines Kanals.

Enum	Sensortyp
PTDIN	PT DIN
PTSAMA	PT SAMA
NIDIN	NI DIN
NISAMA	NI SAMA
CU10	CU10
CU50	CU50
CU53	CU53
NI1000	NI1000 (Landis & Gyr)
NI500	NI500 (Viesmann)
KTY81110	KTY81-100
KTY84	KTY84
KTY81210	KTY81210
POT	POTENTIOMETER (%)
LINEAR400	Linear R: 0 ... 400 Ω
LINEAR20000	Linear R: 0 ... 20000 Ω

Tab. 14-58: Enum IL_INLN_TEMP_SENSOR für FB IL_Temp4_8Rtd

**IL_INLN_TEMP_CONNECTION-
TYPE**

Dieser Aufzählungstyp gestattet die Auswahl des Verbindungstyps des Sensors.

Enum	Beschreibung
THREEWIRE	Verbindungstyp: 3-Leiter-Sensor
TWOWIRE	Verbindungstyp: 2-Leiter-Sensor
FOURWIRE	Reserviert für FB IL_Temp2Rtd

Tab. 14-59: Enum IL_INLN_TEMP_CONNECTIONTYPE für IL_Temp4_8Rtd

Beispiel

Der Funktionsbaustein konfiguriert eine RTD-Temperatur-Klemme zur Temperaturmessung.

In der Konfigurationsvoreinstellung sind vorzunehmen: Konfiguration der Eingänge, Kanalauswahl und Prozessdaten-Mapping zum Gerät.

In diesem Beispiel ist Temp4_8Rtd wie folgt konfiguriert: Kanal 1 und 2 für die Sensoren 'PTDIN' und 'POT'.

Zum Datenlesen steht für jeden Kanal der 'NormalizedValue' der Ausgabe-Struktur zur Verfügung.

Deklaration zum Beispiel

```
PROGRAM PrgIL_Temp4_8Rtd
VAR
    fbIL_Temp4_8Rtd: IL_Temp4_8Rtd;
    wInProcessData_0: WORD;
    wInProcessData_1: WORD;
    wInProcessData_2: WORD;
    wInProcessData_3: WORD;
    wInProcessData_4: WORD;
    wOutProcessData_0: WORD;
    wOutProcessData_1: WORD;
    rNormValCH1: REAL;
    rNormValCH2: REAL;
    enDiagCodeCH1: IL_INLN_DIAG_CODE;
    enDiagCodeCH2: IL_INLN_DIAG_CODE;
END_VAR
```

Implementation zum Beispiel

```
// Configuration
fbIL_Temp4_8Rtd.ChannelConfig[0].ConnectionType := IL_INLN_TEMP_CONNECTIONTYPE.THREEWIRE;
fbIL_Temp4_8Rtd.ChannelConfig[0].NoOfSamples    := BYTE#16;
fbIL_Temp4_8Rtd.ChannelConfig[0].Resistance     := IL_INLN_TEMP_R0.R100;
fbIL_Temp4_8Rtd.ChannelConfig[0].Resolution     := WORD#1;
fbIL_Temp4_8Rtd.ChannelConfig[0].SensorType     := IL_INLN_TEMP_SENSOR.PTDIN;
fbIL_Temp4_8Rtd.ChannelConfig[1].ConnectionType := IL_INLN_TEMP_CONNECTIONTYPE.TWOWIRE;
fbIL_Temp4_8Rtd.ChannelConfig[1].NoOfSamples    := BYTE#32;
fbIL_Temp4_8Rtd.ChannelConfig[1].Resistance     := IL_INLN_TEMP_R0.R1000;
fbIL_Temp4_8Rtd.ChannelConfig[1].Resolution     := WORD#0;
fbIL_Temp4_8Rtd.ChannelConfig[1].SensorType     := IL_INLN_TEMP_SENSOR.POT;
fbIL_Temp4_8Rtd.ChannelSelect                  := BYTE#2#00000011;
fbIL_Temp4_8Rtd.InProcData_0                   := wInProcessData_0;
fbIL_Temp4_8Rtd.InProcData_1                   := wInProcessData_1;
fbIL_Temp4_8Rtd.InProcData_2                   := wInProcessData_2;
fbIL_Temp4_8Rtd.InProcData_3                   := wInProcessData_3;
fbIL_Temp4_8Rtd.InProcData_4                   := wInProcessData_4;
wOutProcessData_0 := fbIL_Temp4_8Rtd.OutProcData_0;
wOutProcessData_1 := fbIL_Temp4_8Rtd.OutProcData_1;

// Call
fbIL_Temp4_8Rtd(Enable:= TRUE );

// Data read
IF fbIL_Temp4_8Rtd.InOperation = TRUE THEN
    rNormValCH1 := fbIL_Temp4_8Rtd.OutputData[0].NormalizedValue;
    rNormValCH2 := fbIL_Temp4_8Rtd.OutputData[1].NormalizedValue;
    enDiagCodeCH1 := fbIL_Temp4_8Rtd.OutputData[0].DiagCode;
    enDiagCodeCH2 := fbIL_Temp4_8Rtd.OutputData[1].DiagCode;
END_IF
```

Prozessdaten-Interface zum Beispiel

Die Prozesseingangs- und -ausgangsworte des Funktionsbausteines sind auf die Prozessdatenworte des einzelnen Feldbussystems zu mappen.

Die Variablen wInProcessData_x und wOutProcessData_x sind im Programm prgIL_Temp4_8Rtd deklariert.

Inline-Module I/O Mapping				
Channels				
Variable	Mapp...	Channel	Address	Type
		Input0	%IW2	
Application.PrgIL_Temp4_8Rtd.wInProcessData_0		Word0	%IW2	WORD
Application.PrgIL_Temp4_8Rtd.wInProcessData_1		Word1	%IW4	WORD
Application.PrgIL_Temp4_8Rtd.wInProcessData_2		Word2	%IW6	WORD
Application.PrgIL_Temp4_8Rtd.wInProcessData_3		Word3	%IW8	WORD
Application.PrgIL_Temp4_8Rtd.wInProcessData_4		Word4	%IW10	WORD
		Output0	%QW2	
Application.PrgIL_Temp4_8Rtd.wOutProcessData_0		Word0	%QW2	WORD
Application.PrgIL_Temp4_8Rtd.wOutProcessData_1		Word1	%QW4	WORD
		Word2	%QW6	WORD
		Word3	%QW8	WORD
		Word4	%QW10	WORD

Abb. 14-17: Prozessdaten-Interface

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle INLINEIO_TABLE, 16#0190. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00003003	16#00000000	Ungültiger 'ChannelSelect'-Eingang
INPUT_INVALID_ERROR (16#0001)	16#00003005	Kanalnummer	'NoOfSamples'-Eingang eines Kanals ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003006	Kanalnummer	'Resistance'-Eingang ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003007	Kanalnummer	'Resolution'-Eingang eines Kanals ist ungültig oder die Auflösung passt nicht zum Sensortyp
INPUT_INVALID_ERROR (16#0001)	16#00003008	Kanalnummer	'SensorType'-Eingang eines Kanals ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003014	Kanalnummer	'ConnectionType' eines Kanals ist ungültig
DEVICE_ERROR (16#0008)	16#00003020	16#00000000	Zeitüberschreitung, Gerät antwortet nicht
DEVICE_ERROR (16#0008)	16#00003020	16#000x8010	Ungültige Konfiguration

Tab. 14-60: IL_Temp4_8Rtd Fehlercodes

14.2.10 IL_Temp4_8RtdEf

Kurzbeschreibung Der Funktionsbaustein **IL_Temp4_8RtdEf** initialisiert und konfiguriert die Inline-Klemme **R-IB IL TEMP 4/8 RTD-EF-PAC** mit acht resistiven Temperatursensor-Eingängen und normalisiert die Daten entsprechend der vom Anwender vorgegebenen Auflösung. Die **R-IB IL TEMP 4/8 RTD-EF-PAC** unterstützt gegenüber der **R-IB IL TEMP 4/8 RTD-PAC** zusätzlich 4-Draht Sensoren.

Kompatibilität Der Funktionsbaustein ist verwendbar für Inlinebus, Profibus und Profinet IO-Feldbus.

Schnittstellenbeschreibung

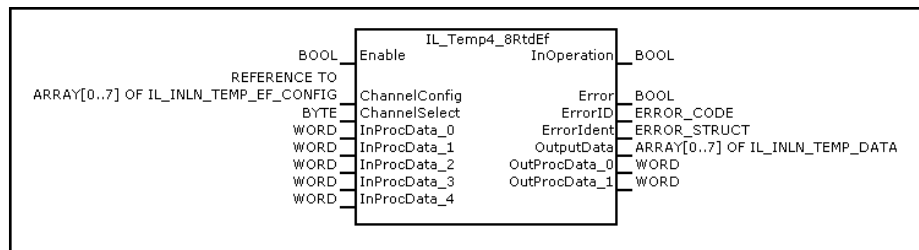


Abb. 14-18: Funktionsbaustein IL_Temp4_8RtdEf

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Kontinuierliche Bearbeitung des FBs, solange "Enable" TRUE ist
	ChannelConfig	REFERENCE TO ARRAY[0..7] OF IL_INLN_TEMP_EF_CONFIG	Enthält die Konfigurationsdaten für die Kanäle 1...8, siehe "IL_INLN_TEMP_EF_CONFIG" auf Seite 619
	ChannelSelect	BYTE	Byte#2#11111111; Individuelle Aktivierung der Kanäle (Bit 0...Bit 7 korrespondieren mit Kanal 1...8)
	InProcData_0	WORD	Prozesseingangswort 0
	InProcData_1	WORD	Prozesseingangswort 1
	InProcData_2	WORD	Prozesseingangswort 2
	InProcData_3	WORD	Prozesseingangswort 3
	InProcData_4	WORD	Prozesseingangswort 4
VAR OUTPUT	InOperation	BOOL	Wenn TRUE, arbeitet der Baustein in seiner aktuellen Task
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb der FB-Abarbeitung eingetreten ist
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Detailinformationen zum Fehler, siehe "Fehlerbehandlung" auf Seite 623
	OutputData	ARRAY[0..7] OF IL_INLN_TEMP_DATA	Enthält für jeden Kanal den normalisierten Wert und den Diagnosecode, siehe "IL_INLN_TEMP_DATA" auf Seite 620
	OutProcData_0	WORD	Prozessausgangswort 0
	OutProcData_1	WORD	Prozessausgangswort 1

Tab. 14-61: Schnittstelle IL_Temp4_8RtdEf

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	kontinuierlich
ChannelSelect	BYTE	1	255	255	Positive Flanke an "Enable"

Tab. 14-62: Eingangsverhalten IL_Temp4_8RtdEf

Funktionsbeschreibung Aktivieren Sie die FB-Instanz erst nach dem ersten SPS-Zyklus durch eine positive Flanke am **Enable**-Eingang.

Um den jeweiligen Kanal zu parametrieren, muss die entsprechende Struktur **IL_INLN_TEMP_EF_CONFIG** mit den erforderlichen Werten belegt sein. Mit der positiven Flanke am **Enable**-Eingang wird der Kanal konfiguriert. Der erfolgreiche Abschluss der Konfiguration wird durch **InOperation** = TRUE angezeigt.

Eine Rekonfiguration kann nur durch eine erneute positive Flanke am **Enable**-Eingang ausgelöst werden.

Bei ungültigen Eingangsparametern generiert der FB einen Fehler und die entsprechenden Informationen gelangen an **Error**, **ErrorID** und **ErrorIdent**.

Die Initialwerte des Funktionsbausteins sind abgestimmt mit den Initialwerten der Analogklemme.

Um Messwerte oder Diagnoseinformationen der Kanäle 1 .. 8 der Klemme zu erhalten, muss **ChannelSelect** entsprechend eingestellt sein.

Die aktivierten Bits des **ChannelSelect** erlauben das Lesen der Kanäle. Bit 0 korrespondiert mit Kanal 1 bis zu Bit 7 mit Kanal 8.

Voreingestellt sind alle Kanäle ausgewählt.

Die gemessenen Werte sind normalisiert bezüglich der vom Anwender vorgegebenen Auflösung und werden in **NormalizedValue** angezeigt. Falls ein Fehler dabei auftreten sollte, wird dieser für den entsprechenden Kanal in **DiagCode** angezeigt.



R-IB IL TEMP4/8-RTD-EF-PAC arbeitet im Multiplex-Betrieb, entsprechend werden die Daten auch multiplex gelesen. Das Update der Daten für jeden einzelnen Zyklus benötigt damit mehr als einen SPS-Zyklus.

IL_INLN_TEMP_EF_CONFIG

Die Struktur enthält die Informationen zur Konfiguration eines Kanals. Der Kanal ist durch **SensorType**, **Resolution**, **Resistance**, **FilterTime** und **ODS-Sensitivity** definiert.

Die Konfiguration kann wie folgt erfolgen:

```
ParameterName[Channel_Number].[Function].
```

z. B. zum Konfigurieren von **SensorType** für den ersten Aufruf des Kanals 1:

```
ChannelConfig[0].SensorType.
```

Name	Typ	Beschreibung	Min-Wert	Max-Wert	Default-Wert	Übernahme
SensorType	IL_INLN_TEMP_EF_SENSOR	Auswahl des Sensortyps, siehe "IL_INLN_TEMP_EF_SENSOR" auf Seite 621 Bsp.: PTSAMA	PTDIN	LINE-AR5000	PTDIN	Positive Flanke an "Enable"
Resolution	BYTE	Auflösung: 0 : 0.1°C/1%/0.1 Ω 1 : 0.01°C/0.1%/0.01Ω 2 : 0.1°F/ Reserviert 3 : 0.01°F/ Reserviert Siehe Auflösungstabelle unten	0	3	0	Positive Flanke an "Enable"
Resistance	IL_INLN_TEMP_EF_R0	Widerstands-Referenzwert R0 [Ω], siehe "IL_INLN_TEMP_EF_R0" auf Seite 621	R100	R10000	R100	Positive Flanke an "Enable"

Name	Typ	Beschreibung	Min-Wert	Max-Wert	Default-Wert	Übernahme
FilterTime	BYTE	0 - 480ms 1 - 120ms 2 - 101ms 3 - 200ms	0	3	0	
ODSSensitivity	IL_INLN_TEMP_EF_ODS	Empfindlichkeit der Drahtbruchererkennung "IL_INLN_TEMP_EF_ODS" auf Seite 621	0	3	0	Positive Flanke an "Enable"

Tab. 14-63: Zulässige Werte von IL_INLN_TEMP_EF_CONFIG für FB IL_Temp4_8RtdEf

Auflösungstabelle

Code		Auflösung für Sensortyp			
dez.	bin.	0 ...11	Linear R0...500 Ω	Linear R0...5 kΩ	Linear R0...30 kΩ
0	00	0.1°C	0.1 Ω	1 Ω	1 Ω
1	01	0.01°C	0.01 Ω	0.1 Ω	Reserviert
2	10	0.1°F	Reserviert	Reserviert	Reserviert
3	11	0.01°F	Reserviert	Reserviert	Reserviert

IL_INLN_TEMP_DATA Die Struktur enthält die Ausgabeparameter eines Kanals.

Die Parameter sind für jeden einzelnen Kanal wie folgt zu ermitteln:

`ParameterName[Channel_Number].[Function]`

z.B. `OutputData[0].NormalizedValue`.

Name	Typ	Beschreibung
NormalizedValue	REAL	Normalisierter Wert in korrespondierender Auflösung
DiagCode	IL_INLN_DIAG_CODE	Diagnosecode, siehe "IL_INLN_DIAG_CODE" auf Seite 620

Tab. 14-64: Strukturelemente IL_INLN_TEMP_DATA

IL_INLN_DIAG_CODE Die Tabelle enthält kanalbezogene Fehler und Diagnosen, die von den Funktionsbausteinen der Bibliothek ausgegeben werden.

DiagCode	ENUM	Beschreibung
16#8001	OVERRANGE	Bereichsüberschreitung
16#8002	OPEN_CIRCUIT	Offener Stromkreis, der Ausgang ist als Stromquelle konfiguriert und der vorgegebene Strom kann nicht fließen
16#8003	OVER_LOAD	Kurzschluß/Überlast, der Ausgang ist als Spannungsquelle konfiguriert und überlastet
16#8004	MEASURED_VALUE_INVALID	Ungültiger Messwert / Messwert nicht verfügbar
16#8008	COLD_JUNCTION_DEFECTIVE	Vergleichsstelle defekt
16#8010	INVALID_CONFIGURATION	Ungültige Konfiguration
16#8020	ERROR_IN_IO_SUPPLY_VOLTAGE	Fehler in der Spannungsversorgung

DiagCode	ENUM	Beschreibung
16#8040	MODULE_FAULTY	Modulfehler
16#8080	UNDERRANGE	Bereichsunterschreitung

Tab. 14-65: Enum IL_INLN_DIAG_CODE

IL_INLN_TEMP_EF_R0 Dieser Aufzählungstyp gestattet die Auswahl des Widerstands-Referenzwertes zur Konfiguration des Kanals.

```

TYPE IL_INLN_TEMP_EF_R0 :
(
  R100  := 0,    // Resistance of Sensor at 0deg : 100 ohm
  R10   := 1,    // Resistance of Sensor at 0deg : 10 ohm
  R20   := 2,    // Resistance of Sensor at 0deg : 20 ohm
  R30   := 3,    // Resistance of Sensor at 0deg : 30 ohm
  R50   := 4,    // Resistance of Sensor at 0deg : 50 ohm
  R120  := 5,    // Resistance of Sensor at 0deg : 120 ohm
  R150  := 6,    // Resistance of Sensor at 0deg : 150 ohm
  R200  := 7,    // Resistance of Sensor at 0deg : 200 ohm
  R240  := 8,    // Resistance of Sensor at 0deg : 240 ohm
  R300  := 9,    // Resistance of Sensor at 0deg : 300 ohm
  R400  := 10,   // Resistance of Sensor at 0deg : 400 ohm
  R500  := 11,   // Resistance of Sensor at 0deg : 500 ohm
  R1000 := 12,   // Resistance of Sensor at 0deg : 1000 ohm
  R1500 := 13,   // Resistance of Sensor at 0deg : 1500 ohm
  R2000 := 14,   // Resistance of Sensor at 0deg : 2000 ohm
  R10000:= 15,   // Resistance of Sensor at 0deg : 10000 ohm
  R3000 := 16,   // Resistance of Sensor at 0deg : 3000 ohm
) WORD;
END_TYPE

```

IL_INLN_TEMP_EF_SENSOR Dieser Aufzählungstyp gestattet die Auswahl des Sensortyps zur Konfiguration eines Kanals.

```

TYPE IL_INLN_TEMP_EF_SENSOR :
(
  PT DIN      := 0,    // Pt DIN SENSOR TYPE
  PT SAMA     := 1,    // Pt SAMA SENSOR TYPE
  Ni DIN      := 2,    // Ni DIN SENSOR TYPE
  Ni SAMA     := 3,    // Ni SAMA SENSOR TYPE
  Cu 10       := 4,    // Cu 10 SENSOR TYPE
  Cu 50       := 5,    // Cu 50 SENSOR TYPE
  Cu 53       := 6,    // Cu 53 SENSOR TYPE
  Ni 1000     := 7,    // Ni 1000 (LANDIS & GYR) SENSOR TYPE
  Ni 500      := 8,    // Ni 500 (VIESSMANN) SENSOR TYPE
  KTY81110    := 9,    // KTY81 -110 SENSOR TYPE
  KTY84       := 16#A, // KTY84 SENSOR TYPE
  KTY81210    := 16#B, // KTY81-210 SENSOR TYPE
  LINEAR30000 := 16#C, // LINEAR R : 0 TO 30000 Ohm SENSOR TYPE

  LINEAR500   := 16#E, // LINEAR R : 0 TO 500 Ohm SENSOR TYPE
  LINEAR5000  := 16#F, // LINEAR R : 0 TO 5000 Ohm SENSOR TYPE
) WORD;
END_TYPE

```

IL_INLN_TEMP_EF_ODS Dieser Aufzählungstyp gestattet die Auswahl der Empfindlichkeit der Drahtbruchererkennung. (ODS - open circuit detection sensitivity): "Hohe Empfindlichkeit"/ "Mittlere Empfindlichkeit"/ "Aus".

Typdefinition

```

TYPE IL_INLN_TEMP_EF_ODS:
(
  HIGH SENSITIVITY := 0    // High Sensitivity
  MEDIUM SENSITIVITY := 1 // Medium Sensitivity
  SWITCHED_OFF := 3       // Switched Off
) BYTE;
END_TYPE

```

Beispiel Der Funktionsbaustein konfiguriert eine RTD-EF-Temperaturklemme zur Temperaturmessung.

In der Konfigurationsvoreinstellung sind vorzunehmen: Konfiguration der Eingänge, Kanalauswahl und Prozessdaten-Mapping zum Gerät.

In diesem Beispiel ist Temp4_8RtdEf wie folgt konfiguriert: Kanal 1 und 2 für die Sensoren 'PTDIN' und 'Linear Resistance 5000 Ohms'.

Zum Datenlesen steht für jeden Kanal der 'NormalizedValue' der Ausgabestruktur zur Verfügung.

Deklaration zum Beispiel

```
PROGRAM PrgIL_Temp4_8RtdEf
VAR
    fbIL_Temp4_8RtdEf: IL_Temp4_8RtdEf;
    bEnable: BOOL;
    byChannelSelect: BYTE := 255;
    wInProcessData_0: WORD;
    wInProcessData_1: WORD;
    wInProcessData_2: WORD;
    wInProcessData_3: WORD;
    wInProcessData_4: WORD;
    wOutProcessData_0: WORD;
    wOutProcessData_1: WORD;
    rNormValCH1: REAL;
    rNormValCH2: REAL;
    enDiagCodeCH1: IL_INLN_DIAG_CODE;
    enDiagCodeCH2: IL_INLN_DIAG_CODE;
END_VAR
```

Implementation zum Beispiel

```
// Configuration Channel 1
fbIL_Temp4_8RtdEf.ChannelConfig[0].SensorType      := IL_INLN_TEMP_EF_SENSOR.PTDIN;
fbIL_Temp4_8RtdEf.ChannelConfig[0].Resolution     := BYTE#0;
fbIL_Temp4_8RtdEf.ChannelConfig[0].Resistance     := IL_INLN_TEMP_EF_R0.R100;
fbIL_Temp4_8RtdEf.ChannelConfig[0].FilterTime     := BYTE#0;
fbIL_Temp4_8RtdEf.ChannelConfig[0].ODSSensitivity:= IL_INLN_TEMP_EF_ODS.HIGH_SENSITIVITY;
// Configuration Channel 2
fbIL_Temp4_8RtdEf.ChannelConfig[1].SensorType      := IL_INLN_TEMP_EF_SENSOR.LINEAR5000;
fbIL_Temp4_8RtdEf.ChannelConfig[1].Resolution     := BYTE#0;
fbIL_Temp4_8RtdEf.ChannelConfig[1].Resistance     := IL_INLN_TEMP_EF_R0.R2000;
fbIL_Temp4_8RtdEf.ChannelConfig[1].FilterTime     := BYTE#0;
fbIL_Temp4_8RtdEf.ChannelConfig[1].ODSSensitivity:= IL_INLN_TEMP_EF_ODS.SWITCHED_OFF;

byChannelSelect := BYTE#2#00000011; // Channel 1 & 2 selected

fbIL_Temp4_8Rtd.InProcData_0      := wInProcessData_0;
fbIL_Temp4_8Rtd.InProcData_1      := wInProcessData_1;
fbIL_Temp4_8Rtd.InProcData_2      := wInProcessData_2;
fbIL_Temp4_8Rtd.InProcData_3      := wInProcessData_3;
fbIL_Temp4_8Rtd.InProcData_4      := wInProcessData_4;

wOutProcessData_0 := fbIL_Temp4_8RtdEf.OutProcData_0;
wOutProcessData_1 := fbIL_Temp4_8RtdEf.OutProcData_1;

// FB call
fbIL_Temp4_8RtdEf(Enable:= TRUE );

// Data read
IF fbIL_Temp4_8RtdEf.InOperation = TRUE THEN
    rNormValCH1 := fbIL_Temp4_8RtdEf.OutputData[0].NormalizedValue;
    rNormValCH2 := fbIL_Temp4_8RtdEf.OutputData[1].NormalizedValue;
    enDiagCodeCH1 := fbIL_Temp4_8RtdEf.OutputData[0].DiagCode;
    enDiagCodeCH2 := fbIL_Temp4_8RtdEf.OutputData[1].DiagCode;
END_IF
```

Prozessdaten-Interface zum Beispiel

Die Prozesseingangs- und -ausgangsworte des Funktionsbausteines sind auf die Prozessdatenworte des einzelnen Feldbussystems zu mappen.

Die Variablen wInProcessData_x und wOutProcessData_x sind im Programm prgIL_Temp4_8RtdEf deklariert.

Variable	Mapp...	Channel	Address	Type
Application.PrgIL_Temp4_8Rtd.wInProcessData_0		Input0	%IW2	
Application.PrgIL_Temp4_8Rtd.wInProcessData_1		Word0	%IW2	WORD
Application.PrgIL_Temp4_8Rtd.wInProcessData_2		Word1	%IW4	WORD
Application.PrgIL_Temp4_8Rtd.wInProcessData_3		Word2	%IW6	WORD
Application.PrgIL_Temp4_8Rtd.wInProcessData_4		Word3	%IW8	WORD
Application.PrgIL_Temp4_8Rtd.wInProcessData_4		Word4	%IW10	WORD
Application.PrgIL_Temp4_8Rtd.wOutProcessData_0		Output0	%QW2	
Application.PrgIL_Temp4_8Rtd.wOutProcessData_1		Word0	%QW2	WORD
Application.PrgIL_Temp4_8Rtd.wOutProcessData_1		Word1	%QW4	WORD
		Word2	%QW6	WORD
		Word3	%QW8	WORD
		Word4	%QW10	WORD

Abb. 14-19: Prozessdaten-Interface

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle INLINEIO_TABLE, 16#0190. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00003003	16#00000000	Ungültiger 'ChannelSelect'-Eingang
INPUT_INVALID_ERROR (16#0001)	16#00003005	Kanalnummer	'NoOfSamples/FilterTime'-Eingang eines Kanals ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003006	Kanalnummer	'Resistance'-Eingang ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003007	Kanalnummer	'Resolution'-Eingang eines Kanals ist ungültig oder die Auflösung passt nicht zum Sensortyp
INPUT_INVALID_ERROR (16#0001)	16#00003008	Kanalnummer	'SensorType'-Eingang eines Kanals ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003019	Kanalnummer	'ODSSensitivity' eines Kanals ist ungültig
DEVICE_ERROR (16#0008)	16#00003020	16#00000000	Zeitüberschreitung, Gerät antwortet nicht
DEVICE_ERROR (16#0008)	16#00003020	16#000x8010	Ungültige Konfiguration

Tab. 14-66: IL_Temp4_8RtdEf Fehlercodes

14.3 Analoge Ausgangsklemmen

14.3.1 IL_AO1Sf

Kurzbeschreibung Der Funktionsbaustein **ILAO1Sf** konvertiert einen normalisierten Wert und übergibt ihn an den Analogausgang der **R-IB IL AO-1/SF**-Klemme (Inline-Klemme mit einem analogen Ausgang).

Für jede Inline-EA-Klemme wird eine Funktionsbausteininstanz benötigt.

Kompatibilität Der Funktionsbaustein ist verwendbar für Inline-, Profibus, sercos und Profinet IO-Feldbus.

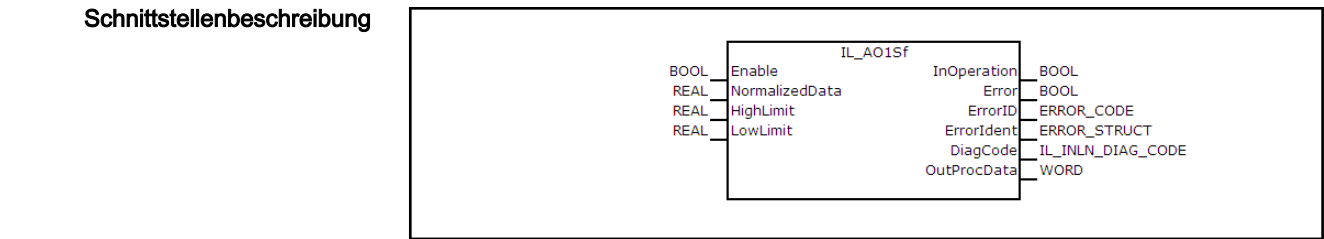


Abb. 14-20: Funktionsbaustein IL_AO1Sf

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Kontinuierliche Bearbeitung des FBs, solange "Enable" TRUE ist
	NormalizedValue	REAL	Normalisierter Eingangswert
	HighLimit	REAL	Obere Grenze für normalisierten Eingangswert
	LowLimit	REAL	Untere Grenze für normalisierten Eingangswert
VAR OUTPUT	InOperation	BOOL	Wenn TRUE, arbeitet der Baustein in seiner aktuellen Task
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb der FB-Abarbeitung eingetreten ist
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Detailinformationen zum Fehler, siehe "Fehlerbehandlung" auf Seite 625
	DiagCode	IL_INLN_DIAG_CODE	Diagnoseinformation, siehe "IL_INLN_DIAG_CODE" auf Seite 624
	OutProcData	WORD	Prozessausgangswort

Tab. 14-67: Schnittstelle IL_AO1Sf

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	kontinuierlich
NormalizedValue	REAL	-65535.0	65535.0	0.0	kontinuierlich mit "Enable"
HighLimit	REAL	-65535.0	65535.0	10.0	kontinuierlich mit "Enable"
LowLimit	REAL	-65535.0	65535.0	0.0	kontinuierlich mit "Enable"

Tab. 14-68: Eingangsverhalten IL_AO1Sf

- Funktionsbeschreibung

Der **IL_AO1Sf**-Funktionsbaustein konvertiert zyklisch den normalisierten Eingangswert in den analogen Ausgangswert des **R-IB-IL-AO-1/SF**-Moduls. Das **R-IB-IL-AO-1/SF**-Modul hat einen Analogausgang zum wahlweisen Anschluss von Spannungs- oder Stromsignalen (0- 10V bzw. 0-20mA / 4-20mA). Die unterschiedlichen Bereiche können durch die Verdrahtung unterschiedlicher Anschlusspunkte am Modul genutzt werden. Für die Verdrahtung sehen Sie bitte die Hardwaredokumentation des Moduls ein.
- IL_INLN_DIAG_CODE

Die Tabelle enthält kanalbezogene Fehler und Diagnosen, die von den Funktionsbausteinen der Bibliothek ausgegeben werden.

DiagCode	ENUM	Beschreibung
16#8001	OVERRANGE	Bereichsüberschreitung
16#8002	OPEN_CIRCUIT	Offener Stromkreis, der Ausgang ist als Stromquelle konfiguriert und der vorgegebene Strom kann nicht fließen
16#8003	OVER_LOAD	Kurzschluß/Überlast, der Ausgang ist als Spannungsquelle konfiguriert und überlastet
16#8004	MEASURED_VALUE_INVALID	Ungültiger Messwert / Messwert nicht verfügbar
16#8008	COLD_JUNCTION_DEFECTIVE	Vergleichsstelle defekt
16#8010	INVALID_CONFIGURATION	Ungültige Konfiguration
16#8020	ERROR_IN_IO_SUPPLY_VOLTAGE	Fehler in der Spannungsversorgung
16#8040	MODULE_FAULTY	Modulfehler
16#8080	UNDERRANGE	Bereichsunterschreitung

Tab. 14-69: Enum IL_INLN_DIAG_CODE

Beispiel In diesem Beispiel konfiguriert der Funktionsbaustein das R-IB-IL-AO-1/SF-Modul für Spannungsausgaben im Bereich von 0 bis 10V.

Deklaration zum Beispiel

```
PROGRAM PrgIL_AO1Sf
VAR
    fbIL_AO1Sf: IL_AO1Sf;
    wOutProcessData: WORD;
    enDiagCode: IL_INLN_DIAG_CODE;
END_VAR
```

Implementation zum Beispiel

```
// Configuration
fbIL_AO1Sf.HighLimit := REAL#10.0 ;
fbIL_AO1Sf.LowLimit := REAL#0.0 ;
wOutProcessData := fbIL_AO1Sf.OutProcData;

// Call
fbIL_AO1Sf( Enable:=TRUE);

// Data output
IF fbIL_AO1Sf.InOperation = TRUE THEN
    fbIL_AO1Sf.NormalizedValue := REAL#5.0 ;
    enDiagCode := fbIL_AO1Sf.DiagCode ;
END_IF
```

Prozessdaten-Interface zum Beispiel

Das Prozessausgangswort des Funktionsbausteines ist auf das Prozessdatenwort des einzelnen Feldbussystems zu mappen.

Die Variable wOutProcessData ist im Programm prgIL_AO1Sf deklariert.

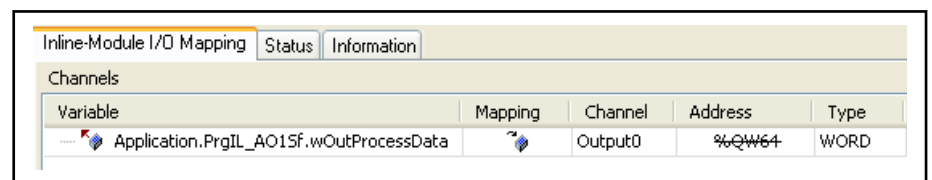


Abb. 14-21: Prozessdaten-Interface

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle INLINEIO_TABLE, 16#0190. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00003009	16#00000000	Ungültige Bereichsgrenzen HighLimit <= LowLimit

Tab. 14-70: IL_AO1Sf Fehlercodes

14.3.2 IL_AO2SfBp

Kurzbeschreibung

Der Funktionsbaustein **IL_AO2SfBp** parametriert die Inline-Klemmen **R IB-IL-AO-2/SF** (Inline-Klemme mit zwei analogen Ausgängen) und **R IB-IL-AO-2/U/BP** und konvertiert und übergibt normalisierte Werte an deren Analog-Ausgänge.

Für jede Inline-EA-Klemme wird eine Funktionsbausteininstanz benötigt.

Kompatibilität

Der Funktionsbaustein ist verwendbar für Inline-, Profibus und Profinet IO-Feldbus.

Schnittstellenbeschreibung

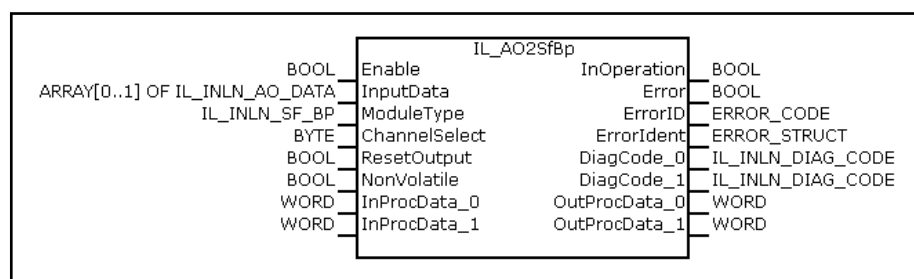


Abb. 14-22: Funktionsbaustein IL_AO2SfBp

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Kontinuierliche Bearbeitung des FBs, solange "Enable" TRUE ist
	InputData	ARRAY[0..1] OF IL_INLN_AO_DATA	Datenübergabe an die Kanäle "IL_INLN_AO_DATA" auf Seite 627
	ModuleType	IL_INLN_SF_BP	Auswahl des Modultyps: Bsp.: SF, "IL_INLN_SF_BP" auf Seite 628
	ChannelSelect	BYTE	Kanalauswahl, BYTE#2#000000yx; x - Kanal 1, y - Kanal 2
	ResetOutput	BOOL	Ausgangsverhalten: FALSE – HOLT, TRUE – RESET
	NonVolatile	BOOL	Ausgangsverhalten: FALSE – flüchtig, TRUE – nicht flüchtig
	InProcData_0	WORD	Prozesseingangswort 0
	InProcData_1	WORD	Prozesseingangswort 1

	Name	Typ	Beschreibung
VAR OUTPUT	InOperation	BOOL	Wenn TRUE, arbeitet der Baustein in seiner aktuellen Task
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb der FB-Abarbeitung eingetreten ist
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Detailinformationen zum Fehler "Fehlerbehandlung" auf Seite 629
	DiagCode_0	IL_INLN_DI-AG_CODE	Diagnoseinformation Kanal 0 "IL_INLN_DIAG_CODE" auf Seite 628
	DiagCode_1	IL_INLN_DI-AG_CODE	Diagnoseinformation Kanal1
	OutProcData_0	WORD	Prozessausgangswort 0
	OutProcData_1	WORD	Prozessausgangswort 1

Tab. 14-71: Schnittstelle IL_AO2SfBp

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	kontinuierlich
ModuleType	ENUM	SF	BPBIPOLAR10V	SF	Positive Flanke von "Enable"
ChannelSelect	BYTE	0	3	3	kontinuierlich mit "Enable"
ResetOutput	BOOL	FALSE	TRUE	FALSE	Positive Flanke von "Enable"
NonVolatile	BOOL	FALSE	TRUE	FALSE	Positive Flanke von "Enable"

Tab. 14-72: Eingangsverhalten IL_AO2SfBp

Funktionsbeschreibung

Der Funktionsbaustein **IL_AO2SfBp** parametriert eine **R IB-IL-AO-2/SF**- oder eine **R IB-IL-AO-2/U/BP**-Klemme, in Abhängigkeit vom gewählten **ModuleType**. Die Parametrierung erfolgt mit einer positiven Flanke am **Enable**-Eingang. Der FB konvertiert und gibt die normalisierten Werte an die Analogausgänge der Kanäle 1 und/ oder 2. Tritt ein Fehler auf, enthalten der Bausteingang **DiagCode** und Fehlerausgänge die entsprechenden Fehlerinformation.

Das Ausgangsverhalten im fehlerfreien Betrieb (Normalbetrieb): Im Normalbetrieb werden der parametrierte Ausgangsbereich und das Datenformat beim Spannungszuschalten der Klemme (Power Up) nichtflüchtig ausgelesen. Es besteht zusätzlich aber auch die Möglichkeit, diese Einstellungen sowie das Verhalten der Klemme im **Fehlerfall** flüchtig zu parametrieren. Diese Parametrierung kann zur Laufzeit durch eine Prozessdatensequenz vorgenommen werden.

Das Ausgangsverhalten im Fehlerfall: Im Fehlerfall werden sich die Ausgänge so verhalten, wie es nichtflüchtig eingestellt oder wie es nachträglich flüchtig parametriert wurde. Das heißt, die Ausgänge halten den letzten Wert (HOLD, Default-Einstellung), oder sie gehen auf Null zurück (RESET).

IL_INLN_AO_DATA

Die Struktur beschreibt die Eingangsdaten eines Kanals.

Name	Typ	Beschreibung	Min-Wert	Max-Wert	Default-Wert	Übernahme
NormalizedValue	REAL	Normalisierter Eingangswert für Analogausgang	LowLimit	HighLimit	0.0	kontinuierlich mit "Enable"
HighLimit	REAL	Obere Grenze des normalisierter Eingangswertes	-	-	10.0	kontinuierlich mit "Enable"
LowLimit	REAL	Untere Grenze des normalisierter Eingangswertes	-	-	0.0	kontinuierlich mit "Enable"

IL_INLN_SF_BP Diese Aufzählung dient der Auswahl des Typs der analogen Ausgangsklemme durch den Funktionsbaustein.

ENUM	Beschreibung
SF	R-IB IL AO 2/SF -Modul mit Ausgabebereich 0..10V, 0..20mA und 4..20mA
BPUNIPOLAR10V	R-IB IL AO 2/U/BP -Modul mit Ausgabebereich 0 ..10V
BPBIPOLAR10V	R-IB IL AO 2/U/BP -Modul mit Ausgabebereich +10 ..-10V

IL_INLN_DIAG_CODE Die Tabelle enthält kanalbezogene Fehler und Diagnosen, die von den Funktionsbausteinen der Bibliothek ausgegeben werden.

DiagCode	ENUM	Beschreibung
16#8001	OVERRANGE	Bereichsüberschreitung
16#8002	OPEN_CIRCUIT	Offener Stromkreis, der Ausgang ist als Stromquelle konfiguriert und der vorgegebene Strom kann nicht fließen
16#8003	OVER_LOAD	Kurzschluß/Überlast, der Ausgang ist als Spannungsquelle konfiguriert und überlastet
16#8004	MEASURED_VALUE_INVALID	Ungültiger Messwert / Messwert nicht verfügbar
16#8008	COLD_JUNCTION_DEFECTIVE	Vergleichsstelle defekt
16#8010	INVALID_CONFIGURATION	Ungültige Konfiguration
16#8020	ERROR_IN_IO_SUPPLY_VOLTAGE	Fehler in der Spannungsversorgung
16#8040	MODULE_FAULTY	Modulfehler
16#8080	UNDERRANGE	Bereichsunterschreitung

Tab. 14-73: Enum IL_INLN_DIAG_CODE

Beispiel Der FB konfiguriert **R IB-IL AO2/Sf** und **R IB-IL AO2/U/BP**-Klemmen. Im **R IB-IL AO2/Sf**-Modul ist die Auswahl **0..10V, 0..20mA** und **4 .. 20mA** durch Hardwarebrücken vorzunehmen.

In nachstehendem Beispiel konfiguriert der FB ein **R IB-IL AO2/Sf-Modul** in Kanal 1 auf **0 ... 10V** und in Kanal 2 auf **4 .. 20mA**. Es wird vorausgesetzt, dass Kanal 2 korrekte Hardwarebrücken enthält.

Der Anwender den angeschlossenen Modultyp über die Variable **Module-Type** vorzugeben. Die obere und die untere Grenze für Kanal 1 sind **10** und **0** und so folgt der Ausgang **0 ...10V** dem normalisierten Wert zwischen **0** und **10**.

Deklaration zum Beispiel

```

PROGRAM PrgIL_AO2SfBp
VAR
  fbIL_AO2SfBp: IL_AO2SfBp;
  wInProcessData_0: WORD;
  wInProcessData_1: WORD;
  wOutProcessData_0: INT;
  wOutProcessData_1: WORD;
  enDiagCode_0: IL_INLN_DIAG_CODE;
  enDiagCode_1: IL_INLN_DIAG_CODE;
END_VAR

```

Implementation zum Beispiel

```

// Configuration
fbIL_AO2SfBp.ModuleType := IL_INLN_SF_BP_SF ;
fbIL_AO2SfBp.ResetOutput := TRUE ;
fbIL_AO2SfBp.NonVolatile := TRUE ;
fbIL_AO2SfBp.InputData[0].HighLimit := REAL#10.0 ;
fbIL_AO2SfBp.InputData[0].LowLimit := REAL#0.0 ;
fbIL_AO2SfBp.InputData[1].HighLimit := REAL#20.0 ;
fbIL_AO2SfBp.InputData[1].LowLimit := REAL#4.0 ;
fbIL_AO2SfBp.ChannelSelect := BYTE#2#00000011 ;
fbIL_AO2SfBp.InProcData_0 := wInProcessData_0 ;
fbIL_AO2SfBp.InProcData_1 := wInProcessData_1 ;
wOutProcessData_0 := fbIL_AO2SfBp.OutProcData_0 ;
wOutProcessData_1 := fbIL_AO2SfBp.OutProcData_1 ;

// FB call
fbIL_AO2SfBp(Enable:= TRUE);

// Data output
(* Once the function block reaches the in operation mode the module starts
   generating voltage and current signals.
   The out can be varying by varying the value at Normalized value input. *)
IF fbIL_AO2SfBp.InOperation = TRUE THEN
  fbIL_AO2SfBp.InputData[0].NormalizedValue := REAL#10.0 ;
  fbIL_AO2SfBp.InputData[1].NormalizedValue := REAL#15.0 ;
  enDiagCode_0 := fbIL_AO2SfBp.DiagCode_0 ;
  enDiagCode_1 := fbIL_AO2SfBp.DiagCode_1 ;
END_IF

```

Prozessdaten-Interface zum Beispiel

Die Prozesseingangs- und -ausgangswerte des Funktionsbausteines sind auf die Prozessdatenworte des einzelnen Feldbussystems zu mappen.

Die Variablen arInProcessData_x und arOutProcessData_x sind im Programm prgIL_AO2SfBp deklariert.

Variable	Mapping	Channel	Address	Type
Application.PrgIL_AO2SfBp.wInProcessData_0		Input0	%IW6	WORD
Application.PrgIL_AO2SfBp.wInProcessData_1		Word1	%IW8	WORD
Application.PrgIL_AO2SfBp.wOutProcessData_0		Output0	%QW6	WORD
Application.PrgIL_AO2SfBp.wOutProcessData_1		Word1	%QW8	WORD

Abb. 14-23: Prozessdaten-Interface

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle INLINEIO_TABLE, 16#0190. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00003003	16#00000000	ChannelSelect ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003009	channel number	Die Bereichsgrenzen sind nicht korrekt HighLimit <= LowLimit.
INPUT_INVALID_ERROR (16#0001)	16#00003010	16#00000000	ModuleType ist ungültig
DEVICE_ERROR (16#0008)	16#00003020	16#00000000	Zeitüberschreitung, das Gerät ist nicht erreichbar. Überprüfen Sie den Modultyp
DEVICE_ERROR (16#0008)	16#00003020	16#00008010	Ungültige Konfiguration oder widersprüchliche Hardwarebrücken

Tab. 14-74: IL_AO2SfBp Fehlercodes

14.3.3 IL_AO2UI

Kurzbeschreibung

Der Funktionsbaustein **IL_AO2UI** unterstützt die Konfiguration der zweikanaligen Inline-Analog-Ausgabeklemme **R-IB IL AO 2/UI-PAC**.

Nach erfolgreicher Konfiguration konvertiert der FB pro Kanal normalisierte Eingangswerte und übergibt sie an die Analog-Ausgänge der Inline-Klemme.

Für jede Inline-EA-Klemme wird eine Funktionsbausteininstanz benötigt.

Kompatibilität

Der Funktionsbaustein ist verwendbar für Inline-, Profibus und Profinet IO-Feldbus.

Schnittstellenbeschreibung

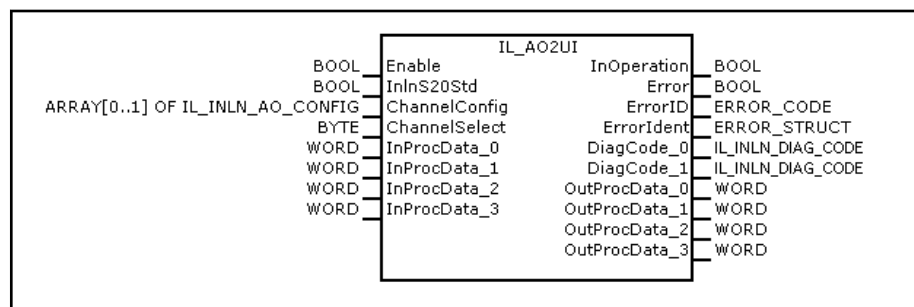


Abb. 14-24: Funktionsbaustein IL_AO2UI

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Kontinuierliche Bearbeitung des FBs, solange "Enable" TRUE ist
	InlnS20Std	BOOL	Auswahl des Modultyps: TRUE: S20 FALSE: Inline
	ChannelConfig	ARRAY[0..1] OF IL_INLN_AO_CONFIG	Analog-Ausgabe-Kanalkonfiguration IL_INLN_AO_CONFIG auf Seite 632
	ChannelSelect	BYTE	Kanalauswahl, BYTE#2#000000yx; x - Kanal 1, y - Kanal 2
	InProcData_0	WORD	Prozesseingangswort 0
	InProcData_1	WORD	Prozesseingangswort 1
	InProcData_2	WORD	Prozesseingangswort 2

	Name	Typ	Beschreibung
	InProcData_3	WORD	Prozesseingangswort 3
VAR OUTPUT	InOperation	BOOL	Wenn TRUE, arbeitet der Baustein in seiner aktuellen Task
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb der FB-Abarbeitung eingetreten ist
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Detailinformationen zum Fehler Fehlerbehandlung auf Seite 634
	DiagCode_0	IL_INLN_DIAG_CODE	Diagnoseinformation Kanal 0 IL_INLN_DIAG_CODE auf Seite 632
	DiagCode_1	IL_INLN_DIAG_CODE	Diagnoseinformation Kanal1
	OutProcData_0	WORD	Prozessausgangswort 0
	OutProcData_1	WORD	Prozessausgangswort 1
	OutProcData_2	WORD	Prozessausgangswort 2
	OutProcData_3	WORD	Prozessausgangswort 3

Tab. 14-75: Schnittstelle IL_AO2UI

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	kontinuierlich
InInS20Std	BOOL	FALSE	TRUE	FALSE	Positive Flanke von "Enable"
ChannelSelect	BYTE	0	3	3	kontinuierlich mit "Enable"

Tab. 14-76: Eingangsverhalten IL_AO2UI

Funktionsbeschreibung

Der IL_AO2UI-Funktionsbaustein konvertiert zyklisch normalisierte Eingabewerte in Analogwerte für das R-IB IL AO 2/UI-PAC-Klemme. Das R-IB IL AO 2/UI-PAC-Klemme besteht aus zwei Analog-Ausgangskanälen mit Spannungsbereichen: 0 V ... 10 V, -10 V ... +10 V bzw. Strombereichen: 0 mA ... 20 mA, 4 mA ... 20 mA, ± 20 mA.

Die Ausgangsbereiche können über den Aufzählungstyp IL_INLN_AO_RANGE, "ChannelConfig.OutputRange" konfiguriert werden.

Der Eingang "InInS20Std", entscheidet zwischen INLINE-Unterstützung (FALSE) und S20-Unterstützung (TRUE).

Die Kanäle werden mit der positiven Flanke an "Enable" konfiguriert.

Nach Abschluss des Vorganges wird der Ausgang "InOperation" auf "TRUE" gesetzt.

Eine Rekonfiguration ist nur durch eine erneute positiven Flanke an "Enable" möglich.

Bei ungültiger Konfiguration generiert der Funktionsbaustein einen Fehler: "Error", "ErrorID" und "ErrorIdent" werden entsprechend gesetzt.

Die Initialwerte der Funktionsbaustein-Parametrierung entsprechen des Initialwerten der Analogklemme.

Der Funktionsbaustein konvertiert normalisierte Eingangswerte in Analogwerte für Kanal 1 und/oder Kanal 2 und gibt sie aus.

Tritt ein Fehler in einem der Kanäle auf, wird die Fehlervariable auf TRUE gesetzt und der "DiagCode" zeigt die Fehlerinformation für den konkreten Kanal an.

IL_INLN_AO_CONFIG Die Struktur beschreibt die Eingangsdaten **eines** Kanals.

Name	Typ	Beschreibung	Min-Wert	Max-Wert	Default-Wert	Übernahme
NormalizedValue	REAL	Normalisierter Eingangswert für den Analogausgang	LowLimit	HighLimit	0.0	kontinuierlich mit "Enable"
HighLimit	REAL	Obere Grenze des normalisierter Eingangswertes	-	-	10.0	Positive Flanke von "Enable"
LowLimit	REAL	Untere Grenze des normalisierter Eingangswertes	-	-	0.0	Positive Flanke von "Enable"
OutputRange	IL_INLN_AO_RANGE	Meßbereich von Analog-Ausgang-Spannung/ - Strom	UNIPO-LAR10V	UNIPO-LAR4TO20 MA	UNIPO-LAR10V	Positive Flanke von "Enable"

IL_INLN_AO_RANGE Diese Aufzählung dient der Auswahl einer geeigneten Ausgangs-Kanal-Spannung oder eines geeigneten Ausgangs-Kanal-Stromes durch den Funktionsbaustein.

ENUM	Wert	Beschreibung
UNIPOLAR10V	0	Meßspannung 0 bis 10 V
BIPOLAR10V	1	Meßspannung -10 bis +10 V
UNIPOLAR20MA	8	Meßstrom 0 bis 20 mA
BIPOLAR20MA	9	Meßstrom -20 bis +20 mA
UNIPOLAR4TO20MA	10	Meßstrom 4 bis 20 mA

IL_INLN_DIAG_CODE Die Tabelle enthält kanalbezogene Fehler und Diagnosen, die von den Funktionsbausteinen der Bibliothek ausgegeben werden.

DiagCode	ENUM	Beschreibung
16#8001	OVERRANGE	Bereichsüberschreitung
16#8002	OPEN_CIRCUIT	Offener Stromkreis, der Ausgang ist als Stromquelle konfiguriert und der vorgegebene Strom kann nicht fließen
16#8003	OVER_LOAD	Kurzschluß/Überlast, der Ausgang ist als Spannungsquelle konfiguriert und überlastet
16#8010	INVALID_CONFIGURATION	Ungültige Konfiguration
16#8020	ERROR_IN_IO_SUPPLY_VOLTAGE	Fehler in der Spannungsversorgung
16#8040	MODULE_FAULTY	Modulfehler
16#8080	UNDERRANGE	Bereichsunterschreitung

Tab. 14-77: Enum IL_INLN_DIAG_CODE

Beispiel Der FB konfiguriert das **R IB-IL AO 2/UI**-Modul für Kanal 1 zu 0 to 10V "arChannelConfig[0].OutputRange" und für Kanal 2 zu +/- 10V "arChannelConfig[1].OutputRange".

Die normalisierten Eingangsdaten "arChannelConfig[x].NormalizedValue" liegen zwischen "arChannelConfig[x].HighLimit" und "arChannelConfig[x].LowLimit" und ergeben die Ausgangsdaten an den Prozeßausgangsdaten des entsprechenden Kanals.

Deklaration zum Beispiel

```
PROGRAM prgIL_AO2UI
VAR
  fbIL_AO2UI:      IL_AO2UI;
  bExecute:        BOOL:= FALSE;
  bInlnS20Std:     BOOL:= FALSE; // FALSE: Inline Standard, TRUE:S20 Standard
  arChannelConfig: ARRAY [0..1] OF IL_INLN_AO_CONFIG; // Analog Channel Configuration
  byChannelSelect: BYTE := 3;    // Channel Selection
  wInProcData_0:   WORD;          // Input Process Data Word 1
  wInProcData_1:   WORD;          // Input Process Data Word 2
  wInProcData_2:   WORD;          // Input Process Data Word 3
  wInProcData_3:   WORD;          // Input Process Data Word 4
  bInOperation:    BOOL;
  bError:          BOOL;
  enErrorID:       ERROR_CODE;
  enErrorIdent:    ERROR_STRUCT;
  enDiagCode_0:    IL_INLN_DIAG_CODE; // Diagnostic code Channel -1
  enDiagCode_1:    IL_INLN_DIAG_CODE; // Diagnostic code Channel -2
  wOutProcData_0:  WORD;          // Output Process Data Word 1
  wOutProcData_1:  WORD;          // Output Process Data Word 2
  wOutProcData_2:  WORD;          // Output Process Data Word 3
  wOutProcData_3:  WORD;          // Output Process Data Word 4

  rNormalizedValue_0: REAL := 5.0; //Normalized Value for Channel 1
  rNormalizedValue_1: REAL := -5.0; //Normalized Value for Channel 2
END_VAR
```

Implementation zum Beispiel

```
//Channel 1 Configuration : UNIPOLAR10V 0 to 10V
arChannelConfig[0].LowLimit      := REAL#0.0;
arChannelConfig[0].HighLimit     := REAL#10.0;
arChannelConfig[0].OutputRange   := IL_INLN_AO_RANGE.UNIPOLAR10V;

//Channel 2 Configuration : BIPOLAR10V -10V to +10V
arChannelConfig[1].LowLimit      := REAL#-10.0;
arChannelConfig[1].HighLimit     := REAL#10.0;
arChannelConfig[1].OutputRange   := IL_INLN_AO_RANGE.BIPOLAR10V;

//FB Call
fbIL_AO2UI(
  Enable:=bExecute ,
  InlnS20Std:=bInlnS20Std ,
  ChannelConfig:= arChannelConfig ,
  ChannelSelect:= byChannelSelect ,
  InProcData_0:= wInProcData_0 ,
  InProcData_1:= wInProcData_1 ,
  InProcData_2:= wInProcData_2 ,
  InProcData_3:= wInProcData_3 ,
  InOperation:= bInOperation ,
  Error=> bError ,
  ErrorID=> enErrorID ,
  ErrorIdent=> enErrorIdent ,
  DiagCode_0=> enDiagCode_0 ,
  DiagCode_1=> enDiagCode_1 ,
  OutProcData_0=> wOutProcData_0 ,
  OutProcData_1=> wOutProcData_1 ,
  OutProcData_2=> wOutProcData_2 ,
  OutProcData_3=> wOutProcData_3);

Data Write:

//Passing Normalized Values to channels
IF fbIL_AO2UI.InOperation = TRUE THEN
  arChannelConfig[0].NormalizedValue := rNormalizedValue_0;
  arChannelConfig[1].NormalizedValue := rNormalizedValue_1;
END_IF
```

Prozessdaten-Interface zum Beispiel

Die Prozesseingangs- und -ausgangsworte des Funktionsbausteines sind auf die Prozessdatenworte des einzelnen Feldbussystems zu mappen.

Die Variablen arInProcessData_x und arOutProcessData_x sind im Programm prgIL_AO2UI deklariert.

Inline-Module I/O Mapping				
Channels				
Variable	Mapping	Channel	Address	Type
Application.prgIL_AO2UI.wInProcData_0 Application.prgIL_AO2UI.wInProcData_1 Application.prgIL_AO2UI.wInProcData_2 Application.prgIL_AO2UI.wInProcData_3		Input0	%IW0	
		Word0	%IW0	WORD
		Word1	%IW2	WORD
		Word2	%IW4	WORD
Application.prgIL_AO2UI.wOutProcData_0 Application.prgIL_AO2UI.wOutProcData_1 Application.prgIL_AO2UI.wOutProcData_2 Application.prgIL_AO2UI.wOutProcData_3		Output0	%QW0	
		Word0	%QW0	WORD
		Word1	%QW2	WORD
		Word2	%QW4	WORD

Abb. 14-25: Prozessdaten-Interface

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle INLINEIO_TABLE, 16#0190. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00003003	16#00000000	ChannelSelect ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003004	channel number	OutputRange ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003009	channel number	Die Bereichsgrenzen sind nicht korrekt HighLimit <= LowLimit.
DEVICE_ERROR (16#0008)	16#00003020	16#00000000	Zeitüberschreitung, das Gerät ist nicht erreichbar. Überprüfen Sie den Modultyp
DEVICE_ERROR (16#0008)	16#00003020	16#00008010	Ungültige Modul-Konfiguration

Tab. 14-78: IL_AO2UI Fehlercodes

14.4 Funktionsklemmen

14.4.1 IL_CntCount

- Kurzbeschreibung

Der Funktionsbaustein **IL_CntCount** parametriert die **R-IB IL CNT**-Klemme (Inline Zählerklemme) in der Betriebsart Ereigniszählung und liest das Zählergebnis.
- Kompatibilität

Für jede Inline-EA-Klemme wird eine Funktionsbausteininstanz benötigt.
- Der Funktionsbaustein ist verwendbar für Inlinebus, Profibus, sercos III und Profinet IO-Feldbus.

Schnittstellenbeschreibung

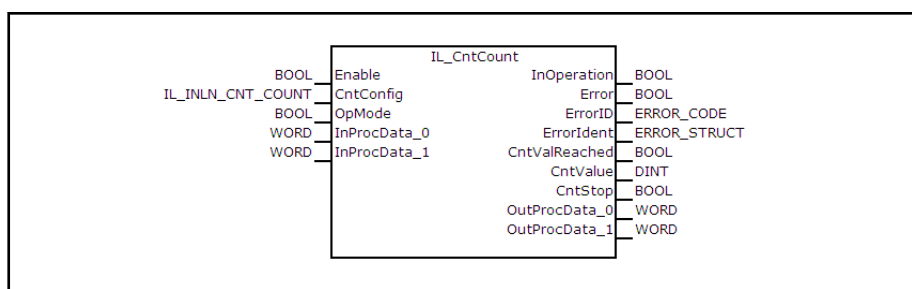


Abb. 14-26: Funktionsbaustein IL_CntCount

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Kontinuierliche Bearbeitung des FBs, solange "Enable" TRUE ist
	CntConfig	IL_INLN_CNT_COUNT	Struktur zur Zählerkonfiguration, siehe "IL_INLN_CNT_COUNT" auf Seite 636
	Pause	BOOL	TRUE = Zählpause FALSE = Fortsetzung des Zählvorgang
	InProcData_0	WORD	Prozesseingangswort 0
	InProcData_1	WORD	Prozesseingangswort 1
VAR OUTPUT	InOperation	BOOL	Wenn TRUE, arbeitet der Baustein in seiner aktuellen Task
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb der FB-Abarbeitung eingetreten ist
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Detailinformationen zum Fehler, siehe "Fehlerbehandlung" auf Seite 640
	CntValReached	BOOL	TRUE - Endwert erreicht, das Ausgangsverhalten hängt von den unter "OutputBehavior" getroffenen Festlegungen ab
	CntValue	DINT	Aktueller Zählerstand
	CntStop	BOOL	Zähler ist gestoppt/pausiert und "CntValue" enthält den aktuellen Zählerstand
	OutProcData_0	WORD	Prozessausgangswort 0
	OutProcData_1	WORD	Prozessausgangswort 1

Tab. 14-79: Schnittstelle IL_CntCount

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	kontinuierlich
Pause	BOOL	FALSE	TRUE	FALSE	kontinuierlich mit "Enable"

Tab. 14-80: Eingangsverhalten IL_CntCount

Funktionsbeschreibung

Aktivieren Sie die FB-Instanz erst nach dem ersten SPS-Zyklus durch eine positive Flanke am **Enable**-Eingang.

Die Parametrierung der **R-IB IL CNT**-Klemme als Zähler setzt eine Belegung der Parameterstruktur **IL_INLN_CNT_COUNT** mit dazu passenden Werten voraus.

Durch eine positive Flanke am **Enable**-Eingang wird die Klemme konfiguriert. Der erfolgreiche Abschluss der Konfiguration wird durch **InOperation** = TRUE angezeigt und die gültigen Zählwerte sind am **CntValue**-Ausgang verfügbar.

Bei ungültigen Eingangsparametern generiert der FB einen Fehler und die entsprechenden Informationen gelangen an **Error**, **ErrorID** und **ErrorIdent**.

Eine positive Flanke am **Enable**-Eingang löscht die Fehler und rekonfiguriert die Klemme.

Eine positive Flanke am **Pause**-Eingang kann den Zählvorgang zu jedem beliebigen Zeitpunkt unterbrechen

Ein 0/1-Übergang am **CntStop**-Ausgang zeigt an, dass der Zähler im Stopp-Zustand ist. Der aktuelle Zählerstand kann am **CntValue**-Ausgang gelesen werden.

Der unterbrochene Zählvorgang kann durch eine positive Flanke am **Pause**-Eingang wieder gestartet werden.

- 
1. Wird die Klemme am Profibus oder Profinet-Feldbus verwendet, muss die Byte-Reihenfolge geändert werden:
Byte 0/1/2/3 = Plug 4/3/2/1.

2. Die Veränderung der Byte-Reihenfolge an der Busklemme ist für alle der Busklemme folgenden Klemmen wirksam!

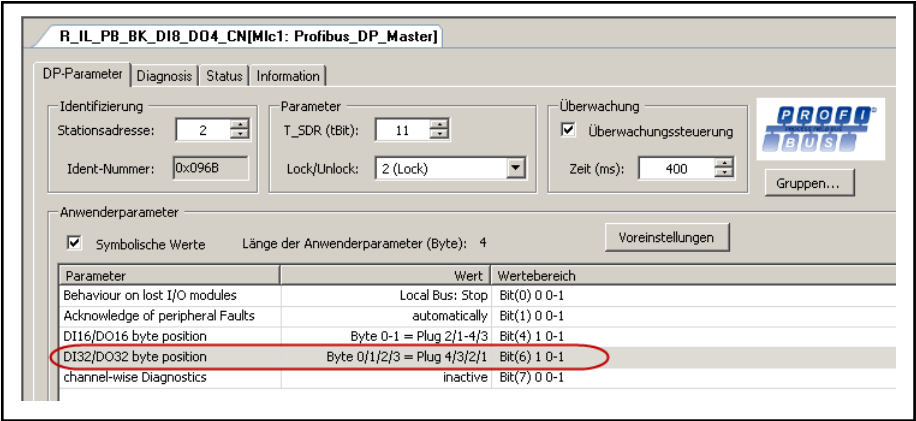


Abb. 14-27: Dialog Busklemme mit geänderter Bytefolge

IL_INLN_CNT_COUNT

Nachstehende Struktur erlaubt die Konfiguration der Zählerklemme in der Betriebsart Ereigniszählung.

Name	Typ	Beschreibung	Min-Wert	Max-Wert	Default-Wert	Übernahme
BusReset	BOOL	FALSE - Bus-Reset setzt den Ausgang zurück, stoppt alle Zähloperationen und löscht die Betriebsarteneinstellungen TRUE – keine Reaktion auf einen Bus-Reset	FALSE	TRUE	FALSE	Positive Flanke an "Enable"
InConfig	IL_INLN_CNT_INCONFIG	Siehe "IL_INLN_CNT_INCONFIG" auf Seite 637	NO_INPUT_CONFIG	SOURCE_GATE_OPERATION	NO_INPUT_CONFIG	Positive Flanke an "Enable"
SourceGate-Operation	IL_INLN_CNT_SG_OPERATION	Siehe "IL_INLN_CNT_SG_OPERATION" auf Seite 638	NONE	SOURCE_OR_GATE	NONE	Positive Flanke an "Enable"
PulseWidth	IL_INLN_CNT_PULSE_WIDTH	Pulslänge des digitalen Ausgangs, siehe "IL_INLN_CNT_PULSE_WIDTH" auf Seite 639	PULSE_10MS	PULSE_1000MS	PULSE_10MS	Positive Flanke an "Enable"
CntRepeat	BOOL	FALSE - einmalige Zählung TRUE - ständige Zählfortsetzung	FALSE	TRUE	FALSE	Positive Flanke an "Enable"
Gate	IL_INLN_CNT_GATE	Siehe "IL_INLN_CNT_GATE" auf Seite 638	NO_FUNCTION	CNT_AT_HIGH_RESET_ON_RISE	NO_FUNCTION	Positive Flanke an "Enable"
CntDir	BOOL	TRUE – aufwärts zählen, FALSE – abwärts zählen	FALSE	TRUE	FALSE	Positive Flanke an "Enable"
OutputBehavior	IL_INLN_CNT_OUTBEHAVE	Siehe "IL_INLN_CNT_OUTBEHAVE" auf Seite 638	HIGH_PULSE	LOW_PULSE	HIGH_PULSE	Positive Flanke an "Enable"
InitialValue	DINT	Initialwert Start Zählvorgang (0-16777215)	0	16777215	0	Positive Flanke an "Enable"
FinalValue	DINT	Endwert Zählvorgang (0-16777215)	0	16777215	0	Positive Flanke an "Enable"

Tab. 14-81: Zulässige Werte von IL_INLN_CNT_COUNT für IL_CntCount

IL_INLN_CNT_INCONFIG

Die Aufzählung gestattet die Konfiguration eines Eingangsfilters oder einer Eingangslogik zur Verknüpfung von zwei Eingangssignalen.

ENUM	Beschreibung
NO_INPUT_CONFIG	Keine Eingangskonfiguration
FILTER_FOR_MECH_SWITCH	RC-Filter an Source und Gate zu schalten, das ermöglicht den Einsatz von mechanischen Schaltern. (z.B. Relais)
SOURCE_GATE_OPERATION	Source Gate Logic Operation, als Zählsignal kann eine Verknüpfung aus Source- und Gate-Signal gewählt werden.

Source-Gate-Logic-Operation, siehe ["IL_INLN_CNT_SG_OPERATION"](#) auf Seite 638

IL_INLN_CNT_SG_OPERATION Bei der Source-Gate-Verknüpfung wird aus den beiden Eingangssignalen Source (S) und Gate (G) ein Signal gebildet, das nach der Verarbeitung entsprechend der logischen Funktion als neues Source-Signal zur Verfügung steht. Es wird im Weiteren als Source' (S') bezeichnet. Das ursprüngliche Source-Signal steht jetzt nicht mehr zur Verfügung.

Das ursprüngliche Gate-Signal kann weiterhin verwendet werden.

Über diese Source-Gate-Verknüpfung können Sie die gebräuchlichen logischen Funktionen realisieren.

ENUM	Beschreibung
NONE	Keine Eingangsverknüpfung
SOURCE_NOR_GATE	"S' = S NOR G". S' wird gebildet aus NOR zwischen Source und Gate
SOURCE_AND_NOT_GATE	"S' = S AND /G". S' wird gebildet aus AND zwischen Source und invertiertem Gate
NOT_GATE	"S' = /G". S' wird gebildet aus invertiertem Gate
NOT_SOURCE_AND_GATE	"S' = /S AND G". S' wird gebildet aus AND zwischen invertierter Source und Gate
NOT_SOURCE	"S' = /S". S' wird gebildet aus invertierter Source.
SOURCE_EXOR_GATE	"S' = S EXOR G". S' wird gebildet aus EXOR zwischen Source und Gate
SOURCE_NAND_GATE	"S' = S NAND G". S' wird gebildet aus NAND zwischen Source and Gate
SOURCE_AND_GATE	"S' = S AND G". S' wird gebildet aus AND zwischen Source and Gate
SOURCE_EXNOR_GATE	"S' = S EXNOR G". S' wird gebildet aus EXNOR zwischen Source und Gate
SOURCE	"S' = S". S' wird gebildet aus Source
SOURCE_OR_NOT_GATE	"S' = S OR /G". S' wird gebildet aus OR zwischen Source und invertiertem Gate
GATE	"S' = G". S' wird gebildet aus Gate
NOT_SOURCE_OR_GATE	"S' = /S OR G". S' wird gebildet aus OR zwischen invertierter Source und Gate
SOURCE_OR_GATE	"S' = S OR G". S' wird gebildet aus OR zwischen Source und Gate

IL_INLN_CNT_GATE Die Aufzählung beschreibt die Bedingung am Gate-Eingang, die für den Zählprozess erfüllt sein muss.

ENUM	Beschreibung
NO_FUNCTION	keine Funktion
CNT_AT_HIGH	Zählung bei High-Pegel an Gate
CNT_AT_LOW	Zählung bei Low-Pegel an Gate
CNT_ON_RISE	Start der Zählung bei steigender Flanke (0/1) an Gate
CNT_ON_FALL	Start der Zählung bei fallender Flanke (1/0) an Gate
CNT_AT_HIGH_RESET_ON_RISE	Zählung bei High-Pegel; bei steigender Flanke wird der Zählwert zurückgesetzt

IL_INLN_CNT_OUTBEHAVE Die Aufzählung definiert das Schaltverhalten des digitalen Ausgangs beim TerminalCount (Erreichen des Endwertes).

ENUM	Beschreibung
INACTIVE	Ausgang nicht aktiv
HIGH_PULSE	Ein positiver Impuls wird generiert

ENUM	Beschreibung
LOW_PULSE	Ein negativer Impuls wird generiert
TOGGLE_LOW	Vorheriger Zustand wird invertiert , initial LOW
TOGGLE_HIGH	Vorheriger Zustand wird invertiert , initial HIGH
HIGH	Ausgang auf HIGH
LOW	Ausgang auf LOW

IL_INLN_CNT_PULSE_WIDTH

Der Aufzählungstyp gibt die Pulslänge des digitalen Ausgangs vor.

ENUM	Beschreibung
PULSEWIDTH_10MS	10 Millisekunden
PULSEWIDTH_50MS	50 Millisekunden
PULSEWIDTH_100MS	100 Millisekunden
PULSEWIDTH_200MS	200 Millisekunden
PULSEWIDTH_300MS	300 Millisekunden
PULSEWIDTH_400MS	400 Millisekunden
PULSEWIDTH_500MS	500 Millisekunden
PULSEWIDTH_1000MS	1000 Millisekunden

Beispiel

Der Funktionsbaustein wird benötigt, um die Systemeinstellungen für eine Zählerklemme in der Betriebsart Frequenzmessung vorzunehmen.

Systemeinstellungen sind Basiseinstellungen für alle Betriebsarten der Klemme.

In den Systemeinstellungen sind enthalten: Eingangs-Konfiguration, Source-Gate-Verknüpfungen, Bus-Reset-Verhalten und Pulslängen-Parameter.

Wenn eine Eingangskonfiguration mit Logic-Source-Gate-Operation gewählt ist, dann kann eine logische Verknüpfung zur Bildung des Source-Signals **S'** aus dem (Eingangs-)Source-Signal **S** und dem Gate-Signal **G** erfolgen.

In diesem Beispiel ist die Zählerklemme so konfiguriert, dass sie wiederholt von einem Startwert von 100 zu einem Endwert von 500 zählt.

Das Ausgangsverhalten ist mit **HIGH_PULSE** festgelegt, es wird bei Erreichen des Zählerendwertes am Ausgang einen 0/1-Übergang generiert.

Das Aktivieren von **Pause** stoppt den Zählvorgang, das Zählen beginnt mit 1/0-Übergang an **Pause** erneut.

Deklaration zum Beispiel

```
PROGRAM PrgIL_CntCount
VAR
    fbIL_CntCount    : IL_CntCount ;
    wInProcessData_0: WORD;
    wInProcessData_1: WORD;
    wOutProcessData_1: WORD;
    wOutProcessData_0: WORD;
    diCntValue: DINT;
    bCntValReached: BOOL;
END_VAR
```

Implementation zum Beispiel

```
// System settings
fbIL_CntCount.CntConfig.BusRese := TRUE;
```

RIL_InlineIOModules.library

```

fbIL_CntCount.CntConfig.Gate          := IL_INLN_CNT_GATE.CNT_AT_HIGH      ;
fbIL_CntCount.CntConfig.InConfig      := IL_INLN_CNT_INCONFIG.SOURCE_GATE_OPERATION;
fbIL_CntCount.CntConfig.SourceGateOperation := IL_INLN_CNT_SG_OPERATION.SOURCE;
fbIL_CntCount.CntConfig.PulseWidth    := IL_INLN_CNT_PULSE_WIDTH.PULSEWIDTH_1000MS;

// Parameter settings for event count mode
fbIL_CntCount.CntConfig.CntDir        := TRUE;
fbIL_CntCount.CntConfig.CntRepeat     := TRUE;
fbIL_CntCount.CntConfig.OutputBehavior := IL_INLN_CNT_OUTBEHAVE.HIGH_PULSE;
fbIL_CntCount.CntConfig.InitialValue  := DINT#100      ;
fbIL_CntCount.CntConfig.FinalValue    := DINT#500;
fbIL_CntCount.InProcData_0            := wInProcessData_0;
fbIL_CntCount.InProcData_1            := wInProcessData_1;
wOutProcessData_0                     := fbIL_CntCount.OutProcData_0;
wOutProcessData_1                     := fbIL_CntCount.OutProcData_1;

// FB call
fbIL_CntCount (Enable:=TRUE);

// Data read
IF fbIL_CntCount.InOperation = TRUE THEN
    diCntValue := fbIL_CntCount.CntValue;
    bCntValReached := fbIL_CntCount.CntValReached;
END_IF

// Stopping the counter operation
IF fbIL_CntCount.InOperation = TRUE THEN
    fbIL_CntCount.Pause := TRUE ;
END_IF

```

Prozessdaten-Interface zum Beispiel

Die Prozesseingangs- und -ausgangsworte des Funktionsbausteines sind auf die Prozessdatenworte des einzelnen Feldbussystems zu mappen.

Die Variablen wInProcessData_x und wOutProcessData_x sind im Programm prgIL_CntCount deklariert.

Variable	Mapping	Channel	Address	Type
Application.PrgIL_CntCount.wInProcessData_0		Input0	%IW24	
Application.PrgIL_CntCount.wInProcessData_1		Word0	%IW24	WORD
		Word1	%IW26	WORD
Application.PrgIL_CntCount.wOutProcessData_0		Output0	%QW24	
Application.PrgIL_CntCount.wOutProcessData_1		Word0	%QW24	WORD
		Word1	%QW26	WORD

Abb. 14-28: Prozessdaten-Interface

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle INLINEIO_TABLE, 16#0190. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000001	'InConfig' ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000002	'SourceGateOperation' ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000003	'PulseWidth' ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000004	'Gate' ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000005	'OutputBehavior' ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000006	'InitialValue' ist ungültig

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000007	'FinalValue' ist ungültig
DEVICE_ERROR (16#0008)	16#00003020	16#00000000	Zeitüberschreitung, Gerät antwortet nicht

Tab. 14-82: IL_CntCount Fehlercodes

14.4.2 IL_CntFrequency

Kurzbeschreibung Der Funktionsbaustein **IL_CntFrequency** parametrisiert die **R-IB IL CNT**-Klemme (Inline Zählerklemme) in der Betriebsart Frequenzmessung und liest die Frequenz (zeitgesteuert oder zustandsgesteuert).

Für jede Inline-EA-Klemme wird eine Funktionsbausteininstanz benötigt.

Kompatibilität Der Funktionsbaustein ist verwendbar für Inlinebus, Profibus, sercos III und Profinet IO-Feldbus.

Schnittstellenbeschreibung

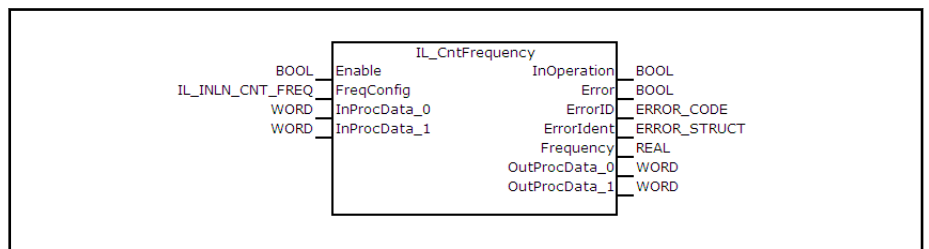


Abb. 14-29: Funktionsbaustein IL_CntFrequency

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Kontinuierliche Bearbeitung des FBs, solange "Enable" TRUE ist
	FreqConfig	IL_INLN_CNT_FREQ	Struktur zur Zählerkonfiguration, siehe "IL_INLN_CNT_FREQ" auf Seite 643
	InProcData_0	WORD	Prozesseingangswort 0
	InProcData_1	WORD	Prozesseingangswort 1

	Name	Typ	Beschreibung
VAR OUTPUT	InOperation	BOOL	Wenn TRUE, arbeitet der Baustein in seiner aktuellen Task
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb der FB-Abarbeitung eingetreten ist
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Detailinformationen zum Fehler, siehe "Fehlerbehandlung" auf Seite 647
	Frequency	REAL	In der zeitgesteuerten Betriebsart wird die normierte Frequenz in [Hz] ausgegeben. Achtung: Die Werte sind nur dann gültig, wenn die Torzeit (Zeit, während der gemessen wird) abgelaufen ist. Bei der zustandsgesteuerten Betriebsart werden die gezählten Impulse der relevanten Messung ausgegeben.
	OutProcData_0	WORD	Prozessausgangswort 0
	OutProcData_1	WORD	Prozessausgangswort 1

Tab. 14-83: Schnittstelle IL_CntFrequency

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	kontinuierlich

Tab. 14-84: Eingangsverhalten IL_CntFrequency

Funktionsbeschreibung

Aktivieren Sie die FB-Instanz erst nach dem ersten SPS-Zyklus durch eine positive Flanke am **Enable**-Eingang.

Die Zählerklemme wird im Frequenz-Mode mit einer positiven Flanke an Enable konfiguriert.

Die Frequenz kann zeit- oder zustandsgesteuert gemessen werden.

Um den Zähler parametrieren zu können, müssen alle Parameter mit korrespondierenden Werten versorgt werden.

Mit der positiven Flanke am **Enable**-Eingang wird die Klemme konfiguriert. Der erfolgreiche Abschluss der Konfiguration wird durch **InOperation** = TRUE angezeigt und die gültigen Zählwerte sind am **Frequency**-Ausgang verfügbar.

Bei ungültigen Eingangsparametern generiert der FB einen Fehler und die entsprechenden Informationen gelangen an **Error**, **ErrorID** und **ErrorIdent**.

Eine positive Flanke am **Enable**-Eingang löscht die Fehler und rekonfiguriert die Klemme.



1. Wird die Klemme am Profibus oder Profinet-Feldbus verwendet, muss die Byte-Reihenfolge geändert werden:
Byte 0/1/2/3 = Plug 4/3/2/1.
2. Die Veränderung der Byte-Reihenfolge an der Busklemme ist für alle der Busklemme folgenden Klemmen wirksam!

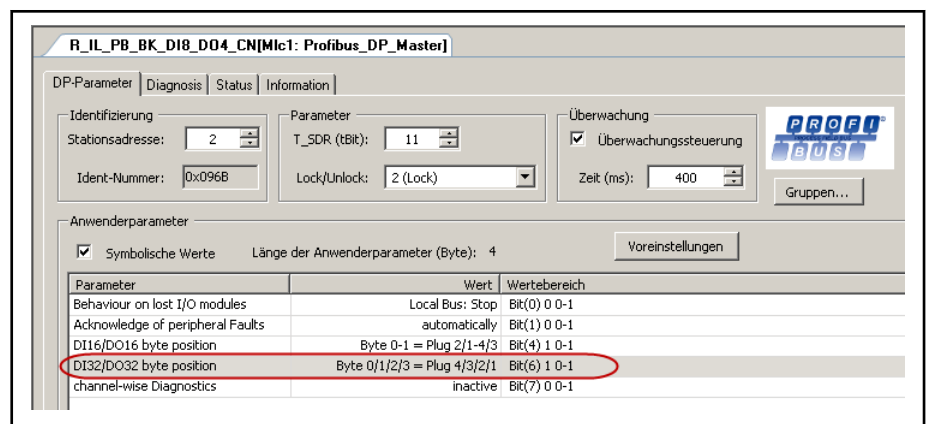


Abb. 14-30: Dialog Busklemme mit geänderter Bytefolge

IL_INLN_CNT_FREQ Die Struktur unterstützt die Konfiguration der Zählerklemme in der Betriebsart Frequenzmessung.

Name	Typ	Beschreibung	Min-Wert	Max-Wert	Default-Wert	Übernahme
BusReset	BOOL	FALSE - Bus-Reset setzt den Ausgang zurück, stoppt alle Zähloperationen und löscht die Betriebsarteneinstellungen. TRUE – keine Reaktion auf einen Bus-Reset.	FALSE	TRUE	FALSE	Kontinuierlich mit "Enable"
InConfig	IL_INLN_CNT_INCONFIG	Siehe " IL_INLN_CNT_INCONFIG " auf Seite 644	NO_INPUT_CONFIG	SOURCE_GATE_OPERATION	NO_INPUT_CONFIG	Positive Flanke an "Enable"
SourceGate-Operation	IL_INLN_CNT_SG_OPERATION	Siehe " IL_INLN_CNT_SG_OPERATION " auf Seite 644	NONE	SOURCE_OR_GATE	NONE	Positive Flanke an "Enable"
FreqControl	IL_INLN_CNT_FREQ_CONTROL	Siehe " IL_INLN_CNT_FREQ_CONTROL " auf Seite 643	TIME_MEASURE	STATE_FALL_EDGE	TIME_MEASURE	Positive Flanke an "Enable"
Factor	INT	Torzeit = Faktor x 10 ms, mögliche Faktoren sind 1, 2, 10, 50, 100, 500, 1000	0	1000	0	Positive Flanke an "Enable"

Tab. 14-85: Zulässige Werte von IL_INLN_CNT_FREQ für IL_CntFrequency

IL_INLN_CNT_FREQ_CONTROL Diese Aufzählung enthält die Optionen zur Frequenzmessung.

ENUM	Beschreibung
TIME_MEASURE	Zeitgesteuerte Frequenzmessung, nach Ablauf der Torzeit wird der Zähler auf den Anfangswert zurückgesetzt.
STATE_HIGH	Bei dieser Messung findet die Zählung während der Zeit statt, in der am Gate ein High-Pegel anliegt. Mit dem Übergang auf einen Low-Pegel stoppt die Zählung. Der letzte Zählwert wird in die Eingangsdaten übernommen. Beim nächsten High- Pegel wird die Zählung wieder bei 0 begonnen.

ENUM	Beschreibung
STATE_LOW	Bei dieser Messung findet die Zählung während der Zeit statt, in der am Gate ein Low-Pegel anliegt. Mit dem Übergang auf einen High-Pegel stoppt die Zählung. Der letzte Zählwert wird in die Eingangsdaten übernommen. Beim nächsten Low- Pegel wird die Zählung wieder bei 0 begonnen.
STATE_RISING_EDGE	Bei dieser Messung beginnt die Zählung sofort mit Übertragung des Kommandos zur Frequenzmessung. Bei jeder steigenden Flanke des Gate-Signals wird der aktuelle Zählwert in die Eingangsdaten übernommen. Der Zähler wird auf 0 zurückgesetzt, und die Zählung wird fortgesetzt.
STATE_FALL_EDGE	Bei dieser Messung beginnt die Zählung sofort mit Übertragung des Kommandos zur Frequenzmessung. Bei jeder fallenden Flanke des Gate-Signals wird der aktuelle Zählwert in die Eingangsdaten übernommen. Der Zähler wird auf 0 zurückgesetzt, und die Zählung wird fortgesetzt.

IL_INLN_CNT_INCONFIG Die Aufzählung gestattet die Konfiguration eines Eingangsfilters oder einer Eingangslogik zur Verknüpfung von zwei Eingangssignalen.

ENUM	Beschreibung
NO_INPUT_CONFIG	Keine Eingangskonfiguration
FILTER_FOR_MECH_SWITCH	RC-Filter an Source und Gate zu schalten, das ermöglicht den Einsatz von mechanischen Schaltern. (z.B. Relais)
SOURCE_GATE_OPERATION	Source Gate Logic Operation, als Zählsignal kann eine Verknüpfung aus Source- und Gate-Signal gewählt werden.

Source-Gate-Logic-Operation, siehe "[IL_INLN_CNT_SG_OPERATION](#)" auf Seite 644

IL_INLN_CNT_SG_OPERATION Bei der Source-Gate-Verknüpfung wird aus den beiden Eingangssignalen Source (S) und Gate (G) ein Signal gebildet, das nach der Verarbeitung entsprechend der logischen Funktion als neues Source-Signal zur Verfügung steht. Es wird im Weiteren als Source' (S') bezeichnet. Das ursprüngliche Source-Signal steht jetzt nicht mehr zur Verfügung.

Das ursprüngliche Gate-Signal kann weiterhin verwendet werden.

Über diese Source-Gate-Verknüpfung können Sie die gebräuchlichen logischen Funktionen realisieren.

ENUM	Beschreibung
NONE	Keine Eingangsverknüpfung
SOURCE_NOR_GATE	" S' = S NOR G ". S' wird gebildet aus NOR zwischen Source und Gate
SOURCE_AND_NOT_GATE	" S' = S AND /G ". S' wird gebildet aus AND zwischen Source und invertiertem Gate
NOT_GATE	" S' = /G ". S' wird gebildet aus invertiertem Gate
NOT_SOURCE_AND_GATE	" S' = /S AND G ". S' wird gebildet aus AND zwischen invertierter Source und Gate
NOT_SOURCE	" S' = /S ". S' wird gebildet aus invertierter Source.
SOURCE_EXOR_GATE	" S' = S EXOR G ". S' wird gebildet aus EXOR zwischen Source und Gate

ENUM	Beschreibung
SOURCE_NAND_GATE	"S' = S NAND G". S' wird gebildet aus NAND zwischen Source and Gate
SOURCE_AND_GATE	"S' = S AND G". S' wird gebildet aus AND zwischen Source and Gate
SOURCE_EXNOR_GATE	"S' = S EXNOR G". S' wird gebildet aus EXNOR zwischen Source und Gate
SOURCE	"S' = S". S' wird gebildet aus Source
SOURCE_OR_NOT_GATE	"S' = S OR /G". S' wird gebildet aus OR zwischen Source und invertiertem Gate
GATE	"S' = G". S' wird gebildet aus Gate
NOT_SOURCE_OR_GATE	"S' = /S OR G". S' wird gebildet aus OR zwischen invertierter Source und Gate
SOURCE_OR_GATE	"S' = S OR G". S' wird gebildet aus OR zwischen Source und Gate

Beispiel Der Funktionsbaustein wird benötigt, um die Systemeinstellungen für eine Zählerklemme in der Betriebsart Frequenzmessung vorzunehmen.

Systemeinstellungen sind Basiseinstellungen für alle Betriebsarten der Klemme.

In den Systemeinstellungen sind enthalten: Eingangs-Konfiguration, Source-Gate-Verknüpfungen, Bus-Reset-Verhalten und Pulslängen-Parameter.

Wenn eine Eingangskonfiguration mit Logic-Source-Gate-Operation gewählt ist, dann kann eine logische Verknüpfung zur Bildung des Source-Signals **S'** aus dem (Eingangs-)Source-Signal **S** und dem Gate-Signal **G** erfolgen.

Zeitgesteuerte Frequenzmessung:

Bei zeitgesteuerter Frequenzmessung wird der Parameter 'FreqControl' als Faktor zur Bestimmung der Torzeit (Zeit, während der gemessen wird) als Vielfaches von 10 ms verwendet. Nach Ablauf der Torzeit wird der Zähler auf den Anfangswert zurückgesetzt.

Aufgabe des Projektanten ist es, den optimalen Faktor auszuwählen, da Auflösung und Torzeit zueinander umgekehrt proportional sind.

Folgende Zusammenhänge bestehen zwischen den einzelnen Größen:

- Torzeit = Faktor x 10 ms
- Auflösung = 1 / Torzeit
- Auflösung = 1 / (Faktor x 10 ms)
- Frequenz = Zählwert x 100 / Faktor

Deklaration zum Beispiel

```
PROGRAM prgIL_CntFrequency
VAR
  fbIL_CntFrequency: IL_CntFrequency;
  wInProcessData_0: WORD;
  wInProcessData_1: WORD;
  wOutProcessData_0: WORD;
  wOutProcessData_1: WORD;
  rFrequency: REAL;
END_VAR
```

Implementation zum Beispiel

```
// Configuration
fbIL_CntFrequency.FreqConfig.BusReset := TRUE ;
fbIL_CntFrequency.FreqConfig.InConfig := IL_INLN_CNT_INCONFIG.SOURCE_GATE_OPERATION ;
fbIL_CntFrequency.FreqConfig.SourceGateOperation := IL_INLN_CNT_SG_OPERATION.SOURCE_AND_GATE ;
fbIL_CntFrequency.FreqConfig.FreqControl := IL_INLN_CNT_FREQ_CONTROL.TIME_MEASURE ;
fbIL_CntFrequency.FreqConfig.Factor := INT#50 ;
fbIL_CntFrequency.InProcData_0 := wInProcessData_0 ;
fbIL_CntFrequency.InProcData_1 := wInProcessData_1 ;
wOutProcessData_0 := fbIL_CntFrequency.OutProcData_0 ;
```

```

wOutProcessData_1 := fbIL_CntFrequency.OutProcData_1 ;

// FB call
fbIL_CntFrequency(Enable:= TRUE);

// Data read
IF fbIL_CntFrequency.InOperation = TRUE THEN
    rFrequency := fbIL_CntFrequency.Frequency ;
END_IF

```

Zustandsgesteuerte Frequenzmessung:

Bei der zustandsgesteuerten Frequenzmessung gibt der Parameter 'Freq-Control' den Zustand des Gate-Einganges an, bei dem gezählt wird oder bei dem der Zählwert übernommen wird.

In diesem Beispiel ist 'FreqControl' ausgewählt zu **STATE_HIGH**, d.h. die Messung läuft so lange das Gate im Zustand **HIGH** ist.

Das Zählen endet, wenn das Gate nach **LOW** wechselt. Der letzte Zählstand wird als Eingangswert übergeben.

Mit dem nächsten **HIGH** startet der Zählvorgang erneut von 0 aus.

Deklaration zum Beispiel

```

PROGRAM prgIL_CntFrequency
VAR
    fbIL_CntFrequency: IL_CntFrequency;
    wInProcessData_0: WORD;
    wInProcessData_1: WORD;
    wOutProcessData_0: WORD;
    wOutProcessData_1: WORD;
    rFrequency: REAL;
END_VAR

```

Implementation zum Beispiel

```

// Configuration
fbIL_CntFrequency.FreqConfig.BusReset := TRUE ;
fbIL_CntFrequency.FreqConfig.InConfig := IL_INLN_CNT_INCONFIG.SOURCE_GATE_OPERATION ;
fbIL_CntFrequency.FreqConfig.SourceGateOperation := IL_INLN_CNT_SG_OPERATION.SOURCE ;
fbIL_CntFrequency.FreqConfig.FreqControl := IL_INLN_CNT_FREQ_CONTROL.STATE_HIGH ;
fbIL_CntFrequency.InProcData_0 := wInProcessData_0 ;
fbIL_CntFrequency.InProcData_1 := wInProcessData_1 ;
wOutProcessData_0 := fbIL_CntFrequency.OutProcData_0 ;
wOutProcessData_1 := fbIL_CntFrequency.OutProcData_1 ;

// FB call
fbIL_CntFrequency(Enable:= TRUE);

// Data read
IF fbIL_CntFrequency.InOperation = TRUE THEN
    rFrequency := fbIL_CntFrequency.Frequency ;
END_IF

```

Prozessdaten-Interface zum Beispiel

Die Prozesseingangs- und -ausgangsworte des Funktionsbausteines sind auf die Prozessdatenworte des einzelnen Feldbussystems zu mappen.

Die Variablen wInProcessData_x und wOutProcessData_x sind im Programm prgIL_CntFrequency deklariert.

Variable	Mapping	Channel	Address	Type
Application.prgIL_CntFrequency.wInProcessData_0		Input0	%IW24	WORD
Application.prgIL_CntFrequency.wInProcessData_1		Word0	%IW24	WORD
		Word1	%IW26	WORD
Application.prgIL_CntFrequency.wOutProcessData_0		Output0	%QW24	WORD
Application.prgIL_CntFrequency.wOutProcessData_1		Word0	%QW24	WORD
		Word1	%QW26	WORD

Abb. 14-31: Prozessdaten-Interface

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle INLINEIO_TABLE, 16#0190. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000001	'InConfig' ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000002	'SourceGateOperation' ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000003	'FreqControl' ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000004	'Factor' ist ungültig
DEVICE_ERROR (16#0008)	16#00003020	16#00000000	Zeitüberschreitung, Gerät antwortet nicht

Tab. 14-86: IL_CntFrequency Fehlercodes

14.4.3 IL_CntPulse

Kurzbeschreibung Der Funktionsbaustein **IL_CntPulse** parametriert die **R-IB IL CNT**-Klemme (Inline Zählerklemme) in der Betriebsart Pulsgenerator und generiert die Impulse.

Für jede Inline-EA-Klemme wird eine Funktionsbausteininstanz benötigt.

Kompatibilität Der Funktionsbaustein ist verwendbar für Inlinebus, Profibus, sercos III und Profinet IO-Feldbus.

Schnittstellenbeschreibung

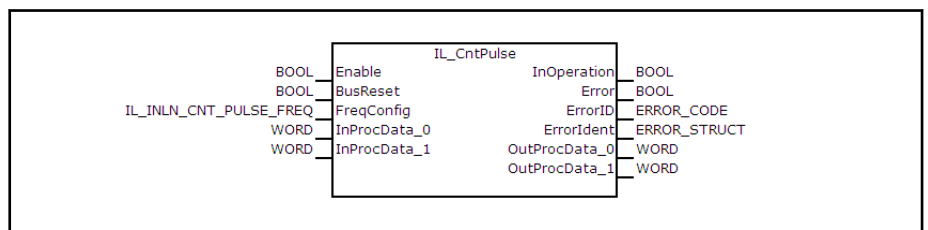


Abb. 14-32: Funktionsbaustein IL_CntPulse

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Kontinuierliche Bearbeitung des FBs, solange "Enable" TRUE ist
	BusReset	BOOL	FALSE - Bus-Reset setzt den Ausgang zurück, stoppt alle Zähloperationen und löscht die Betriebsarteneinstellungen. TRUE – keine Reaktion auf einen Bus-Reset.
	Frequency	IL_INLN_CNT_PULSE_FREQ	Frequenz-Faktor in 500Hz-Inkrementen, siehe "IL_INLN_CNT_PULSE_FREQ" auf Seite 649 $PulseFreq = 1000Hz + (Factor \times 500Hz)$
	InProcData_0	WORD	Prozesseingangswort 0
	InProcData_1	WORD	Prozesseingangswort 1
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, arbeitet der Baustein in seiner aktuellen Task
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb der FB-Abarbeitung eingetreten ist
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Detailinformationen zum Fehler, siehe "Fehlerbehandlung" auf Seite 650
	OutProcData_0	WORD	Prozessausgangswort 0
	OutProcData_1	WORD	Prozessausgangswort 1

Tab. 14-87: Schnittstelle IL_CntPulse

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	kontinuierlich
BusReset	BOOL	FALSE	TRUE	FALSE	Positive Flanke an "Enable"
Frequency	IL_INLN_CNT_PULSE_FREQ	FREQ_1000HZ	FREQ_10000HZ	FREQ_1000HZ	Positive Flanke an "Enable"

Tab. 14-88: Eingangsverhalten IL_CntPulse

Funktionsbeschreibung

Aktivieren Sie die FB-Instanz erst nach dem ersten SPS-Zyklus durch eine positive Flanke am **Enable**-Eingang.

Die Zählerklemme wird mit der positiven Flanke an **Enable** als Pulsgenerator konfiguriert.

Pulsgenerator kann Frequenzen von 1 kHz bis zu 10 kHz in 500Hz-Inkrementen erzeugen.

Mit der positiven Flanke am **Enable**-Eingang wird die Klemme konfiguriert. Der erfolgreiche Abschluss der Konfiguration wird durch **InOperation** = TRUE angezeigt und die Impulserzeugung beginnt.

Bei ungültigen Eingangsparametern generiert der FB einen Fehler und die entsprechenden Informationen gelangen an **Error**, **ErrorID** und **ErrorIdent**.

Eine positive Flanke am **Enable**-Eingang löscht die Fehler und rekonfiguriert die Klemme.



1. Wird die Klemme am Profibus oder Profinet-Feldbus verwendet, muss die Byte-Reihenfolge geändert werden:
Byte 0/1/2/3 = Plug 4/3/2/1.
2. Die Veränderung der Byte-Reihenfolge an der Busklemme ist für alle der Busklemme folgenden Klemmen wirksam!

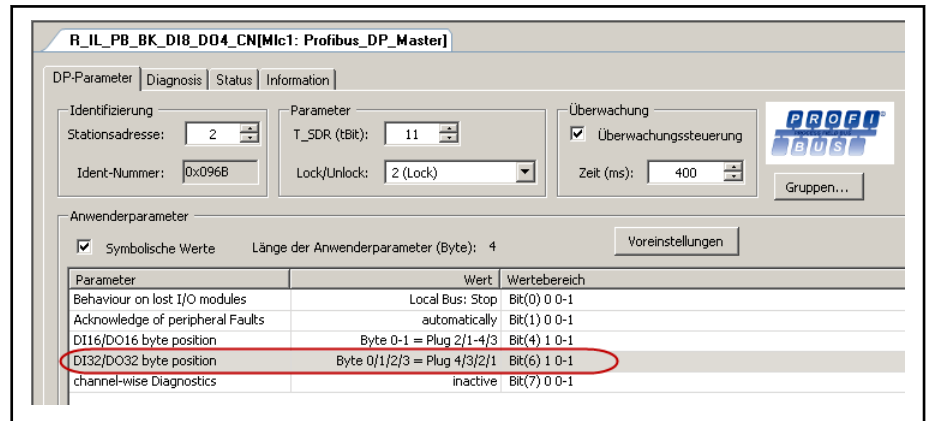


Abb. 14-33: Dialog Busklemme mit geänderter Bytefolge

IL_INLN_CNT_PULSE_FREQ

Die Aufzählung beschreibt die möglichen Frequenzparameter Faktor, Soll-Frequenz F_{soll} , Ist-Frequenz F_{ist} und Fehler bei der Impulserzeugung.

$$\text{Pulse\_Freq} = 1000\text{Hz} + (\text{Faktor} \times 500\text{Hz})$$

ENUM	Beschreibung
FREQ_1000HZ	Faktor = 0, F_{soll} = 1000 Hz, F_{ist} = 1000 Hz, Fehler = 0%
FREQ_1500HZ	Faktor = 1, F_{soll} = 1500 Hz, F_{ist} = 1495 Hz, Fehler = -0.2%
FREQ_2000HZ	Faktor = 2, F_{soll} = 2000 Hz, F_{ist} = 2000 Hz, Fehler = 0%
FREQ_2500HZ	Faktor = 3, F_{soll} = 2500 Hz, F_{ist} = 2500 Hz, Fehler = 0%
FREQ_3000HZ	Faktor = 4, F_{soll} = 3000 Hz, F_{ist} = 3012 Hz, Fehler = 0.4%
FREQ_3500HZ	Faktor = 5, F_{soll} = 3500 Hz, F_{ist} = 3521 Hz, Fehler = 0.6%
FREQ_4000HZ	Faktor = 6, F_{soll} = 4000 Hz, F_{ist} = 4000 Hz, Fehler = 0%
FREQ_4500HZ	Faktor = 7, F_{soll} = 4500 Hz, F_{ist} = 4505 Hz, Fehler = 0.11%
FREQ_5000HZ	Faktor = 8, F_{soll} = 5000 Hz, F_{ist} = 5000 Hz, Fehler = 0%
FREQ_5500HZ	Faktor = 9, F_{soll} = 5500 Hz, F_{ist} = 5495 Hz, Fehler = -0.09%
FREQ_6000HZ	Faktor = 10, F_{soll} = 6000 Hz, F_{ist} = 5998 Hz, Fehler = -0.2%
FREQ_6500HZ	Faktor = 11, F_{soll} = 6500 Hz, F_{ist} = 6494 Hz, Fehler = -0.09%
FREQ_7000HZ	Faktor = 12, F_{soll} = 7000 Hz, F_{ist} = 6993 Hz, Fehler = -0.01%
FREQ_7500HZ	Faktor = 13, F_{soll} = 7500 Hz, F_{ist} = 7519 Hz, Fehler = 0.25%
FREQ_8000HZ	Faktor = 14, F_{soll} = 8000 Hz, F_{ist} = 8000 Hz, Fehler = 0%
FREQ_8500HZ	Faktor = 15, F_{soll} = 8500 Hz, F_{ist} = 8475 Hz, Fehler = -0.29%
FREQ_9000HZ	Faktor = 16, F_{soll} = 9000 Hz, F_{ist} = 9009 Hz, Fehler = 0.1%
FREQ_9500HZ	Faktor = 17, F_{soll} = 9500 Hz, F_{ist} = 9525 Hz, Fehler = 0.25%
FREQ_10000HZ	Faktor = 18, F_{soll} = 10000 Hz, F_{ist} = 10000 Hz, Fehler = 0%

Beispiel Der Funktionsbaustein konfiguriert die Zählerklemme als Pulsgenerator. Der Pulsgenerator kann Frequenzen von 1 kHz bis 10 kHz in 500 Hz-Schritten produzieren.

In diesem Beispiel ist der Funktionsbaustein so konfiguriert, dass Pulse einer Frequenz von 1kHz erzeugt werden.

Deklaration zum Beispiel

```
PROGRAM PrgIL_CntPulse
VAR
    fbIL_CntPulse: IL_CntPulse;
    wInProcessData_0: WORD;
    wInProcessData_1: WORD;
    wOutProcessData_0: WORD;
    wOutProcessData_1: WORD;
END_VAR
```

Implementation zum Beispiel

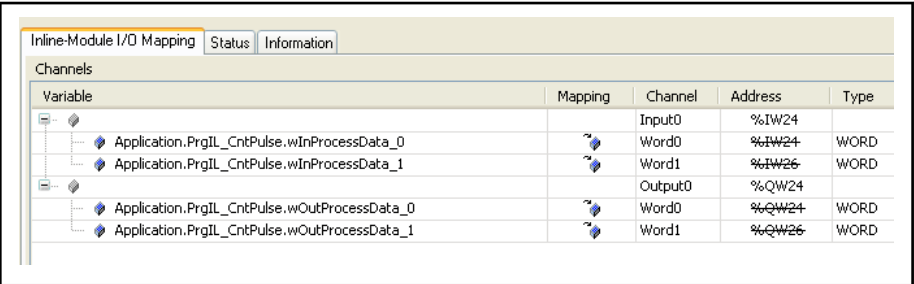
```
// Configuration
fbIL_CntPulse.BusReset := TRUE ;
fbIL_CntPulse.Frequency := IL_INLN_CNT_PULSE_FREQ.FREQ_1000HZ;
fbIL_CntPulse.InProcData_0 := wInProcessData_0 ;
fbIL_CntPulse.InProcData_1 := wInProcessData_1 ;
wOutProcessData_0 := fbIL_CntPulse.OutProcData_0 ;
wOutProcessData_1 := fbIL_CntPulse.OutProcData_1 ;

// FB call
fbIL_CntPulse(Enable:=TRUE) ;
```

Prozessdaten-Interface zum Beispiel

Die Prozesseingangs- und -ausgangsworte des Funktionsbausteines sind auf die Prozessdatenworte des einzelnen Feldbussystems zu mappen.

Die Variablen wInProcessData_x und wOutProcessData_x sind im Programm prgIL_CntPulse deklariert.



Variable	Mapping	Channel	Address	Type
Application.PrgIL_CntPulse.wInProcessData_0		Input0	%IW24	
Application.PrgIL_CntPulse.wInProcessData_1		Word0	%IW24	WORD
		Word1	%IW26	WORD
Application.PrgIL_CntPulse.wOutProcessData_0		Output0	%QW24	
Application.PrgIL_CntPulse.wOutProcessData_1		Word0	%QW24	WORD
		Word1	%QW26	WORD

Abb. 14-34: Prozessdaten-Interface

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle INLINEIO_TABLE, 16#0190. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000001	'Frequency'-Eingang ist ungültig
DEVICE_ERROR (16#0008)	16#00003020	16#00000000	Zeitüberschreitung, Gerät antwortet nicht

Tab. 14-89: IL_CntPulse Fehlercodes

14.4.4 IL_CntTime

Kurzbeschreibung Der Funktionsbaustein IL_CntTime parametriert die R-IB IL CNT-Klemme (Inline Zählerklemme) in der Betriebsart Zeitmessung und liest die gezählten Werte (Periodendauer oder Pulsdauer).

Für jede Inline-EA-Klemme wird eine Funktionsbausteininstanz benötigt.

Kompatibilität

Der Funktionsbaustein ist verwendbar für Inlinebus, Profibus, sercos III und Profinet IO-Feldbus.

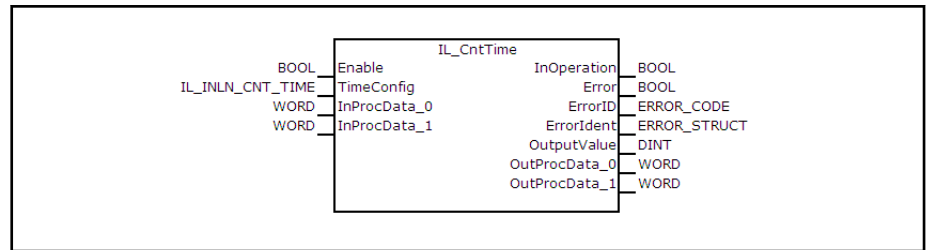
Schnittstellenbeschreibung

Abb. 14-35: Funktionsbaustein IL_CntTime

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Kontinuierliche Bearbeitung des FBs, solange "Enable" TRUE ist
	TimeConfig	IL_INLN_CNT_TIME	Struktur zur Zählerkonfiguration, siehe "IL_INLN_CNT_TIME" auf Seite 652
	InProcData_0	WORD	Prozesseingangswort 0
	InProcData_1	WORD	Prozesseingangswort 1
VAR OUTPUT	InOperation	BOOL	Wenn TRUE, arbeitet der Baustein in seiner aktuellen Task
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb der FB-Abarbeitung eingetreten ist
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Detaillinformationen zum Fehler, siehe "Fehlerbehandlung" auf Seite 656
	OutputValue	DINT	Gemessener Wert, abhängig von der Auflösung: Wertebereiche: bei 2 µs, 0 ... 131070 µs, bei 2 ms, 0 ... 131070 ms und bei 10 ms, 0 ... 655350 ms
	OutProcData_0	WORD	Prozessausgangswort 0
	OutProcData_1	WORD	Prozessausgangswort 1

Tab. 14-90: Schnittstelle IL_CntTime

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	kontinuierlich

Tab. 14-91: Eingangsverhalten IL_CntTime

Funktionsbeschreibung

Aktivieren Sie die FB-Instanz erst nach dem ersten SPS-Zyklus durch eine positive Flanke am **Enable**-Eingang.

Die Parametrierung der **R-IB IL CNT**-Klemme in der Betriebsart Zeitmessung setzt eine Belegung der Parameterstruktur **IL_INLN_CNT_TIME** mit dazu passenden Werten voraus.

Mit einer positiven Flanke am **Enable**-Eingang wird die Klemme konfiguriert. Der erfolgreiche Abschluss der Konfiguration wird durch **InOperation** = TRUE angezeigt und die gültigen Werte sind am **OutputValue**-Ausgang verfügbar.

Bei ungültigen Eingangsparametern generiert der FB einen Fehler und die entsprechenden Informationen gelangen an **Error**, **ErrorID** und **ErrorIdent**.

Eine positive Flanke am **Enable**-Eingang löscht die Fehler und rekonfiguriert die Klemme.

Entsprechend der Vorgabe am Eingang **MeasureLength** ist es möglich die Periodendauer oder Pulsdauer zu messen..



1. Wird die Klemme am Profibus oder Profinet-Feldbus verwendet, muss die Byte-Reihenfolge geändert werden:

Byte 0/1/2/3 = Plug 4/3/2/1.

2. Die Veränderung der Byte-Reihenfolge an der Busklemme ist für alle der Busklemme folgenden Klemmen wirksam!

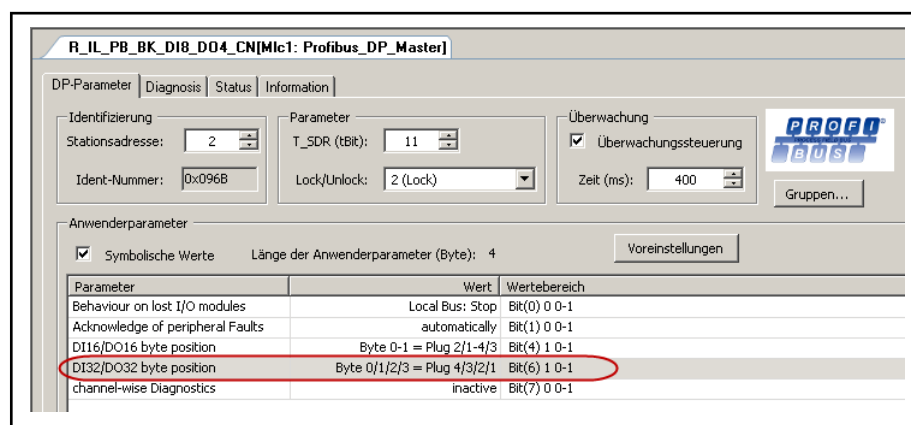


Abb. 14-36: Dialog Busklemme mit geänderter Bytefolge

IL_INLN_CNT_TIME

Die Struktur unterstützt die Konfiguration der Zählerklemme in der Betriebsart Zeitmessung.

Name	Typ	Beschreibung	Min-Wert	Max-Wert	Default-Wert	Übernahme
BusReset	BOOL	FALSE - Bus-Reset setzt den Ausgang zurück, stoppt alle Zähloperationen und löscht die Betriebsarteneinstellungen TRUE – keine Reaktion auf einen Bus-Reset	FALSE	TRUE	FALSE	Positive Flanke an "Enable"
InConfig	IL_INLN_CNT_INCONFIG	Siehe " IL_INLN_CNT_INCONFIG " auf Seite 653	NO_INPUT_CONFIG	SOURCE_GATE_OPERATION	NO_INPUT_CONFIG	Positive Flanke an "Enable"
SourceGate-Operation	IL_INLN_CNT_SG_OPERATION	Siehe " IL_INLN_CNT_SG_OPERATION " auf Seite 653	NONE	SOURCE_OR_GATE	NONE	Positive Flanke an "Enable"
Resolution	IL_INLN_CNT_TIME_RES	Siehe " IL_INLN_CNT_TIME_RES " auf Seite 654	RES_2US	RES_10MS	RES_2MS	Positive Flanke an "Enable"
Measure-Length	BOOL	FALSE – Messung der Periodendauer , TRUE – Messung der Pulsdauer	FALSE	TRUE	FALSE	Positive Flanke an "Enable"

Name	Typ	Beschreibung	Min-Wert	Max-Wert	Default-Wert	Übernahme
OutputRelation	IL_INLN_CNT_TIME_OUT_RELATION	Vergleichsbedingungen, siehe "IL_INLN_CNT_TIME_OUT_RELATION" auf Seite 654	NO_RELATION	CV_LT_IV_HYSTERESIS	NO_RELATION	Positive Flanke an "Enable"
InitialValue	DINT	Initialwert, abhängig von der Auflösung: 2µs- 0 ... 131000 2ms- 0 ... 131000 10ms- 0 ... 655350	0	Auflösung 2µs: 131000 2ms: 131000 10ms: 655000	0	Positive Flanke an "Enable"
FinalValue	DINT	Endwert, abhängig von der Auflösung: 2µs- 0 ... 131000 2ms- 0 ... 131000 10ms- 0 ... 655350	0	Auflösung 2µs: 131000 2ms: 131000 10ms: 655000	0	Positive Flanke an "Enable"

Tab. 14-92: Zulässige Werte von IL_INLN_CNT_COUNT für IL_CntTime

IL_INLN_CNT_INCONFIG Die Aufzählung gestattet die Konfiguration eines Eingangsfilters oder einer Eingangslogik zur Verknüpfung von zwei Eingangssignalen.

ENUM	Beschreibung
NO_INPUT_CONFIG	Keine Eingangskonfiguration
FILTER_FOR_MECH_SWITCH	RC-Filter an Source und Gate zu schalten, das ermöglicht den Einsatz von mechanischen Schaltern. (z.B. Relais)
SOURCE_GATE_OPERATION	Source Gate Logic Operation, als Zählsignal kann eine Verknüpfung aus Source- und Gate-Signal gewählt werden.

Source-Gate-Logic-Operation, siehe ["IL_INLN_CNT_SG_OPERATION"](#) auf Seite 653

IL_INLN_CNT_SG_OPERATION Bei der Source-Gate-Verknüpfung wird aus den beiden Eingangssignalen Source (S) und Gate (G) ein Signal gebildet, das nach der Verarbeitung entsprechend der logischen Funktion als neues Source-Signal zur Verfügung steht. Es wird im Weiteren als Source' (S') bezeichnet. Das ursprüngliche Source-Signal steht jetzt nicht mehr zur Verfügung.

Das ursprüngliche Gate-Signal kann weiterhin verwendet werden.

Über diese Source-Gate-Verknüpfung können Sie die gebräuchlichen logischen Funktionen realisieren.

ENUM	Beschreibung
NONE	Keine Eingangsverknüpfung
SOURCE_NOR_GATE	"S' = S NOR G". S' wird gebildet aus NOR zwischen Source und Gate
SOURCE_AND_NOT_GATE	"S' = S AND /G". S' wird gebildet aus AND zwischen Source und invertiertem Gate
NOT_GATE	"S' = /G". S' wird gebildet aus invertiertem Gate
NOT_SOURCE_AND_GATE	"S' = /S AND G". S' wird gebildet aus AND zwischen invertierter Source und Gate

ENUM	Beschreibung
NOT_SOURCE	" S' = /S ". S' wird gebildet aus invertierter Source.
SOURCE_EXOR_GATE	" S' = S EXOR G ". S' wird gebildet aus EXOR zwischen Source und Gate
SOURCE_NAND_GATE	" S' = S NAND G ". S' wird gebildet aus NAND zwischen Source and Gate
SOURCE_AND_GATE	" S' = S AND G ". S' wird gebildet aus AND zwischen Source and Gate
SOURCE_EXNOR_GATE	" S' = S EXNOR G ". S' wird gebildet aus EXNOR zwischen Source und Gate
SOURCE	" S' = S ". S' wird gebildet aus Source
SOURCE_OR_NOT_GATE	" S' = S OR /G ". S' wird gebildet aus OR zwischen Source und invertiertem Gate
GATE	" S' = G ". S' wird gebildet aus Gate
NOT_SOURCE_OR_GATE	" S' = /S OR G ". S' wird gebildet aus OR zwischen invertierter Source und Gate
SOURCE_OR_GATE	" S' = S OR G ". S' wird gebildet aus OR zwischen Source und Gate

IL_INLN_CNT_TIME_RES Der Aufzählungstyp beschreibt die Auflösung **aller** Werte pro LSB (niederwertiges Bit).

ENUM	Beschreibung
RES_2US	2 Mikrosekunden
RES_2MS	2 Millisekunden
RES_10MS	10 Millisekunden

IL_INLN_CNT_TIME_OUT_RELATION

Der Aufzählungstyp beschreibt die Ausgangs-Vergleichsbedingungen.



Ob ein Grenzwert in die Bedingung eingeschlossen ist oder nicht, hängt von der Bedingung ab. Intern wird jede Bedingung auf einen Vergleich Zählwert kleiner Startwert oder/und Zählwert größer Endwert zurückgeführt.

ENUM	Beschreibung
NO_RELATION	keine Vergleichsbedingung
CV_GE_IV	Zählwert größer oder gleich Startwert
CV_LT_IV	Zählwert kleiner Startwert
CV_WITHIN_IV_AND_FV	Zählwert innerhalb von Startwert und Endwert
CV_OUTSIDE_IV_AND_FV	Zählwert außerhalb von Startwert und Endwert
CV_GT_FV	Zählwert größer Endwert mit Hysterese
CV_LT_IV_HYSTERESIS	Zählwert kleiner Startwert mit Hysterese

Beispiel Der Funktionsbaustein wird benötigt, um die Systemeinstellungen für eine Zählerklemme in der Betriebsart Zeitmessung vorzunehmen.

Systemeinstellungen sind Basiseinstellungen für alle Betriebsarten der Klemme.

In den Systemeinstellungen sind enthalten: Eingangs-Konfiguration, Source-Gate-Verknüpfungen, Bus-Reset-Verhalten und Pulslängen-Parameter.

Wenn eine Eingangskonfiguration mit Logic-Source-Gate-Operation gewählt ist, dann kann eine logische Verknüpfung zur Bildung des Source-Signals **S'** aus dem (Eingangs-)Source-Signal **S** und dem Gate-Signal **G** erfolgen.

Die Messung beginnt mit einer positiven Flanke (0/1-Übergang).

Die Messung der Pulsdauer endet mit dem 1/0-Übergang, die der Periodendauer mit dem nächsten mit dem 0/1-Übergang.

Nur wenn die Messung komplett ist, wird der Zählerstand für die Prozeßdaten akzeptiert.

Tritt kein zählbarer Übergang auf, endet die Messung in einem TimeOut (Zeitüberschreitung), der Ausgang OutputValue ist dann gelöscht.

In diesem Beispiel ist der Funktionsbaustein zur Messung der Periodendauer konfiguriert.

Als Auflösung ist **RES_2MS** gewählt. Mit dieser Auflösung kann eine Periodendauer von 2ms bis 131s gemessen werden.

InitialValue ist 10 und **FinalValue** 100.

Als **OutputRelation** ist **CV_GT_FV** gewählt. Daraus folgt, wenn Zählwert größer als der Endwert mit Hysterese ist, wird der Ausgang **OUT TRUE**.

Deklaration zum Beispiel

```
PROGRAM PrgIL_CntTime
VAR
    fbIL_CntTime: IL_CntTime;
    wInProcessData_0: WORD;
    wInProcessData_1: WORD;
    wOutProcessData_0: WORD;
    wOutProcessData_1: WORD;
    diPulsePeriod: DINT;
END_VAR
```

Implementation zum Beispiel

```
// Configuration
fbIL_CntTime.TimeConfig.BusReset := TRUE;
fbIL_CntTime.TimeConfig.InConfig := IL_INLN_CNT_INCONFIG.SOURCE_GATE_OPERATION;
fbIL_CntTime.TimeConfig.SourceGateOperation := IL_INLN_CNT_SG_OPERATION.SOURCE_OR_GATE ;
fbIL_CntTime.TimeConfig.Resolution := IL_INLN_CNT_TIME_RES.RES_2MS ;
fbIL_CntTime.TimeConfig.InitialValue := DINT#10 ;
fbIL_CntTime.TimeConfig.FinalValue := DINT#100 ;
fbIL_CntTime.TimeConfig.MeasureLength := FALSE ;
fbIL_CntTime.TimeConfig.OutputRelation := IL_INLN_CNT_TIME_OUT_RELATION.CV_GT_FV ;
fbIL_CntTime.InProcData_0 := wInProcessData_0 ;
fbIL_CntTime.InProcData_1 := wInProcessData_1 ;
wOutProcessData_0 := fbIL_CntTime.OutProcData_0 ;
wOutProcessData_1 := fbIL_CntTime.OutProcData_1 ;

// FB call
fbIL_CntTime(Enable:=TRUE);

// Data read
IF fbIL_CntTime.InOperation = TRUE THEN
    diPulsePeriod := fbIL_CntTime.OutputValue ;
END_IF
```

Prozessdaten-Interface zum Beispiel

Die Prozesseingangs- und -ausgangsworte des Funktionsbausteines sind auf die Prozessdatenworte des einzelnen Feldbussystems zu mappen.

Die Variablen wInProcessData_x und wOutProcessData_x sind im Programm prgIL_CntTime deklariert.

Inline-Module I/O Mapping				
Channels				
Variable	Mapping	Channel	Address	Type
Application.PrgIL_CntTime.wInProcessData_0		Input0	%IW24	WORD
Application.PrgIL_CntTime.wInProcessData_1		Word0	%IW26	WORD
Application.PrgIL_CntTime.wOutProcessData_0		Output0	%QW24	WORD
Application.PrgIL_CntTime.wOutProcessData_1		Word0	%QW26	WORD

Abb. 14-37: Prozessdaten-Interface

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle INLINEIO_TABLE, 16#0190. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000001	'InConfig'-Eingang ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000002	'SourceGateOperation'-Eingang ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000003	'Resolution'-Eingang ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000004	'OutputRelation'-Eingang ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000005	'InitialValue'-Eingang ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000006	'FinalValue'-Eingang ist ungültig
DEVICE_ERROR (16#0008)	16#00003020	16#00000000	Zeitüberschreitung, Gerät antwortet nicht

Tab. 14-93: IL_CntTime Fehlercodes

14.4.5 IL_Pwm2Frequency

Kurzbeschreibung Der Funktionsbaustein **IL_Pwm2Frequency** parametriert die **R-IB IL PWM/2-PAC**-Klemme (Inline-Klemme zur Pulsweiten- und Frequenzmodulation) in der Betriebsart **Frequenzgenerator mit konstantem Duty Cycle** zur Erzeugung von Ausgangsimpulsfolgen an der Klemme.

Für jeden Kanal der Inline-EA-Klemme wird eine Funktionsbausteininstanz benötigt. Beide Kanäle arbeiten unabhängig.

 Außer in der Betriebsart Puls-Richtungs-Signal können sie für jeden Kanal eine eigene Betriebsart wählen. Wenn die Klemme in der Betriebsart Puls-Richtungs-Signal arbeitet, werden beide Ausgänge für diese Betriebsart benötigt.

Kompatibilität Der Funktionsbaustein ist verwendbar für Inlinebus, Profibus, sercos III und Profinet IO-Feldbus.

Schnittstellenbeschreibung

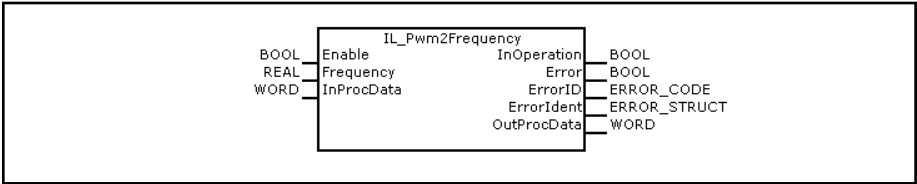


Abb. 14-38: Funktionsbaustein IL_Pwm2Frequency

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Kontinuierliche Bearbeitung des FBs, solange "Enable" TRUE ist
	Frequency	REAL	Frequenz, 12.21Hz ... 50,000 Hz, Schrittweite 12.21Hz.
	InProcData	WORD	Prozesseingangswort
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, arbeitet der Baustein in seiner aktuellen Task
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb der FB-Abarbeitung eingetreten ist
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Detaillinformationen zum Fehler, siehe "Fehlerbehandlung" auf Seite 659
	OutProcData	WORD	Prozessausgangswort

Tab. 14-94: Schnittstelle IL_Pwm2Frequency

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	kontinuierlich
Frequency	REAL	12.21Hz	50000 HZ	12.21Hz	kontinuierlich mit "Enable"

Tab. 14-95: Eingangsverhalten IL_Pwm2Frequency

Funktionsbeschreibung

Aktivieren Sie die FB-Instanz erst nach dem ersten SPS-Zyklus durch eine positive Flanke am **Enable**-Eingang.

Der Funktionsbaustein konfiguriert den Kanal in der Betriebsart Frequenzgenerator mit vorgegebener Frequenz und konstant 50% Puls-Pausen-Verhältnis.

Die Betriebsart unterstützt Frequenzen von 12.21Hz ... 50 kHz, mit einer Schrittweite von 12.21Hz.

Der erfolgreiche Abschluss der Konfiguration wird durch **InOperation** = TRUE angezeigt und die Impulserzeugung beginnt. Es werden kontinuierliche Impulse erzeugt.

Bei ungültigen Eingangsparametern generiert der FB einen Fehler, und die entsprechenden Informationen gelangen an **Error**, **ErrorID** und **ErrorIdent**.

Eine positive Flanke am **Enable**-Eingang löscht die Fehler und rekonfiguriert die Klemme.



1. Wird die Klemme am Profibus oder Profinet-Feldbus verwendet, muss die Byte-Reihenfolge geändert werden:
Byte 0/1/2/3 = Plug 4/3/2/1.
2. Die Veränderung der Byte-Reihenfolge an der Busklemme ist für alle der Busklemme folgenden Klemmen wirksam!

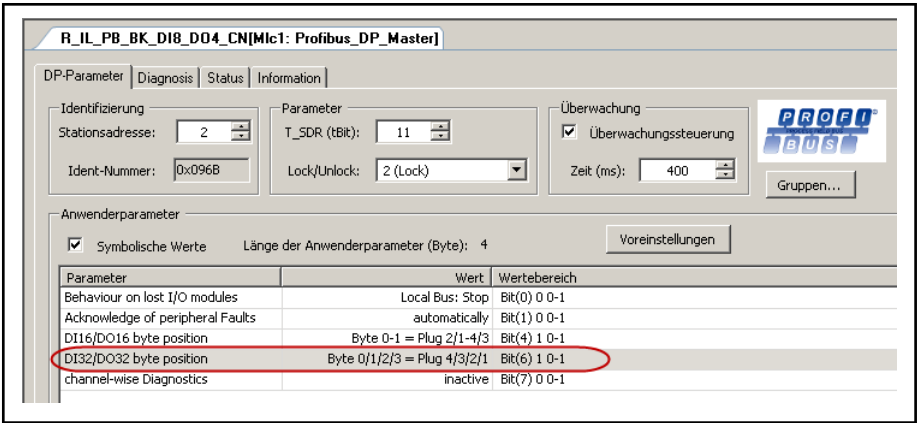


Abb. 14-39: Dialog Busklemme mit geänderter Bytefolge

Beispiel Der Funktionsbaustein wird verwendet, um eine PWM-Klemme in der Betriebsart **Frequenzgenerator mit konstantem Duty Cycle** zu konfigurieren. Vorzugeben ist die Frequenz, ferner ist das Mappen der Prozessdaten ist vorzunehmen.

Im Beispiel ist der Baustein **Pwm2Frequency** einkanalig für die vom Anwender vorgegebene Frequenz (und ein Puls-Pausen-Verhältnis = 50%) konfiguriert.

Deklaration zum Beispiel

```
PROGRAM PrgIL_Pwm2Frequency
VAR
    fbIL_Pwm2Frequency: IL_Pwm2Frequency;
    wInProcessData: WORD;
    wOutProcessData: WORD;
END_VAR
```

Implementation zum Beispiel

```
// Configuration
fbIL_Pwm2Frequency.Frequency := REAL#50;
fbIL_Pwm2Frequency.InProcData := wInProcessData;
wOutProcessData := fbIL_Pwm2Frequency.OutProcData;

// FB call
fbIL_Pwm2Frequency(Enable := TRUE);

// Frequency change during InOperation
IF fbIL_Pwm2Frequency.InOperation = TRUE THEN
    fbIL_Pwm2Frequency.Frequency := REAL#100;
END_IF
```

Prozessdaten-Interface zum Beispiel

Die Prozesseingangs- und -ausgangsworte des Funktionsbausteines sind auf die Prozessdatenworte des einzelnen Feldbussystems zu mappen.

Die Variablen wInProcessData und wOutProcessData sind im Programm prgIL_Pwm2Frequency deklariert.

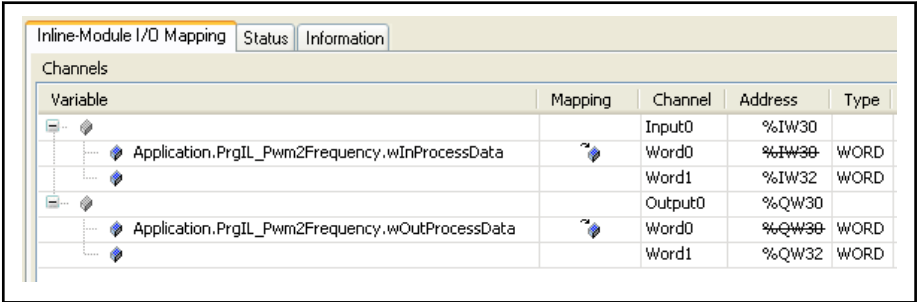


Abb. 14-40: Prozessdaten-Interface

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle INLINEIO_TABLE, 16#0190. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000001	'Frequency'-Eingang ist ungültig
DEVICE_ERROR (16#0008)	16#00003020	16#00000000	Zeitüberschreitung, Gerät antwortet nicht

Tab. 14-96: IL_Pwm2Frequency Fehlercodes

14.4.6 IL_Pwm2Pwm

Kurzbeschreibung Der Funktionsbaustein **IL_Pwm2Pwm** parametriert die **R-IB IL PWM/2-PAC**-Klemme (Inline-Klemme zur Pulsweiten- und Frequenzmodulation) in der Betriebsart **Pulsweitenmodulation (PWM) mit variablem Duty Cycle** zur Erzeugung von Ausgangsimpulsfolgen an der Klemme

Für jeden Kanal der Inline-EA-Klemme wird eine Funktionsbausteininstanz benötigt. Beide Kanäle arbeiten unabhängig.



Außer in der Betriebsart Puls-Richtungs-Signal können sie für jeden Kanal eine eigene Betriebsart wählen. Wenn die Klemme in der Betriebsart Puls-Richtungs-Signal arbeitet, werden beide Ausgänge für diese Betriebsart benötigt.

Kompatibilität Der Funktionsbaustein ist verwendbar für Inlinebus, Profibus, sercos III und Profinet IO-Feldbus.

Schnittstellenbeschreibung

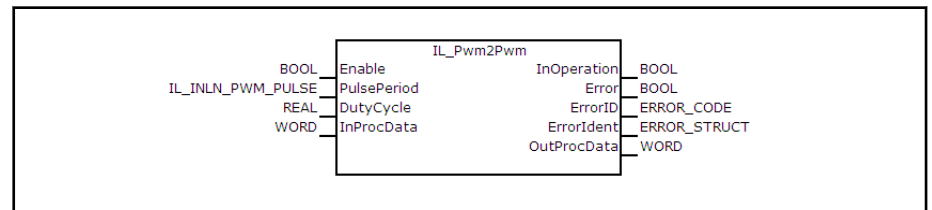


Abb. 14-41: Funktionsbaustein IL_Pwm2Pwm

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Kontinuierliche Bearbeitung des FBs, solange "Enable" TRUE ist
	PulsePeriod	IL_INLN_PWM_PULSE	Pulsedauer, siehe "IL_INLN_PWM_PULSE" auf Seite 661
	DutyCycle	REAL	Duty Cycle in Prozent des PWM-Ausgangs. Der Wert kann zwischen 0% und 99.45%, bei einer Schrittweite von 0.39% vorgegeben werden. z.B. 50% ≈ 49.92% . 0% stoppt die PWM-Funktion.
	InProcData	WORD	Prozesseingangswort

	Name	Typ	Beschreibung
VAR OUTPUT	InOperation	BOOL	Wenn TRUE, arbeitet der Baustein in seiner aktuellen Task
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb der FB-Abarbeitung eingetreten ist
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Detailinformationen zum Fehler, siehe "Fehlerbehandlung" auf Seite 662
	OutProcData	WORD	Prozessausgangswort

Tab. 14-97: Schnittstelle IL_Pwm2Pwm

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	kontinuierlich
PulsePeriod	ENUM	PULSE_100US	PULSE_10S	PULSE_100US	Positive Flanke an "Enable"
DutyCycle	REAL	0	99.45	50	kontinuierlich mit "Enable"

Tab. 14-98: Eingangsverhalten IL_Pwm2Pwm

Funktionsbeschreibung

Aktivieren Sie die FB-Instanz erst nach dem ersten SPS-Zyklus durch eine positive Flanke am **Enable**-Eingang.

Der Funktionsbaustein konfiguriert den Kanal in Pulsweitenmodulation mit der vorgegebenen **PulsePeriod** und dem **DutyCycle** (Puls-Pausen-Verhältnis).

Die Periodendauer können Sie zwischen 100 µs und 10 s vorgeben. Damit wird ein Frequenzbereich von 10 kHz bis 0,1 Hz abgedeckt.

Den Duty Cycle können Sie zwischen 0,39 % und 99,45 % wählen.

Der erfolgreiche Abschluss der Konfiguration wird durch **InOperation** = TRUE angezeigt und die Impulserzeugung beginnt. Es werden kontinuierliche Impulse erzeugt.

Bei ungültigen Eingangsparametern generiert der FB einen Fehler und die entsprechenden Informationen gelangen an **Error**, **ErrorID** und **ErrorIdent**.

Eine positive Flanke am **Enable**-Eingang löscht die Fehler und rekonfiguriert die Klemme.



1. Wird die Klemme am Profibus oder Profinet-Feldbus verwendet, muss die Byte-Reihenfolge geändert werden:
Byte 0/1/2/3 = Plug 4/3/2/1.
2. Die Veränderung der Byte-Reihenfolge an der Busklemme ist für alle der Busklemme folgenden Klemmen wirksam!

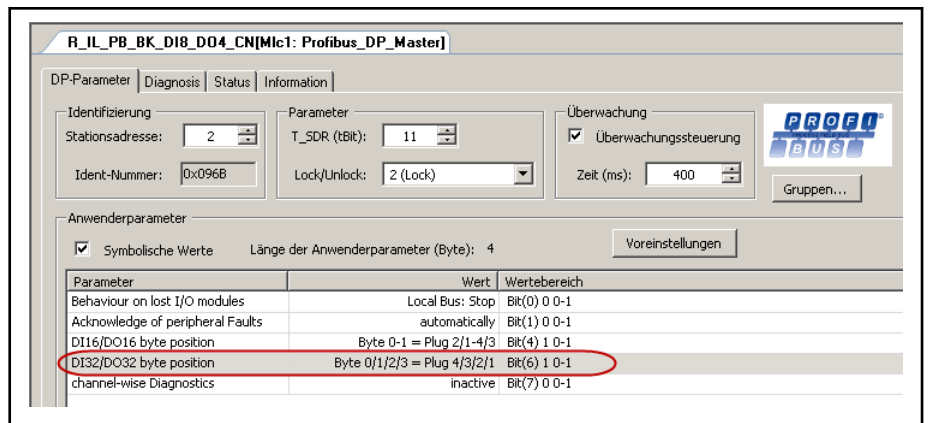


Abb. 14-42: Dialog Busklemme mit geänderter Bytefolge

IL_INLN_PWM_PULSE

Die Aufzählung definiert die Pulsdauer des PWM. 100 Mikrosekunden bis 10 Sekunden können vorgegeben werden. Damit ergibt sich ein Frequenzbereich von 10 kHz bis 0.1Hz.

ENUM	Beschreibung
PULSE_100US	100 Mikrosekunden
PULSE_200US	200 Mikrosekunden
PULSE_400US	400 Mikrosekunden
PULSE_600US	600 Mikrosekunden
PULSE_800US	800 Mikrosekunden
PULSE_1MS	1 Millisekunde
PULSE_2MS	2 Millisekunden
PULSE_4MS	4 Millisekunden
PULSE_6MS	6 Millisekunden
PULSE_8MS	8 Millisekunden
PULSE_10MS	10 Millisekunden
PULSE_20MS	20 Millisekunden
PULSE_40MS	40 Millisekunden
PULSE_60MS	60 Millisekunden
PULSE_80MS	80 Millisekunden
PULSE_100MS	100 Millisekunden
PULSE_200MS	200 Millisekunden
PULSE_400MS	400 Millisekunden
PULSE_600MS	600 Millisekunden
PULSE_800MS	800 Millisekunden
PULSE_1S	1 Sekunde
PULSE_2S	2 Sekunden
PULSE_4S	4 Sekunden
PULSE_6S	6 Sekunden

ENUM	Beschreibung
PULSE_8S	8 Sekunden
PULSE_10S	10 Sekunden

Beispiel Der Funktionsbaustein wird verwendet, um eine PWM-Klemme in der Betriebsart Pulsweitenmodulation zu konfigurieren. Die Einstellungen enthalten **PulsePeriod** und **DutyCycle**, das Mappen der Prozessdaten ist vorzunehmen. Im Beispiel ist der Baustein **Pwm2Pwm** einkanalgig für eine Frequenz = 10ms und ein Puls-Pausen-Verhältnis = 50% konfiguriert.

Deklaration zum Beispiel

```
PROGRAM PrgIL_Pwm2Pwm
VAR
    fbIL_Pwm2Pwm: IL_Pwm2Pwm;
    wInProcessData: WORD;
    wOutProcessData: WORD;
END_VAR
```

Implementation zum Beispiel

```
// Configuration
fbIL_Pwm2Pwm.PulsePeriod := IL_INLN_PWM_PULSE.PULSE_10MS;
fbIL_Pwm2Pwm.DutyCycle   := REAL#50.0;
fbIL_Pwm2Pwm.InProcData  := wInProcessData;
wOutProcessData          := fbIL_Pwm2Pwm.OutProcData;

// FB call
fbIL_Pwm2Pwm(Enable := TRUE);
```

Prozessdaten-Interface zum Beispiel

Die Prozesseingangs- und -ausgangsworte des Funktionsbausteines sind auf die Prozessdatenworte des einzelnen Feldbussystems zu mappen. Die Variablen wInProcessData und wOutProcessData sind im Programm prgIL_Pwm2Pwm deklariert.

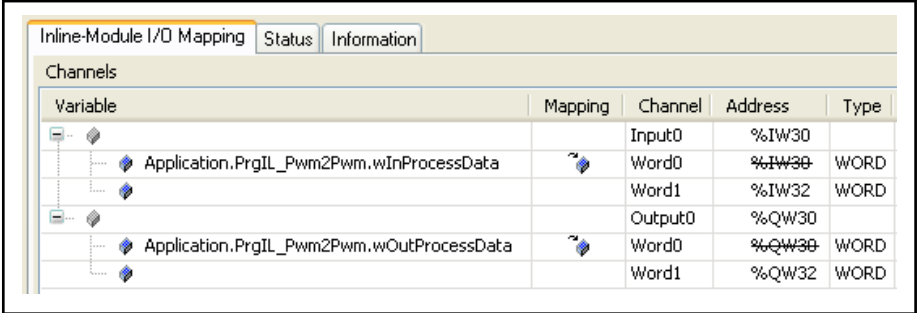


Abb. 14-43: Prozessdaten-Interface

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle INLINEIO_TABLE, 16#0190. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000001	'PulsePeriod' ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000002	'DutyCycle' ist ungültig
DEVICE_ERROR (16#0008)	16#00003020	16#00000000	Zeitüberschreitung, Gerät antwortet nicht

Tab. 14-99: IL_Pwm2Pwm Fehlercodes

14.4.7 IL_Pwm2PulseDirection

Kurzbeschreibung Der Funktionsbaustein **IL_Pwm2PulseDirection** parametrisiert die **R-IB IL PWM/2-PAC**-Klemme (Inline-Klemme zur Pulsweiten- und Frequenzmodulation) in der Betriebsart **Puls-Richtungs-Signal**.

In dieser Betriebsart werden beide Ausgänge der Klemme zusammen verwendet, so dass nur ein Kanal zur Verfügung steht.

Kompatibilität Der Funktionsbaustein ist verwendbar für Inlinebus, Profibus, sercos III und Profinet IO-Feldbus.

Schnittstellenbeschreibung

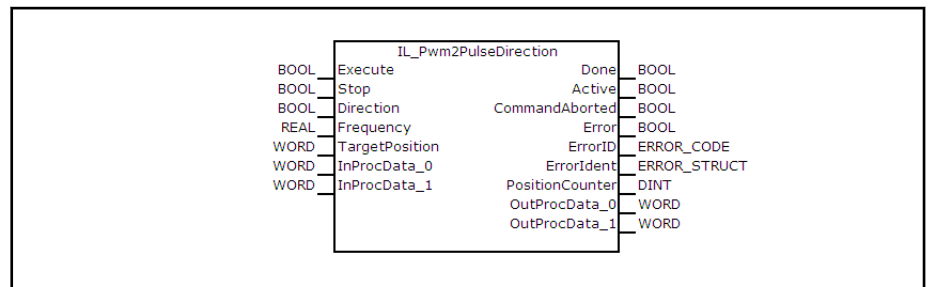


Abb. 14-44: Funktionsbaustein IL_Pwm2PulseDirection

	Name	Typ	Beschreibung
VAR_INPUT	Execute	BOOL	Bei positiver Flanke startet die Bearbeitung des FBs
	Stop	BOOL	Stoppt die Impulsausgabe und setzt den Positionszählerstand auf 0
	Direction	BOOL	Richtung und Ausgang DO2 FALSE = abwärts oder links TRUE = aufwärts oder rechts
	Frequency	REAL	Frequenz, 12.21 Hz ... 25kHz, Schrittweite 12,21Hz.
	TargetPosition	WORD	Zielposition Die Zielposition umfasst einen Wertebereich von 16#0 bis 16#FFFE (0 bis 65534). Der Wert 16#FFFF (65535dez) führt zu einer endlosen Impulsausgabe. Bei einem Wert zwischen 16#0 und 16#FFFE endet die Impulsausgabe dann, wenn die niederwertigen 16-Bit des Positionszählers gleich der Zielposition sind.
	InProcData0	WORD	Prozesseingangswort 0
	InProcData1	WORD	Prozesseingangswort 1

	Name	Typ	Beschreibung
VAR OUTPUT	Done	BOOL	Ende der Pulsfolge erreicht
	Active	BOOL	FB arbeitet, Pulsfolge wird erzeugt
	CommandAborted	BOOL	Abbruch der FB-Bearbeitung
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb der FB-Abarbeitung eingetreten ist
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Detailinformationen zum Fehler, siehe "Fehlerbehandlung" auf Seite 666
	PositionCounter	DINT	Positions-zähler Der Positions-zähler zählt die bisher ausgegebenen Impulse in Abhängigkeit vom Signal 'Direction' aufwärts oder abwärts.
	OutProcData_0	WORD	Prozessausgangswort 0
	OutProcData_1	WORD	Prozessausgangswort 1

Tab. 14-100: Schnittstelle IL_Pwm2PulseDirection

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	kontinuierlich
Stop	BOOL	FALSE	TRUE	FALSE	kontinuierlich
Direction	BOOL	FALSE	TRUE	FALSE	Positive Flanke an 'Execute'
Frequency	REAL	12.21	25000	12.21	kontinuierlich
TargetPosition	WORD	0	65535	0	Positive Flanke an 'Execute'

Tab. 14-101: Eingangsverhalten IL_Pwm2PulseDirection

Funktionsbeschreibung

Der Funktionsbaustein konfiguriert die Klemme in der Betriebsart Puls-Richtungs-Signal unter Vorgabe der Frequenz **Frequency** und der Zielposition **TargetPosition**.

In dieser Betriebsart gibt die Klemme Impulse mit einer Frequenz von 12.21 Hz ... 25KHz, Schrittweite 12.21Hz, aus.

Bei **Frequency**= 0 wird die laufende Impulsausgabe abgebrochen.

Das Puls-Pause-Verhältnis liegt konstant bei 50%.

Der Positions-zähler zählt die absolvierten Schritte in Abhängigkeit von **Direction** auf- oder abwärts, so dass darüber die Position des Antriebs abgelesen werden kann.

Done zeigt an, ob das Ende der Pulsfolge erreicht ist.

Die Zielposition **TargetPosition** umfasst einen Wertebereich von 16#0 bis 16#FFFE (0 bis 65534).

Der Wert 16#FFFF (65535dez) führt zu einer endlosen Impulsausgabe.

Bei einem Wert zwischen 16#0 und 16#FFFE endet die Impulsausgabe dann, wenn die niederwertigen 16-Bit des Positions-zählers gleich der Zielposition sind.

Eine positive Flanke an **Stop** stoppt die Impulsausgabe und setzt den Positionszählerstand auf 0.

Bei ungültigen Eingangsparametern generiert der FB einen Fehler und die entsprechenden Informationen gelangen an **Error**, **ErrorID** und **ErrorIdent**.

Eine positive Flanke am **Execute**-Eingang löscht die Fehler und rekonfiguriert die Klemme.



1. Wird die Klemme am Profibus oder Profinet-Feldbus verwendet, muss die Byte-Reihenfolge geändert werden:
Byte 0/1/2/3 = Plug 4/3/2/1.
2. Die Veränderung der Byte-Reihenfolge an der Busklemme ist für alle der Busklemme folgenden Klemmen wirksam!

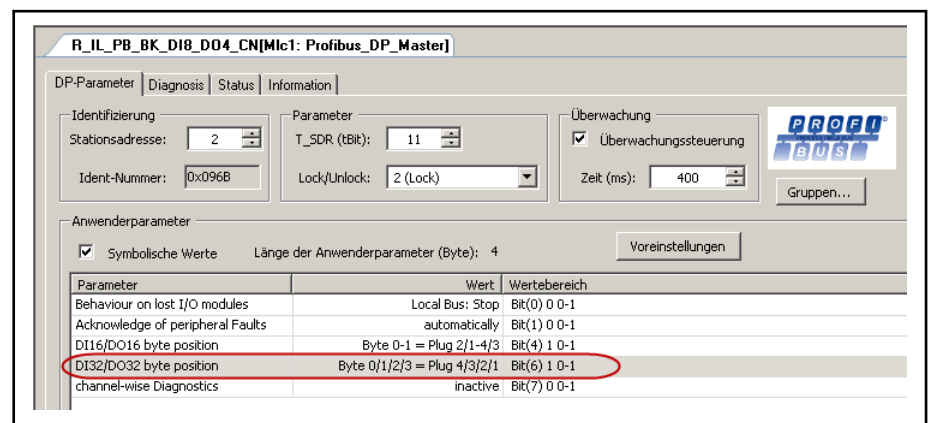


Abb. 14-45: Dialog Busklemme mit geänderter Bytefolge

Beispiel

Der Funktionsbaustein **IL_Pwm2PulseDirection** parametrisiert die **R-IB IL PWM/2-PAC**-Klemme in der Betriebsart **Puls-Richtungs-Signal**.

Die Einstellungen enthalten **Frequency**, **TargetPosition** und **Direction**, das Mappen der Prozessdaten ist vorzunehmen.

Im Beispiel ist der Baustein **Pwm2PulseDirection** konfiguriert für einen Antrieb mit einer Frequenz von 100Hz und einer Zielposition 2000 Impulse.

In dieser Betriebsart werden beide Ausgänge der Klemme zusammen verwendet, so dass nur ein Kanal zur Verfügung steht.

Aktiviert wird der Baustein durch eine positive Flanke an **Execute**.

Der FB konfiguriert dann die Klemme und setzt **Active** auf TRUE.

Die Klemme beginnt mit der Impulsausgabe. Nach Abschluss werden **Done** auf TRUE und **Active** auf FALSE gesetzt.

Deklaration zum Beispiel

```
PROGRAM PrgIL_Pwm2PulseDirection
VAR
    fbIL_Pwm2PulseDirection: IL_Pwm2PulseDirection;
    wInProcessData_0: WORD;
    wInProcessData_1: WORD;
    wOutProcData_0: WORD;
    wOutProcData_1: WORD;
    diPosCount: DINT;
    bDone: BOOL;
END_VAR
```

Implementation zum Beispiel

```
// Configuration
fbIL_Pwm2PulseDirection.Frequency := REAL#100;
```

```
fbIL_Pwm2PulseDirection.Direction      := TRUE;
fbIL_Pwm2PulseDirection.TargetPosition := WORD#2000;
fbIL_Pwm2PulseDirection.Stop           := FALSE;
fbIL_Pwm2PulseDirection.InprocData_0   := wInProcessData_0;
fbIL_Pwm2PulseDirection.InprocData_1   := wInProcessData_1;
wOutProcData_0                         := fbIL_Pwm2PulseDirection.OutProcData_0;
wOutProcData_1                         := fbIL_Pwm2PulseDirection.OutProcData_1;

// FB call
fbIL_Pwm2PulseDirection(Execute := TRUE);

// To find the position value
IF fbIL_Pwm2PulseDirection.Active = TRUE THEN
    diPosCount := fbIL_Pwm2PulseDirection.PositionCounter;
END_IF
```

Prozessdaten-Interface zum Beispiel

Die Prozesseingangs- und -ausgangsworte des Funktionsbausteines sind auf die Prozessdatenworte des einzelnen Feldbussystems zu mappen.

Die Variablen wInProcessData_x und wOutProcessData_x sind im Programm prgIL_Pwm2PulseDirection deklariert.

Inline-Module I/O Mapping				
Channels				
Variable	Mapping	Channel	Address	Type
Application.PrgIL_Pwm2PulseDirection.wInProcessData_0		Input0	%IW30	
Application.PrgIL_Pwm2PulseDirection.wInProcessData_1		Word0	%IW32	WORD
Application.PrgIL_Pwm2PulseDirection.wInProcessData_1		Word1	%IW32	WORD
Application.PrgIL_Pwm2PulseDirection.wOutProcData_0		Output0	%QW30	
Application.PrgIL_Pwm2PulseDirection.wOutProcData_0		Word0	%QW32	WORD
Application.PrgIL_Pwm2PulseDirection.wOutProcData_1		Word1	%QW32	WORD

Abb. 14-46: Prozessdaten-Interface

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle INLINEIO_TABLE, 16#0190. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000001	'Frequency' ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000002	'TargetPosition' ist ungültig
DEVICE_ERROR (16#0008)	16#00003020	16#00000000	Zeitüberschreitung, Gerät antwortet nicht

Tab. 14-102: IL_Pwm2PulseDirection Fehlercodes

14.4.8 IL_Pwm2SinglePulse

Kurzbeschreibung

Der Funktionsbaustein **IL_Pwm2SinglePulse** parametriert die **R-IB IL PWM/2-PAC**-Klemme (Inline-Klemme zur Pulsweiten- und Frequenzmodulation) in der Betriebsart **Single Shot** (Ezelimpulsgenerator) zur Erzeugung von Einzelimpulsen mit variabler Dauer.

Für jeden Kanal der Inline-EA-Klemme wird eine Funktionsbausteininstanz benötigt. Beide Kanäle arbeiten unabhängig.

 Außer in der Betriebsart Puls-Richtungs-Signal können sie für jeden Kanal eine eigene Betriebsart wählen. Wenn die Klemme in der Betriebsart Puls-Richtungs-Signal arbeitet, werden beide Ausgänge für diese Betriebsart benötigt.

Kompatibilität

Der Funktionsbaustein ist verwendbar für Inlinebus, Profibus, sercos III und Profinet IO-Feldbus.

Schnittstellenbeschreibung

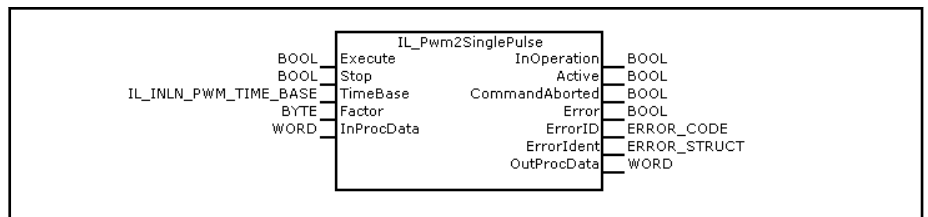


Abb. 14-47: Funktionsbaustein IL_Pwm2SinglePulse

	Name	Typ	Beschreibung
VAR_INPUT	Execute	BOOL	Bei positiver Flanke startet die Bearbeitung des FBs
	Stop	BOOL	Stoppt die Ausgabe des Impulses
	TimeBase	IL_INLN_PWM_TIME_BASE	Zeitbasis, siehe "IL_INLN_PWM_TIME_BASE" auf Seite 668
	Factor	BYTE	Faktor, Impulsdauer = Zeitbasis x Faktor Der Wert 0 stoppt die Single-Shot-Funktion
	InProcData	WORD	Prozesseingangswort
VAR OUTPUT	Done	BOOL	Berechnung abgeschlossen
	Active	BOOL	Berechnung läuft (Impulsausgabe)
	CommandAborted	BOOL	Berechnung durch einen anderen FB oder Stop abgebrochen
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb der FB-Abarbeitung eingetreten ist
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Detailinformationen zum Fehler, siehe "Fehlerbehandlung" auf Seite 669
	OutProcData	WORD	Prozessausgangswort

Tab. 14-103: Schnittstelle IL_Pwm2SinglePulse

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	kontinuierlich
Stop	BOOL	FALSE	TRUE	FALSE	kontinuierlich
TimeBase	ENUM	TIME_10US	TIME_100MS	TIME_10US	Positive Flanke an 'Execute'
Factor	BYTE	0	255	0	Positive Flanke an 'Execute'

Tab. 14-104: Eingangsverhalten IL_Pwm2SinglePulse

Funktionsbeschreibung

Aktivieren Sie die FB-Instanz erst nach dem ersten SPS-Zyklus durch eine positive Flanke am **Execute**-Eingang.

Der Funktionsbaustein konfiguriert den Kanal als Einzelimpulsgenerator unter Vorgabe der Zeitbasis und eines Faktors.

In dieser Betriebsart gibt der Kanal Einzelimpulse einer Länge von 10µs ... 25.5s aus :

$$\text{Pulsdauer} = \text{TimeBase} * \text{Factor}$$

Nach der Übernahme dieser Daten werden **Active** zu TRUE und der Impuls am Klemmenausgang generiert.

Eine positive Flanke am **Stop**-Eingang beendet die Impulsausgabe und aktiviert den **CommandAborted**-Ausgang.

Bei ungültigen Eingangsparametern generiert der FB einen Fehler und die entsprechenden Informationen gelangen an **Error**, **ErrorID** und **ErrorIdent**.

Eine positive Flanke am **Execute**-Eingang löscht die Fehler und rekonfiguriert die Klemme.



1. Wird die Klemme am Profibus oder Profinet-Feldbus verwendet, muss die Byte-Reihenfolge geändert werden:
Byte 0/1/2/3 = Plug 4/3/2/1.
2. Die Veränderung der Byte-Reihenfolge an der Busklemme ist für alle der Busklemme folgenden Klemmen wirksam!

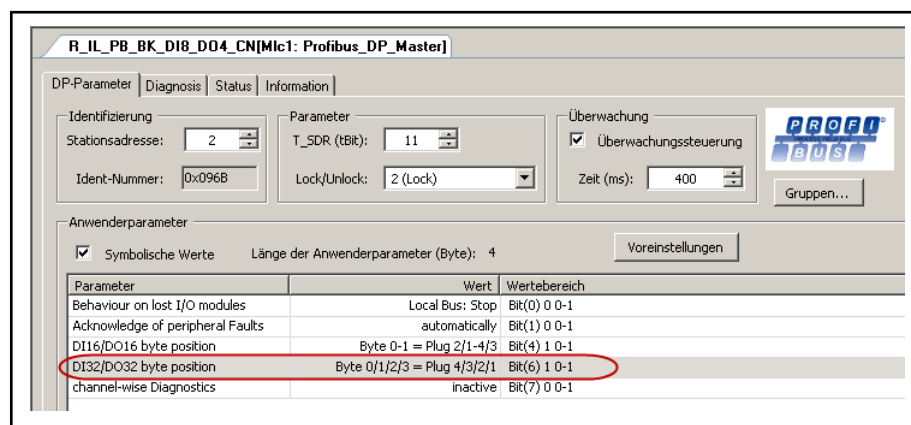


Abb. 14-48: Dialog Busklemme mit geänderter Bytefolge

IL_INLN_PWM_TIME_BASE

Die Aufzählung gibt die Zeitbasis zur Einzelimpulserzeugung vor.

ENUM	Beschreibung
TIME_10US	10 Mikrosekunden
TIME_100US	100 Mikrosekunden
TIME_1MS	1 Millisekunde
TIME_10MS	10 Millisekunden
TIME_100MS	100 Millisekunden

Beispiel Der Funktionsbaustein wird verwendet, um eine PWM-Klemme in der Betriebsart Einzelimpulsgenerator zu konfigurieren.

Die Einstellungen enthalten **TimeBase** und **Factor**, das Mappen der Prozessdaten ist vorzunehmen.

Im Beispiel ist der Baustein **Pwm2SinglePulse** einkanalig für einen Einzelimpuls der Länge 1s konfiguriert.

Deklaration zum Beispiel

```
PROGRAM PrgIL_Pwm2SinglePulse
VAR
    fbIL_Pwm2SinglePulse: IL_Pwm2SinglePulse;
    winProcessData : WORD;
    wOutProcessData: WORD;
```



```

    bDone: BOOL;
END_VAR

```

Implementation zum Beispiel

```

// Configuration
fbIL_Pwm2SinglePulse.TimeBase    := IL_INLN_PWM_TIME_BASE.TIME_10MS;
fbIL_Pwm2SinglePulse.Stop       := FALSE;
fbIL_Pwm2SinglePulse.Factor     := BYTE#100;
fbIL_Pwm2SinglePulse.InProcData := wInProcessData;
wOutProcessData                 := fbIL_Pwm2SinglePulse.OutProcData;

// FB call
fbIL_Pwm2SinglePulse(Execute := TRUE);

// Interrupting the function block
IF fbIL_Pwm2SinglePulse.Active = TRUE THEN
    fbIL_Pwm2SinglePulse.Stop := TRUE;
END_IF

// Confirmation of pulse execution completed
bDone := fbIL_Pwm2SinglePulse.Done;

```

Prozessdaten-Interface zum Beispiel

Die Prozesseingangs- und -ausgangsworte des Funktionsbausteines sind auf die Prozessdatenworte des einzelnen Feldbussystems zu mappen.

Die Variablen wInProcessData und wOutProcessData sind im Programm prgIL_Pwm2SinglePulse deklariert.

Im Programm ist der erste Kanal gemappt.

Variable	Mapping	Channel	Address	Type
Application.PrgIL_Pwm2SinglePulse.wInProcessData		Input0	%IW30	WORD
		Word1	%IW32	WORD
Application.PrgIL_Pwm2SinglePulse.wOutProcessData		Output0	%QW30	WORD
		Word1	%QW32	WORD

Abb. 14-49: Prozessdaten-Interface

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle INLINEIO_TABLE, 16#0190. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000001	'Frequency' ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000002	'Factor' ist ungültig
DEVICE_ERROR (16#0008)	16#00003020	16#00000000	Zeitüberschreitung, Gerät antwortet nicht

Tab. 14-105: IL_Pwm2SinglePulse Fehlercodes

14.4.9 IL_Ssi

Kurzbeschreibung

Der Funktionsbaustein **IL_Ssi** konfiguriert die Positionierklemme für Absolutwertgeber **R-IB IL SSI-PAC**. Die Positionierklemme enthält neben der kompletten Funktionslogik für die Steuerung des Positioniervorgangs die Ist-Positionserfassung (Eingang für Absolutwertgeber) und die Ansteuerung des Antriebs (vier digitale Ausgänge). Sie kann sowohl für die Punkt-zu-Punkt-Positionierung bei Rundachsen als auch bei Linearachsen eingesetzt werden.

Für jede Inline-EA-Klemme wird eine Funktionsbausteininstanz benötigt.

Kompatibilität Der Funktionsbaustein ist verwendbar für Inlinebus, Profibus, sercos III und Profinet IO-Feldbus.

Schnittstellenbeschreibung

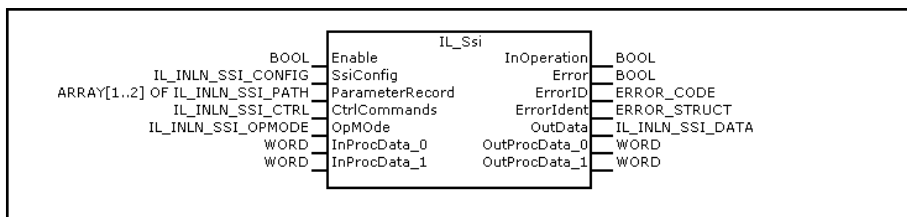


Abb. 14-50: Funktionsbaustein IL_Ssi

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Kontinuierliche Bearbeitung des FBs, solange "Enable" TRUE ist
	SsiConfig	IL_INLN_SSI_CONFIG	Konfiguration der Klemme, siehe "IL_INLN_SSI_CONFIG" auf Seite 672
	ParameterRecord	ARRAY[1..2] OF IL_INLN_SSI_PATH	Parametrierung des Verfahrensweges, siehe "IL_INLN_SSI_CONFIG" auf Seite 672
	CtrlCommands	IL_INLN_SSI_CTRL	Kommandoparameter zur Steuerung der Positionierklemme, siehe "IL_INLN_SSI_CTRL" auf Seite 675
	OpMode	IL_INLN_SSI_OP- MODE	Betriebsartenauswahl, siehe "IL_INLN_SSI_OP- MODE" auf Seite 677
	InProcData_0	WORD	Prozesseingangswort 0
	InProcData_1	WORD	Prozesseingangswort 1
VAR OUTPUT	InOperation	BOOL	Wenn TRUE, arbeitet der Baustein in seiner aktuellen Task
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb der FB-Abarbeitung eingetreten ist
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Detailinformationen zum Fehler, siehe "Fehlerbehandlung" auf Seite 682
	OutData	IL_INLN_SSI_DATA	Status der Positionierklemme, siehe "IL_INLN_SSI_DATA" auf Seite 677
	OutProcData_0	WORD	Prozessausgangswort 0
	OutProcData_1	WORD	Prozessausgangswort 1

Tab. 14-106: Schnittstelle IL_Ssi

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	kontinuierlich
OpMode	IL_INLN_SSI_O P- MODE	0	2	0	kontinuierlich mit "Enable"

Tab. 14-107: Eingangsverhalten IL_Ssi

Funktionsbeschreibung Der Funktionsbaustein konfiguriert die **R-IB IL SSI**-Positionierklemme, parametrisiert Verfahrenwege und steuert die Positionierung.

1. **Konfiguration:** Die Struktur ["IL_INLN_SSI_CONFIG" auf Seite 672](#) enthält alle Eingangsvariablen zum Initialisieren der Klemme.

Bei positiver Flanke an **Enable** wird die Klemme initialisiert und entsprechend der Anwendervorgaben konfiguriert.

Nach erfolgreichem Initialisieren setzt der Baustein **InOperation** auf TRUE.

2. **Parametrierung des Verfahrenweges:** Die Struktur "**IL_INLN_SSI_PATH**" auf Seite 675 enthält alle Daten, die erforderlich sind den Verfahrenweg zu parametrieren. Die Positionierklemme bietet die Möglichkeit, mit zwei Positionierdatensätzen zwei voneinander unabhängige Verfahrenwege zu parametrieren.

Die Variable **ParameterRecord** enthält zwei Elemente des Typs **IL_INLN_SSI_PATH** und erlaubt somit die unabhängige Parametrierung von zwei Verfahrenwegen.

Eine positive Flanke an **DefineParameterRecord** übernimmt die Daten.

3. **Position steuern:** Die Positionierklemme unterstützt drei Betriebsarten zum Position steuern und Status lesen .
 - **Position steuern und Position lesen:** In dieser Betriebsart können Sie die Positionierung steuern und gleichzeitig die aktuelle Position lesen
 - **Position steuern und Geschwindigkeit lesen:** In dieser Betriebsart können Sie die Positionierung steuern und gleichzeitig die aktuelle Geschwindigkeit lesen
 - **Position steuern und Status lesen:** In dieser Betriebsart können Sie die Positionierung steuern und gleichzeitig den aktuellen Status lesen

Die unterschiedlichen Betriebsarten "**IL_INLN_SSI_OPMODE**" auf Seite 677 können über den **OpMode**-Eingang ausgewählt werden. Nach Auswahl der Betriebsart sind die Kommandoparameter "**IL_INLN_SSI_CTRL**" auf Seite 675 zur Steuerung der Positionierklemme zu wählen.

Die Struktur "**IL_INLN_SSI_DATA**" auf Seite 677 enthält alle Status- und Positionsinformationen.

Falls irgend ein Fehler auftritt, generiert der FB die entsprechenden Informationen an **Error**, **ErrorID** und **ErrorIdent**.



1. Definieren Sie den Geber-Offset **vor** dem Übersetzungsfaktor
2. Das Verhältnis zwischen Zähler und Nenner des Übersetzungsfaktors müssen Sie entsprechend der Mechanik selbst bestimmen.
Der Übersetzungsfaktor muss ≤ 1 sein!
3. Jetzt müssen Sie alle weiteren Werte in der Maßeinheit des Übersetzungsfaktors angeben:
 - Software-Endschalter
 - Parametersätze (Anlauffenster, Schnellanlauffenster, Vorabschaltfenster, Abschaltfenster, Zielfenster, Zielposition)
 - Antriebsstillstand
 - Logischer Offset

Außerdem meldet die Klemme alle Werte (z. B. Istposition) in dieser Maßeinheit.



1. Wird die Klemme am Profibus oder Profinet-Feldbus verwendet, muss die Byte-Reihenfolge geändert werden:
Byte 0/1/2/3 = Plug 4/3/2/1.
2. Die Veränderung der Byte-Reihenfolge an der Busklemme ist für alle der Busklemme folgenden Klemmen wirksam!

Abb. 14-51: Dialog Busklemme mit geänderter Bytefolge

IL_INLN_SSI_CONFIG

Die Struktur IL_INLN_SSI_CONFIG enthält alle Parameter zum Initialisieren der Klemme.

Name	Typ	Beschreibung	Min-Wert	Max-Wert	Default-Wert	Übernahme
Parity	BYTE	Geber-Parität 0 – keine 1 – gerade 2 – ungerade	0	2	0	Positive Flanke an "Enable"
EncRotation	BOOL	Einbaurichtung des Gebers FALSE – Normal, TRUE – Invertiert	FALSE	TRUE	FALSE	Positive Flanke an "Enable"
Resolution	BYTE	Geber-Auflösung (8 .. 26)	8	26	8	Positive Flanke an "Enable"
GrayCode	BOOL	FALSE – Binär-Code TRUE – Gray-Code	FALSE	TRUE	FALSE	Positive Flanke an "Enable"
AxType	BYTE	Achstyp (Rundachse/Linearachse) 0 – Linearachse 1 – Rundachse: Bewegung nur in negativer Drehrichtung möglich 2 – Rundachse: Bewegung nur in positiver Drehrichtung möglich 3 – Rundachse: negative und positive Drehrichtung	0	3	0	Positive Flanke an "Enable"

Name	Typ	Beschreibung	Min-Wert	Max-Wert	Default-Wert	Übernahme
BusReset	BOOL	Verhalten der Ausgänge bei Bus-Reset FALSE – Bei einem Bus-Reset werden die Schaltausgänge zurückgesetzt und damit ein laufender Positioniervorgang gestoppt. TRUE – Ein Bus-Reset hat keine Auswirkung auf den Positioniervorgang, d.h. durch die Eingänge können noch Positioniervorgänge gestartet werden.	FALSE	TRUE	FALSE	Positive Flanke an "Enable"
OutputFunctions	BYTE	Schaltausgänge (Ausgangsvarianten) 0 – Ungültig 1 – Version1 2 – Version2 3 – Version3 4 – Version4 5 – Version 5 Näheres im Datenblatt "Versions-details"	0	5	0	Positive Flanke an "Enable"
Initiator3Function	BYTE	IN3F: Funktion Initiator 3 1 - 0/1 Positionierung mit Parametersatz 2 starten 2 - 0/1 Positionierung mit Parametersatz 1 starten 3 – keine (nur digitaler Eingang)	1	3	1	Positive Flanke an "Enable"
Initiator2Function	BYTE	IN2F: Funktion Initiator 2 0 - maximaler Endschalter 1 - 1/0 maximaler Endschalter 2 - 0/1 Positionierung mit Parametersatz 2 starten 3 – keine (nur digitaler Eingang)	0	3	0	Positive Flanke an "Enable"
Initiator1Function	BYTE	IN1F: Funktion Initiator 1 0 - minimaler Endschalter 1 - 1/0 minimaler Endschalter 2 - 0/1 Positionierung mit Parametersatz 1 starten 3 – keine (nur digitaler Eingang)	0	3	0	Positive Flanke an "Enable"
DefineEncOffset	BOOL	Definition Geber-Offset	FALSE	TRUE	FALSE	Kontinuierlich mit "Enable"

Name	Typ	Beschreibung	Min-Wert	Max-Wert	Default-Wert	Übernahme
EncOffset	DWORD	Geber-Offset 0 - kein Geber-Offset 1 ... ($2^{26} - 1$) Inkremente	0	67108863	0	Positive Flanke an "DefineEncOffset"
DefineGear-Ratio	BOOL	Definition Übersetzungsfaktor	FALSE	TRUE	FALSE	Kontinuierlich mit "Enable"
NumeratorGearRatio	WORD	Übersetzungsfaktor Zähler z.B. Verfahrstrecke in cm (Zähler < Nenner)	1	1023	1	Positive Flanke an "DefineGearRatio"
DenominatorGearRatio	WORD	Übersetzungsfaktor Nenner z.B. Verfahrstrecke in Inkrementen	1	65535	1	Positive Flanke an "DefineGearRatio"
DefineLogicOffset	BOOL	Definition des logischen Offset Verschiebung der Positionswerte	FALSE	TRUE	FALSE	Kontinuierlich mit "Enable"
LogicOffset	DINT	Logischer Offset	-33554432	33554431	0	Positive Flanke an "DefineLogicOffset"
DefineMin-MaxSWLimSwitch	BOOL	Definition Software-Endschalter Maximum-Software-Endschalter, Minimum-Software-Endschalter	FALSE	TRUE	FALSE	Kontinuierlich mit "Enable"
MinSWLimSwitch	DINT	Minimum-Software-Endschalter	-33554432	33554431	0	Positive Flanke an "DefineMin-MaxSWLimSwitch"
MaxSWLimSwitch	DINT	Maximum-Software-Endschalter	-33554432	33554431	0	Positive Flanke an "DefineMin-MaxSWLimSwitch"
DefineDsdOd	BOOL	Definition Wartezeit für Richtungs- und Stillstandserkennung und Ausgangsverzögerungszeit	FALSE	TRUE	FALSE	Kontinuierlich mit "Enable"
DrvStarting-Delay	BYTE	Wartezeit für Richtungs- und Stillstandserkennung	0	255	0	Positive Flanke an "DefineDsdOd"
OutDelayTime	WORD	Ausgangsverzögerungszeit	0	4095	0	Positive Flanke an "DefineDsdOd"
DefineJogMode	BOOL	Definition Tipp-Betrieb	FALSE	TRUE	FALSE	Kontinuierlich mit "Enable"
JogDelayTimeCount	BYTE	Verzögerungszeit für Dauersignal bei Tipp-Betrieb	0	255	0	Positive Flanke an "DefineJogMode"
JogMonoflop-Time	WORD	Monoflop-Zeit für Tipp-Betrieb	0	65535	0	Positive Flanke an "DefineJogMode"
DefineDrvStop	BOOL	Definition Antriebsstillstand	FALSE	TRUE	FALSE	Kontinuierlich mit "Enable"
DrvStopTime	BYTE	Zeitintervall Antriebsstillstand	0	31	0	Positive Flanke an "DefineDrvStop"
DrvStopDist	WORD	Strecke Antriebsstillstand	0	65535	0	Positive Flanke an "DefineDrvStop"

IL_INLN_SSI_PATH Die Positionierklemme bietet die Möglichkeit, mit zwei Positionierdatensätzen zwei voneinander unabhängige Verfahrwege zu parametrieren. Diese Struktur enthält die erforderlichen Parameter zur Konfiguration des Verfahrweges.

Name	Typ	Beschreibung	Min-Wert	Max-Wert	Default-Wert	Übernahme
DefineParameterRecord	BOOL	Positive Flanke schreibt die Werte, die zur Parametrierung des Verfahrweges erforderlich sind.	FALSE	TRUE	FALSE	Kontinuierlich mit "Enable"
DefineTargetPos	BOOL	Positive Flanke definiert die Zielposition. Die gleiche Wirkung hat eine positive Flanke an DefineParameterRecord	FALSE	TRUE	FALSE	Kontinuierlich mit "Enable"
StartWindow	DWORD	"Anlaufenster" des Verfahrweges 0 ... (2 ²⁶) - 1 Inkremente	0	67108863	0	Positive Flanke an "DefineParameterRecord"
RapidStartWindow	DWORD	"Schnellanlaufenster" des Verfahrweges 0 ... (2 ²⁶) - 1 Inkremente	0	67108863	0	Positive Flanke an "DefineParameterRecord"
RapidStopWindow	DWORD	"Schnellabschaltfenster" des Verfahrweges 0 ... (2 ²⁶) - 1 Inkremente	0	67108863	0	Positive Flanke an "DefineParameterRecord"
PreStopWindow	DWORD	"Vorabschaltfenster" des Verfahrweges 0 ... (2 ²⁶) - 1 Inkremente	0	67108863	0	Positive Flanke an "DefineParameterRecord"
StopWindow	DWORD	"Abschaltfenster" des Verfahrweges 0 ... (2 ²⁶) - 1 Inkremente	0	67108863	0	Positive Flanke an "DefineParameterRecord"
RepeatCounter	BYTE	Zielfahrt-Wiederholungszähler 0 ... 15 Wiederholungen	0	15	0	Positive Flanke an "DefineParameterRecord"
TargetWindow	WORD	Zielfenster (0 ... 2 ¹⁶ - 1) Inkremente	0	65535	0	Positive Flanke an "DefineParameterRecord"
TargetPos	DINT	Zielposition -2 ²⁵ ... +(2 ²⁵ - 1) Inkremente	-33554432	33554431	0	Positive Flanke an "DefineParameterRecord/ DefineTargetPos"
LubFricCompValue	DINT	Schmier- und Reibungskorrekturwert -2 ²⁵ ... +(2 ²⁵ - 1) Inkremente	-33554432	33554431	0	Positive Flanke an "DefineParameterRecord"

IL_INLN_SSI_CTRL Die Struktur enthält die Kommandoparameter zur Steuerung der Positionierklemme.

Name	Typ	Beschreibung	Min-Wert	Max-Wert	Default-Wert	Übernahme
CtrlOut4	BOOL	Ausgang 4 Falls die Ausgangsvariante 5 aktiviert ist, sonst ohne Bedeutung	FALSE	TRUE	FALSE	Kontinuierlich mit "Enable"
CtrlOut3	BOOL	Ausgang 3. Falls die Ausgangsvariante 5 aktiviert ist, sonst ohne Bedeutung	FALSE	TRUE	FALSE	Kontinuierlich mit "Enable"
CtrlOut2	BOOL	Ausgang 2. Falls die Ausgangsvariante 5 aktiviert ist, sonst ohne Bedeutung	FALSE	TRUE	FALSE	Kontinuierlich mit "Enable"
CtrlOut1	BOOL	Ausgang 1. Falls die Ausgangsvariante 5 aktiviert ist, sonst ohne Bedeutung	FALSE	TRUE	FALSE	Kontinuierlich mit "Enable"
JogNegative	BOOL	Tipp-Betrieb in negativer Richtung	FALSE	TRUE	FALSE	Kontinuierlich mit "Enable"
JogPositive	BOOL	Tipp-Betrieb in positiver Richtung	FALSE	TRUE	FALSE	Kontinuierlich mit "Enable"
EnableLubFricComp	BOOL	Schmier- und Reibungskompensation aktivieren	FALSE	TRUE	FALSE	Kontinuierlich mit "Enable"
EnableLooping	BOOL	Schleifenfahrt erlauben	FALSE	TRUE	FALSE	Kontinuierlich mit "Enable"
DefDrvDir	BOOL	Anfahrriichtung der Zielposition bei aktiver Getriebelose-Kompensation oder Schleifenfahrt festlegen 0: Positive Richtung 1: Negative Richtung bei aktiver Getriebelose-Kompensation oder Schleifenfahrt	FALSE	TRUE	FALSE	Kontinuierlich mit "Enable"
EnableBacklashComp	BOOL	Getriebelose-Kompensation aktivieren Die Richtung ist in DefDrvDir vorgegeben	FALSE	TRUE	FALSE	Kontinuierlich mit "Enable"
EnableLocalCtrl2	BOOL	Freigabe der lokalen Ansteuerung eines Positioniervorganges mit dem Parametersatz 2 durch die digitalen Eingänge (Initiatoren)	FALSE	TRUE	FALSE	Kontinuierlich mit "Enable"
StartPosDrv2	BOOL	0/1 Positionierung mit Parametersatz 2 starten	FALSE	TRUE	FALSE	Kontinuierlich mit "Enable"
EnableLocalCtrl1	BOOL	Freigabe der lokalen Ansteuerung eines Positioniervorganges mit dem Parametersatz 1 durch die digitalen Eingänge (Initiatoren)	FALSE	TRUE	FALSE	Kontinuierlich mit "Enable"

Name	Typ	Beschreibung	Min-Wert	Max-Wert	Default-Wert	Übernahme
StartPosDrv1	BOOL	0/1 Positionierung mit Parametersatz 1 starten	FALSE	TRUE	FALSE	Kontinuierlich mit "Enable"
Stop	BOOL	Die Positionierung sofort stoppen. Das aktuell ausgeführte Kommando wird abgebrochen. Ein neuer Positioniervorgang kann erst nach dem Zurücksetzen des Stopp-Bits gestartet werden.	FALSE	TRUE	FALSE	Kontinuierlich mit "Enable"

IL_INLN_SSI_DATA Die Struktur enthält die Statusparameter der Positionierklemme.

Name	Typ	Beschreibung
ActPos	DINT	Ist-Position
Speed	DINT	Ist-Geschwindigkeit
StatusOfOut4	BOOL	Status des Ausgangs 4
StatusOfOut3	BOOL	Status des Ausgangs 3
StatusOfOut2	BOOL	Status des Ausgangs 2
StatusOfOut1	BOOL	Status des Ausgangs 1
StatusOfIn4	BOOL	Status des Eingangs 4
StatusOfIn3	BOOL	Status des Eingangs 3
StatusOfIn2	BOOL	Status des Eingangs 2
StatusOfIn1	BOOL	Status des Eingangs 1
PosP2InProgress	BOOL	Positionierung mit dem Parametersatz 2 ist in Bearbeitung
PosP2Complete	BOOL	Positionierung mit dem Parametersatz 2 wurde erfolgreich beendet
PosP1InProgress	BOOL	Positionierung mit dem Parametersatz 1 ist in Bearbeitung
PosP1Complete	BOOL	Positionierung mit dem Parametersatz 1 wurde erfolgreich beendet
InitNotComplete	BOOL	Klemme ist noch nicht vollständig initialisiert
DrvStop	BOOL	Antriebsstillstand wurde erkannt oder die Überwachung des Antriebsstillstandes ist ausgeschaltet.
ParameterizationInProgress	BOOL	TRUE - Parametrierung in Bearbeitung

IL_INLN_SSI_OPMODE Die Aufzählung unterstützt die Betriebsartenauswahl.

ENUM	Beschreibung
CTRL_POS_READ_POS	Position steuern und Position lesen
CTRL_POS_READ_VEL	Position steuern und Geschwindigkeit lesen
CTRL_POS_READ_STATUS	Position steuern und Status lesen

Beispiel Für die Positionierfahrt nutzt die Positioniersteuerung das Eilgang-/Schleichgang-Verfahren.

Dabei werden die Antriebe mit zwei festen Geschwindigkeiten (erst Eilgang, dann Schleichgang) bis zum Erreichen des Abschaltpunktes gefahren. Dann

wird der Antrieb abgeschaltet. Das heißt, der Positioniervorgang erfolgt beim Eilgang-/Schleichgang-Verfahren, indem die Positioniersteuerung die Ausgangssignale zur Ansteuerung des Antriebs nach dem Start der Positionierfahrt einschaltet und umschaltet, wenn die Achse die vordefinierten Positionsschwellen überschreitet. Die Umschaltung erfolgt zunächst von Eilgang auf Schleichgang und dann von Schleichgang auf Stopp.

Wenn die Zielposition erreicht wurde, ist der Antrieb abgeschaltet; der Antrieb liefert also in der Zielposition kein Drehmoment.

Wesentliche Schritte zur Konfiguration und Bedienung der Positionierklemme:

1. Geber konfigurieren (obligatorisch) – Parität, Auflösung, Code, Drehrichtung
2. Achstyp konfigurieren (obligatorisch) – Linear oder rotatorisch, Initiator-Funktionen 1, 2 & 3, Ausgangsfunktionen
3. Geber-Offset konfigurieren – Geber-Offset vor Übersetzungsfaktor konfigurieren
4. Übersetzungsfaktor definieren – Übersetzungsfaktor sollte ≤ 1 sein
5. Logischen Offset definieren – Verschiebung von Positionswerten. Alle Werte haben die gleiche Maßeinheit wie der Übersetzungsfaktor
6. Software-Endschalter definieren – Alle Werte haben die gleiche Maßeinheit wie der Übersetzungsfaktor
7. Antriebsstillstand definieren – Maximale Strecke und Zeit bis zum Antriebsstillstand
8. Wartezeit für Richtungs- und Stillstandserkennung definieren
9. Positionierdatensätzen für zwei voneinander unabhängige Fahrwege definieren
10. Position steuern und Position/Geschwindigkeit/Status lesen – Betriebsartenauswahl
11. Kommandoparameter zur Steuerung der Positionierklemme auswählen, um die Aktion durchzuführen



Es kann immer nur ein Auftrag ausgeführt werden, also folgen Sie der Liste Schritt für Schritt.

Z.B. Geber-Offset konfigurieren, Übersetzungsfaktor definieren, Software-Endschalter definieren u.s.w.

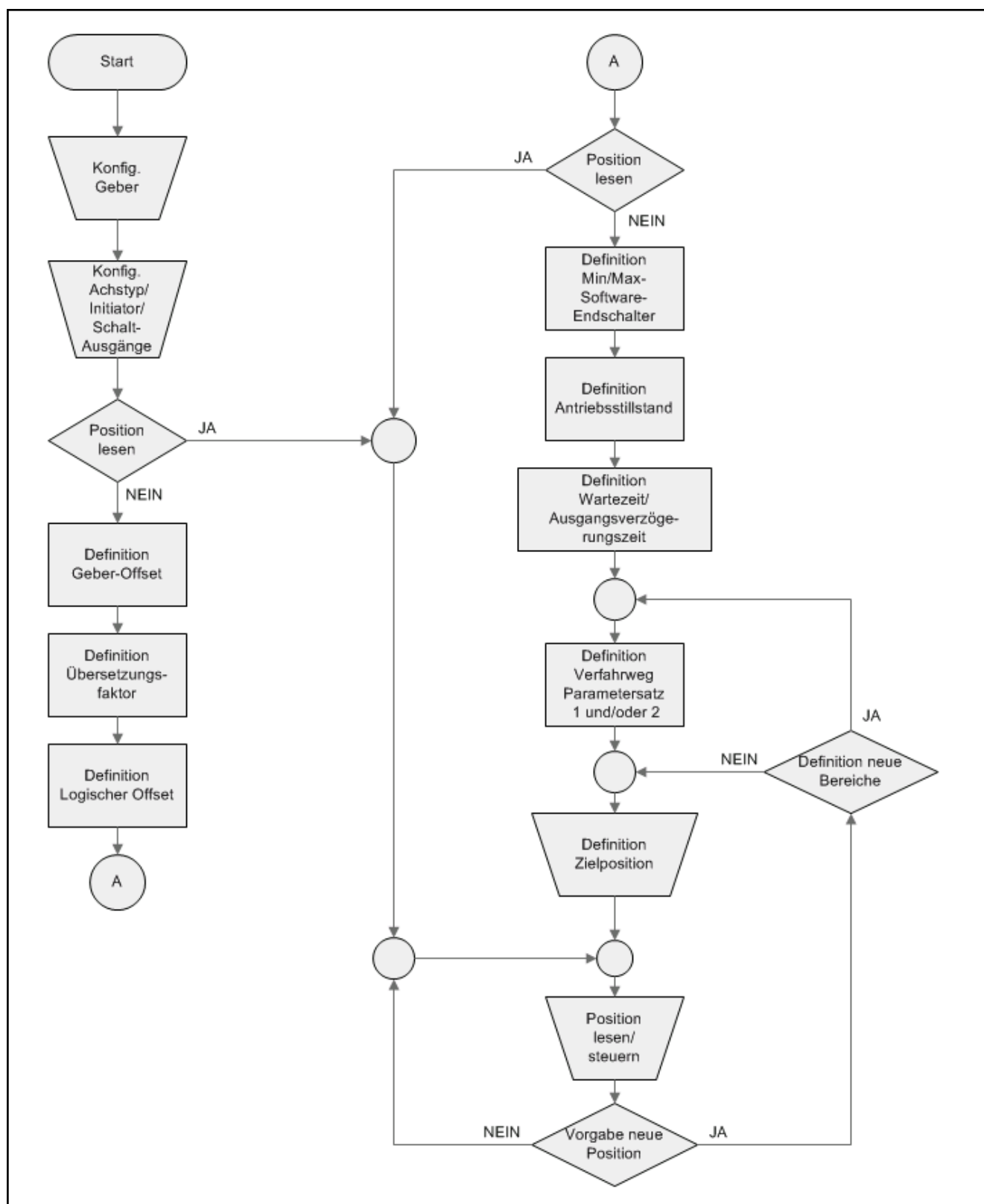


Abb. 14-52: Überblick über die Kommandoabfolge

Deklaration zum Beispiel

```

PROGRAM PrgIL_Ssi
VAR
    fbIL_Ssi: IL_Ssi;
    diActualPos: DINT;
    bDefineEncOffset: BOOL;
    RTrig_GearRatio: R_TRIG;
    bDefineGearRatio: BOOL;
    bDefineLogicOffset: BOOL;
    bDefineSWLimSwitch: BOOL;
    bDefineDsdOd: BOOL;
    bDefineDrvStop: BOOL;
    bDefineJogMode: BOOL;
    bDefineParameterRecord1: BOOL;
    bDefineParameterRecord2: BOOL;
    bStartPosDrv1: BOOL;
    bStartPosDrv2: BOOL;
    bJogPositive: BOOL;
    bJogNegative: BOOL;
    bStop: BOOL;
    bEnableLooping: BOOL;
    bEnableLocalCtrl1: BOOL;
    bEnableLocalCtrl2: BOOL;
    bEnableBacklashComp: BOOL;
    dwEncOffset: DWORD;
    bEnableLubFricComp: BOOL;
    bPosP1Complete: BOOL;
    bPosP2Complete: BOOL;
END_VAR

```

Implementation zum Beispiel*Minimale Konfiguration zum Position-Lesen*

```

fbIL_Ssi.SsiConfig.Parity := BYTE#0;
fbIL_Ssi.SsiConfig.Resolution:= BYTE#12;
fbIL_Ssi.SsiConfig.GrayCode:= TRUE;
fbIL_Ssi.SsiConfig.EncRotation:= FALSE;
fbIL_Ssi.SsiConfig.AxType:= BYTE#3;
fbIL_Ssi.SsiConfig.Initiator1Function:= BYTE#0;
fbIL_Ssi.SsiConfig.Initiator2Function:= BYTE#0;
fbIL_Ssi.SsiConfig.Initiator3Function:= BYTE#2;
fbIL_Ssi.SsiConfig.OutputFunctions:= BYTE#1;
fbIL_Ssi.InProcData_0:= InputProcessData_0;
fbIL_Ssi.InProcData_1:= InputProcessData_1;
OutputProcessData_0:= fbIL_Ssi.OutProcData_0;
OutputProcessData_1:= fbIL_Ssi.OutProcData_1 ;

fbIL_Ssi(Enable:=TRUE);

IF    fbIL_Ssi.InOperation = TRUE THEN
    fbIL_Ssi.OpMode:= IL_INLN_SSI_OPMODE.CTRL_POS_READ_POS;
    diActualPos := fbIL_Ssi.OutData.ActPos;
END_IF

```

Geber-Offset konfigurieren. 0/1 an bDefineEncOffset setzt Geber-Offset

```

IF    fbIL_Ssi.InOperation = TRUE THEN
    fbIL_Ssi.SsiConfig.EncOffset:= dwEncOffset;
    fbIL_Ssi.SsiConfig.DefineEncOffset:= bDefineEncOffset;
END_IF

```

Klemme parametrieren. Positive Flanke am jeweiligen Steuerbit zur Übernahme der Daten

```

IF    fbIL_Ssi.InOperation = TRUE THEN
    fbIL_Ssi.SsiConfig.NumeratorGearRatio:= WORD#1000;
    fbIL_Ssi.SsiConfig.DenominatorGearRatio:= WORD#4096;
    fbIL_Ssi.SsiConfig.MinSWLimSwitch:= DINT#-200;
    fbIL_Ssi.SsiConfig.MaxSWLimSwitch:= DINT#33445531;
    fbIL_Ssi.SsiConfig.DrvStartingDelay:= BYTE#50;
    fbIL_Ssi.SsiConfig.OutDelayTime:= WORD#10;
    fbIL_Ssi.SsiConfig.DrvStopDist:= WORD#200;
    fbIL_Ssi.SsiConfig.DrvStopTime:= BYTE#10;

```

```

fbIL_Ssi.SsiConfig.LogicOffset:= DINT#0;
fbIL_Ssi.SsiConfig.JogDelayTimeCount:= BYTE#20;
fbIL_Ssi.SsiConfig.JogMonoflopTime:= WORD#10;

fbIL_Ssi.SsiConfig.DefineGearRatio := bDefineGearratio;
fbIL_Ssi.SsiConfig.DefineLogicOffset:= bDefineLogicOffset;
fbIL_Ssi.SsiConfig.DefineMinMaxSWLimSwitch:= bDefineSWLimSwitch;
fbIL_Ssi.SsiConfig.DefineDsdOd:= bDefineDsdOd;
fbIL_Ssi.SsiConfig.DefineDrvStop:= bDefineDrvStop;
fbIL_Ssi.SsiConfig.DefineJogMode:= bDefineJogMode;

END_IF

```

*Positionierdatensätze parametrieren. Positive Flanke am jeweiligen Defini-
onseingang zur Übernahme der Daten*

```

IF      fbIL_Ssi.InOperation = TRUE THEN

    fbIL_Ssi.ParameterRecord[1].StartWindow:= DWORD#10;
    fbIL_Ssi.ParameterRecord[1].RapidStartWindow:= DWORD#250;
    fbIL_Ssi.ParameterRecord[1].RapidStopWindow:= DWORD#20;
    fbIL_Ssi.ParameterRecord[1].PreStopWindow:= DWORD#10;
    fbIL_Ssi.ParameterRecord[1].StopWindow:= DWORD#5;
    fbIL_Ssi.ParameterRecord[1].LubFricCompValue:= 0;
    fbIL_Ssi.ParameterRecord[1].TargetWindow:= WORD#2;
    fbIL_Ssi.ParameterRecord[1].TargetPos:=DINT#550;

    fbIL_Ssi.ParameterRecord[1].DefineParameterRecord:=bDefineParameterRecord1;

    fbIL_Ssi.ParameterRecord[2].StartWindow:= DWORD#10;
    fbIL_Ssi.ParameterRecord[2].RapidStartWindow:= DWORD#200;
    fbIL_Ssi.ParameterRecord[2].RapidStopWindow:= DWORD#20;
    fbIL_Ssi.ParameterRecord[2].PreStopWindow:= DWORD#10;
    fbIL_Ssi.ParameterRecord[2].StopWindow:= DWORD#5;
    fbIL_Ssi.ParameterRecord[2].LubFricCompValue:= 0;
    fbIL_Ssi.ParameterRecord[2].TargetWindow:= WORD#2;
    fbIL_Ssi.ParameterRecord[2].TargetPos:=DINT#245;

    fbIL_Ssi.ParameterRecord[2].DefineParameterRecord:=bDefineParameterRecord2;
END_IF

```

*Steuerkommandos. Setzen der entsprechenden Steuerbit zur Steuerung der
Positionierklemme. Während der Positionierung umschalten auf Betriebsart
Position steuern und Status lesen.*

```

IF      fbIL_Ssi.InOperation = TRUE THEN
    fbIL_Ssi.OpMode:= IL_INLN_SSI_OPMODE.CTRL_POS_READ_STATUS;
    fbIL_Ssi.CtrlCommands.StartPosDrv1:= bStartPosDrv1;
    fbIL_Ssi.CtrlCommands.StartPosDrv2:= bStartPosDrv2;
    fbIL_Ssi.CtrlCommands.JogPositive:= bJogPositive;
    fbIL_Ssi.CtrlCommands.JogNegative:= bJogNegative;
    fbIL_Ssi.CtrlCommands.Stop:= bStop;
    fbIL_Ssi.CtrlCommands.EnableLooping:= bEnableLooping;
    fbIL_Ssi.CtrlCommands.EnableLocalCtrl1:= bEnableLocalCtrl1;
    fbIL_Ssi.CtrlCommands.EnableLocalCtrl2:= bEnableLocalCtrl2;
    fbIL_Ssi.CtrlCommands.EnableBacklashComp:= bEnableBacklashComp;
    fbIL_Ssi.CtrlCommands.EnableLubFricComp:= bEnableLubFricComp;

END_IF

```

Status lesen

```

IF      fbIL_Ssi.OpMode = IL_INLN_SSI_OPMODE.CTRL_POS_READ_STATUS THEN
    bPosP1Complete:= fbIL_Ssi.OutData.PosP1Complete;
    bPosP2Complete:= fbIL_Ssi.OutData.PosP2Complete;
END_IF

```

Prozessdaten-Interface zum Beispiel

Die Prozesseingangs- und -ausgangsworte des Funktionsbausteines sind auf die Prozessdatenworte des einzelnen Feldbussystems zu mappen.

Die Variablen wInProcessData_x und wOutProcessData_x sind im Programm prgIL_Ssi deklariert.

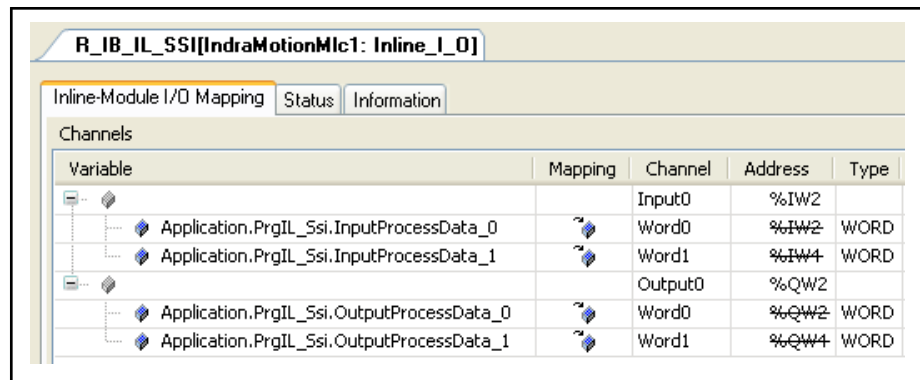


Abb. 14-53: Prozessdaten-Interface

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle INLINEIO_TABLE, 16#0190. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000001	Parität ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000002	Encoder-Auflösung ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000003	Achstyp ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000004	Ausgangsfunktion ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000005	Initiator 3-Funktion ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000006	Initiator 2-Funktion ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000007	Initiator 1-Funktion ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000008	Zeit bis zum Antriebsstillstand ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000009	Übersetzungsfaktor ist ungültig oder Zähler des Übersetzungsfaktors ist größer als der Nenner
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#0000000A	Nenner Übersetzungsfaktor ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#0000000B	Logischer Offset ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#0000000C	Minimum-Software-Endschalter ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#0000000D	Maximum-Software-Endschalter ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#0000000E	Ausgangsverzögerungszeit ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#0000000F	"Anlauffenster" des Verfahrenweges Parametersatz 1 ist ungültig

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000010	"Schnellanlauffenster" des Verfahrweges Parametersatz 1 ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000011	"Schnellabschaltfenster" des Verfahrweges Parametersatz 1 ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000012	"Vorabschaltfenster" des Verfahrweges Parametersatz 1 ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000013	"Abschaltfenster" des Verfahrweges Parametersatz 1 ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000014	Zielfahrt-Wiederholungszähler Parametersatz 1 ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000015	Zielposition Parametersatz 1 ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000016	Schmier- und Reibungskorrekturwert Parametersatz 1 ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000017	"Anlauffenster" des Verfahrweges Parametersatz 2 ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000018	"Schnellanlauffenster" des Verfahrweges Parametersatz 2 ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000019	"Schnellabschaltfenster" des Verfahrweges Parametersatz 2 ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#0000001A	"Vorabschaltfenster" des Verfahrweges Parametersatz 2 ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#0000001B	"Abschaltfenster" des Verfahrweges Parametersatz 2 ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#0000001C	Zielfahrt-Wiederholungszähler Parametersatz 2 ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#0000001D	Zielposition Parametersatz 2 ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#0000001E	Schmier- und Reibungskorrekturwert Parametersatz 2 ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#0000001F	Geber-Offset ist ungültig oder Geber-Offset ist größer als Geberauflösung
DEVICE_ERROR (16#0008)	16#00003020	16#00000000	Zeitüberschreitung, Gerät antwortet nicht
DEVICE_ERROR (16#0008)	16#00003020	16#00000001	Klemme ist im HHOP-Mode, Steuerung ist in diesem Mode über den Bus nicht möglich
DEVICE_ERROR (16#0008)	16#00003020	16#00000002	Ausgangstreiber überlastet oder Kurzschluss
DEVICE_ERROR (16#0008)	16#00003020	16#00000003	Fehler in der Gebersversorgung. Keine Versorgungsspannung oder Kurzschluss
DEVICE_ERROR (16#0008)	16#00003020	16#00000004	Geberkonfiguration ist ungültig
DEVICE_ERROR (16#0008)	16#00003020	16#00000005	Paritätsfehler Verbindung zum Sensor ist defekt oder Sensor-Konfiguration ist fehlerhaft
DEVICE_ERROR (16#0008)	16#00003020	16#00000006	Initiator- oder Schaltausgangs-Konfiguration ist ungültig

ErrorID	Additional1	Additional2	Beschreibung
DEVICE_ERROR (16#0008)	16#00003020	16#00000007	Reserviert
DEVICE_ERROR (16#0008)	16#00003020	16#00000008	Steuerkommando ist ungültig
DEVICE_ERROR (16#0008)	16#00003020	16#00000009	Zielposition ist außerhalb des möglichen Bereiches oder nicht definiert
DEVICE_ERROR (16#0008)	16#00003020	16#0000000A	Funktion kann nicht ausgeführt werden da ein Software-Endschalter überfahren würde
DEVICE_ERROR (16#0008)	16#00003020	16#0000000B	Motor bewegt sich nicht oder bewegt sich in die falsche Richtung
DEVICE_ERROR (16#0008)	16#00003020	16#0000000C	Reserviert
DEVICE_ERROR (16#0008)	16#00003020	16#0000000D	Entfernung zur Zielposition ist geringer als die Summe von Anlauf- und Abschaltfenster
DEVICE_ERROR (16#0008)	16#00003020	16#0000000E	Antriebshalt festgestellt
DEVICE_ERROR (16#0008)	16#00003020	16#0000000F	Software-Endschalter erreicht
DEVICE_ERROR (16#0008)	16#00003020	16#00000010	Hardware-Endschalter erreicht
DEVICE_ERROR (16#0008)	16#00003020	16#00000011	Zielfenster konnte nicht erreicht werden
DEVICE_ERROR (16#0008)	16#00003020	16#00000012	Zählerüberlauf
DEVICE_ERROR (16#0008)	16#00003020	16#00000013	Ungültige Aktion
DEVICE_ERROR (16#0008)	16#00003020	16#00000014	Bus-Reset
DEVICE_ERROR (16#0008)	16#00003020	16#00000015	Ungültige Systemkonfiguration

Tab. 14-108: IL_Ssi Fehlercodes

14.4.10 IL_Ssiln

Kurzbeschreibung

Der Funktionsbaustein **IL_Ssiln** konfiguriert die Inline-Klemme **R-IB IL SSI IN-PAC** für Absolutwertgeber mit SSI-Schnittstelle und liest die Daten vom Absolutwertgeber SSI-Schnittstelle bis 25 Bit (Singleturn, Multiturn oder Längensmesssysteme) mit Gray- und Binary-Code und wandelt dabei Gray-Code in Binary-Code um.

Für jede Inline-EA-Klemme wird eine Funktionsbausteininstanz benötigt.

Kompatibilität

Der Funktionsbaustein ist verwendbar für Inlinebus, Profibus, sercos III und Profinet IO-Feldbus.

Schnittstellenbeschreibung

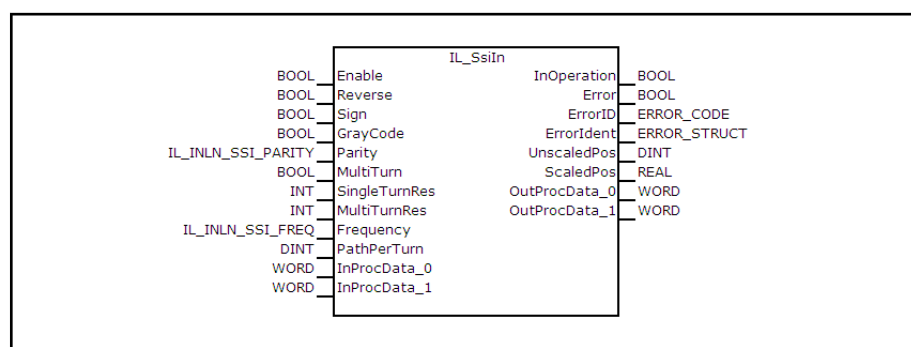


Abb. 14-54: Funktionsbaustein IL_Ssiln

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Kontinuierliche Bearbeitung des FBs, solange "Enable" TRUE ist
	Reverse	BOOL	Drehrichtungsumkehr FALSE = Aus TRUE = Ein
	Sign	BOOL	FALSE = Geberwert ist ohne Vorzeichenbit TRUE = Geberwert ist mit Vorzeichenbit (MSB)
	GrayCode	BOOL	Parameter gibt die Codierung an, in der der Geber die Informationen ausgibt. FALSE = Die Klemme gibt die Daten unverändert zum Lokalbus weiter. TRUE = Die Klemme wandelt die Daten von Gray-Code in Binary-Code um und gibt sie dann zum Lokalbus weiter.
	Parity	IL_INLN_SSI_PARITY	Parität, siehe "IL_INLN_SSI_PARITY" auf Seite 687
	MultiTurn	BOOL	TRUE = Multi turn FALSE = Single turn
	SingleTurnRes	INT	Auflösung Singleturn: 8 bis 25 Bit Hinweis: (SingleTurnRes + MultiTurnRes) <= 25
	MultiTurnRes	INT	Auflösung Multiturn: 0 bis 25 Bit Hinweis: (SingleTurnRes + MultiTurnRes) <= 25
	Frequency	IL_INLN_SSI_FREQ	Übertragungsrate mit der die Geberdaten über die SSI-Schnittstelle gelesen werden, siehe "IL_INLN_SSI_FREQ" auf Seite 687
	PathPerTurn	DINT	Entfernung, die innerhalb einer Geberumdrehung zurückgelegt wird
	InProcData_0	WORD	Prozesseingangswort 0
	InProcData_1	WORD	Prozesseingangswort 1

	Name	Typ	Beschreibung
VAR OUTPUT	InOperation	BOOL	Wenn TRUE, arbeitet der Baustein in seiner aktuellen Task
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb der FB-Abarbeitung eingetreten ist
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Detailinformationen zum Fehler, siehe "Fehlerbehandlung" auf Seite 688
	UnScaledPos	DINT	Nicht-skaliertes Positionswert (Originalwert)
	ScaledPos	REAL	Skalierter Positionswert
	OutProcData_0	WORD	Prozessausgangswort 0
	OutProcData_1	WORD	Prozessausgangswort 1

Tab. 14-109: Schnittstelle IL_SsIn

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	kontinuierlich
Reverse	BOOL	FALSE	TRUE	FALSE	Positive Flanke an "Enable"
Sign	BOOL	FALSE	TRUE	FALSE	kontinuierlich mit Enable
GrayCode	BOOL	FALSE	TRUE	FALSE	Positive Flanke an "Enable"
Parity	IL_INLN_SSI_PARITY	None	Even	None	Positive Flanke an "Enable"
MultiTurn	BOOL	FALSE	TRUE	FALSE	Positive Flanke an "Enable"
SingleTurnRes	INT	8	25	0	Positive Flanke an "Enable"
MultiTurnRes	INT	0	25	0	Positive Flanke an "Enable"
Frequency	IL_INLN_SSI_FREQ	FREQ_100KHZ	FREQ_1MHZ	FREQ_100KHZ	Positive Flanke an "Enable"
PathPerTurn	DINT	0	65535	0	Positive Flanke an "Enable"

Tab. 14-110: Eingangsverhalten IL_SsIn

Funktionsbeschreibung

Der Funktionsbaustein dient zum Auslesen der Daten von Absolutwertgebern mit SSI-Schnittstelle bis 25 Bit (Singleturn, Multiturn oder Längenmesssysteme). Er unterstützt Geber mit Gray- und Binary-Code und wandelt dabei Gray-Code in Binary-Code um.

Der Funktionsbaustein unterstützt folgende Eigenschaften:

- (Singleturn, Multiturn oder Längenmesssysteme)
- 8 ... 25 Bits (konfigurierbar)
- Gray-Code, Binary-Code
- Keine, gerade oder ungerade Paritätsüberwachung

- Übertragungsraten 100 kHz, 200 kHz, 400 kHz, 800 kHz, 1 MHz
- Drehrichtungsumkehr
- Zuweisung eines Vorzeichenbits (sofern der Geber dies unterstützt)
- Offset-Voreinstellung

Eine positive Flanke an **Enable** weist die Eingangsparameter zu und aktiviert den Funktionsbaustein.

Eine Rekonfiguration kann nur durch einen erneute positive Flanke am **Enable**-Eingang ausgelöst werden.

Bei ungültigen Eingangsparametern generiert der FB einen Fehler und die entsprechenden Informationen gelangen an **Error**, **ErrorID** und **ErrorIdent**.

Ein Fehler kann durch eine erneute positive Flanke am **Enable**-Eingang gelöscht werden.

IL_INLN_SSI_PARITY

Mit diesem Parameter wird die Paritätsüberwachungs-Funktion für die SSI-Schnittstelle ausgewählt.

ENUM	Parität
NONE	Keine
ODD	Ungerade Parität
EVEN	Gerade Parität

IL_INLN_SSI_FREQ

Der Aufzählungstyp gestattet die Auswahl der Übertragungsrate mit der die Geberdaten über die SSI-Schnittstelle gelesen werden.

ENUM	Übertragungsrate
FREQ_100KHZ	100 kHz
FREQ_200KHZ	200 kHz
FREQ_400KHZ	400 kHz
FREQ_800KHZ	800 kHz
FREQ_1MHZ	1 MHz

Beispiel

Der Funktionsbaustein liest die Daten eines Absolutwertgeber mit SSI-Schnittstelle bis 25 Bit (Singleturn, Multiturn oder Längenmesssysteme) mit Gray- und Binary-Code und wandelt dabei Gray-Code in Binary-Code um.

In diesem Beispiel ist der Absolutgeber mit einer Singleturn-Auflösung von 8 bit, einer Multiturn-Auflösung von 12 bit, keine Parität, Gray-Code und 100kHz Übertragungsrate konfiguriert. **PathPerTurn** ist die Distance die der Geber zurücklegt bei einer Geberauflösung und im Beispiel zu 100 angenommen. Der Originalwert, der vom Decoder gelesen wurde, wird in **UnScaledPos** und der skalierte Wert in **ScaledPos** angezeigt.

Ist die Drehrichtungsumkehr eingeschaltet, invertiert die Klemme alle vom Geber eingelesenen Positionswerte, d. h. der Positionswert ändert sich durch Aktivieren dieser Funktion.

Das **Sign**-Bit kann verwendet werden, die vom Geber gelesenen Daten zu invertieren.

Deklaration zum Beispiel

```
PROGRAM PrgIL_SsiIn
VAR
    fbIL_SsiIn: IL_SsiIn;
    wInProcessData_0: WORD;
```

RIL_InlineIOModules.library

```

wInProcessData_1: WORD;
wOutProcessData_0: WORD;
wOutProcessData_1: WORD;
diUnScaledPos: DINT;
rScaledPos: REAL;
END_VAR

```

Implementation zum Beispiel

```

// Configuration
fbIL_SsiIn.Frequency      := IL_INLN_SSI_FREQ.FREQ_100KHZ;
fbIL_SsiIn.GrayCode       := TRUE;
fbIL_SsiIn.Parity         := IL_INLN_SSI_PARITY.NONE;
fbIL_SsiIn.SingleTurnRes  := INT#8;
fbIL_SsiIn.MultiTurn      := TRUE;
fbIL_SsiIn.MultiTurnRes   := INT#12;
fbIL_SsiIn.Sign           := FALSE;
fbIL_SsiIn.Reverse        := FALSE;
fbIL_SsiIn.InProcData_0   := wInProcessData_0;
fbIL_SsiIn.InProcData_1   := wInProcessData_1;
wOutProcessData_0         := fbIL_SsiIn.OutProcData_0;
wOutProcessData_1         := fbIL_SsiIn.OutProcData_1;
fbIL_SsiIn.PathPerTurn    := DINT#100;

// FB call
fbIL_SsiIn(Enable := TRUE);

// Data read
IF fbIL_SsiIn.InOperation = TRUE THEN
    diUnScaledPos := fbIL_SsiIn.UnScaledPos;
    rScaledPos    := fbIL_SsiIn.ScaledPos;
END_IF

// For inverting the incoming data
IF fbIL_SsiIn.InOperation = TRUE THEN
    fbIL_SsiIn.Sign := TRUE;
END_IF

// For reversing the encoder pulses
fbIL_SsiIn.Reverse := TRUE;

```

Prozessdaten-Interface zum Beispiel

Die Prozesseingangs- und -ausgangsworte des Funktionsbausteines sind auf die Prozessdatenworte des einzelnen Feldbussystems zu mappen.

Die Variablen wInProcessData_x und wOutProcessData_x sind im Programm prgIL_SsiIn deklariert.

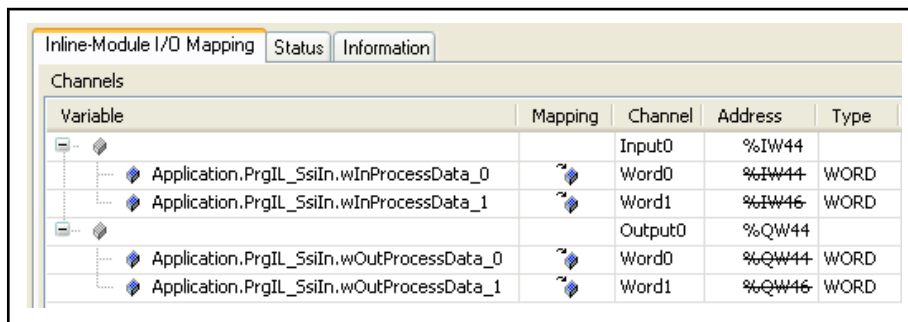


Abb. 14-55: Prozessdaten-Interface

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle INLINEIO_TABLE, 16#0190. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000001	'Parity' ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000002	'SingleTurnRes' ist ungültig.

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000003	'MultiTurnRes' ist ungültig. (0< MultiTurnRes >25)
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000004	Parameter (SingleTurnRes + MultiTurnRes) > 25
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000005	'Frequency' ist ungültig.
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000006	'PathPerTurn' ist ungültig
DEVICE_ERROR (16#0008)	16#00003020	16#00000000	Zeitüberschreitung, Gerät antwortet nicht
DEVICE_ERROR (16#0008)	16#00003020	16#00000042	Gebersversorgung nicht vorhanden oder kurzgeschlossen
DEVICE_ERROR (16#0008)	16#00003020	16#00000044	Paritätsfehler, ein Übertragungsfehler ist bei der seriellen Datenübertragung vom Absolutgeber zur Klemme aufgetreten
DEVICE_ERROR (16#0008)	16#00003020	16#00000048	Ungültige Konfiguration
DEVICE_ERROR (16#0008)	16#00003020	16#00000050	Unbekannter Steuer-Code, ein unbekannter Steuer-Code wurde übertragen

Tab. 14-111: IL_SsIn Fehlercodes

14.4.11 IL_IncIn

Kurzbeschreibung

Der Funktionsbaustein **IL_IncIn** konfiguriert die Inline-Klemme **R-IB IL INC-IN-PAC** (Inline-Klemme für Inkrementalwertgeber) und normalisiert deren Positionswerte.

Der Funktionsbaustein rechnet alle notwendigen Parameter und Messergebnisse in durch den Anwender spezifizierte Maßeinheiten um.

Kompatibilität

Der Funktionsbaustein ist verwendbar für Inlinebus, Profibus, sercos und Profinet IO-Feldbus.

Schnittstellenbeschreibung

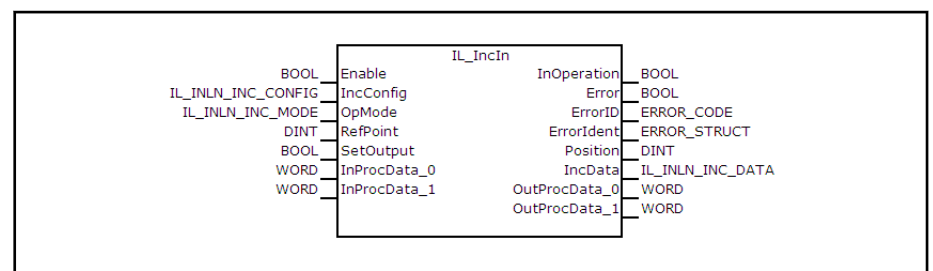


Abb. 14-56: Funktionsbaustein IL_IncIn

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Kontinuierliche Bearbeitung des FBs, solange "Enable" TRUE ist
	IncConfig	IL_INLN_INC_CONFIG	Konfigurationsdaten, siehe "IL_INLN_INC_CONFIG" auf Seite 691
	OpMode	IL_INLN_INC_MODE	Betriebsarten, siehe "IL_INLN_INC_MODE" auf Seite 692
	RefPoint	DINT	Referenzpunkt. Dieser Wert wird als Initialwert beim Homing übernommen
	SetOutput	BOOL	Dieser Eingang steuert die digitale Ausgabe der Klemme
	InProcData_0	WORD	Prozesseingangswort 0
	InProcData_1	WORD	Prozesseingangswort 1
VAR_OUTPUT	InOperation	BOOL	Wenn TRUE, arbeitet der Baustein in seiner aktuellen Task
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb der FB-Abarbeitung eingetreten ist
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Detailinformationen zum Fehler, siehe "Fehlerbehandlung" auf Seite 693
	Position	DINT	Skalierte Position im "ReadPos"-Mode
	IncData	IL_INLN_INC_DATA	Status_I1/ Status_I2 / Status_I3/ IncValue und DeviceHomed, siehe "IL_INLN_INC_DATA" auf Seite 691
	OutProcData_0	WORD	Prozessausgangswort 0
	OutProcData_1	WORD	Prozessausgangswort 1

Tab. 14-112: Schnittstelle IL_IncIn

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	kontinuierlich
RefPoint	DINT	0	-----	0	kontinuierlich
SetOutput	BOOL	FALSE	TRUE	FALSE	kontinuierlich

Tab. 14-113: Eingangsverhalten IL_IncIn

Funktionsbeschreibung

Der Funktionsbaustein kennt zwei Betriebsarten:

1. **ReadPos:** dient dem Lesen und Skalieren der Positionswerte der Klemme.
2. **Reference:** unterstützt das Homing des Gebers. Insgesamt sind 6 Arten des Homing möglich.

Der **ReadPos**-Mode startet, wenn dies für **OpMode** ausgewählt wurde und der **Enable**-Eingang aktiviert wird (positive Flanke).

Mit einer positiven Flanke ist der **IncConfig**-Parameter (IL_INLN_INC_CONFIG-Struktur) automatisch übernommen. Änderungen der Parameterwerte haben nun keinen Einfluss mehr auf die Funktion der Klemme. Werden geänderte Werte benötigt, ist eine erneute positive Flanke an **Enable** erforderlich.

Der 0/1-Übergang an **InOperation** zeigt an, dass der Funktionsbaustein arbeitet, gültige skalierte Messwerte werden an **Position** ausgegeben. Zusätzliche Statusinformationen der Klemme sind in der strukturierten Variablen **IncData** verfügbar.

Der **Reference**-Mode startet, wenn der Funktionsbaustein am **OpMode**-Eingang mit einer der 6 Homing-Arten aktiviert wurde. Während des Homings sind die Positionswerte ungültig. Ein erfolgreiches Homing der Klemme wird durch einen 0/1-Übergang an **IncData.DeviceHomed** angezeigt.

Tritt ein Fehler in einer der beiden Betriebsarten auf, kann er über **Error**, **ErrorID** und **ErrorIdent** identifiziert und beseitigt werden.

IL_INLN_INC_CONFIG

Die Struktur enthält die Parameter zur Konfiguration der Inline-Klemme **R-IB IL INC-IN**.

Name	Typ	Beschreibung	Min-Wert	Max-Wert	Default-Wert	Übernahme
EncoderType	BOOL	Angeschlossener Gebertyp TRUE: Asymmetrischer Geber FALSE: Symmetrischer Geber	FALSE	TRUE	FALSE	Positive Flanke an "Enable"
PulseFactor	INT	Mögliche Pulsfaktoren für Geberpulse 1- Einfach-Auswertung, 2- Zweifach-Auswertung, 4- Vierfach-Auswertung (sollte > 0 sein)	0	4	0	Positive Flanke an "Enable"
IncPerTurn	INT	Inkrement pro Geberumdrehung (für Geberauflösung siehe Geberdatenblatt) (sollte > 0 sein)	0	65535	0	Positive Flanke an "Enable"
PathPerTurn	DINT	Weg der durch die Maschine pro Geberumdrehung zurückgelegt wird. Der Weg sollte in der kleinsten zu messenden Maßeinheit angegeben werden und muss kleiner sein als (IncPerTurn * PulseFactor). (sollte > 0 sein)	0	-----	0	Positive Flanke an "Enable"
Offset	DINT	Offset des Positionswertes Der Wert kann zum Verschieben des Zählbereiches genutzt werden	0	-----	0	kontinuierlich

Tab. 14-114: Zulässige Werte von **IL_INLN_INC_CONFIG** für **IL_Incln**

IL_INLN_INC_DATA

Die Struktur enthält die Ausgangsparameter der Klemme **R-IB IL INC-IN**.

Name	Typ	Beschreibung
Status_I1	BOOL	Status Eingangskanal 1
Status_I2	BOOL	Status Eingangskanal 2
Status_I3	BOOL	Status Eingangskanal 3

Name	Typ	Beschreibung
DeviceHomed	BOOL	Referenzpunkt erreicht
IncValue	REAL	Inhalt des Positionszählers (nicht skaliert).

IL_INLN_INC_MODE Die Aufzählung unterstützt die Auswahl des Betriebszustandes der Klemme.

ENUM	Beschreibung
READPOS	Positions-Istwert lesen
SETTO_REFERENCE	Zähler auf Referenzpunktwert setzen (Referenzieren im Stillstand)
HOME_RISE_DI	Referenzieren auf die nächste positive Flanke von Eingang I1, I2 oder I3 ohne Z
HOME_FALL_DI	Referenzieren auf die nächste negative Flanke von Eingang I1, I2 oder I3 ohne Z
HOME_Z_RISE_DI	Referenzieren auf Z nach positiver Flanke von Eingang I1, I2 oder I3
HOME_Z_FALL_DI	Referenzieren auf Z nach negativer Flanke von Eingang I1, I2 oder I3
HOME_DELTA	Referenzieren abstandcodierter Inkrementalwertgeber
TESTMODE	Testmodus

Beispiel Dieser Funktionsbaustein konfiguriert eine Inline-Klemme für Inkrementalwertgeber zur Messung von Entfernungen oder zur Bestimmung von Positionen.

Die Einstellungen enthalten die Konfiguration der Eingabe und die Betriebsartenauswahl; das Mappen der Prozessdaten ist vorzunehmen.

Im Beispiel ist die **R-IB IL INC-IN-PAC**-Klemme für Einfach-Auswertung und asymmetrischen Gebersignaltyp konfiguriert.

ReadPos oder **Homing** kann durch Auswahl der Betriebsart in **OpMode** vorgegeben werden.

Im **ReadPos**-Mode, liest der Funktionsbaustein die Position. Die Position ist skaliert als 'diPosition' und unskaliert als 'rUnScaledVal' verfügbar.

Im **'SETTO_REFERENCE'**-Mode werden der Antrieb referenziert und die Initialposition nach 'RefPoint' übertragen.

Deklaration zum Beispiel

```
PROGRAM PrgIL_IncIn
VAR
  fbIL_IncIn: IL_IncIn;
  wOutProcessData_0: WORD;
  wOutProcessData_1: WORD;
  wInProcessData_0: WORD;
  wInProcessData_1: WORD;
  diPosition: DINT;
  rUnScaledVal: REAL;
  bStatus_I1: BOOL;
  bStatus_I2: BOOL;
  bStatus_I3: BOOL;
  bDeviceHomed: BOOL;
END_VAR
```

Implementation zum Beispiel

```
// Configuration
fbIL_IncIn.IncConfig.EncoderType := TRUE;
fbIL_IncIn.IncConfig.IncPerTurn  := INT#100;
fbIL_IncIn.IncConfig.Offset      := DINT#45;
fbIL_IncIn.IncConfig.PathPerTurn := DINT#50;
fbIL_IncIn.IncConfig.PulseFactor := INT#1;
fbIL_IncIn.OpMode                 := IL_INLN_INC_MODE.READPOS;
fbIL_IncIn.RefPoint               := DINT#20;
```



```

fbIL_IncIn.SetOutput           := FALSE;
wOutProcessData_0             := fbIL_IncIn.OutProcData_0;
wOutProcessData_1             := fbIL_IncIn.OutProcData_1;
fbIL_IncIn.InProcData_0       := wInProcessData_0;
fbIL_IncIn.InProcData_1       := wInProcessData_1;

// Activating the function block
fbIL_IncIn(Enable :=TRUE);

// Read position
IF fbIL_IncIn.InOperation = TRUE THEN
  fbIL_IncIn.OpMode := IL_INLN_INC_MODE.READPOS;
  diPosition        := fbIL_IncIn.Position;
  rUnScaledVal       := fbIL_IncIn.IncData.IncValue;
  bStatus_I1         := fbIL_IncIn.IncData.Status_I1;
  bStatus_I2         := fbIL_IncIn.IncData.Status_I2;
  bStatus_I3         := fbIL_IncIn.IncData.Status_I3;
  bDeviceHomed       := fbIL_IncIn.IncData.DeviceHomed;
END IF
// Device homing
IF fbIL_IncIn.InOperation = TRUE THEN
  fbIL_IncIn.RefPoint := DINT#30;
  fbIL_IncIn.OpMode   := IL_INLN_INC_MODE.SETTO_REFERENCE;
  bStatus_I1          := fbIL_IncIn.IncData.Status_I1;
  bStatus_I2          := fbIL_IncIn.IncData.Status_I2;
  bStatus_I3          := fbIL_IncIn.IncData.Status_I3;
  bDeviceHomed        := fbIL_IncIn.IncData.DeviceHomed;
END IF

```

Prozessdaten-Interface zum Beispiel

Die Prozesseingangs- und -ausgangsworte des Funktionsbausteines sind auf die Prozessdatenworte des einzelnen Feldbussystems zu mappen.

Die Variablen wInProcessData_x und wOutProcessData_x sind im Programm prgIL_IncIn deklariert.

Variable	Mapping	Channel	Address	Type
Application.PrgIL_IncIn.wInProcessData_0	↔	Input0	%IW20	
Application.PrgIL_IncIn.wInProcessData_1	↔	Word0	%IW20	WORD
Application.PrgIL_IncIn.wInProcessData_1	↔	Word1	%IW22	WORD
Application.PrgIL_IncIn.wOutProcessData_0	↔	Output0	%QW20	
Application.PrgIL_IncIn.wOutProcessData_0	↔	Word0	%QW20	WORD
Application.PrgIL_IncIn.wOutProcessData_1	↔	Word1	%QW22	WORD

Abb. 14-57: Prozessdaten-Interface

Fehlerbehandlung Der Funktionsbaustein nutzt die Fehlertabelle INLINEIO_TABLE, 16#0190. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000001	'PulseFactor' ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000002	'IncPerTurn' ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000003	'PathPerTurn' ist ungültig (≤ 0) oder ($> \text{PulseFactor} * \text{IncPerTurn}$)
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000004	Homing wurde mehrfach angestoßen
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000005	'RefPoint' ist kleiner als der Messbereich

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000006	'RefPoint' ist größer als der Messbereich
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000007	'OpMode' ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#00000008	Istposition ist ungültig (Spannungsversorgung)
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#0000000A	Istposition ist ungültig (Geberfehler)
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#0000000C	Homing nicht möglich, Parametrierfehler
INPUT_INVALID_ERROR (16#0001)	16#00003002	16#0000000E	Istposition ist ungültig (Randbedingungen während der Berechnung geändert)

Tab. 14-115: IL_Incln Fehlercodes

14.5 Kommunikationsmodule

14.5.1 IL_RsCom

Kurzbeschreibung

Der Funktionsbaustein **IL_RSCom** parametriert die Kommunikationsmodule **R-IB IL RS232 PRO**, **R-IB IL RS485/422 PRO** und **R-IB IL RS-Uni** (Inline-Klemmen zur seriellen Datenübertragung) zum Senden und Empfangen von Daten.

Kompatibilität

Der Funktionsbaustein ist verwendbar für Inlinebus, Profibus, Sercos III und Profinet IO-Feldbus.

Schnittstellenbeschreibung

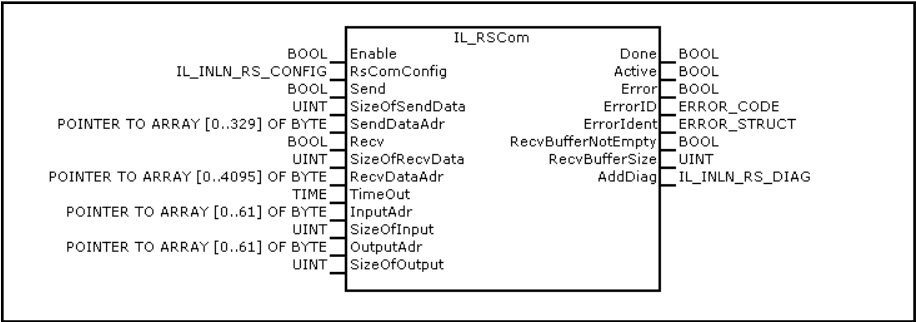


Abb. 14-58: Funktionsbaustein IL_RsCom

	Name	Typ	Beschreibung
VAR_INPUT	Enable	BOOL	Kontinuierliche Bearbeitung des FBs, solange "Enable" TRUE ist
	RsComConfig	IL_INLN_RS_CONFIG	Struktur mit Variablen zur Parametrierung des Moduls. IL_INLN_RS_CONFIG, Seite 707
	Send	BOOL	Die Daten werden bei einer positiven Flanke gesendet. Die Sendezeit ist durch "TimeOut" überwacht
	SendDataAdr	POINTER TO ARRAY [0..329] OF BYTE	Pointer auf das Send-Datenfeld (SendDataAdr <= Send-Datenfeld)
	SizeOfSendData	UINT	Anzahl der zu sendenden Bytes.
	Recv	BOOL	Die Daten werden bei einer positiven Flanke gelesen. Die Lesezeit ist durch "TimeOut" überwacht
	SizeOfRecvData	UINT	Anzahl der zu empfangenden Bytes.
	RecvDataAdr	POINTER TO ARRAY [0..4095] OF BYTE	Pointer auf das Empfangs-Datenfeld (RecvDataAdr <= Empfangs-Datenfeld)
	TimeOut	TIME	Zeit zur Überwachung des Sende-/Empfangsvorgangs
	InputAdr	POINTER TO ARRAY [0..61] OF BYTE	Adresse des Prozesseingangs-Datenfeldes
	SizeOfInput	UINT	Größe des Prozesseingangs-Datenfeldes
	OutputAdr	POINTER TO ARRAY [0..61] OF BYTE	Adresse des Prozessausgangs-Datenfeldes
	SizeOfOutput	UINT	Größe des Prozessausgangs-Datenfeldes
VAR OUTPUT	Done	BOOL	Wenn TRUE, Datenübertragung abgeschlossen
	Active	BOOL	Wenn TRUE, Datenübertragung läuft
	Error	BOOL	Zeigt an, dass ein Fehler innerhalb der FB-Abarbeitung eingetreten ist
	ErrorID	ERROR_CODE	Fehlerkennung
	ErrorIdent	ERROR_STRUCT	Detailinformationen zum Fehler Fehlerbehandlung, Seite 701
	RecvBufferNotEmpty	BOOL	TRUE: Daten im Puffer zum Lesen verfügbar
	RecvBufferSize	UINT	Anzahl der Bytes, die im Puffer zum Lesen verfügbar sind
	AddDiag	IL_INLN_RS_DIAG	Struktur zur erweiterten Diagnose IL_INLN_RS_DIAG, Seite 710

Tab. 14-116: Schnittstelle IL_RsCom

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	kontinuierlich
Send	BOOL	FALSE	TRUE	FALSE	kontinuierlich mit "Enable"
SizeOfSendData	UINT	0	330	0	kontinuierlich mit "Enable"
Recv	BOOL	FALSE	TRUE	FALSE	kontinuierlich mit "Enable"
SizeOfRecvData	UINT	0	4096	0	kontinuierlich mit "Enable"
TimeOut	TIME			T#2s	kontinuierlich mit "Enable"
SizeOfInput	UINT	12	62	0	Positive Flanke an "Enable"
SizeOfOutput	UINT	12	62	0	Positive Flanke an "Enable"

Tab. 14-117: Eingangsverhalten IL_RsCom

Besonderheiten beim Handling

Die Prozessbildbreite der R-IB IL RSUNI Klemme wird durch die DIP Schal-terposition bestimmt und enthält jeweils 7, 15 oder 31 Datenworte in Richtung des Eingangs und des Ausgangs. Die DIP Schalter müssen korrekt gesetzt sein, bevor der Modultyp ausgewählt werden kann.

- Die RecvBufferSize ist immer gleich Null im „End to End“ Protokoll. RecvBufferNotEmpty zeigt an, dass die Daten jetzt gelesen werden können.
- RecvBufferNotEmpty wird im Transparentprotokoll auf TRUE gesetzt, sobald Daten im Eingangspuffer vorhanden sind. Das Modul benötigt einige Zyklen, bis alle Daten im Puffer angekommen sind. Während dieses Zeitraums ändert sich die RecvBufferSize immer wieder.

Empfohlener Workaround für den Anwender beim Empfangen der Daten:

Den Ausgang RecvBufferSize auf Änderungen überprüfen. Wenn sich die Anzahl der empfangenen Bytes nicht mehr ändert, können die Daten empfangen werden.

Oder den Ausgang RecvBufferSize mit der zu erwartenden Datenmenge (Anzahl an Bytes) vergleichen bis die zu erwartende Datenmenge (Anzahl an Bytes) erreicht ist und dann die Daten empfangen.

Werden die Daten ohne Überprüfung des Ausgangs RecvBufferSize empfangen, erhält man bei mehreren Bausteinaufrufen die Daten in mehreren aufgeteilten Teilsequenzen.

Funktionsbeschreibung

Der Funktionsbaustein parametrisiert die Kommunikationsmodule **R-IB IL RS232 PRO**, **R-IB IL RS485/422 PRO** und **R-IB IL RS-Uni** zum Senden und Empfangen von Daten.

- **Transparent:** Beim Transparent-Protokoll werden die seriellen Daten durch die Klemme so weitergereicht, wie sie von der seriellen Schnittstelle oder von der Bus-Seite geliefert wurden
- **XON/XOFF:** Dieses Protokoll arbeitet ähnlich dem Transparent-Protokoll. Im Unterschied dazu wird die Übertragung durch XON/XOFF-Zeichen gesteuert

- **END-END:** Für das Ende-Ende-Protokoll werden die seriellen Daten aufbereitet.

Werden serielle Daten von der Bus-Seite her gesendet, so werden zur Übertragung an die serielle Schnittstelle zwei zusätzliche Zeichen, der First und der Second Delimiter, angehängt. First und Second Delimiter definieren Sie bei der Konfiguration der Klemme.

Empfangene Daten können nur gelesen werden, wenn beide Delimiter empfangen wurden

- **Wechsellpuffer-Protokoll:** Hier wird immer der zuletzt empfangene Datenblock gespeichert.

Ein Datenblock ist definiert als eine Folge von Zeichen mit den Endezeichen First und Second Delimiter, wie beim Ende-Ende-Protokoll. Sobald ein neuer Datenblock empfangen wird, wird der vorherige überschrieben. Erreicht wird das durch zwei Puffer, die abwechselnd beschrieben werden. Damit steht ein Puffer immer zum Empfang der seriellen Daten bereit, während der zweite den zuletzt empfangenen Datenblock speichert. Erst wenn die beiden Delimiter nacheinander erkannt wurden, gilt ein Datenblock als abgeschlossen und kann von der Bus-Seite her gelesen werden. (nicht verfügbar mit **R-IB IL RS-Uni**)

- **3964R:** Dieses Protokoll ist das aufwendigste und wurde von der Firma Siemens entwickelt. Es verwendet Anfangs- und Endekennung, eine Checksumme und eine Zeitüberwachung. Zum Senden stehen 3 Puffer und zum Empfangen stehen 3 Puffer zur Verfügung. (nicht verfügbar mit **R-IB IL RS-Uni**)

Nachstehende Tabelle gibt einen Überblick über die Sende- und Empfangspuffergrößen sowohl für das RS232- als auch für das RS485-Modul und RSUni-Modul.

Protokoll	RS232 Empfangspuffergröße	RS232 Sendepuffergröße	RS485 Empfangspuffergröße	RS485 Sendepuffergröße
TRANSPARENT	4096 Bytes	330 Bytes	4096 Bytes	330 Bytes
END-END	3 Puffer zu 330 Bytes	330 Bytes	25 Puffer zu 11 Bytes	330 Bytes
DUAL BUFFER (Wechsellpuffer)	2 Puffer zu 330 Bytes	330 Bytes	2 Puffer zu 11 Bytes	330 Bytes
3964R	3 Puffer zu 330 Bytes	255 Bytes	25 Puffer zu 11 Bytes	255 Bytes
XON/XOFF	4096 Bytes	330 Bytes	4096 Bytes	330 Bytes



Bei dem **R-IB IL RS-Uni** Modul ist zu beachten, dass die Prozessdaten von der Schalterstellung der DIP-Switch abhängig ist. Dabei ist zu kontrollieren, welche DIP-Switch Stellung, 7 Word, 15 Word oder 31 Word gewählt ist.

Beispiel 1

Der Funktionsbaustein wird verwendet um RS232 PRO-, RS485 PRO- und RSUni-Module für unterschiedliche Protokolle wie Transparent-, End-End-, Wechsellpuffer-, 3964R- und XON-XOFF-Protokoll zu parametrieren.

Im nachstehenden Beispiel wird ein RS232-Modul mit Transparent-Protokoll konfiguriert. das Transparent-Protokoll hat 4096 Bytes Empfangspuffer und 330 Bytes Sendepuffer. Es ist zwingend, die Adressen des Sende- und Empfangs-Arrays und der Prozessdaten vor Aufruf des Funktionsbausteins vorzugeben, sonst ergeben sich NULL-Pointerzugriffe und der Baustein generiert

RIL_InlineIOModules.library

einen Fehler. Die Menge der Empfangsdaten muss gleich oder kleiner als das Empfangsdatenfeld sein.

Nach "Enable" = TRUE konfiguriert der Funktionsbaustein das Modul und startet die Abarbeitung: der Funktionsbaustein liest den Gerätestatus. Wenn kein Fehler dabei auftritt, wird mit dem Datenhandling begonnen.

Deklaration zum Beispiel

```
PROGRAM PrgIL_RsCom
VAR
  fbIL_RsCom : IL_RsCom;
  arSenddataArray: ARRAY[1..330] OF BYTE;
  arRecvdataArray: ARRAY[1..4096] OF BYTE;
  arInProcessData: ARRAY[0..11] OF BYTE;
  arOutProcessData: ARRAY[0..11] OF BYTE;
  bRecvBufferNotEmpty: BOOL;
  uiRecvBufferSize: UINT;
  bRecvBufferFull: BOOL;
  bSendBufferFull: BOOL;
  bSendBufferNotEmpty: BOOL;
  wFirmware: WORD;
  uiExpectedDataSize: UINT := UINT#330;
  bRecvError: BOOL;
  bSendError: BOOL;
  bSend: BOOL;
  bRecv: BOOL;
  bDone: BOOL;
  bActive: BOOL;
  bError: BOOL;
  enErrorID: ERROR_CODE;
  stErrorIdent: ERROR_STRUCT;
END_VAR
```

Implementation zum Beispiel

```
// Configuration
fbIL_RsCom.RsComConfig.Baudrate := IL_INLN_RS_BAUDRATE.BAUDRATE_9600;
fbIL_RsCom.RsComConfig.DataWidth := IL_INLN_RS_DATAWIDTH.DATABIT7_EVEN_STOPBIT1;
fbIL_RsCom.RsComConfig.Protocol := IL_INLN_RS_PROTOCOL.TRANSPARENT ;
fbIL_RsCom.RsComConfig.ErrorPattern := 16#24;
fbIL_RsCom.RsComConfig.FirstDelimiter := 16#0D ;
fbIL_RsCom.RsComConfig.SecondDelimiter := 16#0A ;
fbIL_RsCom.RsComConfig.InterfaceType := IL_INLN_RS_INTERFACE.RS232 ;
fbIL_RsCom.RsComConfig.ModuleType := IL_INLN_RS_MODULETYPE.RS232PRO ;

// FB invocation
fbIL_RsCom(
  Enable:=TRUE,
  Send:= bSend,
  SizeOfSendData:= SIZEOF(arSenddataArray),
  SendDataAdr:= ADR(arSenddataArray),
  Recv:= bRecv,
  SizeOfRecvData:= SIZEOF(arRecvdataArray),
  RecvDataAdr:= ADR(arRecvdataArray),
  Timeout:= T#2S,
  InputAdr:= ADR(arInProcessData),
  SizeOfInput:= SIZEOF(arInProcessData),
  OutputAdr:= ADR(arOutProcessData),
  SizeOfOutput:= SIZEOF(arOutProcessData),
  Done=> bDone,
  Active=> bActive,
  Error=> bError,
  ErrorID=> enErrorID,
  ErrorIdent=> stErrorIdent,
  RecvBufferNotEmpty=> bRecvBufferNotEmpty,
  RecvBufferSize=> uiRecvBufferSize);

// Status monitoring
IF fbIL_RsCom.Active = TRUE THEN
  bRecvBufferFull := fbIL_RsCom.AddDiag.RecvBufferFull ;
  bSendBufferFull := fbIL_RsCom.AddDiag.SendBufferFull ;
  bSendBufferNotEmpty := fbIL_RsCom.AddDiag.SendBufferNotEmpty ;
  wFirmware := fbIL_RsCom.AddDiag.Firmware ;
  bRecvError := fbIL_RsCom.AddDiag.ErrorRecv ;
  bSendError := fbIL_RsCom.AddDiag.ErrorSend ;
END_IF

// Receive cycle
```

```

IF      (fbIL_RSCom.Active = TRUE) AND (fbIL_RSCom.RecvBufferNotEmpty = TRUE)
  AND (fbIL_RSCom.RecvBufferSize >= uiExpectedDataSize) THEN
  bRecv := TRUE ;
END_IF

```

Prozessdaten-Interface zum Beispiel

Die Prozesseingangs- und -ausgangsworte des Funktionsbausteines sind auf die Prozessdatenworte des einzelnen Feldbussystems zu mappen.

Die Variablen arInProcessData und arOutProcessData sind im Programm PrgIL_RsCom deklariert.

Variable	Mapping	Channel	Address	Type
Input0				
Application.PrgIL_RsCom.arInProcessData[0]		Byte0	%IB32	BYTE
Application.PrgIL_RsCom.arInProcessData[1]		Byte1	%IB33	BYTE
Application.PrgIL_RsCom.arInProcessData[2]		Byte2	%IB34	BYTE
Application.PrgIL_RsCom.arInProcessData[3]		Byte3	%IB35	BYTE
Application.PrgIL_RsCom.arInProcessData[4]		Byte4	%IB36	BYTE
Application.PrgIL_RsCom.arInProcessData[5]		Byte5	%IB37	BYTE
Application.PrgIL_RsCom.arInProcessData[6]		Byte6	%IB38	BYTE
Application.PrgIL_RsCom.arInProcessData[7]		Byte7	%IB39	BYTE
Application.PrgIL_RsCom.arInProcessData[8]		Byte8	%IB40	BYTE
Application.PrgIL_RsCom.arInProcessData[9]		Byte9	%IB41	BYTE
Application.PrgIL_RsCom.arInProcessData[10]		Byte10	%IB42	BYTE
Application.PrgIL_RsCom.arInProcessData[11]		Byte11	%IB43	BYTE
Output0				
Application.PrgIL_RsCom.arOutProcessData[0]		Byte0	%QB32	BYTE
Application.PrgIL_RsCom.arOutProcessData[1]		Byte1	%QB33	BYTE
Application.PrgIL_RsCom.arOutProcessData[2]		Byte2	%QB34	BYTE
Application.PrgIL_RsCom.arOutProcessData[3]		Byte3	%QB35	BYTE
Application.PrgIL_RsCom.arOutProcessData[4]		Byte4	%QB36	BYTE
Application.PrgIL_RsCom.arOutProcessData[5]		Byte5	%QB37	BYTE
Application.PrgIL_RsCom.arOutProcessData[6]		Byte6	%QB38	BYTE
Application.PrgIL_RsCom.arOutProcessData[7]		Byte7	%QB39	BYTE
Application.PrgIL_RsCom.arOutProcessData[8]		Byte8	%QB40	BYTE
Application.PrgIL_RsCom.arOutProcessData[9]		Byte9	%QB41	BYTE
Application.PrgIL_RsCom.arOutProcessData[10]		Byte10	%QB42	BYTE
Application.PrgIL_RsCom.arOutProcessData[11]		Byte11	%QB43	BYTE

Abb. 14-59: Prozessdaten-Interface

Beispiel 2

Im nachstehenden Beispiel wird ein RSUni-Modul mit Transparent-Protokoll konfiguriert. Das Transparent-Protokoll hat 4096 Bytes Empfangspuffer und 330 Bytes Sendepuffer. Es ist zwingend, die Adressen des Sende- und Empfangs-Arrays und der Prozessdaten vor Aufruf des Funktionsbausteines vorzugeben, sonst ergeben sich NULL-Pointerzugriffe und der Baustein generiert einen Fehler. Die Menge der Empfangsdaten muss gleich oder kleiner als das Empfangsdatenfeld sein.

Nach "Enable" = TRUE konfiguriert der Funktionsbaustein das Modul und startet die Abarbeitung: der Funktionsbaustein liest den Gerätestatus. Wenn kein Fehler dabei auftritt, wird mit dem Datenhandling begonnen.

Deklaration zum Beispiel

```

PROGRAM PrgIL_RsCom
VAR
  fbIL_RSComUni : IL_RSCom;
  bEnableUni: BOOL;
  arSendDataArrayUni: ARRAY[1..330] OF BYTE;

```

RIL_InlineIOModules.library

```

arRecvdataArrayUni: ARRAY[1..4096] OF BYTE;
arInProcessDataUni: ARRAY[0..13] OF BYTE;
arOutProcessDataUni: ARRAY[0..13] OF BYTE;
bRecvBufferNotEmptyUni: BOOL;
uiRecvBufferSizeUni: UINT;
uiExpectedDataSize: UINT := UINT#330;
bRecvBufferFullUni: BOOL;
bSendBufferFullUni: BOOL;
bSendBufferNotEmptyUni: BOOL;
bSendUni: BOOL;
bRecvUni: BOOL;
bDoneUni: BOOL;
bActiveUni: BOOL;
bErrorUni: BOOL;
enErrorIDUni: ERROR CODE;
stErrorIdentUni: ERROR_STRUCT;
END_VAR

```

Implementation zum Beispiel

```

//Configuration
fbIL_RSComUni.RsComConfig.Baudrate := IL_INLN_RS_BAUDRATE.BAUDRATE_9600;
fbIL_RSComUni.RsComConfig.DataWidth := IL_INLN_RS_DATAWIDTH.DATABIT8_EVEN_
                                     STOPBIT1;
fbIL_RSComUni.RsComConfig.Protocol := IL_INLN_RS_PROTOCOL.TRANSPARENT ;
fbIL_RSComUni.RsComConfig.ErrorPattern := 16#24;
fbIL_RSComUni.RsComConfig.FirstDelimiter := 16#0D ;
fbIL_RSComUni.RsComConfig.SecondDelimiter := 16#0A ;
fbIL_RSComUni.RsComConfig.InterfaceType := IL_INLN_RS_INTERFACE.RS485 ;
fbIL_RSComUni.RsComConfig.ModuleType := IL_INLN_RS_MODULETYPE.RS485PRO ;

//Invocation
fbIL_RSComUni(
  Enable:= bEnableUni,
  Send:= bSendUni,
  SizeOfSendData:= SIZEOF(arSenddataArrayUni),
  SendDataAdr:= ADR(arSenddataArrayUni),
  Recv:= bRecvUni,
  SizeOfRecvData:= SIZEOF(arRecvdataArrayUni),
  RecvDataAdr:= ADR(arRecvdataArrayUni),
  Timeout:= T#2S,
  InputAdr:= ADR(arInProcessDataUni),
  SizeOfInput:= SIZEOF(arInProcessDataUni),
  OutputAdr:= ADR(arOutProcessDataUni),
  SizeOfOutput:= SIZEOF(arOutProcessDataUni),
  Done=> bDoneUni,
  Active=> bActiveUni,
  Error=> bErrorUni,
  ErrorID=> enErrorIDUni,
  ErrorIdent=> stErrorIdentUni,
  RecvBufferNotEmpty=> bRecvBufferNotEmptyUni,
  RecvBufferSize=> uiRecvBufferSizeUni );

// Receive cycle
IF (fbIL_RSComUni.Active = TRUE) AND (fbIL_RSComUni.RecvBufferNotEmpty = TRUE)
  AND (fbIL_RSComUni.RecvBufferSize >= uiExpectedDataSize) THEN
  bRecv := TRUE ;
END_IF

```

Prozessdaten-Interface zum Beispiel

Die Prozesseingangs- und -ausgangsworte des Funktionsbausteines sind auf die Prozessdatenworte des einzelnen Feldbussystems zu mappen.

Die Variablen arInProcessDataUni und arOutProcessDataUni sind im Programm PrgIL_RsCom deklariert.

R_IB_IL_RS_UNI_11_Byte_user_data			
Sercos-Modul	Funktionsgruppen	Benutzerdefinierte Parameter	sercos Module E/A-Abbild
Kanäle			
Variable	Mapping	Kanal	Adresse
Complex Protocol			
Application.prgIL_RSCom.arOutProcessDataUni		DiagOut	%QB85
Application.prgIL_RSCom.arInProcessDataUni		Complex Protocol	%QB87
Application.prgIL_RSCom.arInProcessDataUni		DiagIn	%IB116
Application.prgIL_RSCom.arInProcessDataUni		Complex Protocol	%IB118

Abb. 14-60: Prozessdaten-Interface

Zeitverläufe

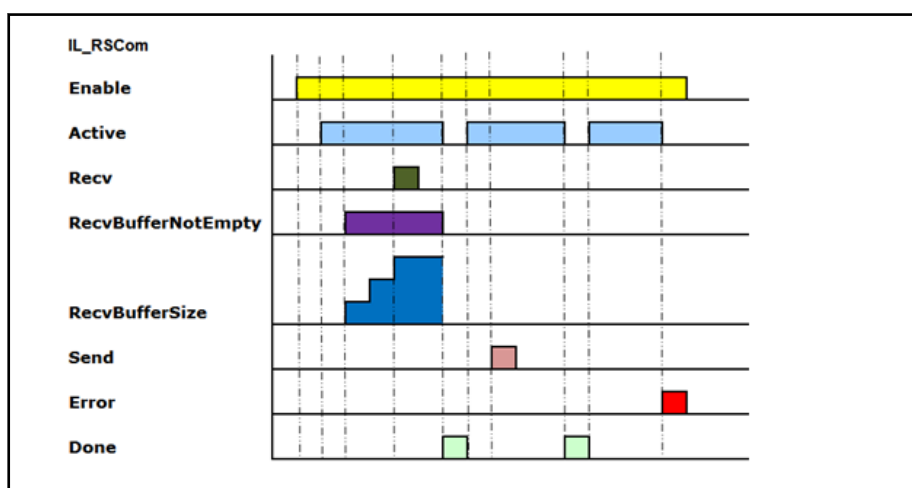


Abb. 14-61: Zeitverläufe IL_RSCom

Fehlerbehandlung

Der Funktionsbaustein nutzt die Fehlertabelle INLINEIO_TABLE, 16#0190. Er kann in Additional1 und Additional2 folgende Fehlermeldungen erzeugen:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR (16#0001)	16#3002	16#0001	Baudrate ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#3002	16#0002	Datenbreite ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#3002	16#0003	Protokoll ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#3002	16#0004	Interface-Eingang ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#3002	16#0005	SizeOfSendData ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#3002	16#0006	SizeOfRecvData ist ungültig
INPUT_INVALID_ERROR (16#0001)	16#3010	16#0000	Falsches Modul angeschlossen
COMMUNICATION_ERROR (16#0002)	16#3011	16#0000	Sende-Zeitüberschreitung
COMMUNICATION_ERROR (16#0002)	16#3011	16#0001	Fehler beim Datensenden

ErrorID	Additional1	Additional2	Beschreibung
COMMUNICATION_ERROR (16#0002)	16#3011	16#0002	Fehler bei der Sende-Datenübergabe zum Puffer
COMMUNICATION_ERROR (16#0002)	16#3011	16#0003	Sende-Fehler im 3964R-Protokoll
COMMUNICATION_ERROR (16#0002)	16#3012	16#0000	Empfangs-Zeitüberschreitung
COMMUNICATION_ERROR (16#0002)	16#3012	16#0001	Fehler beim Datenempfangen
COMMUNICATION_ERROR (16#0002)	16#3012	16#0002	Empfangs-Fehler im 3964R-Protokoll
RESOURCE_ERROR (16#0003)	16#3013	16#0002	Nullzeiger, ungültige Adresse eines Eingangs
DEVICE_ERROR (16#0008)	16#3020	16#0000	Zeitüberschreitung, Gerät antwortet nicht
DEVICE_ERROR (16#0008)	16#3020	16#0001	Ungültiger Parameter für das angegebene Kommando oder Ausfall der Peripheriespannung

Tab. 14-118: IL_RsCom Fehlercodes

14.6 Datentypen

14.6.1 IL_INLN_AI_CONFIG

Die Struktur unterstützt die Konfiguration eines Kanals eines Analog-Eingangsmoduls.

Typdefinition

```

TYPE IL_INLN_AI_CONFIG :
STRUCT
  HighLimit : REAL := 10.0 ; // High limit of the normalized value
  LowLimit : REAL := 0.0; // Low limit of the normalized value
  NoOfSamples : BYTE := BYTE#16 ; // Average value 0, 4, 16, 32
  InputRange : IL_INLN_AI_RANGE ; // Enum to set the Measuring range
END_STRUCT
END_TYPE

```

Genutzte Strukturen und Aufzählungstypen

- IL_INLN_AI_RANGE

Verwendet in

- IL_AI2Sf
- IL_AI4Ef
- IL_AI4IU
- IL_AI8Sfls

14.6.2 IL_INLN_AI_RANGE

Diese Aufzählung ermöglicht das Festlegen des Messbereiches einzelner Analog-Eingangskanäle.

Typdefinition

```

TYPE IL_INLN_AI_RANGE :
(
  UNIPOLAR10V      := 0, // Measuring Voltage 0 to 10 V
  BIPOLAR10V       := 1, // Measuring Voltage -10 to +10 V
  UNIPOLAR5V       := 2, // Measuring Voltage 0 to 5 V
  BIPOLAR5V        := 3, // Measuring Voltage -5 to +5 V
  UNIPOLAR25V      := 4, // Measuring Voltage 0 to 25 V
  BIPOLAR25V       := 5, // Measuring Voltage -25 to + 25 V
)

```

```

UNIPOLAR50V      :=6,  // Measuring Voltage 0 to 50 V
UNIPOLAR20MA     :=8,  // Measuring Current 0 to 20 mA
BIPOLAR20MA      :=9,  // Measuring Current -20 to +20 mA
UNIPOLAR4TO20MA :=10,  // Measuring Current 4 to 20 mA
UNIPOLAR40MA     :=12,  // Measuring Current 0 to 40 mA
BIPOLAR40MA      :=13  // Measuring Current -40 to +40 mA
);
END_TYPE

```

Verwendet in

- IL_AI2Sf
- IL_AI4Ef
- IL_AI4IU
- IL_AI8Sfls

14.6.3 IL_INLN_AI_DATA

IL_INLN_AI_DATA Die Struktur enthält die Ausgabeparameter eines Kanals.

Die Parameter sind für jeden einzelnen Kanal wie folgt zu ermitteln:

ParameterName[Channel_Number].[Function]

z.B. OutputData[0].NormalizedValue.

Name	Typ	Beschreibung
NormalizedValue	REAL	Normalisierter Wert zwischen dem oberen und dem unteren Grenzwert
DiagCode	IL_INLN_DIAG_CODE	Diagnosecode

Tab. 14-119: Strukturelemente IL_INLN_AI_DATA

14.6.4 IL_INLN_DIAG_CODE

Die Tabelle enthält kanalbezogene Fehler und Diagnosen, die von den Funktionsbausteinen der Bibliothek ausgegeben werden.

DiagCode	ENUM	Beschreibung
16#8001	OVERRANGE	Bereichsüberschreitung
16#8002	OPEN_CIRCUIT	Offener Stromkreis, der Ausgang ist als Stromquelle konfiguriert und der vorgegebene Strom kann nicht fließen
16#8003	OVER_LOAD	Kurzschluß/Überlast, der Ausgang ist als Spannungsquelle konfiguriert und überlastet
16#8004	MEASURED_VALUE_INVALID	Ungültiger Messwert / Messwert nicht verfügbar
16#8008	COLD_JUNCTION_DEFECTIVE	Vergleichsstelle defekt
16#8010	INVALID_CONFIGURATION	Ungültige Konfiguration
16#8020	ERROR_IN_IO_SUPPLY_VOLTAGE	Fehler in der Spannungsversorgung
16#8040	MODULE_FAULTY	Modulfehler
16#8080	UNDERRANGE	Bereichsunterschreitung

Tab. 14-120: Enum IL_INLN_DIAG_CODE

Verwendet in

- IL_AI2Sf
- IL_AI4Ef
- IL_AI4IU

- IL_AI8Sfls
- IL_AO1Sf
- IL_AO2SfBp
- IL_Sgi2f
- IL_Temp2Rtd
- IL_Temp2Uth
- IL_Temp4_8Rtd

14.6.5 IL_INLN_CNT_INCONFIG

Die Aufzählung gestattet die Konfiguration eines Eingangsfilters oder einer Eingangslogik zur Verknüpfung von zwei Eingangssignalen.

Typdefinition

```

TYPE IL_INLN_CNT_INCONFIG :
(
  NO_INPUT_CONFIG := 0, //No Input Configuration
  FILTER_FOR_MECH_SWITCH := 1, //Source and Gate filter for Mechanical Contacts
  PULSE_GENERATOR := 2, //Setting for pulse generator
  SOURCE_GATE_OPERATION := 3 //Souce Gate Logic Operation as per "IL_INLN_CNT_SGLOGIC"
);
END_TYPE

```

Verwendet in

- IL_CntCount
- IL_CntFrequency
- IL_CntTime

14.6.6 IL_INLN_TEMP_CONFIG

Die Struktur unterstützt die Konfiguration der Kanäle von Modulen zur Erfassung von Temperaturen.

Typdefinition

```

TYPE IL_INLN_TEMP_CONFIG :
STRUCT
  NoOfSamples : BYTE := BYTE#16 ; // Average value 0, 4, 16, 32
  SensorType : IL_INLN_TEMP_SENSOR; // Sensor Type Selection
  ConnectionType : IL_INLN_TEMP_CONNECTIONTYPE; // Sensor Wire Connection Selection
  Resolution : WORD := 0; // Resolution of value display
  Resistance : IL_INLN_TEMP_R0 := IL_INLN_TEMP_R0.R100; // Value of reference Resistance.
END_STRUCT
END_TYPE

```

Genutzte Strukturen und Aufzählungstypen

- IL_INLN_TEMP_SENSOR, ENUM, Sensortypen
- IL_INLN_TEMP_CONNECTIONTYPE, ENUM, Verbindungstypen der Sensoren
- IL_INLN_TEMP_R0, ENUM, Auswahl des Widerstands-Referenzwerte

Verwendet in

- IL_INLN_TEMP_CONFIG, IL_Temp2Rtd
- IL_INLN_TEMP_CONFIG, IL_Temp4_8Rtd

14.6.7 IL_INLN_TEMP_CONNECTIONTYPE

Der Aufzählungstyp gestattet die Auswahl des Sensor-Verbindungstyps zur Konfiguration eines Kanals.

Typdefinition

```

TYPE IL_INLN_TEMP_CONNECTIONTYPE :
(
  THREEWIRE := 0, // Connection type sensor is 3 wire type
  TWOWIRE := 16#1000, // Connection type sensor is 2 wire type
  FOURWIRE := 16#2000 // Connection type sensor is 4 wire type
);
END_TYPE

```

Verwendet in

- IL_INLN_TEMP_SENSOR, IL_Temp2Rtd
- IL_INLN_TEMP_SENSOR, IL_Temp4_8Rtd

14.6.8 IL_INLN_TEMP_R0

Dieser Aufzählungstyp gestattet die Auswahl des Widerstands-Referenzwertes zur Konfiguration des Kanals.

Typdefinition

```

TYPE IL_INLN_TEMP_R0 :
(
  R100 := 0, // Resistance of Sensor at 0deg : 100 ohm
  R10 := 1, // Resistance of Sensor at 0deg : 10 ohm
  R20 := 2, // Resistance of Sensor at 0deg : 20 ohm
  R30 := 3, // Resistance of Sensor at 0deg : 30 ohm
  R50 := 4, // Resistance of Sensor at 0deg : 50 ohm
  R120 := 5, // Resistance of Sensor at 0deg : 120 ohm
  R150 := 6, // Resistance of Sensor at 0deg : 150 ohm
  R200 := 7, // Resistance of Sensor at 0deg : 200 ohm
  R240 := 8, // Resistance of Sensor at 0deg : 240 ohm
  R300 := 9, // Resistance of Sensor at 0deg : 300 ohm
  R400 := 10, // Resistance of Sensor at 0deg : 400 ohm
  R500 := 11, // Resistance of Sensor at 0deg : 500 ohm
  R1000 := 12, // Resistance of Sensor at 0deg : 1000 ohm
  R1500 := 13, // Resistance of Sensor at 0deg : 1500 ohm
  R2000 := 14, // Resistance of Sensor at 0deg : 2000 ohm
  R10000 := 15, // Resistance of Sensor at 0deg : 10000 ohm
  R3000 := 16 // Resistance of Sensor at 0deg : 3000 ohm
) WORD;
END_TYPE

```

Verwendet in

- IL_INLN_TEMP_SENSOR, IL_Temp2Rtd
- IL_INLN_TEMP_SENSOR, IL_Temp4_8Rtd

14.6.9 IL_INLN_TEMP_SENSOR

Der Aufzählungstyp gestattet die Auswahl des Sensortyps zur Konfiguration eines Kanals.

Typdefinition

```

TYPE IL_INLN_TEMP_SENSOR :
(
  PTDIN := 0, // PT DIN SENSOR TYPE
  PTSAMA := 1, // PT SAMA SENSOR TYPE
  NIDIN := 2, // NI DIN SENSOR TYPE
  NISAMA := 3, // NI SAMA SENSOR TYPE
  CU10 := 4, // CU 10 SENSOR TYPE
  CU50 := 5, // CU 50 SENSOR TYPE
  CU53 := 6, // CU 53 SENSOR TYPE
  NI1000 := 7, // NI 1000 (LANDIS & GYR) SENSOR TYPE
  NI500 := 8, // NI 500 (VIESSMANN) SENSOR TYPE
  KTY81110 := 9, // KTY81 -110 SENSOR TYPE
  KTY84 := 10, // KTY84 SENSOR TYPE
  KTY81210 := 11, // KTY81-210 SENSOR TYPE
  POT := 13, // POTENTIOMETER(%) SENSOR TYPE
  LINEAR400 := 14, // LINEAR R : 0 TO 400 Ohm SENSOR TYPE
  LINEAR20000 := 15, // LINEAR R : 0 TO 20000 Ohm SENSOR TYPE
  LINEAR4000 := 16 // LINEAR R : 0 TO 4000 Ohm SENSOR TYPE
);
END_TYPE

```

Verwendet in

- IL_INLN_TEMP_SENSOR, IL_Temp2Rtd
- IL_INLN_TEMP_SENSOR, IL_Temp4_8Rtd

14.6.10 IL_INLN_TEMP_EF_CONFIG

Die Struktur unterstützt die Konfiguration der Kanäle von Modulen zur Erfassung von Temperaturen.

Typdefinition

```

TYPE IL_INLN_TEMP_EF_CONFIG :
STRUCT
  SensorType      : IL_INLN_TEMP_EF_SENSOR:= IL_INLN_TEMP_EF_SENSOR.PTDIN; // Sensor Type Selection
  Resolution      : BYTE:= BYTE#0; // Resolution of value display
  Resistance      : IL_INLN_TEMP_EF_R0 := IL_INLN_TEMP_EF_R0.R100; // Value of reference Resistance
  FilterTime      : BYTE:= BYTE#0; // Filter Time
  ODSensitivity   : IL_INLN_TEMP_EF_ODS; //Open circuit detection sensitivity
END_STRUCT
END_TYPE

```

Genutzte Strukturen und Aufzählungstypen

- IL_INLN_TEMP_EF_SENSOR, ENUM, Sensortypen
- IL_INLN_TEMP_EF_R0, ENUM, Auswahl des Widerstands-Referenzwerte
- IL_INLN_TEMP_EF_ODS, ENUM, Empfindlichkeit

Verwendet in

- IL_INLN_TEMP_EF_CONFIG, IL_Temp4_8RtdEf

14.6.11 IL_INLN_TEMP_EF_ODS

Dieser Aufzählungstyp gestattet die Auswahl der Empfindlichkeit der Drahtbruchererkennung. (ODS - open circuit detection sensitivity) : "Hohe Empfindlichkeit"/ "Mittlere Empfindlichkeit"/ "Aus".

Typdefinition

```

TYPE IL_INLN_TEMP_EF_ODS:
(
  HIGH SENSITIVITY := 0 // High Sensitivity
  MEDIUM SENSITIVITY := 1 // Medium Sensitivity
  SWITCHED_OFF := 3 // Switched Off
) BYTE;
END_TYPE

```

Verwendet in

- IL_INLN_TEMP_EF_CONFIG, IL_Temp4_8RtdEf

14.6.12 IL_INLN_TEMP_EF_R0

Dieser Aufzählungstyp gestattet die Auswahl des Widerstands-Referenzwertes zur Konfiguration des Kanals.

Typdefinition

```

TYPE IL_INLN_TEMP_EF_R0 :
(
  R100 := 0, // Resistance of Sensor at 0deg : 100 ohm
  R10 := 1, // Resistance of Sensor at 0deg : 10 ohm
  R20 := 2, // Resistance of Sensor at 0deg : 20 ohm
  R30 := 3, // Resistance of Sensor at 0deg : 30 ohm
  R50 := 4, // Resistance of Sensor at 0deg : 50 ohm
  R120 := 5, // Resistance of Sensor at 0deg : 120 ohm
  R150 := 6, // Resistance of Sensor at 0deg : 150 ohm
  R200 := 7, // Resistance of Sensor at 0deg : 200 ohm
  R240 := 8, // Resistance of Sensor at 0deg : 240 ohm
  R300 := 9, // Resistance of Sensor at 0deg : 300 ohm
  R400 := 10, // Resistance of Sensor at 0deg : 400 ohm
  R500 := 11, // Resistance of Sensor at 0deg : 500 ohm

```

```

R1000 := 12,    // Resistance of Sensor at 0deg : 1000 ohm
R1500 := 13,    // Resistance of Sensor at 0deg : 1500 ohm
R2000 := 14,    // Resistance of Sensor at 0deg : 2000 ohm
R10000 := 15,   // Resistance of Sensor at 0deg : 10000 ohm
R3000 := 16     // Resistance of Sensor at 0deg : 3000 ohm
) WORD;
END_TYPE

```

Verwendet in

- IL_INLN_TEMP_EF_CONFIG, IL_Temp4_8RtdEf

14.6.13 IL_INLN_TEMP_EF_SENSOR

Der Aufzählungstyp gestattet die Auswahl des Sensortyps zur Konfiguration eines Kanals.

Typdefinition

```

TYPE IL_INLN_TEMP_EF_SENSOR :
(
    PTDIN      := 0,    // Pt DIN SENSOR TYPE
    PTSAMA     := 1,    // Pt SAMA SENSOR TYPE
    NIDIN      := 2,    // Ni DIN SENSOR TYPE
    NISAMA     := 3,    // Ni SAMA SENSOR TYPE
    CU10       := 4,    // Cu 10 SENSOR TYPE
    CU50       := 5,    // Cu 50 SENSOR TYPE
    CU53       := 6,    // Cu 53 SENSOR TYPE
    NI1000     := 7,    // Ni 1000 (LANDIS & GYR) SENSOR TYPE
    NI500      := 8,    // Ni 500 (VIESSMANN) SENSOR TYPE
    KTY81110   := 9,    // KTY81 -110 SENSOR TYPE
    KTY84      := 16#A, // KTY84 SENSOR TYPE
    KTY81210   := 16#B, // KTY81-210 SENSOR TYPE
    LINEAR30000:= 16#C, // LINEAR R : 0 TO 30000 Ohm SENSOR TYPE

    LINEAR500  := 16#E, // LINEAR R :0 TO 500 Ohm SENSOR TYPE
    LINEAR5000 := 16#F, // LINEAR R :0 TO 5000 Ohm SENSOR TYPE
) WORD;
END_TYPE

```

Verwendet in

- IL_INLN_TEMP_EF_CONFIG, IL_Temp4_8RtdEf

14.6.14 IL_INLN_RS_CONFIG

Die Struktur beschreibt die Konfigurationsparameter des **R-IB IL RS232 PRO**, **R-IB IL RS485/422 PRO** und **R-IB IL RS-Uni**-Moduls. Sie enthält ferner die Standard-Daten einer COM-Schnittstelle.

Name	Typ	Beschreibung	Min-Wert	Max-Wert	Default-Wert	Übernahme
Protocol	IL_INLN_RS_PROTOCOL	Protokolltyp IL_INLN_RS_PROTOCOL , Seite 708	TRANSPARENT	XON_XOFF	TRANSPARENT	Positive Flanke an "Enable"
Baudrate	IL_INLN_RS_BAUDRATE	Baudrate IL_INLN_RS_BAUDRATE , Seite 708	BAUDRATE_110	BAUDRATE_250000	BAUDRATE_9600	Positive Flanke an "Enable"
DataWidth	IL_INLN_RS_DATAWIDTH	Datenbreite IL_INLN_RS_DATAWIDTH , Seite 709	DATA_BIT7_EVEN_STOP_BIT1	DATA_BIT6_NO_NE_STOP_BIT1	DATA_BIT7_EVEN_STOPBIT1	Positive Flanke an "Enable"
ErrorPattern	BYTE	beliebiges Zeichen			16#24	Positive Flanke an "Enable"

Name	Typ	Beschreibung	Min-Wert	Max-Wert	Default-Wert	Übernahme
FirstDelimiter	BYTE	beliebiges Zeichen (verwendet bei End-End & Dual Buffer-Protokoll)			16#0D	Positive Flanke an "Enable"
SecondDelimiter	BYTE	beliebiges Zeichen (verwendet bei End-End & Dual Buffer-Protokoll)			16#0A	Positive Flanke an "Enable"
InterfaceType	IL_INLN_RS_INTERFACE	Interfacetyp, (RS232, RS485 oder RS422) IL_INLN_RS_INTERFACE, Seite 709	RS232	RS422	RS232	Positive Flanke an "Enable"
ModuleType	IL_INLN_RS_MODULE-TYPE	Modultyp (RS232, RS485 oder RSUni) IL_INLN_RS_MODULETYPE, Seite 709	RS232PRO	RSU-NI_31WORLD	RS232PRO	Positive Flanke an "Enable"

Tab. 14-121: IL_INLN_RS_CONFIG, IL_RsCom

14.6.15 IL_INLN_RS_PROTOCOL

Der Aufzählungstyp dient der Auswahl des Protokolltyps.

ENUM	Beschreibung
TRANSPARENT	Transparent-Protokoll
END_TO_END	End-End-Protokoll
DUAL_BUFFER	Wechsellpuffer-Protokoll (nicht möglich mit R-IB IL RS-Uni)
LOW_PRIORITY_3964R	3964R mit niedriger Priorität (nicht möglich mit R-IB IL RS-Uni)
HIGH_PRIORITY_3964R	3964R mit hoher Priorität (nicht möglich mit R-IB IL RS-Uni)
XON_XOFF	XON/XOFF-Protokoll

14.6.16 IL_INLN_RS_BAUDRATE

Der Aufzählungstyp unterstützt die Auswahl der Baudrate.

ENUM	Beschreibung
BAUDRATE_110	110 Baud
BAUDRATE_300	300 Baud
BAUDRATE_600	600 Baud
BAUDRATE_1200	1200 Baud
BAUDRATE_1800	1800 Baud
BAUDRATE_2400	2400 Baud
BAUDRATE_4800	4800 Baud
BAUDRATE_9600	9600 Baud
BAUDRATE_15625	15625 Baud (nur R-IB IL RS-Uni)

ENUM	Beschreibung
BAUDRATE_19200	19200 Baud
BAUDRATE_38400	38400 Baud
BAUDRATE_57600	57600 Baud (nur R-IB IL RS-Uni)
BAUDRATE_115200	115200 Baud (nur R-IB IL RS-Uni)
BAUDRATE_230400	230400 Baud (nur R-IB IL RS-Uni)
BAUDRATE_250000	250000 Baud (nur R-IB IL RS-Uni)

14.6.17 IL_INLN_RS_DATAWIDTH

Der Aufzählungstyp wird zur Vorgabe der Datenbreite genutzt.

ENUM	Beschreibung
DATABIT7_EVEN_STOPBIT1	7 Datenbits, gerade Parität, 1 Stoppbit
DATABIT7_ODD_STOPBIT1	7 Datenbits, ungerade Parität, 1 Stoppbit
DATABIT8_EVEN_STOPBIT1	8 Datenbits, gerade Parität, 1 Stoppbit
DATABIT8_ODD_STOPBIT1	8 Datenbits, ungerade Parität, 1 Stoppbit
DATABIT8_NONE_STOPBIT1	8 Datenbits, ohne Parität, 1 Stoppbit
DATABIT7_NONE_STOPBIT1	7 Datenbits, ohne Parität, 1 Stoppbit
DATABIT7_EVEN_STOPBIT2	7 Datenbits, gerade Parität, 2 Stoppbits
DATABIT7_ODD_STOPBIT2	7 Datenbits, ungerade Parität, 2 Stoppbits
DATABIT8_EVEN_STOPBIT2	8 Datenbits, gerade Parität, 2 Stoppbits
DATABIT8_ODD_STOPBIT2	8 Datenbits, ungerade Parität, 2 Stoppbits
DATABIT8_NONE_STOPBIT2	8 Datenbits, ohne Parität, 2 Stoppbits
DATABIT7_NONE_STOPBIT2	7 Datenbits, ohne Parität, 2 Stoppbits
DATABIT8_CONSTANT0_STOPBIT1	8 Datenbits, Konstant bei 0, 1 Stoppbit (nur R-IB IL RS-Uni)
DATABIT8_CONSTANT1_STOPBIT1	8 Datenbits, Konstant bei 1, 1 Stoppbit (nur R-IB IL RS-Uni)
DATABIT6_NONE_STOPBIT1	6 Datenbits, ohne Parität, 1 Stoppbit (nur R-IB IL RS-Uni)

14.6.18 IL_INLN_RS_INTERFACE

Der Aufzählungstyp unterstützt die Auswahl des Interfacetyps.

ENUM	Beschreibung
RS232	RS-232-Interfacetyp
RS485	RS-485-Interfacetyp
RS422	RS-422-Interfacetyp

14.6.19 IL_INLN_RS_MODULETYPE

Der Aufzählungstyp unterstützt die Auswahl des Modultyps, der verwendet wird.

ENUM	Beschreibung
RS232PRO	Verwendung Modul R-IB IL RS232 PRO
RS485PRO	Verwendung Modul R-IB IL RS485/422 PRO
RSUNI_7WORDS	Verwendung Modul R-IB IL RS-Uni, siehe Tabelle mit DIP-Switch Einstellung
RSUNI_15WORDS	Verwendung Modul R-IB IL RS-Uni, siehe Tabelle mit DIP-Switch Einstellung
RSUNI_31WORDS	Verwendung Modul R-IB IL RS-Uni, siehe Tabelle mit DIP-Switch Einstellung

Tab. 14-122: IL_INLN_RS_MODULETYPE

DIP-Switch 2	DIP-Switch 1	Datenbreite	Nutzdaten
OFF	OFF	7 Word	11 Bytes
OFF	ON	15 Word	27 Bytes
ON	OFF	31 Word	59 Bytes
ON	ON	reserviert	reserviert

Tab. 14-123: DIP-Switch Einstellung

14.6.20 IL_INLN_RS_DIAG

Die Struktur enthält zusätzliche Diagnoseparameter wie **ErrorRecv**, **ErrorSend**, **RecvBufferfull**, **SendBufferfull**, **SendBufferNotEmpty** und die Firmware-Version des Moduls.

Name	Typ	Beschreibung
ErrorRecv	BOOL	Fehler beim Empfang eines Telegramms
ErrorSend	BOOL	Fehler beim Senden eines Telegramms
RecvBufferfull	BOOL	Empfangspuffer ist voll
SendBufferfull	BOOL	Sendepuffer ist voll
SendBufferNotEmpty	BOOL	Sendepuffer ist nicht leer
RecvCounter	UINT	Anzahl der empfangenen Daten
Firmware	WORD	Firmware-Version des Moduls

15 RIH_Safety.library

Die Bibliothek RIH_Safety bietet Zugriffsmöglichkeiten aus der Standard-Applikation auf Daten der Sicherheitssteuerung SafeLogic.

Die Bibliothek RIH_Safety steht automatisch im Bibliotheksverwalter der Standard-Applikation zur Verfügung, sobald eine SafeLogic an der Standard-Steuerung angeschlossen ist. Auf die Informationen der SafeLogic kann in der Standard-Applikation über die globale Variable "SafetyData" zugegriffen werden. Der Einstiegspunkt ist `SafetyData.SafetyFM`.

Weiterhin beinhaltet die Bibliothek RIH_Safety im Ordner "DiagTypes" folgende Typen:

- IH_S20_DIAG_ADD_VALUE beinhaltet den zusätzlichen Diagnosecode und ausgewählte Texte hierfür
- IH_S20_SAFETY_CHANNEL beinhaltet den Kanal und die Zuordnung zum Eingang
- IH_SAFETY_IO_DIAG_TEXT beinhaltet Diagnosetext und Diagnosecode

Diese Typen werden von den Quittierungs-Funktionsbausteinen in den Bibliotheken RIH_Safety_ProfibusDP, RIH_Safety_ProfinetIO und RIH_Safety_SercosIII verwendet.

16 RIH_Safety_ProfibusDP.library

Einführung und Überblick:
 RIH_Safety_ProfibusDP, RIH_Safety_ProfinetIO und RIH_Safety_SercosIII

In diesen Bibliotheken werden feldbusspezifische Funktionalitäten zur Nutzung in der Standard-Applikation gebündelt, die im Zusammenhang mit der SafeLogic stehen. An dem Postfix _ProfibusDP, _ProfinetIO und _SercosIII ist die Zuordnung zum Feldbus zu erkennen. Die Bibliotheken stehen im Bibliotheksrepository ab IndraWorks Version 14V16 zur Verfügung. Die von Ihnen benötigte Bibliothek muss zum Bibliotheksverwalter der Standard-Applikation hinzugefügt werden.



Fügen Sie nur die Bibliotheken zum Bibliotheksverwalter hinzu, die Sie tatsächlich benötigen. Das spart Ladezeit des IndraWorks-Projekts.

Die Bibliotheken enthalten Quittierungsfunktionsbausteine für die Safety-EA-Module von Bosch Rexroth, die an der SafeLogic betrieben werden. Die Quittierungsfunktionsbausteine werden in der Standardapplikation zum Auslesen und Quittieren von Diagnosen der Safety-EA-Module von Bosch Rexroth verwendet. In der Regel wird eine Quittierung durch den Anwender benötigt, damit nach Beseitigung der Fehlerursache der Kanal/das Modul den Betrieb wieder aufnehmen kann. Das Quittierungsverfahren ist herstellerspezifisch und richtet sich nach dem Feldbus und dem lokalen Bus der EA-Station.

Hieraus ergeben sich folgende Quittierungsfunktionsbausteine, die in den feldbusspezifischen Bibliotheken zur Verfügung gestellt werden:

Bibliothek für Profibus: RIH_Safety_ProfibusDP

- IH_PBDPAck_PSDx_Inline Quittierungsfunktionsbaustein für Profibus Inline Safety-EA-Module auf Basis PROFIsafe
- IH_PBDPAck_PSDx_S20 Quittierungsfunktionsbaustein für Profibus S20 Safety-EA-Module auf Basis PROFIsafe

Bibliothek für Profinet: RIH_Safety_ProfinetIO

- IH_PNIOAck_PSDx_Inline Quittierungsfunktionsbaustein für Profinet Inline Safety-EA-Module auf Basis PROFIsafe
- IH_PNIOAck_PSDx_S20 Quittierungsfunktionsbaustein für Profinet S20 Safety-EA-Module auf Basis PROFIsafe

Bibliothek für Sercos: RIH_Safety_SercosIII

- IH_SIIIAck SSDx_S20 Quittierungsfunktionsbaustein für Sercos S20 Safety-EA-Module auf Basis Csos

Die Funktionsbausteine verwenden dabei Datentypen der Bibliothek RIH_Safety.

16.1 IH_PBDPAck_PSDx_Inline

Kurzbeschreibung

Der Funktionsbaustein **IH_PBDPAck_PSDx_Inline** dient zum Auslesen und Quittieren von Diagnosen von PSDx-Modulen (PROFIsafe Inline-Module) am Profibus Inline-Buskoppler (R-IL PB BK DI8 DO4/CN).

Der Funktionsbaustein wird mit steigender Flanke an "Execute" aktiviert und kehrt nach erfolgreicher Ausführung mit TRUE am Ausgang "Done" zurück.

Danach kann der Funktionsbaustein wieder mit steigenden Flanke an "Execute" aktiviert werden. Wie oft der Funktionsbaustein aktiviert wird, bestimmt der Anwender.

Der Ausgang "Active" zeigt an, dass Diagnosen abgerufen werden und verfügbar sind.

Der Ausgang "Done" wird gesetzt, wenn alle Diagnosen quittiert sind oder es sich um nicht-quittierungspflichtige bzw. Standarddiagnosen handelt. Der Ausgang "Done" wird auch gesetzt, wenn keine Diagnose vorhanden war.



Der Funktionsbaustein **IH_PBDPAck_PSDx_Inline** benötigt mindestens den **Profibus Inline-Buskoppler R-IL PB BK DI8 DO4/CN** mit **Änderungsindex AD1** oder höher!

Zielsystem	Bibliothek/Ordner
MLC	RIH_Safety_ProfibusDP
MTX	RIH_Safety_ProfibusDP

Tab. 16-1: Referenztabelle

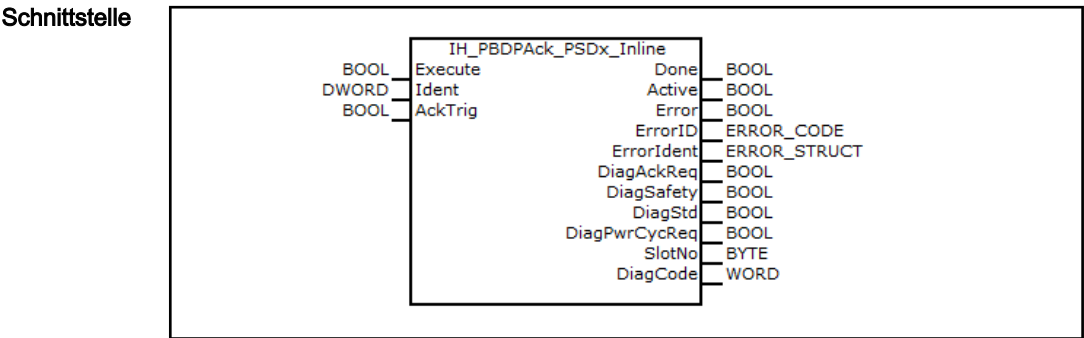


Abb. 16-1: Schnittstellenbeschreibung FB IH_PBDPAck_PSDx_Inline

I/O-Typ	Name	Typ	Beschreibung
VAR_INPUT	Execute	BOOL	Startet bei steigender Flanke die Funktion
	Ident	DWORD	Identifiziert vom DP-Slave R-IL PB BK DI8 DO4/CN (Profibus Ident Handle der Funktion IL_DPIdent , Seite 39) Identifiziert vom DP-Slave R-IL PB BK DI8 DO4/CN (Profibus Ident Handle der Funktion IL_DPIdent) Hinweis: Als Slot ist "0" zu verwenden!
	AckTrig	BOOL	Quittierungseingang: steigende Flanke notwendig, wenn "DiagAckReq" ansteht

I/O-Typ	Name	Typ	Beschreibung
VAR_OUTPUT	Done	BOOL	TRUE zeigt an, dass alle Diagnosen quittiert sind oder dass es sich um nicht-quittierungspflichtige bzw. Standarddiagnosen handelt oder dass keine Diagnose vorhanden war
	Active	BOOL	TRUE zeigt an, dass der Funktionsbaustein aktiv ist
	Error	BOOL	TRUE zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerinformation Siehe auch " Fehlerbehandlung " auf Seite 717
	DiagAckReq	BOOL	TRUE zeigt an, dass eine quittierungspflichtige Diagnose ansteht und über "AckTrig" bestätigt werden muss
	DiagSafety	BOOL	TRUE zeigt an, dass es sich um eine safety-relevante Diagnose handelt
	DiagStd	BOOL	TRUE zeigt an, dass es sich um eine Standarddiagnose handelt
	DiagPwrCycReq	BOOL	TRUE zeigt an, dass ein Neustart (Power Cycle) notwendig ist, damit die PSDx-Module durch den Power On Selbsttest laufen. Hinweis: Der PowerCycle ist am Buskoppler inklusive evtl. zusätzlicher Spannungseinspeisungsklemmen durchzuführen
	SlotNo	BYTE	Slot Nummer des diagnosemeldenden Moduls
	DiagCode	WORD	Diagnosecode

Tab. 16-2: I/O-Schnittstelle des FB IH_PBDPAck_PSDx_Inline

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	kontinuierlich
AckTrig	BOOL			FALSE	kontinuierlich wenn "DiagAckReq" ansteht
Ident	DWORD		16777214	0	steigende Flanke an "Execute"

Tab. 16-3: Min-, Max- und Default-Werte der IH_PBDPAck_PSDx_Inline-Eingänge

Typen und Konstanten

Die Typen und Konstanten ermöglichen es detaillierte Diagnoseinformationen zu erhalten und Diagnosetexte für den Anwender zusammenzubauen.

Die Konstante "IH_DIAGCODE_NO_EXTENDED_DIAG_AVAILABLE" beinhaltet den Diagnosecode für den Fall, dass keine erweiterte Diagnose verfügbar ist.

Die Konstante "IH_DIAGCODE_PROFIBUS_SLAVE_NOT_AVAILABLE" beinhaltet den Diagnosecode für den PROFIBUS-Fehler "Gerät nicht erreichbar".

Die Struktur "IH_SAFETY_IO_DIAG_TEXT" beinhaltet Diagnosetext und Diagnosecode.

Name	Typ	Kommentar
DiagCode	WORD	Diagnosis Code
Text	STRING(255)	Diagnosis Text

Tab. 16-4: "IH_SAFETY_IO_DIAG_TEXT"

Die Konstante "SafetyDiagTexts" beinhaltet 2 Arrays vom Typ "IH_SAFETY_IO_DIAG_TEXT" mit Diagnosetexten für die INLINE-PSDx-Module:

- Deutsch: arInlinePnbSafetyDiagTexts_De
- Englisch: arInlinePbSafetyDiagTexts_En

Die Konstante "IH_PNIO_INLINE_SAFETY_DIAG_TEXT_AMOUNT" gibt die Anzahl der Diagnosetexte an.

Sie können in "arInlinePbSafetyDiagTexts_xx" nach dem "DiagCode" suchen und so den zugehörigen Diagnosetext ermitteln.

```
//Get DE texts from global text array
IF FB_PBDPACK_PSDx_Inline.DiagCode <> 0 THEN
  FOR i:=1 TO IH_PBDP_INLINE_SAFETY_DIAG_TEXT_AMOUNT DO
    IF arInlinePbSafetyDiagTexts_De[i].DiagCode = FB_PBDPACK_PSDx_Inline.DiagCode THEN
      szString := concat (szString , arInlinePbSafetyDiagTexts_De[i].Text);
      EXIT; //Text found
    END_IF
  END_FOR
END_IF
```

Abb. 16-2: Beispiel zum Ermitteln der Diagnosetexte aus "arInlinePbSafetyDiagTexts_xx"

Funktionsbeschreibung

Der Funktionsbaustein **IH_PBDPACK_PSDx_Inline** ermittelt und quittiert Diagnosen von PSDx-Modulen an einem Profibus Inline-Buskoppler (R-IL PB BK DI8 DO4/CN) mittels DPV1 Read/Write-Diensten.

Der Funktionsbaustein **IH_PBDPACK_PSDx_Inline** basiert auf dem Basisfunktionsbaustein **IL_ExecuteDoneBase**.

In dem Funktionsbaustein werden verschiedene DPV1-Funktionsbausteine gebündelt, z. B. [IL_DPReadDiag, Seite 43](#), [IL_DPV1Read, Seite 40](#), [IL_DPV1Write, Seite 42](#).

Der Funktionsbaustein "IH_PBDPACK_PSDx_Inline" ist unter Angabe der Profibus-Adresse des Profibus Inline-Buskopplers und Slot "0" zu verwenden. Der Funktionsbaustein behandelt alle Module, die sich an dem zugehörigen Profibus Inline-Buskoppler befinden.



Um die Kommunikationslast auf dem PROFIBUS gering zu halten, verwenden sie möglichst wenige Funktionsbausteine "IH_PBDPACK_PSDx_Inline" gleichzeitig.

Haben mehrere Kanäle oder Module quittierungspflichtige Fehler, werden diese Fehler im Funktionsbaustein nacheinander behandelt. Die Diagnosen müssen einzeln quittiert werden. Die DiagnoseCodes werden ausgegeben und es werden dann separate Flanken am Eingang "AckTrig" benötigt. Der Zustand TRUE am Ausgang "Done" zeigt an, dass alle Diagnosen quittiert wurden.

Es wird die Quittierung vom Diagnosen für folgende PROFIsafe Inline-Module unterstützt:

- R-IB IL 24 PSDI 16
- R-IB IL 24 PSDI 8

- R-IB IL 24 PSDO 8
- R-IB IL 24 PSDO 4/4
- R-IB IL 24 PSDOR 4

Bei den quittierungspflichtigen Safety-Diagnosen handelt es sich meist um kanalspezifische Diagnosen wie z. B. "Symmetrieverletzung am Eingang x Kanal y". Nach Beseitigung der Fehlerursache müssen diese Diagnosen quittiert werden, bevor der Kanal / das Modul den Betrieb wieder aufnimmt.

WARNUNG

Dokumentation der PSDx-Module beachten!

Die Bedeutung gerätespezifischen Diagnosen und Fehlercodes des Profibus Inline-Buskopplers (R-IL PB BK DI8 DO4/CN) und der PROFIsafe Inline-Module entnehmen Sie der entsprechenden Dokumentation dieser Geräte. In der jeweiligen Dokumentation finden Sie Hinweise zu Fehlerursachen sowie Anweisungen zur Fehlerbeseitigung und Quittierung.

WARNUNG

Quittierung eines Fehlers kann zum gefährlichen Zustand führen

Das Quittieren eines Fehlers führt bis auf die angegebenen Ausnahmen sofort zur Rückkehr des sicheren Ausgangs / sicheren Eingangs in den Betriebszustand. Stellen Sie deshalb vor der Quittierung eines Fehlers sicher, dass die Quittierung nicht zum gefährlichen Zustand der Maschine führen kann! Berücksichtigen Sie bei der Planung der Maschine oder Anlage, dass das Quittieren nur dann möglich sein darf, wenn der Gefahrenbereich einsehbar ist.

Der Prozess der Quittierung von Safety-relevanten Diagnosen wird mit steigender Flanke an "Execute" initiiert.

Zuerst wird implizit der **IL_DPReadDiag** ausgeführt, um zu erkennen, ob am Buskoppler bzw. seinen Modulen eine Diagnose ansteht.

Auf Basis der Diagnose-Rückmeldung wird die Diagnoseklasse ermittelt. Diese Diagnoseklasse gibt Rückschluss ob es sich um eine Standard-Diagnose oder Safety-Diagnose handelt. Auf dieser Basis werden entsprechend die Ausgänge "DiagSafety", "DiagStd" und "DiagPwrCycReq" gesetzt.

Der Ausgang "DiagAckReq" zeigt an, dass eine quittierungspflichtige Safety-Diagnose ansteht und diese Diagnose mit einer positiven Flanke am Eingang "AckTrig" nach Beseitigung der Fehlerursache quittiert werden muss.

Sind alle quittierungspflichtigen Diagnosen abgearbeitet, wird der Ausgang "Done" auf TRUE gesetzt. Am Ausgang "DiagCode" wird die Diagnose "Finished" angezeigt.



Der Quittierungsvorgang im PSDx-Modul nimmt mehrere Sekunden in Anspruch.

Diagnosen

Der Ausgang "DiagCode" gibt den Zustand des Profibus Inline-Buskopplers und der Inline-Modulen zurück.

Zusätzlich zeigt der Ausgang "SlotNo" den diagnosemeldenden Slot bzw. die Inline-Modul Nummer an.

Fehlerbehandlung

Der Funktionsbaustein **IH_PBDPAck_PSDx_Inline** reicht Fehlercodes von verwendeten DPV1-Funktionsbausteinen am eigenen Error-Ausgang weiter (siehe Tabelle **PB_DP_TABLE (16#0130)** Profibus-DP Fehler).

Der Funktionsbaustein **IH_PBDPAck_PSDx_Inline** erzeugt folgende Fehlercodes:

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#000000F9	0	Fehler in der State Machine beim Lesen der Bestätigung für die Quittierung
STATE_MACHINE_ERROR	16#000000FA	0	Fehler in der State Machine beim Schreiben, um den Diagnosepuffer zu aktualisieren
INPUT_RANGE_ERROR	16#000000FB	0	Falscher Slot: Die Funktion "IL DPIdent" mit Slot "0" verwenden
DEVICE_ERROR	16#000000FC	0	Falsches Diagnose-Format im R-IL PB BK DI8 DO4/CN ausgewählt. Das Format "Status-PDU" konfigurieren
DEVICE_ERROR	16#000000FE	0	Nicht unterstütztes Gerät. Es ist zwingend der Buskoppler R-IL PB BK DI8 DO4/CN zu verwenden
STATE_MACHINE_ERROR	16#000000FF	0	Fehler in der State Machine. Wenn dieser Fehler erscheint, hat der Funktionsbaustein nichts weiter getan als diesen Fehler zu generieren

Tab. 16-5: Fehlercodes, die von **IH_PBDPAck_PSDx_Inline** generiert werden

16.2 IH_PBDPAck_PSDx_S20

Kurzbeschreibung Der Funktionsbaustein **IH_PBDPAck_PSDx_S20** dient zum Auslesen und Quittieren von Diagnosen der S20-PSDx-Module (PROFIsafe-Module) am Profibus-S20-Buskoppler (S20-PB-BK).

Der Funktionsbaustein wird mit steigender Flanke am Eingang "Execute" aktiviert und kehrt nach erfolgreicher Ausführung mit TRUE am Ausgang "Done" zurück.

Danach kann der Funktionsbaustein wieder mit steigender Flanke am Eingang "Execute" aktiviert werden. Wie oft der Funktionsbaustein aktiviert wird, bestimmt der Anwender.

Der Ausgang "Active" zeigt an, ob der Funktionsbaustein in Arbeit ist, z. B. Diagnosen abrufen.

Der Ausgang "Done" wird gesetzt, wenn alle Diagnosen quittiert sind oder es sich um nicht-quittierungspflichtige bzw. Standarddiagnosen handelt. Der Ausgang "Done" wird auch gesetzt, wenn keine Diagnose vorhanden war.



Der Funktionsbaustein **IH_PBDPAck_PSDx_S20** benötigt den **Profibus-S20-Buskoppler S20-PB-BK** (R911173247) ab Ausgabe-stand AB1.

Zielsystem	Bibliothek/Ordner
MLC	RIH_Safety_ProfibusDP
MTX	RIH_Safety_ProfibusDP

Tab. 16-6: Referenztable

Schnittstelle

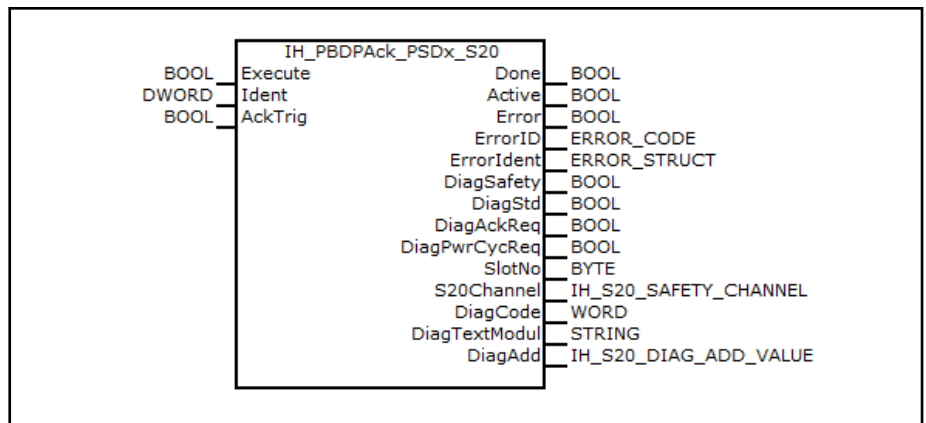


Abb. 16-3: Schnittstellenbeschreibung Funktionsbaustein
IH_PBDPAck_PSDx_S20

	Name	Typ	Beschreibung
VAR_INPUT	Execute	BOOL	Startet bei steigender Flanke die Funktion
	Ident	DWORD	Identifiziert vom DP-Slave "S20-PB-BK" (Profibus Ident Handle der Funktion IL_DPIdent, Seite 39) Identifiziert vom DP-Slave "S20-PB-BK" (Profibus Ident Handle der Funktion IL_DPIdent) Hinweis: Als Slot ist "0" zu verwenden!
	AckTrig	BOOL	Quittierungseingang: Steigende Flanke notwendig, wenn "DiagAckReq" ansteht
VAR_OUTPUT	Done	BOOL	TRUE zeigt an, dass alle Diagnosen quittiert sind oder dass es sich um nicht-quittierungspflichtige bzw. Standarddiagnosen handelt oder dass keine Diagnose vorhanden war
	Active	BOOL	TRUE zeigt an, dass der Funktionsbaustein aktiv ist
	Error	BOOL	TRUE zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerinformation Siehe auch Fehlerbehandlung, Seite 723
	DiagSafety	BOOL	TRUE zeigt an, dass es sich um eine Diagnose von einem Safety-Modul handelt (Safety-Diagnose)
	DiagStd	BOOL	TRUE zeigt an, dass es sich um eine Standarddiagnose handelt.
	DiagAckReq	BOOL	TRUE zeigt an, dass eine quittierungspflichtige Diagnose ansteht und über "AckTrig" bestätigt werden muss
	DiagPwrCycReq	BOOL	TRUE zeigt an, dass ein Neustart/PowerCycle (Spannung aus- und wieder einschalten) notwendig ist, damit die S20-PSDx-Module durch den Power On Selbsttest laufen. Hinweis: Der PowerCycle ist am Buskoppler inklusive evtl. zusätzlicher Spannungseinspeisungsklemmen durchzuführen
	SlotNo	BYTE	Slotnummer des diagnosemeldenden Moduls

	Name	Typ	Beschreibung
	S20Channel	IH_S20_SAFE-TY_CHANNEL	Kanal, für den die Diagnose ansteht. Der Wert 255 bedeutet, dass das ganze Modul betroffen ist. Hinweis: SafetyIoNo, SafetyChannelNo, SafetyIoChannel-Name sind nur gültig, wenn "DiagSafety" TRUE ist.
	DiagCode	WORD	S20 Diagnosecode
	DiagTextModul	STRING	S20 Diagnosetext direkt vom Modul, max. 50 Zeichen. Default-Sprache ist Englisch
	DiagAdd	IH_S20_DI-AG_ADD_VALUE	S20 zusätzliche Diagnose (Code und Text)

Tab. 16-7: Ein-/Ausgänge Funktionsbaustein IH_PBDPAck_PSDx_S20

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	Kontinuierlich
Ident	DWORD		16777214	0	Steigende Flanke an "Execute"
AckTrig	BOOL			FALSE	Kontinuierlich wenn "DiagAckReq" ansteht

Tab. 16-8: Min-/Max- und Default-Werte der Bausteineingänge IH_PBDPAck_PSDx_S20

Typen und Konstanten

Die Typen und Konstanten ermöglichen es, detaillierte Diagnoseinformationen zu erhalten und Diagnosetexte für den Anwender zusammenzubauen.

Die Struktur "IH_S20_DIAG_ADD_VALUE" beinhaltet den zusätzlichen Diagnosecode und ausgewählte Texte hierfür.

Name	Typ	Kommentar
Value	DWORD	S20 Additional Diagnosis Value
Text	STRING	Text For The Additional Diagnosis Value

Tab. 16-9: "IH_S20_DIAG_ADD_VALUE"

Die Struktur "IH_S20_SAFETY_CHANNEL" beinhaltet den Kanal und die Zuordnung zum Eingang.

Name	Typ	Wertebereich
S20ChannelNo	BYTE	0 - 7, 255 (255: whole module is affected)
SafetyIoNo	BYTE	0 - 3, 255 (255: whole module is affected)
SafetyChannelNo	BYTE	1, 2, 255 (255: whole module is affected)
SafetyIoChannelName	STRING	'OUTx_CHy', 'INx_CHy', "

Tab. 16-10: "IH_S20_SAFETY_CHANNEL"

Die Struktur "IH_SAFETY_IO_DIAG_TEXT" beinhaltet Diagnosetext und Diagnosecode.

Name	Typ	Kommentar
DiagCode	WORD	Diagnosis Code
Text	STRING(255)	Diagnosis Text

Tab. 16-11: "IH_SAFETY_IO_DIAG_TEXT"

Die Konstante "IH_DIAGCODE_NO_EXTENDED_DIAG_AVAILABLE" beinhaltet den Diagnosecode für den Fall, dass keine erweiterte Diagnose verfügbar ist.

Die Konstante "IH_DIAGCODE_PROFIBUS_SLAVE_NOT_AVAILABLE" beinhaltet den Diagnosecode für den Profibus-Fehler "Gerät nicht erreichbar".

Die Konstante "SafetyDiagTexts" beinhaltet zwei Arrays vom Typ "IH_SAFETY_IO_DIAG_TEXT" mit Diagnosetexten für die S20-PSDx-Module:

- Deutsch: arS20PbSafetyDiagTexts_De
- Englisch: arS20PbSafetyDiagTexts_En

Die Konstante "IH_PBDP_S20_SAFETY_DIAG_TEXT_AMOUNT" gibt die Anzahl der Diagnosetexte an.

Sie können in "arS20PbSafetyDiagTexts_xx" nach dem "DiagCode" suchen und so den zugehörigen Diagnosetext ermitteln.

Beispiel zum Ermitteln der Diagnosetexte aus "arS20PbSafetyDiagTexts_xx"

Programm:

```
//Get DE diagnosis text from global text array
IF fbIH_PBDPack_PSDx_S20.DiagCode <> 0 THEN
  FOR i:=1 TO IH_PBDP_S20_SAFETY_DIAG_TEXT_AMOUNT DO
    IF arS20PbSafetyDiagTexts_De[i].DiagCode = fbIH_PBDPack_PSDx_S20.DiagCode THEN
      szStringS20 := concat (szStringS20 , arS20PbSafetyDiagTexts_De[i].Text);
      bTextFoundS20 := TRUE;
      EXIT; //Text found
    END IF
  END FOR
END IF
IF bTextFoundS20 = FALSE THEN
  // If no text found use text from modul
  szStringS20 := concat (szStringS20 , fbIH_PBDPack_PSDx_S20.DiagTextModul);
END IF
```

Funktionsbeschreibung

Der Funktionsbaustein "IH_PBDPack_PSDx_S20" ermittelt und quittiert Diagnosen der S20-PSDx-Module (PROFIsafe Module) am Profibus-S20-Buskoppler (S20-PB-BK).

Der Funktionsbaustein "IH_PBDPack_PSDx_S20" basiert auf dem Basisfunktionsbaustein "IL_ExecuteDoneBase".

In dem Funktionsbaustein werden verschiedene DPV1-Funktionsbausteine gebündelt, z. B. [IL_DPReadDiag, Seite 43](#), [IL_DPV1Read, Seite 40](#), [IL_DPV1Write, Seite 42](#).

Der Funktionsbaustein "IH_PBDPack_PSDx_S20" ist unter Angabe der Profibus-Adresse des Profibus-S20-Buskopplers und Slot "0" zu verwenden. Der Funktionsbaustein behandelt alle S20-PSDx-Module, die sich an dem Profibus-S20-Buskoppler befinden.



Um die Kommunikationslast auf dem Profibus gering zu halten, verwenden Sie möglichst wenig Funktionsbausteine "IH_PBDPack_PSDx_S20" gleichzeitig.

Haben mehrere Kanäle oder mehrere Module quittierungspflichtige Fehler, werden diese im Funktionsbaustein nacheinander behandelt. Die Diagnosen

müssen einzeln quittiert werden. D.h. der Diagnosecode wird ausgegeben und es wird dann eine separate Flanke am Eingang "AckTrig" benötigt. Der Zustand TRUE am Ausgang "Done" und der Zustand FALSE am Ausgang "DiagAckReq" zeigen am Ende an, dass die Diagnosen quittiert wurden.

Es wird die Quittierung von Diagnosen für folgende PROFIsafe S20-PSDx-Module unterstützt:

- S20-PSDI-8/4
- S20-PSDO-8/3

Bei den quittierungspflichtigen Safety-Diagnosen handelt es sich meist um kanalspezifische Diagnosen, wie z. B. "Symmetrieverletzung am Eingang x Kanal y". Nach Beseitigung der Fehlerursache müssen diese Diagnosen quittiert werden, damit der Kanal/das Modul den Betrieb wieder aufnehmen kann.

WARNUNG

Es ist die Dokumentation der S20-PSDx-Module zu beachten

Die Bedeutung der gerätespezifischen Diagnosen und Fehlercodes des Profibus-S20-Buskopplers (S20-PB-BK) und der PROFIsafe-S20-Module entnehmen Sie der entsprechenden Dokumentation dieser Geräte. Dort finden Sie Hinweise zu Fehlerursachen sowie Anweisungen zur Fehlerbeseitigung und Quittierung.

WARNUNG

Quittierung kann zum gefährlichen Zustand führen

Das Quittieren eines Fehlers führt bis auf die angegebenen Ausnahmen sofort zur Rückkehr des sicheren Ausgangs/sicheren Eingangs in den Betriebszustand. Stellen Sie deshalb vor der Quittierung eines Fehlers sicher, dass die Quittierung nicht zum gefährlichen Zustand der Maschine führen kann! Berücksichtigen Sie bei der Planung der Maschine oder Anlage, dass das Quittieren nur dann möglich sein darf, wenn der Gefahrenbereich einsehbar ist.

Der Prozess des Auslesens und der Quittierung von Safety-Diagnosen wird mit steigender Flanke am Eingang "Execute" initiiert.

Am Anfang wird einmalig die Sollkonfiguration am S20-PB-BK ausgelesen. Damit ist bekannt, auf welchen Slots sich S20-PSDx-Module befinden. Ändern sich während der Laufzeit des Funktionsbausteins die Eingangsparameter Ident (Master, Segment oder Station) wird die Sollkonfiguration am S20-PB-BK erneut ausgelesen.

Zuerst wird implizit der **IL_DPReadDiag** ausgeführt, um zu erkennen, ob am Buskoppler bzw. seinen Modulen eine Diagnose ansteht.

Der Funktionsblock sucht nach dem ersten S20-PSDx-Modul, welches eine Diagnose hat. (Die Suche beginnt bei der niedrigsten Slot-Nummer.)

Wenn die Safety-Diagnose erfolgreich quittiert wurde, wird die nächste Diagnose von diesem Modul behandelt (meist von einem anderen Kanal).

Gibt es keine Diagnosen mehr an diesem Modul, dann wird das nächste S20-PSDx-Modul gesucht, welches Diagnosen hat.

Gibt es keine S20-PSDx-Module mit Diagnosen, wird das erste Standard-Module behandelt, welches Diagnosen hat.

Auf dieser Basis werden die Ausgänge "DiagSafety", "DiagStd", "DiagAckReq", "DiagPwrCycReq", "SlotNo", "S20Channel" und "DiagCode" gesetzt.

Der Ausgang "DiagAckReq" zeigt an, dass eine quittierungspflichtige Safety-Diagnose ansteht und diese Diagnose mit einer positiven Flanke an dem Eingang "AckTrig" nach Beseitigung der Fehlerursache quittiert werden muss. Zum Quittieren wird intern das "ResetDiag-Objekt 0x0019" verwendet.

Sind alle quittierungspflichtigen Diagnosen abgearbeitet, wird der Ausgang "Done" auf TRUE gesetzt.

Diagnosen Der Ausgang "DiagCode" gibt den Zustand des Profibus-S20-Buskopplers und der S20-Module an.

Der Ausgang "SlotNo" zeigt den diagnosemeldenden Slot bzw. die S20-Modul Nummer an.

Der Ausgang "S20Channel" zeigt den diagnosemeldenden Kanal an. Der Wert 255 bedeutet, dass das ganze Modul betroffen ist. Basis hierfür ist das Diagnoseobjekt "DiagState-Objekt 0x0018, Index 3: Channel / Group / Module".

Der Ausgang "DiagTextModul" gibt den Diagnosetext direkt vom Modul aus. Das entspricht Diagnoseobjekt "DiagState-Objekt 0x0018, Index 5: Text".

Der Ausgang "DiagAdd" gibt zusätzliche Diagnose (Code und Text) aus. Hierfür wird das Diagnoseobjekt "DiagStateAddValue-Objekt 0x0034" ausgelesen.

Fehlerbehandlung Der Funktionsbaustein "IH_PBDPAck_PSDx_S20" reicht die Fehlercodes von den intern verwendeten Funktionsbausteinen am eigenen Error-Ausgang weiter. Siehe Tabelle **PB_DP_TABLE (16#0130)** Profibus-DP Fehler.

Der Funktionsbaustein IH_PBDPAck_PSDx_S20 erzeugt folgende Fehlercodes:

ErrorID	Additional1	Additional2	Beschreibung
DEVICE_ERROR	16#000000FC	0	Falsches Diagnoseformat im S20-PB-BK ausgewählt. Bitte das Format "Common Format" konfigurieren
INPUT_RANGE_ERROR	16#000000FB	0	Falscher Slot: Die Funktion "IL DPIdent" mit Slot "0" verwenden
DEVICE_ERROR	16#000000FD	0	Die maximale Anzahl von 20 aufeinanderfolgenden Quittierungen für den gleichen Slot ist erreicht
DEVICE_ERROR	16#000000FE	0	Nicht unterstütztes Gerät, es ist der Buskoppler S20-PB-BK zu verwenden
STATE_MACHINE_ERROR	16#000000FF	0	Fehler in der State Machine. Wenn dieser Fehler erscheint, hat der Funktionsbaustein nur diesen Fehler generiert

Tab. 16-12: Fehlercodes, die von IH_PBDPAck_PSDx_S20 generiert werden

17 RIH_Safety_ProfinetIO.library

Einführung und Überblick:
RIH_Safety_ProfibusDP, RIH_Safety_ProfinetIO und RIH_Safety_SercosIII

In diesen Bibliotheken werden feldbusspezifische Funktionalitäten zur Nutzung in der Standard-Applikation gebündelt, die im Zusammenhang mit der SafeLogic stehen. An dem Postfix _ProfibusDP, _ProfinetIO und _SercosIII ist die Zuordnung zum Feldbus zu erkennen. Die Bibliotheken stehen im Bibliotheksrepository ab IndraWorks Version 14V16 zur Verfügung. Die von Ihnen benötigte Bibliothek muss zum Bibliotheksverwalter der Standard-Applikation hinzugefügt werden.



Fügen Sie nur die Bibliotheken zum Bibliotheksverwalter hinzu, die Sie tatsächlich benötigen. Das spart Ladezeit des IndraWorks-Projekts.

Die Bibliotheken enthalten Quittierungsfunktionsbausteine für die Safety-EA-Module von Bosch Rexroth, die an der SafeLogic betrieben werden. Die Quittierungsfunktionsbausteine werden in der Standardapplikation zum Auslesen und Quittieren von Diagnosen der Safety-EA-Module von Bosch Rexroth verwendet. In der Regel wird eine Quittierung durch den Anwender benötigt, damit nach Beseitigung der Fehlerursache der Kanal/das Modul den Betrieb wieder aufnehmen kann. Das Quittierungsverfahren ist herstellerspezifisch und richtet sich nach dem Feldbus und dem lokalen Bus der EA-Station.

Hieraus ergeben sich folgende Quittierungsfunktionsbausteine, die in den feldbusspezifischen Bibliotheken zur Verfügung gestellt werden:

Bibliothek für Profibus: RIH_Safety_ProfibusDP

- IH_PBDPAck_PSDx_Inline Quittierungsfunktionsbaustein für Profibus Inline Safety-EA-Module auf Basis PROFIsafe
- IH_PBDPAck_PSDx_S20 Quittierungsfunktionsbaustein für Profibus S20 Safety-EA-Module auf Basis PROFIsafe

Bibliothek für Profinet: RIH_Safety_ProfinetIO

- IH_PNIOAck_PSDx_Inline Quittierungsfunktionsbaustein für Profinet Inline Safety-EA-Module auf Basis PROFIsafe
- IH_PNIOAck_PSDx_S20 Quittierungsfunktionsbaustein für Profinet S20 Safety-EA-Module auf Basis PROFIsafe

Bibliothek für Sercos: RIH_Safety_SercosIII

- IH_SIIIAck SSDx_S20 Quittierungsfunktionsbaustein für Sercos S20 Safety-EA-Module auf Basis Csos

Die Funktionsbausteine verwenden dabei Datentypen der Bibliothek RIH_Safety.

17.1 IH_PNIOAck_PSDx_S20

Kurzbeschreibung

Der Funktionsbaustein **IH_PNIOAck_PSDx_S20** dient zum Auslesen und Quittieren von Diagnosen der S20-PSDx-Module (PROFIsafe-Module) am PROFINET-S20-Buskoppler (S20-PN-BK+).

Der Funktionsbaustein wird mit steigender Flanke am Eingang "Execute" aktiviert und kehrt nach erfolgreicher Ausführung mit TRUE am Ausgang "Done" zurück.

Danach kann der Funktionsbaustein wieder mit steigenden Flanke am Eingang "Execute" aktiviert werden. Wie oft der Funktionsbaustein aktiviert wird, bestimmt der Anwender.

Der Ausgang "Active" zeigt an, ob der Funktionsbaustein in Arbeit ist, z.B. Diagnosen abrufen.

Der Ausgang "Done" wird gesetzt, wenn alle Diagnosen quittiert sind oder es sich um nicht-quittierungspflichtige bzw. Standarddiagnosen handelt. Der Ausgang "Done" wird auch gesetzt, wenn keine Diagnose vorhanden war.



Der Funktionsbaustein IH_PNIOAck_PSDx_S20 benötigt den **PROFINET-S20-Buskoppler S20-PN-BK+** (R911173359).

Zielsystem	Bibliothek /Ordner
MLC	RIH_Safety_ProfinetIO
MTX	RIH_Safety_ProfinetIO

Tab. 17-1: Referenztabelle

Schnittstelle

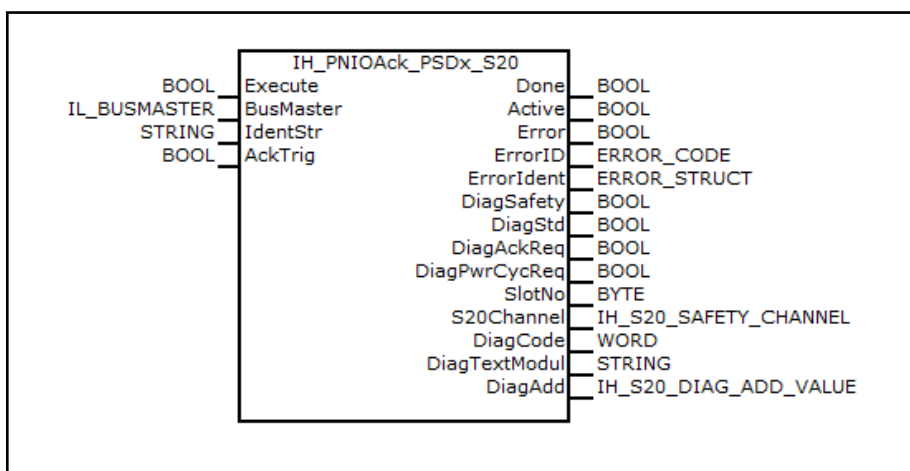


Abb. 17-1: Schnittstellenbeschreibung Funktionsbaustein IH_PNIO-Ack_PSDx_S20

	Name	Typ	Beschreibung
VAR_INPUT	Execute	BOOL	Startet bei steigender Flanke die Funktion
	BusMaster	IL_BUSMASTER	Instanz des PROFINET IO Controllers in der Reihenfolge des Konfigurationsbaums. Ist nur ein PROFINET IO Controller vorhanden, besitzt dieser immer den Wert IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 189
	IdentStr	STRING	Identifikationsstring (Stationsname) des PN-Devices S20-PN-BK+
	AckTrig	BOOL	Quittierungseingang: steigende Flanke notwendig, wenn "DiagAckReq" ansteht
VAR_OUTPUT	Done	BOOL	TRUE zeigt an, dass alle Diagnosen quittiert sind oder dass es sich um nicht-quittierungspflichtige bzw. Standarddiagnosen handelt oder dass keine Diagnose vorhanden war.
	Active	BOOL	TRUE zeigt an, dass der Funktionsbaustein aktiv ist
	Error	BOOL	TRUE zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse

	Name	Typ	Beschreibung
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerinformation Siehe auch Fehlerbehandlung, Seite 731
	DiagSafety	BOOL	TRUE zeigt an, dass es sich um eine Diagnose von einem Safety-Modul handelt (Safety-Diagnose).
	DiagStd	BOOL	TRUE zeigt an, dass es sich um eine Standard-Diagnose handelt.
	DiagAckReq	BOOL	TRUE zeigt an, dass eine quittierungspflichtige Diagnose ansteht und über "AckTrig" bestätigt werden muss.
	DiagPwrCycReq	BOOL	TRUE zeigt an, dass ein Neustart / PowerCycle (Spannung aus- und wieder einschalten) notwendig ist, damit die S20-PSDx-Module durch den Power On Selbsttest laufen. Hinweis: Der PowerCycle ist am Buskoppler inklusive evtl. zusätzlicher Spannungseinspeisungsklemmen durchzuführen.
	SlotNo	BYTE	Slot-Nummer des diagnosemeldenden Moduls
	S20Channel	IH_S20_SAFE-TY_CHANNEL	Kanal für den die Diagnose ansteht. Der Wert 255 bedeutet, dass das ganze Modul betroffen ist. Hinweis: SafetyIoNo, SafetyChannelNo, SafetyIoChannelName sind nur gültig, wenn "DiagSafety" TRUE ist.
	DiagCode	WORD	S20 Diagnosecode
	DiagTextModul	STRING	S20 Diagnosetext direkt vom Modul, max. 50 Zeichen. Default-Sprache ist Englisch.
	DiagAdd	IH_S20_DIAG_ADD_VALUE	S20 zusätzliche Diagnose (Code und Text)

Tab. 17-2: Ein-/Ausgänge Funktionsbaustein IH_PNIOAck_PSDx_S20

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4		steigende Flanke an "Execute"
IdentStr	STRING			"	steigende Flanke an "Execute"
AckTrig	BOOL			FALSE	kontinuierlich wenn "DiagAckReq" ansteht

Tab. 17-3: Min-/Max- und Default-Werte der Bausteineingänge IH_PNIOAck_PSDx_S20

Typen und Konstanten

Die Typen und Konstanten ermöglichen es detaillierte Diagnoseinformationen zu erhalten und Diagnosetexte für den Anwender zusammenzubauen.

Die Struktur "IH_S20_DIAG_ADD_VALUE" beinhaltet den zusätzlichen Diagnosecode und ausgewählte Texte hierfür.

Name	Typ	Kommentar
Value	DWORD	S20 Additional Diagnosis Value
Text	STRING	Text For The Additional Diagnosis Value

Tab. 17-4: "IH_S20_DIAG_ADD_VALUE"

Die Struktur "IH_S20_SAFETY_CHANNEL" beinhaltet den Kanal und die Zuordnung zum Eingang.

Name	Typ	Wertebereich
S20ChannelNo	BYTE	0 - 7, 255 (255: whole module is affected)
SafetyIoNo	BYTE	0 - 3, 255 (255: whole module is affected)
SafetyChannelNo	BYTE	1, 2, 255 (255: whole module is affected)
SafetyIoChannelName	STRING	'OUTx_CHy', 'INx_CHy', "

Tab. 17-5: "IH_S20_SAFETY_CHANNEL"

Die Struktur "IH_SAFETY_IO_DIAG_TEXT" beinhaltet Diagnosetext und Diagnosecode.

Name	Typ	Kommentar
DiagCode	WORD	Diagnosis Code
Text	STRING(255)	Diagnosis Text

Tab. 17-6: "IH_SAFETY_IO_DIAG_TEXT"

Die Konstante "IH_DIAGCODE_MISSING_MODULE" beinhaltet den Diagnosecode für den PROFINET-Fehler "Modul nicht vorhanden".

Die Konstante "IH_DIAGCODE_WRONG_MODULE" beinhaltet den Diagnosecode für den PROFINET-Fehler "Falsches Modul".

Die Konstante "SafetyDiagTexts" beinhaltet 2 Arrays vom Typ "IH_SAFETY_IO_DIAG_TEXT" mit Diagnosetexten für die S20-PSDx-Module:

- Deutsch: arS20PnSafetyDiagTexts_De
- Englisch: arS20PnSafetyDiagTexts_En

Die Konstante "IH_PNIO_S20_SAFETY_DIAG_TEXT_AMOUNT" gibt die Anzahl der Diagnosetexte an.

Sie können in "arS20PnSafetyDiagTexts_xx" nach dem "DiagCode" suchen und so den zugehörigen Diagnosetext ermitteln.

```

//Get DE diagnosis text from global text array
IF FB_PNIOAck_PSDx_S20.DiagCode <> 0 THEN
  FOR i:=1 TO IH_PNIO_S20_SAFETY_DIAG_TEXT_AMOUNT DO
    IF arS20PnSafetyDiagTexts_De[i].DiagCode = FB_PNIOAck_PSDx_S20.DiagCode THEN
      szStringS20 := concat (szStringS20 , arS20PnSafetyDiagTexts_De[i].Text);
      bTextFoundS20 := TRUE;
      EXIT; //Text found
    END_IF
  END_FOR
END_IF
IF bTextFoundS20 = FALSE THEN
  // If no text found use text from modul
  szStringS20 := concat (szStringS20 , FB_PNIOAck_PSDx_S20.DiagTextModul);
END_IF

```

Abb. 17-2: Beispiel zum Ermitteln der Diagnosetexte aus "arS20PnSafetyDiagTexts_xx"

Funktionsbeschreibung

Der Funktionsbaustein "IH_PNIOAck_PSDx_S20" ermittelt und quittiert Diagnosen der S20-PSDx-Module (PROFIsafe Module) am PROFINET-S20-Buskoppler (S20-PN-BK+).

Der Funktionsbaustein "IH_PNIOAck_PSDx_S20" basiert auf dem Basisfunktionsbaustein "IL_ExecuteDoneBase".

In dem Funktionsbaustein werden verschiedene Funktionsbausteine gebündelt: [131 auf Seite 131](#), [73 auf Seite 73](#) und [81 auf Seite 81](#).

Der Funktionsbaustein „IH_PNIOAck_PSDx_S20“ ist unter Angabe von "Stationsname" und "PNIO-Controller" des PROFINET-S20-Buskopplers zu verwenden. Der Funktionsbaustein behandelt alle S20-PSDx-Module, die sich an dem PROFINET-S20-Buskoppler befinden.



Um die Kommunikationslast auf dem PROFINET gering zu halten, verwenden Sie möglichst wenige Funktionsbausteine "IH_PNIOAck_PSDx_S20" gleichzeitig.

Haben mehrere Kanäle oder mehrere Module quittierungspflichtige Fehler, werden diese im Funktionsbaustein nacheinander behandelt. Die Diagnosen müssen einzeln quittiert werden. D.h. der Diagnosecode wird ausgegeben und es wird dann eine separate Flanke am Eingang "AckTrig" benötigt. Der Zustand TRUE am Ausgang "Done" und der Zustand FALSE am Ausgang "DiagAckReq" zeigen am Ende an, dass die Diagnosen quittiert wurden.

Es wird die Quittierung vom Diagnosen für folgende PROFIsafe S20-PSDx-Module unterstützt:

- S20-PSDI-8/4
- S20-PSDO-8/3

Bei den quittierungspflichtigen Safety-Diagnosen handelt es sich meist um kanalspezifische Diagnosen wie z.B. "Symmetrieverletzung am Eingang x Kanal y". Nach Beseitigung der Fehlerursache müssen diese Diagnosen quittiert werden, damit der Kanal / das Modul den Betrieb wieder aufnehmen kann.

⚠ WARNUNG

Es ist die Dokumentation der S20-PSDx-Module zu beachten

Die Bedeutung der gerätespezifischen Diagnosen und Fehlercodes des PROFINET-S20-Buskopplers (S20-PN-BK+) und der PROFIsafe-S20-Module entnehmen Sie der entsprechenden Dokumentation dieser Geräte. Dort finden Sie Hinweise zu Fehlerursachen sowie Anweisungen zur Fehlerbeseitigung und Quittierung.

⚠ WARNUNG

Quittierung kann zum gefährlichen Zustand führen

Das Quittieren eines Fehlers führt bis auf die angegebenen Ausnahmen sofort zur Rückkehr des sicheren Ausgangs / sicheren Eingangs in den Betriebszustand. Stellen Sie deshalb vor der Quittierung eines Fehlers sicher, dass die Quittierung nicht zum gefährlichen Zustand der Maschine führen kann! Berücksichtigen Sie bei der Planung der Maschine oder Anlage, dass das Quittieren nur dann möglich sein darf, wenn der Gefahrenbereich einsehbar ist.

Der Prozess des Auslesens und der Quittierung von Safety-Diagnosen wird mit steigender Flanke am Eingang "Execute" initiiert.

Am Anfang wird einmalig die Soll-Konfiguration am S20-PN-BK+ ausgelesen. Damit ist bekannt, auf welchen Slots sich S20-PSDx-Module befinden. Ändern sich während der Laufzeit des FBs die Eingangsparameter BusMaster oder IdentStr (Stationsname) wird die Soll-Konfiguration am S20-S3-BK+ erneut ausgelesen.

Zuerst wird der FB [131 auf Seite 131](#) ausgeführt, um zu erkennen, ob am Buskoppler bzw. seinen Modulen Diagnosen anstehen.

Der Funktionsblock sucht nach dem ersten S20-PSDx-Modul, welches eine Diagnose hat. (Die Suche beginnt bei der niedrigsten Slot-Nummer.)

Wenn die Safety-Diagnose erfolgreich quittiert wurde, wird die nächste Diagnose von diesem Modul behandelt (meist von einem anderen Kanal).

Gibt es keine Diagnosen mehr an diesem Modul, dann wird das nächste S20-PSDx-Modul gesucht, welches Diagnosen hat.

Gibt es keine S20-PSDx-Module mit Diagnosen, wird das erste Standard-Module behandelt, welches Diagnosen hat.

Auf dieser Basis werden die Ausgänge "DiagSafety", "DiagStd", "DiagAckReq", "DiagPwrCycReq", "SlotNo", "S20Channel" und "DiagCode" gesetzt.

Der Ausgang "DiagAckReq" zeigt an, dass eine quittierungspflichtige Safety-Diagnose ansteht und diese Diagnose mit einer positiven Flanke an dem Eingang "AckTrig" nach Beseitigung der Fehlerursache quittiert werden muss. Zum Quittieren wird intern das "ResetDiag-Objekt 0x0019" verwendet.

Sind alle quittierungspflichtigen Diagnosen abgearbeitet, wird der Ausgang "Done" auf TRUE gesetzt.

Diagnosen

Der Ausgang "DiagCode" gibt den Zustand des PROFINET-S20-Buskopplers und der S20-Modulen an.

Der Ausgang "SlotNo" zeigt den diagnosemeldenden Slot bzw. die S20-Modul Nummer an.

Der Ausgang "S20Channel" zeigt den diagnosemeldenden Kanal an. Der Wert 255 bedeutet, dass das ganze Modul betroffen ist. Basis hierfür ist das

Diagnoseobjekt "DiagState-Objekt 0x0018, Index 3: Channel / Group / Module".

Der Ausgang "DiagTextModul" gibt den Diagnosetext direkt vom Modul aus. Das entspricht Diagnoseobjekt "DiagState-Objekt 0x0018, Index 5: Text".

Der Ausgang "DiagAdd" gibt zusätzliche Diagnose (Code und Text) aus. Hierfür wird das Diagnoseobjekt "DiagStateAddValue-Objekt 0x0034" ausgelesen.

Fehlerbehandlung Der Funktionsbaustein "IH_PNIOAck_PSDx_S20" reicht die Fehlercodes von den intern verwendeten Funktionsbausteinen am eigenen Error-Ausgang weiter. Siehe Tabelle **PROFINETIO_TABLE (16#0200)**.

ErrorID	Additional1	Additional2	Beschreibung
DEVICE_ERROR	16#00000003	16#C00C0056	Dienst nicht verfügbar; Device ist nicht im IO-Datenaustausch.

Tab. 17-7: *Beispiel Fehlercodes von den intern verwendeten Funktionsbausteinen*

Der Funktionsbaustein IH_PNIOAck_PSDx_S20 erzeugt folgende Fehlercodes:

ErrorID	Additional1	Additional2	Beschreibung
DEVICE_ERROR	16#000000FD	0	Die maximale Anzahl von 20 aufeinanderfolgenden Quittierungen für den gleichen Slot ist erreicht.
DEVICE_ERROR	16#000000FE	0	Nicht unterstütztes Gerät, es ist der Buskoppler S20-PN-BK+ zu verwenden.
STATE_MACHINE_ERROR	16#000000FF	0	Fehler in der State Machine, wenn dieser Fehler erscheint, hat der Funktionsbaustein nichts weiter getan als diesen Fehler zu generieren.

Tab. 17-8: *Fehlercodes, die von IH_PNIOAck_PSDx_S20 generiert werden*

17.2 IH_PNIOAck_PSDx_Inline

Kurzbeschreibung Der Funktionsbaustein **IH_PNIOAck_PSDx_Inline** dient zum Auslesen und Quittieren von Diagnosen der PROFIsafe INLINE-Module (PSDx) am PROFINET-INLINE-Buskoppler (R-IL PN BK DI8 DO4).

Der Funktionsbaustein wird mit steigender Flanke am Eingang "Execute" aktiviert und kehrt nach erfolgreicher Ausführung mit TRUE am Ausgang "Done" zurück.

Danach kann der Funktionsbaustein wieder mit steigenden Flanke am Eingang "Execute" aktiviert werden. Wie oft der Funktionsbaustein aktiviert wird, bestimmt der Anwender.

Der Ausgang "Active" zeigt an, ob der Funktionsbaustein in Arbeit ist, z.B. Diagnosen abrufen.

Der Ausgang "Done" wird gesetzt, wenn alle Diagnosen quittiert sind oder es sich um nicht-quittierungspflichtige bzw. Standarddiagnosen handelt. Der Ausgang "Done" wird auch gesetzt, wenn keine Diagnose vorhanden war.



Der Funktionsbaustein IH_PNIOAck_PSDx_Inline benötigt den **PROFINET-INLINE-Buskoppler R-IL PN BK DI8 DO4** mit Änderungsindex AD1 (FWA-Version 3.41) oder höher.

Zielsystem	Bibliothek /Ordner
MLC	RIH_Safety_ProfinetIO
MTX	RIH_Safety_ProfinetIO

Tab. 17-9: Referenztable

Schnittstelle

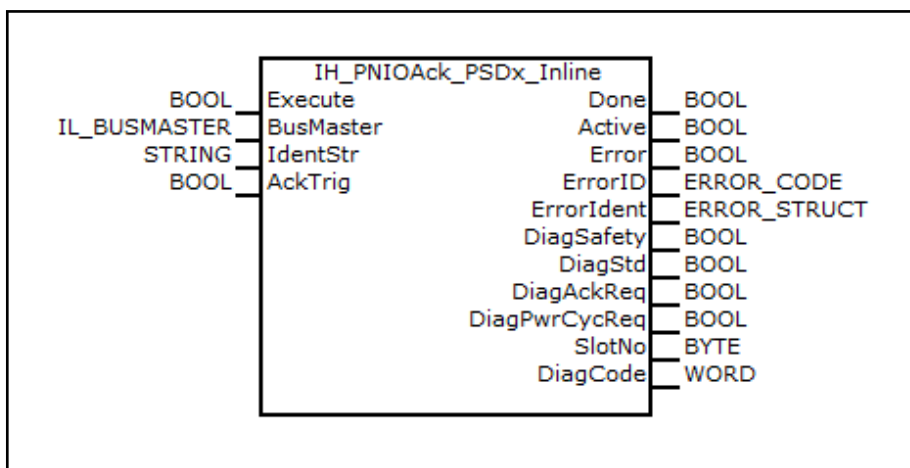


Abb. 17-3: Schnittstellenbeschreibung Funktionsbaustein IH_PNIO-Ack_PSDx_Inline

	Name	Typ	Beschreibung
VAR_INPUT	Execute	BOOL	Startet bei steigender Flanke die Funktion
	BusMaster	IL_BUSMASTER	Instanz des PROFINET IO Controllers in der Reihenfolge des Konfigurationsbaums. Ist nur ein PROFINET IO Controller vorhanden, besitzt dieser immer den Wert IL_BUSMASTER_0. siehe auch IL_BUSMASTER auf Seite 189
	IdentStr	STRING	Identifikationsstring (Stationsname) des PN-Devices R-IL PN BK DI8 DO4
	AckTrig	BOOL	Quittierungseingang: steigende Flanke notwendig, wenn "DiagAckReq" ansteht
VAR_OUTPUT	Done	BOOL	TRUE zeigt an, dass alle Diagnosen quittiert sind oder dass es sich um nicht-quittierungspflichtige bzw. Standarddiagnosen handelt oder dass keine Diagnose vorhanden war.
	Active	BOOL	TRUE zeigt an, dass der Funktionsbaustein aktiv ist
	Error	BOOL	TRUE zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerinformation Siehe auch Fehlerbehandlung, Seite 735
	DiagSafety	BOOL	TRUE zeigt an, dass es sich um eine Diagnose von einem Safety-Modul handelt (Safety-Diagnose).
	DiagStd	BOOL	TRUE zeigt an, dass es sich um eine Standard-Diagnose handelt.

	Name	Typ	Beschreibung
	DiagAckReq	BOOL	TRUE zeigt an, dass eine quittierungspflichtige Diagnose ansteht und über "AckTrig" bestätigt werden muss.
	DiagPwrCycReq	BOOL	TRUE zeigt an, dass ein Neustart / PowerCycle (Spannung aus- und wieder einschalten) notwendig ist, damit die PROFIsafe INLINE-Module durch den Power On-Selbsttest laufen. Hinweis: Der PowerCycle ist am Buskoppler inklusive evtl. zusätzlicher Spannungseinspeisungsklemmen durchzuführen.
	SlotNo	BYTE	Slot-Nummer des diagnosemeldenden Moduls
	DiagCode	WORD	Diagnose-Code

Tab. 17-10: Ein-/Ausgänge Funktionsbaustein IH_PNIOAck_PSDx_Inline

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4		steigende Flanke an "Execute"
IdentStr	STRING			"	steigende Flanke an "Execute"
AckTrig	BOOL			FALSE	kontinuierlich wenn "DiagAckReq" ansteht

Tab. 17-11: Min-/Max- und Default-Werte der Bausteineingänge IH_PNIOAck_PSDx_Inline

Typen und Konstanten

Die Typen und Konstanten ermöglichen es detaillierte Diagnoseinformationen zu erhalten und Diagnosetexte für den Anwender zusammenzubauen.

Die Konstante "IH_DIAGCODE_MISSING_MODULE" beinhaltet den Diagnosecode für den PROFINET-Fehler "Modul nicht vorhanden".

Die Konstante "IH_DIAGCODE_WRONG_MODULE" beinhaltet den Diagnosecode für den PROFINET-Fehler "Falsches Modul".

Die Struktur "IH_SAFETY_IO_DIAG_TEXT" beinhaltet Diagnosetext und Diagnosecode.

Name	Typ	Kommentar
DiagCode	WORD	Diagnosis Code
Text	STRING(255)	Diagnosis Text

Tab. 17-12: "IH_SAFETY_IO_DIAG_TEXT"

Die Konstante "SafetyDiagTexts" beinhaltet 2 Arrays vom Typ "IH_SAFETY_IO_DIAG_TEXT" mit Diagnosetexten für die INLINE-PSDx-Module:

- Deutsch: arInlinePnSafetyDiagTexts_De
- Englisch: arInlinePnSafetyDiagTexts_En

Die Konstante "IH_PNIO_INLINE_SAFETY_DIAG_TEXT_AMOUNT" gibt die Anzahl der Diagnosetexte an.

Sie können in "arInlinePnSafetyDiagTexts_xx" nach dem "DiagCode" suchen und so den zugehörigen Diagnosetext ermitteln.

```

//Get DE texts from global text array
IF FB_PNIOAck_PSDx_Inline.DiagCode <> 0 THEN
  FOR i:=1 TO IH_PNIO_INLINE_SAFETY_DIAG_TEXT_AMOUNT DO
    IF arInlinePnSafetyDiagTexts_De[i].DiagCode = FB_PNIOAck_PSDx_Inline.DiagCode THEN
      szString := concat (szString , arInlinePnSafetyDiagTexts_De[i].Text);
      EXIT; //Text found
    END_IF
  END_FOR
END_IF

```

Abb. 17-4: Beispiel zum Ermitteln der Diagnosetexte aus "arInlinePnSafetyDiagTexts_xx"

Funktionsbeschreibung

Der Funktionsbaustein "IH_PNIOAck_PSDx_Inline" ermittelt und quittiert Diagnosen der PROFIsafe INLINE-Module (PROFIsafe Module) am PROFINET-INLINE-Buskoppler (R-IL PN BK DI8 DO4).

Der Funktionsbaustein "IH_PNIOAck_PSDx_Inline" basiert auf dem Basisfunktionsbaustein "IL_ExecuteDoneBase".

In dem Funktionsbaustein werden verschiedene Funktionsbausteine gebündelt: [131 auf Seite 131](#), [73 auf Seite 73](#) und [81 auf Seite 81](#).

Der Funktionsbaustein „IH_PNIOAck_PSDx_Inline“ ist unter Angabe von "Stationsname" und "PNIO-Controller" des PROFINET-INLINE-Buskopplers zu verwenden. Der Funktionsbaustein behandelt alle PROFIsafe INLINE-Module, die sich an dem PROFINET-INLINE-Buskoppler befinden.



Um die Kommunikationslast auf dem PROFINET gering zu halten, verwenden Sie möglichst wenige Funktionsbausteine "IH_PNIOAck_PSDx_Inline" gleichzeitig.

Haben mehrere Kanäle oder mehrere Module quittierungspflichtige Fehler, werden diese im Funktionsbaustein nacheinander behandelt. Die Diagnosen müssen einzeln quittiert werden. D.h. der Diagnosecode wird ausgegeben und es wird dann eine separate Flanke am Eingang "AckTrig" benötigt. Der Zustand TRUE am Ausgang "Done" und der Zustand FALSE am Ausgang "DiagAckReq" zeigen am Ende an, dass die Diagnosen quittiert wurden.

Es wird die Quittierung vom Diagnosen für folgende PROFIsafe INLINE-Module unterstützt:

- R-IB IL 24 PSDI 16
- R-IB IL 24 PSDI 8
- R-IB IL 24 PSDO 8
- R-IB IL 24 PSDO 4/4
- R-IB IL 24 PSDOR 4

Bei den quittierungspflichtigen Safety-Diagnosen handelt es sich meist um kanalspezifische Diagnosen wie z.B. "Symmetrieverletzung am Eingang x Kanal y". Nach Beseitigung der Fehlerursache müssen diese Diagnosen quittiert werden, damit der Kanal / das Modul den Betrieb wieder aufnehmen kann.

⚠ WARNUNG**Es ist die Dokumentation der PROFIsafe IN-LINE-Module zu beachten**

Die Bedeutung der gerätespezifischen Diagnosen und Fehlercodes des PROFINET-INLINE-Buskopplers (R-IL PN BK D18 DO4) und der PROFIsafe IN-LINE-Module entnehmen Sie der entsprechenden Dokumentation dieser Geräte. Dort finden Sie Hinweise zu Fehlerursachen sowie Anweisungen zur Fehlerbeseitigung und Quittierung.

⚠ WARNUNG**Quittierung kann zum gefährlichen Zustand führen**

Das Quittieren eines Fehlers führt bis auf die angegebenen Ausnahmen sofort zur Rückkehr des sicheren Ausgangs / sicheren Eingangs in den Betriebszustand. Stellen Sie deshalb vor der Quittierung eines Fehlers sicher, dass die Quittierung nicht zum gefährlichen Zustand der Maschine führen kann! Berücksichtigen Sie bei der Planung der Maschine oder Anlage, dass das Quittieren nur dann möglich sein darf, wenn der Gefahrenbereich einsehbar ist.

Der Prozess des Auslesens und der Quittierung von Safety-Diagnosen wird mit steigender Flanke am Eingang "Execute" initiiert.

Am Anfang wird einmalig die Soll-Konfiguration am R-IL PN BK D18 DO4 ausgelesen. Damit ist bekannt, auf welchen Slots sich PROFIsafe IN-LINE-Module befinden. Ändern sich während der Laufzeit des FBs die Eingangsparameter BusMaster oder IdentStr (Stationsname) wird die Soll-Konfiguration am R-IL PN BK D18 DO4 erneut ausgelesen.

Zuerst wird der FB [131 auf Seite 131](#) ausgeführt, um zu erkennen, ob am Buskoppler bzw. seinen Modulen Diagnosen anstehen.

Der Funktionsblock sucht nach dem ersten IN-LINE-PSDx-Modul, welches eine Diagnose hat. (Die Suche beginnt bei der niedrigsten Slot-Nummer.)

Wenn die Safety-Diagnose erfolgreich quittiert wurde, wird die nächste Diagnose von diesem Modul behandelt (meist von einem anderen Kanal).

Gibt es keine Diagnosen mehr an diesem Modul, dann wird das nächste IN-LINE-PSDx-Modul gesucht, welches Diagnosen hat.

Gibt es keine PROFIsafe IN-LINE-Module mit Diagnosen, wird das erste Standard-Module behandelt, welches Diagnosen hat.

Auf dieser Basis werden die Ausgänge "DiagSafety", "DiagStd", "DiagAckReq", "DiagPwrCycReq", "SlotNo" und "DiagCode" gesetzt.

Der Ausgang "DiagAckReq" zeigt an, dass eine quittierungspflichtige Safety-Diagnose ansteht und diese Diagnose mit einer positiven Flanke an dem Eingang "AckTrig" nach Beseitigung der Fehlerursache quittiert werden muss.

Sind alle quittierungspflichtigen Diagnosen abgearbeitet, wird der Ausgang "Done" auf TRUE gesetzt.

Diagnosen

Der Ausgang "DiagCode" gibt den Zustand des PROFINET-IN-LINE-Buskopplers und der IN-LINE-Modulen an.

Der Ausgang "SlotNo" zeigt den diagnosemeldenden Slot bzw. die IN-LINE-Modul Nummer an.

Fehlerbehandlung

Der Funktionsbaustein "IH_PNIOAck_PSDx_Inline" reicht die Fehlercodes von den intern verwendeten Funktionsbausteinen am eigenen Error-Ausgang weiter. Siehe Tabelle **PROFINETIO_TABLE (16#0200)**.

ErrorID	Additional1	Additional2	Beschreibung
DEVICE_ERROR	16#00000003	16#C00C0056	Dienst nicht verfügbar; Device ist nicht im IO-Datenaustausch.

Tab. 17-13: *Beispiel Fehlercodes von den intern verwendeten Funktionsbausteinen*

Der Funktionsbaustein "IH_PNIOAck_PSDx_Inline" erzeugt folgende Fehlercodes:

ErrorID	Additional1	Additional2	Beschreibung
DEVICE_ERROR	16#000000FD	0	Die maximale Anzahl von 20 aufeinanderfolgenden Quitierungen für den gleichen Slot ist erreicht.
DEVICE_ERROR	16#000000FE	0	Nicht unterstütztes FWA-Version des Geräts, es ist der Buskoppler R-IL PN BK DI8 DO4 mit FW-Version 3.41 oder höher zu verwenden.
STATE_MACHINE_ERROR	16#000000FF	0	Fehler in der State Machine, wenn dieser Fehler erscheint, hat der Funktionsbaustein nichts weiter getan als diesen Fehler zu generieren.

Tab. 17-14: *Fehlercodes, die von "IH_PNIOAck_PSDx_Inline" generiert werden*

18 RIH_Safety_SercosIII.library

Einführung und Überblick:
 RIH_Safety_ProfibusDP, RIH_Safety_ProfinetIO und RIH_Safety_SercosIII

In diesen Bibliotheken werden feldbusspezifische Funktionalitäten zur Nutzung in der Standard-Applikation gebündelt, die im Zusammenhang mit der SafeLogic stehen. An dem Postfix _ProfibusDP, _ProfinetIO und _SercosIII ist die Zuordnung zum Feldbus zu erkennen. Die Bibliotheken stehen im Bibliotheksrepository ab IndraWorks Version 14V16 zur Verfügung. Die von Ihnen benötigte Bibliothek muss zum Bibliotheksverwalter der Standard-Applikation hinzugefügt werden.



Fügen Sie nur die Bibliotheken zum Bibliotheksverwalter hinzu, die Sie tatsächlich benötigen. Das spart Ladezeit des IndraWorks-Projekts.

Die Bibliotheken enthalten Quittierungsfunktionsbausteine für die Safety-EA-Module von Bosch Rexroth, die an der SafeLogic betrieben werden. Die Quittierungsfunktionsbausteine werden in der Standardapplikation zum Auslesen und Quittieren von Diagnosen der Safety-EA-Module von Bosch Rexroth verwendet. In der Regel wird eine Quittierung durch den Anwender benötigt, damit nach Beseitigung der Fehlerursache der Kanal/das Modul den Betrieb wieder aufnehmen kann. Das Quittierungsverfahren ist herstellerspezifisch und richtet sich nach dem Feldbus und dem lokalen Bus der EA-Station.

Hieraus ergeben sich folgende Quittierungsfunktionsbausteine, die in den feldbusspezifischen Bibliotheken zur Verfügung gestellt werden:

Bibliothek für Profibus: RIH_Safety_ProfibusDP

- IH_PBDPAck_PSDx_Inline Quittierungsfunktionsbaustein für Profibus Inline Safety-EA-Module auf Basis PROFIsafe
- IH_PBDPAck_PSDx_S20 Quittierungsfunktionsbaustein für Profibus S20 Safety-EA-Module auf Basis PROFIsafe

Bibliothek für Profinet: RIH_Safety_ProfinetIO

- IH_PNIOAck_PSDx_Inline Quittierungsfunktionsbaustein für Profinet Inline Safety-EA-Module auf Basis PROFIsafe
- IH_PNIOAck_PSDx_S20 Quittierungsfunktionsbaustein für Profinet S20 Safety-EA-Module auf Basis PROFIsafe

Bibliothek für Sercos: RIH_Safety_SercosIII

- IH_SIIIAck_SSDx_S20 Quittierungsfunktionsbaustein für Sercos S20 Safety-EA-Module auf Basis Csos

Die Funktionsbausteine verwenden dabei Datentypen der Bibliothek RIH_Safety.

18.1 IH_SIIIAck_SSDx_S20

Kurzbeschreibung

Der Funktionsbaustein **IH_SIIIAck_SSDx_S20** dient zum Auslesen und Quittieren von Diagnosen der S20-SSDx-Module (Csos-Module) am Sercos-S20-Buskoppler (S20-S3-BK+). Csos steht für "CIP Safety on Sercos".

Der Funktionsbaustein wird mit steigender Flanke am Eingang "Execute" aktiviert und kehrt nach erfolgreicher Ausführung mit TRUE am Ausgang "Done" zurück.

Danach kann der Funktionsbaustein wieder mit steigenden Flanke am Eingang "Execute" aktiviert werden. Wie oft der Funktionsbaustein aktiviert wird, bestimmt der Anwender.

Der Ausgang "Active" zeigt an, ob der Funktionsbaustein in Arbeit ist, z.B. Diagnosen abrufen.

Der Ausgang "Done" wird gesetzt, wenn die Diagnose quittiert ist oder es sich um eine nicht-quittierungspflichtige bzw. Standarddiagnose handelt. Der Ausgang "Done" wird auch gesetzt, wenn keine Diagnose vorhanden war.



Der Funktionsbaustein IH_SIIIAck_SSDx_S20 benötigt den Sercos-S20-Buskoppler S20-S3-BK+ (R911173318).

Zielsystem	Bibliothek /Ordner
MLC	RIH_Safety_SercosIII
MTX	RIH_Safety_SercosIII

Tab. 18-1: Referenztablelle

Schnittstelle

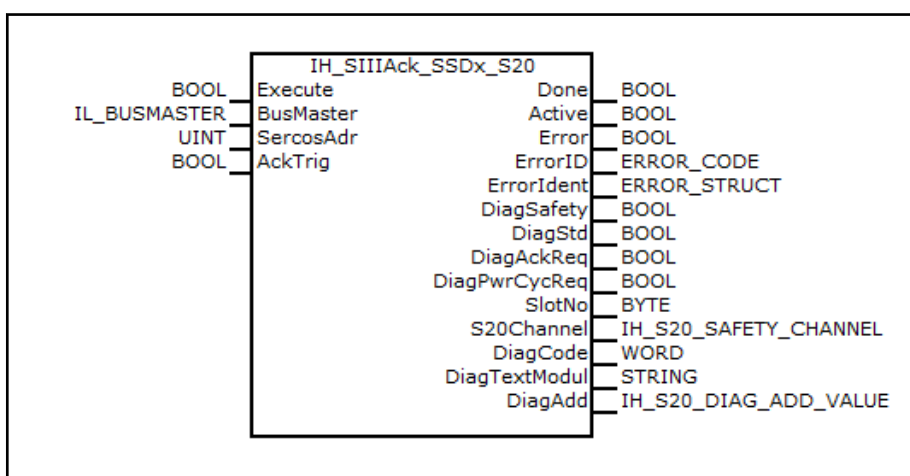


Abb. 18-1: Schnittstellenbeschreibung Funktionsbaustein IH_SIII-Ack_SSDx_S20

	Name	Typ	Beschreibung
VAR_INPUT	Execute	BOOL	Startet bei steigender Flanke die Funktion
	BusMaster	IL_BUSMASTER	Master-ID. Wenn auf der Steuerung mehrere Master gleichzeitig betrieben werden, werden diese durch die Eingangsvariable "BusMaster" unterschieden. Wird nur ein Master betrieben, so hat dieser die Master-ID="IL_BUSMASTER_0". siehe auch IL_BUSMASTER auf Seite 429
	SercosAdr	UINT	Sercos III-Adresse des Slaves S20-S3-BK+
	AckTrig	BOOL	Quittierungseingang: steigende Flanke notwendig, wenn "DiagAckReq" ansteht
VAR_OUTPUT	Done	BOOL	TRUE zeigt an, dass die Diagnose quittiert wurde, oder dass es sich um eine nicht-quittierungspflichtige bzw. Standarddiagnose handelt, oder dass keine Diagnose vorhanden war.
	Active	BOOL	TRUE zeigt an, dass der Funktionsbaustein aktiv ist
	Error	BOOL	TRUE zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse

	Name	Typ	Beschreibung
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerinformation Siehe auch Fehlerbehandlung, Seite 742
	DiagSafety	BOOL	TRUE zeigt an, dass es sich um eine Diagnose von einem Safety-Modul handelt (Safety-Diagnose).
	DiagStd	BOOL	TRUE zeigt an, dass es sich um eine Standard-Diagnose handelt.
	DiagAckReq	BOOL	TRUE zeigt an, dass eine quittierungspflichtige Diagnose ansteht und über "AckTrig" bestätigt werden muss.
	DiagPwrCycReq	BOOL	TRUE zeigt an, dass ein Neustart / PowerCycle (Spannung aus- und wieder einschalten) notwendig ist, damit die S20-SSDx-Module durch den Power On Selbsttest laufen. Hinweis: Der PowerCycle ist am Buskoppler inklusive evtl. zusätzlicher Spannungseinspeisungsklemmen durchzuführen.
	SlotNo	BYTE	Slot-Nummer des diagnosemeldenden Moduls
	S20Channel	IH_S20_SAFE-TY_CHANNEL	Kanal für den die Diagnose ansteht. Der Wert 255 bedeutet, dass das ganze Modul betroffen ist. Hinweis: SafetyIoNo, SafetyChannelNo, SafetyIoChannel-Name sind nur gültig, wenn "DiagSafety" TRUE ist.
	DiagCode	WORD	S20 Diagnosecode
	DiagTextModul	STRING	Der Diagnosetext wird über den Parameter S-0-0095 am S20-S3-BK+ ausgelesen. Default-Sprache ist Englisch.
	DiagAdd	IH_S20_DIAG_ADD_VALUE	S20 zusätzliche Diagnose (Code und Text)

Tab. 18-2: Ein-/Ausgänge Funktionsbaustein IH_SIII-Ack_SSDx_S20

Name	Typ	Min-Wert	Max-Wert	Default-Wert	Übernahme
Execute	BOOL			FALSE	kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	steigende Flanke an "Execute"
SercosAdr	UINT	1	511	0	steigende Flanke an "Execute"
AckTrig	BOOL			FALSE	kontinuierlich wenn "DiagAckReq" ansteht

Tab. 18-3: Min-/Max- und Default-Werte der Bausteineingänge IH_SIII-Ack_SSDx_S20

Typen und Konstanten

Die Typen und Konstanten ermöglichen es, detaillierte Diagnoseinformationen zu erhalten und Diagnosetexte für den Anwender zusammenzubauen.

Die Struktur "IH_S20_DIAG_ADD_VALUE" beinhaltet den zusätzlichen Diagnosecode und ausgewählte Texte hierfür.

Name	Typ	Kommentar
Value	DWORD	S20 Additional Diagnosis Value
Text	STRING	Text For The Additional Diagnosis Value

Tab. 18-4: "IH_S20_DIAG_ADD_VALUE"

Die Struktur "IH_S20_SAFETY_CHANNEL" beinhaltet den Kanal und die Zuordnung zum Eingang.

Name	Typ	Wertebereich
S20ChannelNo	BYTE	0 - 7, 255 (255: whole module is affected)
SafetyIoNo	BYTE	0 - 3, 255 (255: whole module is affected)
SafetyChannelNo	BYTE	1, 2, 255 (255: whole module is affected)
SafetyIoChannelName	STRING	'OUTx_CHy', 'INx_CHy', "

Tab. 18-5: "IH_S20_SAFETY_CHANNEL"

Die Struktur "IH_SAFETY_IO_DIAG_TEXT" beinhaltet Diagnosetext und Diagnosecode.

Name	Typ	Kommentar
DiagCode	WORD	Diagnosis Code
Text	STRING(255)	Diagnosis Text

Tab. 18-6: "IH_SAFETY_IO_DIAG_TEXT"

Die Konstante "SafetyDiagTexts" beinhaltet 2 Arrays vom Typ "IH_SAFETY_IO_DIAG_TEXT" mit Diagnosetexten für die S20-SSDx-Module:

- Deutsch: arS20S3SafetyDiagTexts_De
- Englisch: arS20S3SafetyDiagTexts_En

Die Konstante "IH_SIII_S20_SAFETY_DIAG_TEXT_AMOUNT" gibt die Anzahl der Diagnosetexte an.

Sie können in "arS20S3SafetyDiagTexts_xx" nach dem "DiagCode" suchen und so den zugehörigen Diagnosetext ermitteln.

```
// Get DE diagnosis text from global text array
IF FB_SIIIAck_SSDx_S20.DiagCode <> 0 THEN
  FOR i:=1 TO IH_SIII_S20_SAFETY_DIAG_TEXT_AMOUNT DO
    IF arS20S3SafetyDiagTexts_De[i].DiagCode = FB_SIIIAck_SSDx_S20.DiagCode THEN
      szStringS20 := concat (szStringS20 , arS20S3SafetyDiagTexts_De[i].Text);
      bTextFoundS20 := TRUE;
      EXIT; // Text found
    END_IF
  END_FOR
END_IF
IF bTextFoundS20 = FALSE THEN
  // If no text found use text from modul
  szStringS20 := concat (szStringS20 , FB_SIIIAck_SSDx_S20.DiagTextModul);
END_IF
```

Abb. 18-2: Beispiel zum Ermitteln der Diagnosetexte aus "arS20S3SafetyDiagTexts_xx"

Funktionsbeschreibung Der Funktionsbaustein "IH_SIIIAck_SSDx_S20" ermittelt und quittiert Diagnosen der S20-SSDx-Module (Csos Module) am Sercos-S20-Buskoppler (S20-S3-BK+).

Der Funktionsbaustein "IH_SIIIAck_SSDx_S20" basiert auf dem Basisfunktionsbaustein "IL_ExecuteDoneBase".

In dem Funktionsbaustein werden verschiedene Funktionsbausteine gebündelt: IL_SIIIRemoteSlaveState, IL_SIIIRemoteSlaveStateDetails, IL_SIIISvcRead, IL_SIIISvcWrite.

Der Funktionsbaustein "IH_SIIIAck_SSDx_S20" ist unter Angabe der Sercos-Adresse und des Sercos-Masters des Sercos-S20-Buskopplers zu verwenden. Der Funktionsbaustein behandelt alle S20-SSDx-Module, die sich an dem Sercos-S20-Buskoppler befinden.



Um die Kommunikationslast auf dem Sercos gering zu halten, verwenden Sie möglichst wenige Funktionsbausteine "IH_SIIIAck_PSDx_S20" gleichzeitig.

Haben mehrere Kanäle oder mehrere Module quittierungspflichtige Fehler, werden diese im Funktionsbaustein nacheinander behandelt. Die Diagnosen müssen einzeln quittiert werden. D.h. der Diagnosecode wird ausgegeben und es wird dann eine separate Flanke am Eingang "AckTrig" benötigt. Der Zustand TRUE am Ausgang "Done" und der Zustand FALSE an den Ausgängen "DiagAckReq" und "DiagStd" zeigen am Ende an, dass alle Diagnosen quittiert wurden.

Es wird die Quittierung vom Diagnosen für folgende Csos S20-SSDx-Module unterstützt:

- S20-SSDI-8/4
- S20-SSDO-8/3

Bei den quittierungspflichtigen Safety-Diagnosen handelt es sich meist um kanalspezifische Diagnosen wie z.B. "Symmetrieverletzung am Eingang x Kanal y". Nach Beseitigung der Fehlerursache müssen diese Diagnosen quittiert werden, damit der Kanal / das Modul den Betrieb wieder aufnehmen kann.

WARNUNG

Es ist die Dokumentation der S20-SSDx-Module zu beachten

Die Bedeutung der gerätespezifischen Diagnosen und Fehlercodes des Sercos-S20-Buskopplers (S20-S3-BK+) und der Csos-S20-Module entnehmen Sie der entsprechenden Dokumentation dieser Geräte. Dort finden Sie Hinweise zu Fehlerursachen sowie Anweisungen zur Fehlerbeseitigung und Quittierung.

WARNUNG

Quittierung kann zum gefährlichen Zustand führen

Das Quittieren eines Fehlers führt bis auf die angegebenen Ausnahmen sofort zur Rückkehr des sicheren Ausgangs / sicheren Eingangs in den Betriebszustand. Stellen Sie deshalb vor der Quittierung eines Fehlers sicher, dass die Quittierung nicht zum gefährlichen Zustand der Maschine führen kann! Berücksichtigen Sie bei der Planung der Maschine oder Anlage, dass das Quittieren nur dann möglich sein darf, wenn der Gefahrenbereich einsehbar ist.

Der Prozess des Auslesens und der Quittierung von Safety-Diagnosen wird mit steigender Flanke am Eingang "Execute" initiiert.

Am Anfang wird einmalig die Soll-Konfiguration am S20-S3-BK+ ausgelesen. Damit ist bekannt, auf welchen Slots sich S20-SSDx-Module befinden. Ändern sich während der Laufzeit des FBs die Eingangsparameter "BusMaster" oder "SercosAdr" wird die Soll-Konfiguration am S20-S3-BK+ erneut ausgelesen.

Zuerst wird der FB "IL_SIIIRemoteSlaveState" ausgeführt, um zu erkennen, ob am Sercos-S20-Buskoppler bzw. seinen Modulen Diagnosen anstehen.

Wenn eine Diagnose ansteht, prüft der Funktionsblock, ob die Diagnose von einem S20-SSDx-Modul stammt und quittierungspflichtig ist. Falls ja wird, intern der Quittierungsvorgang vorbereitet.

Wenn die Safety-Diagnose erfolgreich quittiert wurde, wird die nächste Diagnose behandelt, die am Sercos-S20-Buskoppler ansteht.

Auf dieser Basis werden die Ausgänge "DiagSafety", "DiagStd", "DiagAckReq", "DiagPwrCycReq", "SlotNo", "S20Channel" und "DiagCode" gesetzt.

Der Ausgang "DiagAckReq" zeigt an, dass eine quittierungspflichtige Safety-Diagnose ansteht und diese Diagnose mit einer positiven Flanke an dem Eingang "AckTrig" nach Beseitigung der Fehlerursache quittiert werden muss. Zum Quittieren wird intern das "ResetDiag-Objekt 0x0019" verwendet.

Diagnosen Der Ausgang "DiagCode" gibt den Zustand des Sercos-S20-Buskopplers und der S20-Modulen an.

Der Ausgang "SlotNo" zeigt den diagnosemeldenden Slot bzw. die S20-Modul Nummer an.

Der Ausgang "S20Channel" zeigt den diagnosemeldenden Kanal an. Der Wert 255 bedeutet, dass das ganze Modul betroffen ist. Basis hierfür ist das Diagnoseobjekt "DiagState-Objekt 0x0018, Index 3: Channel / Group / Module".

Der Ausgang "DiagTextModul" gibt den Diagnosetext direkt vom Modul aus (Parameter S-0-0095).

Der Ausgang "DiagAdd" gibt zusätzliche Diagnose (Code und Text) aus. Hierfür wird das Diagnoseobjekt "DiagStateAddValue-Objekt 0x0034" ausgelesen.

Fehlerbehandlung Der Funktionsbaustein "IH_SIIIAck_SSDx_S20" reicht die Fehlercodes von den intern verwendeten Funktionsbausteinen am eigenen Error-Ausgang weiter. Siehe Tabelle **SERCOS_TABLE(16#0010)**, **F_RELATED_TABLE(16#0170)**.

ErrorID	Additional1	Additional2	Beschreibung
DEVICE_ERROR	16#00008097	0	Sercos: Gerät nicht vorhanden

Tab. 18-7: Beispiel Fehlercode aus der SERCOS_TABLE von den intern verwendeten Funktionsbausteinen

Der Funktionsbaustein "IH_SIIIAck_SSDx_S20" erzeugt folgende Fehlercodes:

ErrorID	Additional1	Additional2	Beschreibung
DEVICE_ERROR	16#000000FB	0	Die Modul-ID unterscheidet sich von der im IndraWorks-Projekt konfigurierten Modul-ID
DEVICE_ERROR	16#000000FC	0	Die Anzahl der Module unterscheidet sich von der im IndraWorks-Projekt konfigurierten Modulen

ErrorID	Additional1	Additional2	Beschreibung
DEVICE_ERROR	16#000000FD	0	Die maximale Anzahl von 20 aufeinanderfolgenden Quit- tierungen für den gleichen Slot ist erreicht.
DEVICE_ERROR	16#000000FE	0	Nicht unterstütztes Gerät, es ist der Buskoppler S20-S3- BK+ zu verwenden.
STATE_MACHINE_ERROR	16#000000FF	0	Fehler in der State Machine, wenn dieser Fehler er- scheint, hat der Funktionsbaustein nichts weiter getan als diesen Fehler zu generieren.

Tab. 18-8: Fehlercodes, die von IH_SIIIack_SSDx_S20 generiert werden

19 RIH_Dali.library

19.1 RIH_Dali, Allgemeines

DALI - Digital Addressable Lighting Interface ist ein Protokoll für die Licht- und Gebäudeautomation gemäß der technischen Norm IEC 62386.

Die Bibliothek **RIH_DALI** ist mit den Inline-Modulen **R-IB IL DALI/PWR-PAC** (R911172212) und **R-IB IL DALI-PAC** (R911172213) zu verwenden. Das Inline-Modul kann an die Steuerung oder einen Inline-Buskoppler angeschlossen werden (Profibus, Profinet, sercos).

Mit einer DALI-Steuerung (**R-IB IL DALI/PWR-PAC**) kann auf bis zu 64 Geräte zugegriffen werden (z. B. Lampen, Vorschaltgeräte, Sensoren oder Dimmer). An die DALI-Steuerung können zur Erweiterung des DALI-Netzwerks DALI-Module (**R-IB IL DALI-PAC**) angeschlossen werden. Bosch Rexroth bietet Lampen mit DALI-Unterstützung, wie SL36 und SL78 electronic. Das DALI-Netzwerk kann Linien- oder Sterntopologie haben. Jedes Gerät kann einzeln oder über Sammelbefehle angesprochen werden. Anordnung und Zugriff auf DALI-Vorschaltgeräte können in Gruppen (max. 16 Gruppen) bzw. Szenen (max. 16 Szenen) erfolgen.

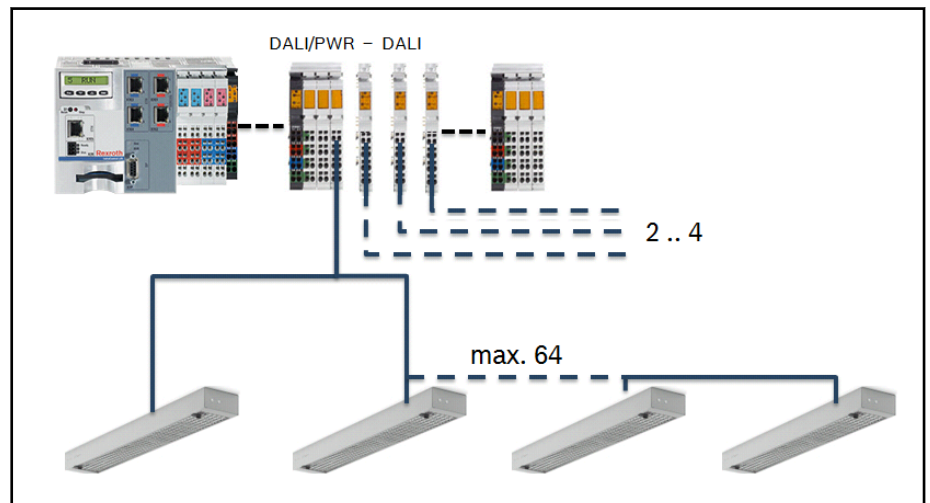


Abb. 19-1: Beispiel eines DALI-Netzwerks mit Lx5-Steuerung, R-IB IL DALI/PWR-PAC und SL36 electronic

Die Bibliothek **RIH_DALI** enthält einen grundlegenden (alles für den Start mit DALI) und einen erweiterten Teil (zusätzliche Unterstützung). Mit der DALI-Funktionsbausteinlösung von Bosch Rexroth kann ein DALI-Netzwerk auch ohne weitere Tools gesteuert werden (für die Ersteinstellung der DALI-Adressen ist kein externes Tool notwendig, dies kann mit Hilfe des Funktionsbausteins **IH_DaliProgramShortAddress** erfolgen).

Um ein SPS-Prgramm mit **RIH_DALI** zu starten, muss der Funktionsbaustein **IH_DaliReadWrite** für jedes DALI-Inline-Modul instanziiert werden, er stellt den Kommunikation-Handle für alle anderen Funktionsbausteine in der **RIH_DALI**-Bibliothek bereit, z. B. **IH_DALION**, **IH_DALIDim**.

Funktionsbausteine der Bibliothek RIH_Dali

Typ	Beschreibung
Basisfunktionsbausteine (Basic)	
IH_DaliReadWrite, Seite 763	Master-Funktionsbaustein, gewährleistet die Kommunikation mit dem DALI-Inline-Modul und stellt den Kommunikation-Handle IOstrToken für die anderen Funktionsbausteine bereit
IH_DaliDim, Seite 747	Steuert die Helligkeit des Vorschaltgeräts, kann für Gruppen oder Broadcast verwendet werden (ggf. Vorschaltgerät einschalten), die Helligkeit ist festgelegt
IH_DaliOff, Seite 774	Vorschaltgerät ausschalten, kann für Gruppen oder Broadcast verwendet werden
IH_DaliOn, Seite 776	Vorschaltgerät einschalten, kann für Gruppen oder Broadcast verwendet werden
IH_DaliDirectPower, Seite 755	Steuert die Helligkeit des Vorschaltgeräts, kann für Gruppen oder Broadcast verwendet werden (ggf. Vorschaltgerät einschalten), die Helligkeit kann eingestellt werden
IH_DaliAddress, Seite 749	Weist dem Vorschaltgerät eine neue Adresse zu
IH_DaliCallScene, Seite 751	Setzt die Helligkeit der Szene auf den festgelegten Wert
IH_DaliAddToGroup, Seite 770	Fügt das Vorschaltgerät zur Gruppe hinzu
IH_DaliAddToScene, Seite 772	Fügt das Vorschaltgerät oder eine Gruppe zur Szene hinzu
IH_DaliRemoveFromGroup, Seite 753	Entfernt das Vorschaltgerät aus der Gruppe
IH_DaliRemoveFromScene, Seite 758	Entfernt das Vorschaltgerät oder eine Gruppe aus der Szene
Erweiterte Funktionsbausteine (Advanced)	
IH_DaliQueryStatus, Seite 760	Liest den Status aller angeschlossenen DALI.Vorschaltgeräte
IH_DaliQueryActualLevel, Seite 766	Liest die Istleistung des Vorschaltgeräts
IH_DaliQuerySceneLevel, Seite 781	Liest die Istleistung der Szene
IH_DaliQueryDTR, Seite 794	Liest das Data Transfer Register (DTR) des Vorschaltgeräts
IH_DaliStoreParameter, Seite 788	Speichert den Inhalt des Data Transfer Register (DTR) in einen festgelegten Parameter des Vorschaltgeräts
IH_DaliQueryParameter, Seite 796	Liest die Parameter Fade Rate, Fade Time, Min Level, Max Level, Power On Level, Physical Min Level und System Failure Level des Vorschaltgeräts
IH_DaliQueryGroup, Seite 800	Liest die zugewiesene Gruppe des Vorschaltgeräts
IH_DaliQueryRandomAddress, Seite 803	Liest die Zufallsadresse des Vorschaltgeräts
IH_DaliSearchAddress, Seite 779	Speichert die Suchadresse des Vorschaltgeräts
IH_DaliProgramShortAddress, Seite 783	Weist dem Vorschaltgerät eine Kurzadresse durch zufällige Adressvergabe oder physikalische Auswahl während der Initialisierung zu
IH_DaliReset, Seite 791	Setzt Variablen im nichtflüchtigen Speicher des Vorschaltgeräts zurück

Tab. 19-1: Funktionsbausteine der Bibliothek RIH_Dali

19.2 IH_DaliDim

Kurzbeschreibung

Der Funktionsbaustein **IH_DaliDim** dient zur Steuerung der Helligkeit des Vorschaltgeräts. Der Funktionsbaustein muss im Funktionsbaustein **IH_DaliReadWrite**, Seite 763 aufgerufen werden.

Zielsystem	Bibliothek/Ordner
MLC	RIH_Dali/Basic
MTX	RIH_Dali/Basic
MLD (IL 2G)	RIH_Dali/Basic

Tab. 19-2: Referenztabelle

Schnittstellenbeschreibung

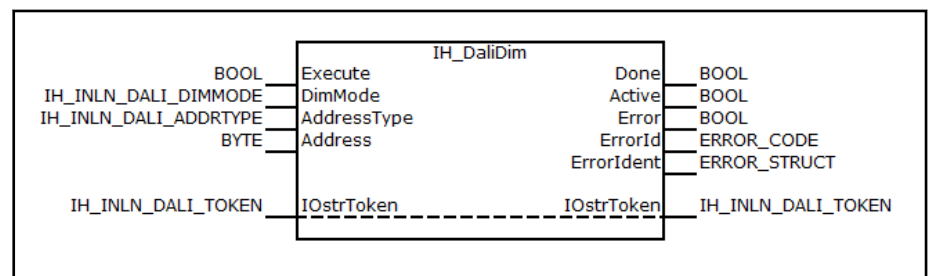


Abb. 19-2: Funktionsbaustein IH_DaliDim

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Steigende Flanke startet Berechnung
	DimMode	IH_INLN_DALI_DIMMODE IH_INLN_DALI_DIMMODE, Seite 806	Aufzählung zur Auswahl des DimMode IH_INLN_DALI_DIMMODE, Seite 806
	AddressType	IH_INLN_DALI_ADDRTYPE IH_INLN_DALI_ADDRTYPE, Seite 806	Aufzählung zur Auswahl des Adresstyps. IH_INLN_DALI_ADDRTYPE, Seite 806
	Adress	BYTE	Die Adresse hängt vom ausgewählten Typ unter AddressType ab. Für Kurzadressen sind Werte von 0 bis 63 und für Gruppenadressen von 0 bis 15 möglich.
VAR_OUTPUT	Done	BOOL	Berechnung beendet
	Active	BOOL	Berechnung findet statt
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Dali, Seite 809
VAR_IN_OUT	IOstrToken	IH_INLN_DALI_TOKEN IH_INLN_DALI_TOKEN, Seite 806	Struktur zum Umgang mit Daten zwischen IH_DaliReadWrite und anderen Dali-Funktionsbausteinen. IH_INLN_DALI_TOKEN, Seite 806

Tab. 19-3: Schnittstellenvariablen von Funktionsbaustein IH_DaliDim

Min-/Max- und Standard-Werte In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
DimMode	IH_INLN_DALI_DIMMODE	UP	STEP-DOWN_AND_OFF	UP	Steigende Flanke an Execute
AddressType	IH_INLN_DALI_ADDRTYPE	SHORTADDR	BROADCAST	SHORTADDR	Steigende Flanke an Execute
Adress	BYTE	0	Kurzadresse = 63 Gruppenadresse = 15	0	Steigende Flanke an Execute

Tab. 19-4: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_DaliDim

Funktionsbeschreibung

Mit diesem Funktionsbaustein wird abhängig von der Auswahl unter Address-Type die Helligkeit eines einzelnen Vorschaltgeräts, einer Gruppe oder aller angeschlossenen Vorschaltgeräte geändert. Unter DimMode kann der Nutzer unterschiedliche Abblendmodi einstellen. Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Execute" auf TRUE gesetzt wird. Geänderte Eingangswerte werden erst mit einer neuen steigenden Flanke am Eingang "Execute" angewendet. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Der Funktionsbaustein kann mit einer neuen steigenden Flanke am Eingang "Execute" reinitialisiert werden. Falls während des Betriebs ein Fehler auftritt, wird das Bit "Error" auf TRUE gesetzt. ErrorID und ErrorIDent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Execute" quittiert werden.

Nicht-funktionale Kriterien

Dieser Funktionsbaustein muss zur Kommunikation mit dem Dali-Modul über den Funktionsbaustein **IH_DaliReadWrite** aufgerufen werden.

Kompatibilität

Kompatibel mit Inline-, Profibus- und Profinet-Feldbussen.

Beispiel eines Funktionsbausteins

Zunächst muss der Funktionsbaustein IH_DaliReadWrite instanziiert werden, bevor der Funktionsbaustein IH_DaliDim zur Kommunikation mit dem Dali-Modul instanziiert werden kann. Die Eingangs-/Ausgangsvariable stDaliToken tauscht die Daten zwischen den Funktionsbausteinen IH_DaliReadWrite und IH_daliDim aus. Deshalb muss diese Variable in beiden Funktionsbausteinen IOstrToken zugewiesen werden.

In diesem Beispiel ist der Funktionsbaustein IH_DaliDim im Modus ON_AND_STEPUP für ein einziges Vorschaltgerät konfiguriert. Ist das Vorschaltgerät in diesem Modus ausgeschaltet, wird es von dem Befehl zunächst eingeschaltet und auf Min Level gesetzt. Eine neue steigende Flanke am Eingang "bExecute" erhöht die Helligkeit um einen Schritt.

Variablendeklaration

```
VAR
    fbIH_DaliDim: IH_DaliDim;
    bExecute: BOOL;
    stDaliToken: IH_INLN_DALI_TOKEN;
```



```

bDone: BOOL;
bActive: BOOL;
bError: BOOL;
enErrorID: ERROR_CODE;
stErrorIdent: ERROR_STRUCT;
END_VAR

```

Beispiel

```

fbIH_DaliDim(
    Execute:= bExecute,
    DimMode:= IH_INLN_DALI_DIMMODE.ON AND STEPUP,
    AddressType:= IH_INLN_DALI_ADDRTYPE.SHORTADDR,
    Address:= BYTE#2,
    IOstrToken:= stDaliToken,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent);

```

19.3 IH_DaliAddress

Kurzbeschreibung Der Funktionsbaustein **IH_DaliAddress** weist einem Vorschaltgerät eine neue Kurzadresse zu. Der Funktionsbaustein muss mit dem Funktionsbaustein [IH_DaliReadWrite](#), [Seite 763](#) aufgerufen werden.

Zielsystem	Bibliothek/Ordner
MLC	RIH_Dali/Basic
MTX	RIH_Dali/Basic
MLD (IL 2G)	RIH_Dali/Basic

Tab. 19-5: Referenztabelle

Schnittstellenbeschreibung

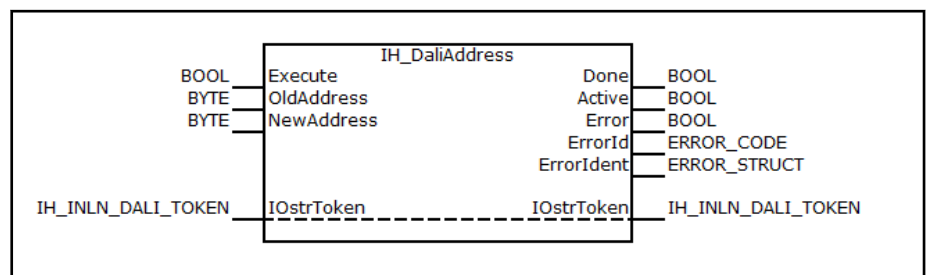


Abb. 19-3: Funktionsbaustein IH_DaliAddress

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Steigende Flanke startet Berechnung
	OldAddress	BYTE	Alte Adresse des Vorschaltgeräts (0 bis 63)
	NewAddress	BYTE	Neue Adresse des Vorschaltgeräts (0 bis 63)

E/A-Typ	Name	Datentyp	Beschreibung
VAR_OUTPUT	Done	BOOL	Berechnung beendet
	Active	BOOL	Berechnung findet statt
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Dali, Seite 809
VAR_IN_OUT	IOstrToken	IH_INLN_DALI_TOKEN IH_INLN_DALI_TOKEN, Seite 806	Struktur zum Umgang mit Daten zwischen IH_DaliReadWrite und anderen Dali-Funktionsbausteinen. IH_INLN_DALI_TOKEN, Seite 806

Tab. 19-6: Schnittstellenvariablen von Funktionsbaustein IH_DaliAddress

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Ausführen	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
OldAddress	BYTE	0	63	0	Steigende Flanke von Execute
NewAddress	BYTE	0	63	0	Steigende Flanke an Execute

Tab. 19-7: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_DaliAddress

Funktionsbeschreibung

Der Funktionsbaustein weist einem Vorschaltgerät eine neue Kurzadresse zu. Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Execute" auf TRUE gesetzt wird. Geänderte Eingangswerte werden erst mit einer neuen steigenden Flanke am Eingang "Execute" angewendet. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Der Funktionsbaustein kann mit einer neuen steigenden Flanke am Eingang "Execute" reinitialisiert werden. Falls während des Betriebs ein Fehler auftritt, wird das Bit "Error" auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an.

Der Fehler kann dann mit einer steigenden Flanke am Eingang "Execute" quittiert werden.

Nicht-funktionale Kriterien

Dieser Funktionsbaustein muss zur Kommunikation mit dem Dali-Modul über den Funktionsbaustein **IH_DaliReadWrite** aufgerufen werden.

Kompatibilität

Kompatibel mit Inline-, Profibus- und Profinet-Feldbussen.

Beispiel eines Funktionsbausteins

Zunächst muss der Funktionsbaustein IH_DaliReadWrite instanziiert werden, bevor der Funktionsbaustein IH_DaliAddress zur Kommunikation mit dem Dali-Modul instanziiert werden kann. Die Eingangs-/Ausgangsvariable stDali-Token tauscht die Daten zwischen den Funktionsbausteinen IH_DaliReadW-

rite und IH_DaliAddress aus. Deshalb muss diese Variable in beiden Funktionsbausteinen IOstrToken zugewiesen werden.

In diesem Beispiel ist der Funktionsbaustein IH_DaliAddress so konfiguriert, dass er die Adresse des Vorschaltgeräts von 2 auf 4 ändert. Mit einer neuen steigenden Flanke am Eingang "bExecute" wird die neue Adresse dem Vorschaltgerät zugewiesen.

Variablendeklaration

```
VAR
    fbDaliAddress : IH_DaliAddress;
    bExecute      : BOOL;
    stDaliToken   : IH_INLN_DALI_TOKEN;
    bDone         : BOOL;
    bActive       : BOOL;
    bError        : BOOL;
    enErrorID     : ERROR_CODE;
    stErrorIdent  : ERROR_STRUCT;
END_VAR
```

Beispiel

```
fbDaliAddress (
    Execute:= bExecute,
    OldAddress:= BYTE#2,
    NewAddress:= BYTE#4,
    IOstrToken:= stDaliToken,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent);
```

19.4 IH_DaliCallScene

Kurzbeschreibung Der Funktionsbaustein **IH_DaliCallScene** dient dem Aufruf der bereits definierten Szene. Der Funktionsbaustein muss mit dem Funktionsbaustein **IH_DaliReadWrite**, Seite 763 aufgerufen werden.

Zielsystem	Bibliothek/Ordner
MLC	RIH_Dali/Basic
MTX	RIH_Dali/Basic
MLD (IL 2G)	RIH_Dali/Basic

Tab. 19-8: Referenztablelle

Schnittstellenbeschreibung

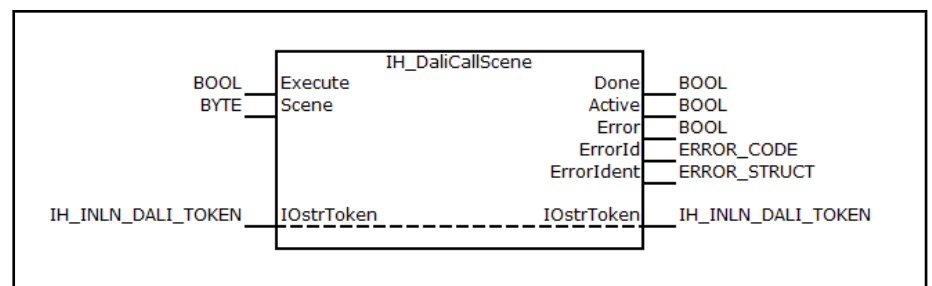


Abb. 19-4: Funktionsbaustein IH_DaliCallScene

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Steigende Flanke startet Berechnung
	Scene	BYTE	Szenennummer Wertebereich von 0 bis 15

E/A-Typ	Name	Datentyp	Beschreibung
VAR_OUTPUT	Done	BOOL	Berechnung beendet
	Active	BOOL	Berechnung findet statt
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Dali, Seite 809
VAR_IN_OUT	IOstrToken	IH_INLN_DALI_TOKEN IH_INLN_DALI_TOKEN, Seite 806	Struktur zum Umgang mit Daten zwischen IH_Dali-ReadWrite und anderen Dali-Funktionsbausteinen. IH_INLN_DALI_TOKEN, Seite 806

Tab. 19-9: Schnittstellenvariablen von Funktionsbaustein IH_DaliCallScene

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
Scene	BYTE	0	15	0	Steigende Flanke von Execute

Tab. 19-10: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_DaliCallScene

Funktionsbeschreibung

Dieser Funktionsbaustein ruft die bereits definierte Szene auf. Es können insgesamt 16 Szenen aufgerufen werden. Der Wert von "Scene" liegt zwischen 0 und 15. Nach dem ersten SPS-Zyklus wird der Funktionsbaustein aktiviert, indem der Eingang "Execute" auf TRUE gesetzt wird. Geänderte Eingabewerte werden erst mit einer neuen steigenden Flanke am Eingang "Execute" angewendet. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Der Funktionsbaustein kann mit einer neuen steigenden Flanke am Eingang "Execute" reinitialisiert werden. Falls während des Betriebs ein Fehler auftritt, wird das Bit "Error" auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Execute" quittiert werden.

Nicht-funktionale Kriterien

Dieser Funktionsbaustein muss zur Kommunikation mit dem Dali-Modul über den Funktionsbaustein **IH_DaliReadWrite** aufgerufen werden.

Kompatibilität

Kompatibel mit Inline-, Profibus- und Profinet-Feldbussen.

Beispiel eines Funktionsbausteins

Zunächst muss der Funktionsbaustein IH_DaliReadWrite instanziiert werden, bevor der Funktionsbaustein IH_DaliCallScene zur Kommunikation mit dem Dali-Modul instanziiert werden kann. Die Eingangs-/Ausgangsvariable stDali-Token tauscht die Daten zwischen den Funktionsbausteinen IH_DaliReadWrite und IH_DaliCallScene aus. Deshalb muss diese Variable in beiden Funktionsbausteinen IOstrToken zugewiesen werden.

In diesem Beispiel wird mit IH_DaliCallScene die bereits definierte Szene Nummer 5 aufgerufen. Bei einer steigenden Flanke am Eingang "Execute" wird Szene 5 auf allen definierten Vorschaltgeräten aktiviert.

Variablendeklaration

```
VAR
    fbIH_DaliCallScene: IH_DaliCallScene;
    bExecute: BOOL;
    stDaliToken: IH_INLN_DALI_TOKEN;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
END_VAR
```

Beispiel

```
fbIH_DaliCallScene(
    Execute:= bExecute,
    Scene:= BYTE#5,
    IOstrToken:= stDaliToken,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent);
```

19.5 IH_DaliRemoveFromGroup

Kurzbeschreibung Mit dem Funktionsbaustein **IH_DaliRemoveFromGroup** wird ein Vorschaltgerät aus der angegebenen Gruppe entfernt. Der Funktionsbaustein muss mit dem Funktionsbaustein **IH_DaliReadWrite**, [Seite 763](#) aufgerufen werden.

Zielsystem	Bibliothek/Ordner
MLC	RIH_Dali/Basic
MTX	RIH_Dali/Basic
MLD (IL 2G)	RIH_Dali/Basic

Tab. 19-11: Referenztablelle

Schnittstellenbeschreibung

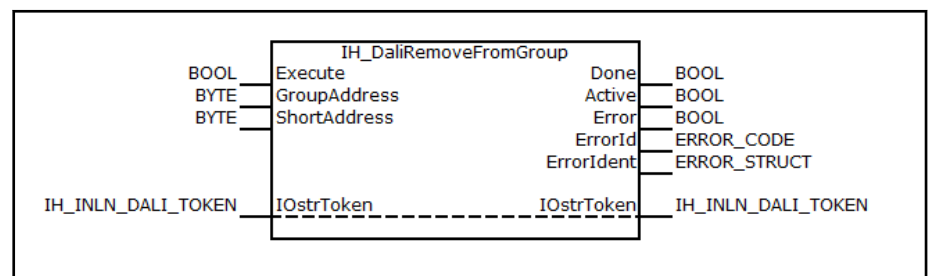


Abb. 19-5: Funktionsbaustein IH_DaliRemoveFromGroup

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Steigende Flanke startet Berechnung
	GroupAddress	BYTE	Adresse der Gruppe (0 bis 15)
	ShortAddress	BYTE	Adresse des Vorschaltgeräts (0 bis 63)

E/A-Typ	Name	Datentyp	Beschreibung
VAR_OUTPUT	Done	BOOL	Berechnung beendet
	Active	BOOL	Berechnung findet statt
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Dali, Seite 809
VAR_IN_OUT	IOstrToken	IH_INLN_DALI_TOKEN IH_INLN_DALI_TOKEN, Seite 806	Struktur zum Umgang mit Daten zwischen IH_Dali-ReadWrite und anderen Dali-Funktionsbausteinen. IH_INLN_DALI_TOKEN, Seite 806

Tab. 19-12: Schnittstellenvariablen von Funktionsbaustein IH_DaliRemoveFromGroup

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Ausführen	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
GroupAddress	BYTE	0	15	0	Steigende Flanke von Execute
ShortAddress	BYTE	0	63	0	Steigende Flanke von Execute

Tab. 19-13: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_DaliRemoveFromGroup

Funktionsbeschreibung

Mit dem Funktionsbaustein wird ein Vorschaltgerät aus der angegebenen Gruppe entfernt. Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Execute" auf TRUE gesetzt wird. Geänderte Eingangswerte werden erst mit einer neuen steigenden Flanke am Eingang "Execute" angewendet. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Der Funktionsbaustein kann mit einer neuen steigenden Flanke am Eingang "Execute" reinitialisiert werden. Falls während des Betriebs ein Fehler auftritt, wird das Bit "Error" auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Execute" quittiert werden.

Nicht-funktionale Kriterien

Dieser Funktionsbaustein muss zur Kommunikation mit dem Dali-Modul über den Funktionsbaustein **IH_DaliReadWrite** aufgerufen werden.

Kompatibilität

Kompatibel mit Inline-, Profibus- und Profinet-Feldbussen.

Beispiel eines Funktionsbausteins

Zunächst muss der Funktionsbaustein IH_DaliReadWrite instanziiert werden, bevor der Funktionsbaustein IH_DaliRemoveFromGroup zur Kommunikation mit dem Dali-Modul instanziiert werden kann. Die Eingangs-/Ausgangsvariable stDaliToken tauscht die Daten zwischen den Funktionsbausteinen IH_Dali-

ReadWrite und IH_DaliRemoveFromGroup aus. Deshalb muss diese Variable in beiden Funktionsbausteinen IOstrToken zugewiesen werden.

Im Beispiel ist der Funktionsbaustein IH_DaliRemoveFromGroup so konfiguriert, dass er das Vorschaltgerät (Adresse = 2) aus Gruppe Nummer 5 entfernt. Bei einer neuen steigenden Flanke am Eingang "bExecute" wird das Vorschaltgerät aus der angegebenen Gruppe entfernt.

Variablendeklaration

Programm:

```
VAR
    fbDaliRemoveFromGroup: IH_DaliRemoveFromGroup;
    bExecute: BOOL;
    stDaliToken: IH_INLN_DALI_TOKEN;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
END_VAR
```

Beispiel

```
fbDaliRemoveFromGroup(
    Execute:= bExecute,
    GroupAddress:= BYTE#5,
    ShortAddress:= BYTE#2,
    IOstrToken:= stDaliToken,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent);
```

19.6 IH_DaliDirectPower

Kurzbeschreibung Der Funktionsbaustein **IH_DaliDirectPower** dient zur direkten Eingabe der Helligkeit. Der Funktionsbaustein muss mit dem Funktionsbaustein **IH_DaliReadWrite**, Seite 763 aufgerufen werden.

Zielsystem	Bibliothek/Ordner
MLC	RIH_Dali/Basic
MTX	RIH_Dali/Basic
MLD (IL 2G)	RIH_Dali/Basic

Tab. 19-14: Referenztablelle

Schnittstellenbeschreibung

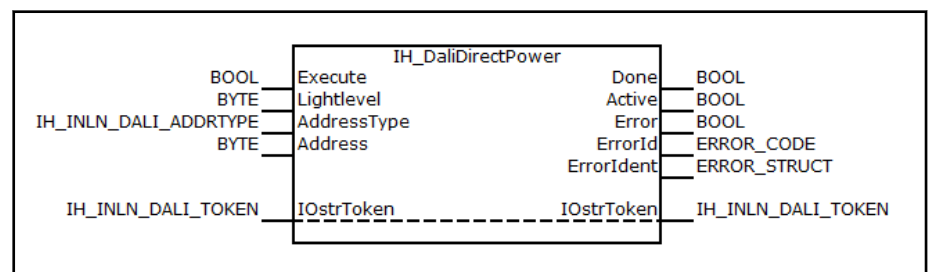


Abb. 19-6: Funktionsbaustein IH_DaliDirectPower

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Steigende Flanke startet Berechnung
	LightLevel	BYTE	Der Wert gibt die Helligkeit an. Der Wertebereich reicht von 0 bis 254.
	AddressType	IH_INLN_DALI_ADDR- TYPE IH_INLN_DALI_ADDR- TYPE, Seite 806	Aufzählung zur Auswahl des Adresstyps. IH_INLN_DALI_ADDRTYPE, Seite 806
	Adress	BYTE	Die Adresse hängt vom ausgewählten Typ unter AddressType ab. Für Kurzadressen sind Werte von 0 bis 63 und für Gruppenadressen von 0 bis 15 möglich.
VAR_OUTPUT	Done	BOOL	Berechnung beendet
	Active	BOOL	Berechnung findet statt
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Dali, Seite 809
VAR_IN_OUT	IOstrToken	IH_INLN_DALI_TO- KEN IH_INLN_DALI_TO- KEN, Seite 806	Struktur zum Umgang mit Daten zwischen IH_Dali-ReadWrite und anderen Dali-Funktionsbausteinen. IH_INLN_DALI_TOKEN, Seite 806

Tab. 19-15: Schnittstellenvariablen von Funktionsbaustein IH_DaliDirectPower

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Type	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
LightLevel	BYTE	0	254	254	Steigende Flanke von Execute
AddressType	IH_INLN_DALI_ADDR- TYPE	SHORTADDR	BROADCAST	SHORTADDR	Steigende Flanke von Execute
Adress	BYTE	0	Kurzadresse = 63 Gruppenadresse = 15	0	Steigende Flanke von Execute

Tab. 19-16: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_DaliDirectPower

Funktionsbeschreibung

Mit diesem Funktionsbaustein wird abhängig von der Auswahl unter AddressType ein Helligkeitswert für ein einzelnes Vorschaltgerät, eine Gruppe oder alle angeschlossenen Vorschaltgeräte direkt eingegeben. Die Leistung wird mit folgender Formel berechnet:

$$P_x = 10^{\left(\frac{x-1}{255/3}\right)} \cdot \frac{P_{100\%}}{1000}$$

Abb. 19-7: Leistungsberechnung

Dabei steht X für den Wert des Eingangs "LightLevel".

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Execute" auf TRUE gesetzt wird. Geänderte Eingangswerte werden erst mit einer neuen steigenden Flanke am Eingang "Execute" angewendet. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Der Funktionsbaustein kann mit einer neuen steigenden Flanke am Eingang "Execute" reinitialisiert werden. Falls während des Betriebs ein Fehler auftritt, wird das Bit "Error" auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Execute" quittiert werden.

Nicht-funktionale Kriterien

Dieser Funktionsbaustein muss zur Kommunikation mit dem Dali-Modul über den Funktionsbaustein **IH_DaliReadWrite** aufgerufen werden.

Kompatibilität

Kompatibel mit Inline-, Profibus-, Sercos III- und Profinet-Feldbussen.

Beispiel eines Funktionsbausteins

Zunächst muss der Funktionsbaustein IH_DaliReadWrite instanziiert werden, bevor der Funktionsbaustein IH_DaliDirectPower zur Kommunikation mit dem Dali-Modul instanziiert werden kann. Die Eingangs-/Ausgangsvariable stDaliToken tauscht die Daten zwischen den Funktionsbausteinen IH_DaliReadWrite und IH_DaliDirectPower aus. Deshalb muss diese Variable in beiden Funktionsbausteinen IOstrToken zugewiesen werden.

Im Beispiel wird mit IH_DaliDirectPower die Helligkeit eines einzelnen Vorschaltgeräts auf 254 gesetzt.

Variablendeklaration

```
VAR
    fbIH_DaliDirectPower: IH_DaliDirectPower;
    bExecute: BOOL;
    stDaliToken: IH_INLN_DALI_TOKEN;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
END_VAR
```

Beispiel

```
fbIH_DaliDirectPower(
    Execute:= bExecute,
    LightLevel:= BYTE#254,
    AddressType:= IH_INLN_DALI_ADDRTYPE.SHORTADDR,
    Address:= BYTE#2,
    IOstrToken:= stDaliToken,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent);
```

19.7 IH_DaliRemoveFromScene

Kurzbeschreibung Mit dem Funktionsbaustein **IH_DaliRemoveFromScene** wird ein Vorschaltgerät aus der angegebenen Szene entfernt. Der Funktionsbaustein muss mit dem Funktionsbaustein **IH_DaliReadWrite**, [Seite 763](#) aufgerufen werden.

Zielsystem	Bibliothek/Ordner
MLC	RIH_Dali/Basic
MTX	RIH_Dali/Basic
MLD (IL 2G)	RIH_Dali/Basic

Tab. 19-17: Referenztabelle

Schnittstellenbeschreibung

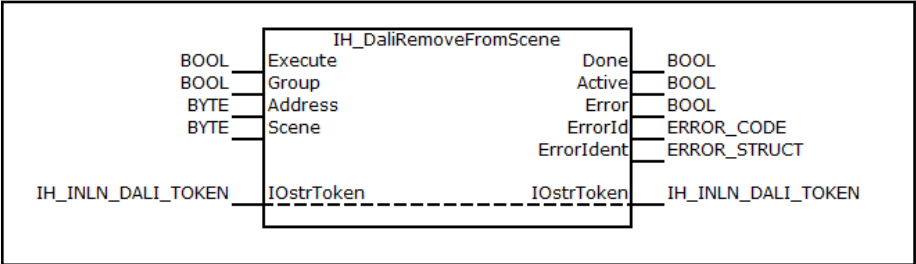


Abb. 19-8: Funktionsbaustein IH_DaliRemoveFromScene

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Steigende Flanke startet Berechnung
	Group	BOOL	TRUE = Adresse ist Gruppenadresse FALSE = Adresse ist Vorschaltgeräteadresse
	Adress	BYTE	Adresse des Vorschaltgeräts/der Gruppe (hängt vom Eingang „Group“ ab)
	Scene	BYTE	Szenennummer (0 bis 15)
VAR_OUTPUT	Done	BOOL	Berechnung beendet
	Active	BOOL	Berechnung findet statt
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Dali, Seite 809
VAR_IN_OUT	IOstrToken	IH_INLN_DALI_TOKEN IH_INLN_DALI_TOKEN, Seite 806	Struktur zum Umgang mit Daten zwischen IH_Dali-ReadWrite und anderen Dali-Funktionsbausteinen. IH_INLN_DALI_TOKEN, Seite 806

Tab. 19-18: Schnittstellenvariablen von Funktionsbaustein IH_DaliRemoveFromScene

Min-/Max- und Standard-Werte In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
Group	BOOL	FALSE	TRUE	FALSE	Steigende Flanke von Execute
Adress	BYTE	0	Gruppe = 15 Vorschaltgerät = 63	0	Steigende Flanke von Execute
Scene	BYTE	0	15	0	Steigende Flanke von Execute

Tab. 19-19: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_DaliRemoveFromScene

Funktionsbeschreibung

Mit dem Funktionsbaustein wird ein Vorschaltgerät/eine Gruppe aus der angegebenen Szene entfernt.

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Execute" auf TRUE gesetzt wird. Geänderte Eingabewerte werden erst mit einer neuen steigenden Flanke am Eingang "Execute" angewendet. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Der Funktionsbaustein kann mit einer neuen steigenden Flanke am Eingang "Execute" reinitialisiert werden. Falls während des Betriebs ein Fehler auftritt, wird das Bit "Error" auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Execute" quittiert werden.

Nicht-funktionale Kriterien

Dieser Funktionsbaustein muss zur Kommunikation mit dem Dali-Modul über den Funktionsbaustein **IH_DaliReadWrite** aufgerufen werden.

Kompatibilität

Kompatibel mit Inline-, Profibus- und Profinet-Feldbussen.

Beispiel eines Funktionsbausteins

Zunächst muss der Funktionsbaustein IH_DaliReadWrite instanziiert werden, bevor der Funktionsbaustein IH_DaliRemoveFromScene zur Kommunikation mit dem Dali-Modul instanziiert werden kann. Die Eingangs-/Ausgangsvariable stDaliToken tauscht die Daten zwischen den Funktionsbausteinen IH_DaliReadWrite und IH_DaliRemoveFromScene aus. Deshalb muss diese Variable in beiden Funktionsbausteinen IOstrToken zugewiesen werden.

Im Beispiel ist der Funktionsbaustein IH_DaliRemoveFromScene so konfiguriert, dass er das Vorschaltgerät (Adresse = 2) aus Szene Nummer 5 entfernt. Bei einer neuen steigenden Flanke am Eingang "bExecute" wird das Vorschaltgerät aus der angegebenen Szene entfernt.

Variablendeklaration

```
VAR
    fbDaliRemoveFromScene: IH_DaliRemoveFromScene;
    bExecute: BOOL;
    stDaliToken: IH_INLN_DALI_TOKEN;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
    bGroup: BOOL;
END_VAR
```

Beispiel

```
fbDaliRemoveFromScene(  
    Execute:= bExecute,  
    Group:= bGroup,  
    Address:= BYTE#2,  
    Scene:= BYTE#5,  
    IOstrToken:= stDaliToken,  
    Done=> bDone,  
    Active=> bActive,  
    Error=> bError,  
    ErrorID=> enErrorID,  
    ErrorIdent=> stErrorIdent);
```

19.8 IH_DaliQueryStatus

Kurzbeschreibung Der Funktionsbaustein **IH_DaliQueryStatus** dient zum Auslesen des Status der an einen Dali-Bus angeschlossenen Vorschaltgeräte. Dieser Funktionsbaustein muss mit dem Funktionsbaustein [IH_DaliReadWrite](#), Seite 763 aufgerufen werden.

Zielsystem	Bibliothek/Ordner
MLC	RIH_Dali/Advanced
MTX	RIH_Dali/Advanced
MLD (IL 2G)	RIH_Dali/Advanced

Tab. 19-20: Referenztable

Schnittstellenbeschreibung

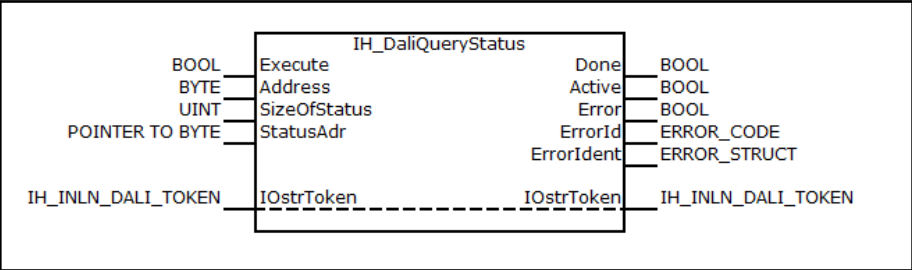


Abb. 19-9: Funktionsbaustein IH_DaliQueryStatus

E/A-Art	Name	Datentyp	Beschreibung		
VAR_INPUT	Execute	BOOL	Steigende Flanke startet Berechnung		
	Adress	BYTE	Adresse des Vorschaltgeräts (0 bis 63 oder 255)		
			Adresse	SizeOfStatus	Anz. gelesener Vorschaltgeräte
			0-63	1	1 Vorschaltgerät
			255	64	64 Vorschaltgeräte
			255	Anz. Vorschaltgeräte von 1 bis 64	Anz. Vorschaltgeräte in SizeOfStatus
	SizeOfStatus	UINT	Statusgröße (1 bis 64)		
	StatusAdr	POINTER_TO_BYTE	Adresse des Status-Array		
			Adresswert	Größe des Status-Array	
			0 bis 63	1 Byte	
255			Anz. Vorschaltgeräte (1 bis 64)		
VAR_OUTPUT	Done	BOOL	Berechnung beendet		
	Active	BOOL	Berechnung findet statt		
	Error	BOOL	Zeigt einen Fehler an		
	ErrorID	ERROR_CODE	Fehlerklasse		
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Dali, Seite 809		
VAR_IN_OUT	IOstrToken	IH_INLN_DALI_TOKEN IH_INLN_DALI_TOKEN, Seite 806	Struktur zum Umgang mit Daten zwischen IH_Dali-ReadWrite und anderen Dali-Funktionsbausteinen. IH_INLN_DALI_TOKEN, Seite 806		

Tab. 19-21: Schnittstellenvariablen von Funktionsbaustein IH_DaliQueryStatus

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
Adress	BYTE	0	255	0	Steigende Flanke an Execute
SizeOfStatus	UINT	1	64	1	Steigende Flanke an Execute

Tab. 19-22: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_DaliQueryStatus

Funktionsbeschreibung

Mit dem Funktionsbaustein wird der Status der an den Dali-Bus angeschlossenen Vorschaltgeräte abgerufen. Liegt der eingegebene Adresswert zwischen 0 und 63 wird der Status des jeweiligen Vorschaltgeräts gelesen. Beträgt der Adresswert 255, liest der FB den Status der in SizeOfStatus ange-

gebenen Anzahl Vorschaltgeräte. Die Statusinformationen stehen im Status-Array bereit. Es ist darauf zu achten, dass der in `SizeOfStatus` festgelegte Wert mit der in `StatusAdr` festgelegten Größe des Arrays übereinstimmt.

Der Status eines einzelnen Vorschaltgeräts ist in einem Byte enthalten, welches folgende Statusinformationen bereitstellt:

Bit	Statusinformationen
0	Status des Vorschaltgeräts 0 = OK 1= NOT OK
1	Lampenausfall. 0 = OK
2	Lampe eingeschaltet. 0 = OFF
3	Grenzwertfehler. 0 = die zuletzt abgerufene Lampenleistung lag entweder zwischen MIN LEVEL und MAX LEVEL oder war OFF
4	Abblenden abgeschlossen: 0 = Abblenden beendet 1 = Abblenden findet statt
5	Status zurücksetzen. 0 = Nein
6	Kurzadresse fehlt. 0 = Nein 1 = Ja
7	Fehler bei Stromversorgung. 0 = Nein, seit dem letzten Einschalten wurde ein Reset- oder Steuerbefehl für die Lampenleistung empfangen

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Execute" auf TRUE gesetzt wird. Geänderte Eingabewerte werden erst mit einer neuen steigenden Flanke am Eingang "Execute" angewendet. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Der Funktionsbaustein kann mit einer neuen steigenden Flanke am Eingang "Execute" reinitialisiert werden.

Falls während des Betriebs ein Fehler auftritt, wird das Bit "Error" auf TRUE gesetzt. `ErrorID` und `ErrorIdent` zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Execute" quittiert werden.

Nicht-funktionale Kriterien

Dieser Funktionsbaustein muss zur Kommunikation mit dem Dali-Modul über den Funktionsbaustein **IH_DaliReadWrite** aufgerufen werden.

Kompatibilität

Kompatibel mit Inline-, Profibus- und Profinet-Feldbussen.

Beispiel eines Funktionsbausteins

Zunächst muss der Funktionsbaustein `IH_DaliReadWrite` instanziiert werden, bevor der Funktionsbaustein `IH_DaliQueryStatus` zur Kommunikation mit dem Dali-Modul instanziiert werden kann. Die Eingangs-/Ausgangsvariable `stDaliToken` tauscht die Daten zwischen den Funktionsbausteinen `IH_DaliReadWrite` und `IH_DaliQueryStatus` aus. Deshalb muss diese Variable in beiden Funktionsbausteinen `IOstrToken` zugewiesen werden.

Beispiel 1

Der Funktionsbaustein `IH_DaliQueryStatus` ist so konfiguriert, dass er den Status eines einzelnen Vorschaltgeräts (Adresse = 2) abruft. Der Status wird im Byte "byAddressStatus" gespeichert.

Variablendeklaration

```

VAR
    fbDaliQueryStatus: IH_DaliQueryStatus;
    bExecute: BOOL;
    stDaliToken: IH_INLN_DALI_TOKEN;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
    byAddressStatus: BYTE;
END_VAR

```

Implementation

```

fbDaliQueryStatus(
    Execute:= bExecute,
    Address:= BYTE#2,
    SizeOfStatus:= SIZEOF(byAddressStatus),
    StatusAdr:= ADR(byAddressStatus),
    IOstrToken:= stDaliToken,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent);

```

Beispiel 2

Der Funktionsbaustein IH_DaliQueryStatus ist so konfiguriert, dass er den Status aller Vorschaltgeräte (von Adresse 0 bis 63) abrufen. Der Status wird im Array "arAddressStatus" gespeichert.

Variablendeklaration

```

VAR
    fbDaliQueryStatus: IH_DaliQueryStatus;
    bExecute: BOOL;
    stDaliToken: IH_INLN_DALI_TOKEN;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
    arAddressStatus: ARRAY[1..64] OF BYTE;
END_VAR

```

Implementation

```

fbDaliQueryStatus(
    Execute:= bExecute,
    Address:= BYTE#16#FF,
    SizeOfStatus:= SIZEOF(arAddressStatus),
    StatusAdr:= ADR(arAddressStatus),
    IOstrToken:= stDaliToken,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent);

```

19.9 IH_DaliReadWrite

Kurzbeschreibung

Der Funktionsbaustein **IH_DaliReadWrite** gewährleistet die Kommunikation mit dem Dali-Terminal. Er erkennt Befehle in der Warteschlange anderer Dali-Bausteine im Programm und gewährleistet deren Ausführung. Der Funktionsbaustein muss für jedes Dali-Terminal einmal instanziiert werden.

Zielsystem	Bibliothek/Ordner
MLC	RIH_Dali/Basic
MTX	RIH_Dali/Basic
MLD (IL 2G)	RIH_Dali/Basic

Tab. 19-23: Referenztable

Schnittstellenbeschreibung

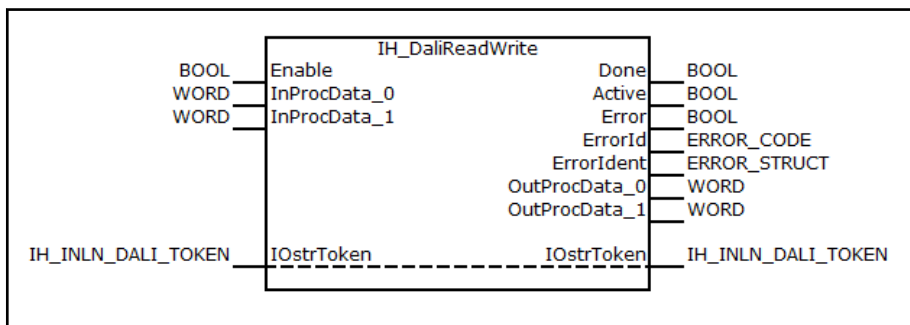


Abb. 19-10: Funktionsbaustein IH_DaliReadWrite

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Funktionsbaustein aktiv, falls TRUE
	InProcData_0	WORD	Eingabe Prozessdatenwort 0
	InProcData_1	WORD	Eingabe Prozessdatenwort 1
VAR_OUTPUT	Done	BOOL	Berechnung ist beendet und startet im nächsten Zyklus neu
	Active	BOOL	Funktionsbaustein arbeitet
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Dali, Seite 809
	OutProcData_0	WORD	Ausgabe Prozessdatenwort 0
	OutProcData_1	WORD	Ausgabe Prozessdatenwort 1
VAR_IN_OUT	IOstrToken	IH_INLN_DALI_TOKEN IH_INLN_DALI_TOKEN, Seite 806	Struktur zum Umgang mit Daten zwischen IH_DaliReadWrite und anderen Dali-Funktionsbausteinen. IH_INLN_DALI_TOKEN, Seite 806

Tab. 19-24: Schnittstellenvariablen von Funktionsbaustein IH_DaliReadWrite

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	Kontinuierlich

Tab. 19-25: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_DaliReadWrite

Funktionsbeschreibung

Der Funktionsbaustein gewährleistet die Kommunikation mit dem Dali-Terminal. "InProcData" und "OutProcData" müssen im Dali-Terminal abgebildet

werden. "IostrToken" muss mit anderen Dali-Bausteinen im Programm verbunden werden. Die Struktur "IostrToken" kommuniziert mit anderen Dali-Bausteinen und tauscht Daten aus. Optional wird vom Modul ein Antwortbefehl empfangen und zu den anderen Dali-Funktionsbausteinen weitergeleitet. In den entsprechenden Dali-Funktionsbausteinen werden Befehle bewertet und verarbeitet. Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Enable" auf TRUE gesetzt wird. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Werden in der Warteschlange Befehle erkannt, beginnt der Funktionsbaustein mit der Verarbeitung, bis das Bit "Enable" auf TRUE gesetzt wird. Falls während des Betriebs ein Fehler auftritt, wird das Bit "Error" auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Enable" quittiert werden.

Besonderheiten bei der Handhabung

Wird das Dali-Modul im Profibus- oder Profinet-Feldbus verwendet, muss die Reihenfolge der Bytes wie folgt geändert werden: Byte 0/1/2/3 = Plug 4/3/2/1.

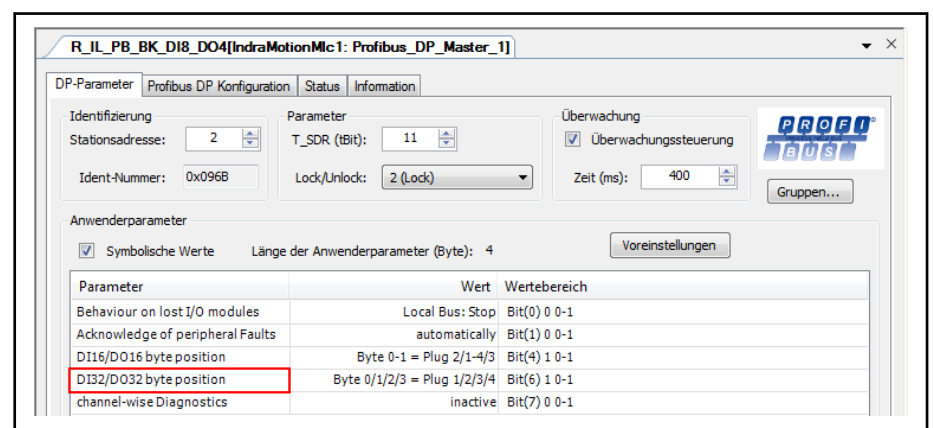


Abb. 19-11: Ändern der Reihenfolge der Bytes

Nicht-funktionale Kriterien

Der Funktionsbaustein IH_DaliReadWrite ist zur Kommunikation mit dem Dali-Modul gemeinsam mit den anderen Funktionsbausteinen zu instanziiieren.

Der aktuelle Status der Vorschaltgeräts wird nicht automatisch aktualisiert. Zur Aktualisierung des aktuellen Status ist eine neue Abfrage erforderlich.

Kompatibilität

Kompatibel mit Inline-, Profibus- und Profinet-Feldbussen.

Beispiel eines Funktionsbausteins

Der Funktionsbaustein IH_DaliReadWrite muss instanziiert werden, bevor andere Dali-Funktionsbausteine zur Kommunikation mit dem Dali-Modul instanziiert werden können. Die Eingangs-/Ausgangsvariable stDaliToken tauscht Daten zwischen IH_DaliReadWrite und anderen Funktionsbausteinen aus. Deshalb muss diese Variable in beiden Funktionsbausteinen IostrToken zugewiesen werden.

Im Beispiel ist der Funktionsbaustein IH_DaliReadWrite so konfiguriert, dass er Daten mit der Dali-Einheit und anderen Dali-Funktionsbausteinen austauscht. Die Variablen "wInProcessData" und "wOutProcessData" müssen der Dali-Einheit zur Kommunikation zugewiesen werden.



Einige Geräte benötigen möglicherweise zunächst eine Dali-Adresse für die Adressierung. Zur Adressierung ist der Funktionsbaustein **IH_DaliProgramShortAddress** zu verwenden.

Variablendeklaration

```
VAR
    fbDaliReadWrite: IH_DaliReadWrite;
    bEnable: BOOL;
    wInProcessData_0: WORD;
    wInProcessData_1: WORD;
    stDaliToken: IH_INLN_DALI_TOKEN;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
    wOutProcessData_0: WORD;
    wOutProcessData_1: WORD;
END_VAR
```

Beispiel

```
fbDaliReadWrite (
    Enable:= bEnable,
    InprocData_0:= wInProcessData_0,
    InprocData_1:= wInProcessData_1,
    IOstrToken:= stDaliToken,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent,
    OutProcData_0=> wOutProcessData_0,
    OutProcData_1=> wOutProcessData_1);
```

Abbildung von Prozessdaten

Channels					
Variable	Mapping	Channel	Address	Type	
Application.POU_Dali.wInProcessData_0		Input0	%IW102		
Application.POU_Dali.wInProcessData_1		Word0	%IW102	WORD	
Application.POU_Dali.wInProcessData_1		Word1	%IW104	WORD	
Application.POU_Dali.wOutProcessData_0		Output0	%QW102		
Application.POU_Dali.wOutProcessData_0		Word0	%QW102	WORD	
Application.POU_Dali.wOutProcessData_1		Word1	%QW104	WORD	

Abb. 19-12: Abbildung von Prozessdaten von IH_DaliReadWrite

19.10 IH_DaliQueryActualLevel

Kurzbeschreibung Der Funktionsbaustein **IH_DaliQueryActualLevel** dient zum Auslesen der tatsächlichen Lampenleistung der an einen Dali-Bus angeschlossenen Vorschaltgeräte. Dieser Funktionsbaustein muss mit dem Funktionsbaustein **IH_DaliReadWrite**, Seite 763 aufgerufen werden.

Zielsystem	Bibliothek/Ordner
MLC	RIH_Dali/Advanced
MTX	RIH_Dali/Advanced
MLD (IL 2G)	RIH_Dali/Advanced

Tab. 19-26: Referenztafel

Schnittstellenbeschreibung

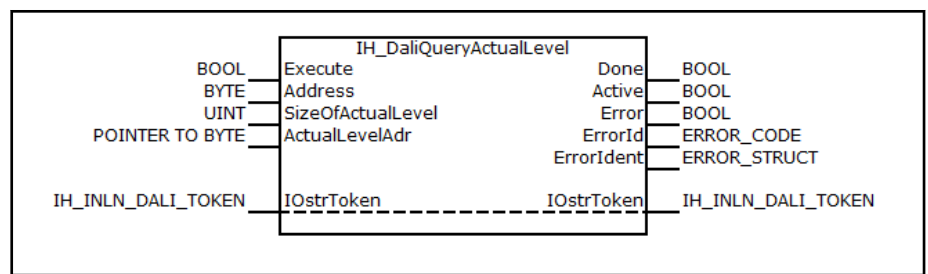


Abb. 19-13: Funktionsbaustein IH_DaliQueryActualLevel

E/A-Typ	Name	Datentyp	Beschreibung		
VAR_INPUT	Execute	BOOL	Steigende Flanke startet Berechnung		
	Adress	BYTE	Adresse des Vorschaltgeräts (0 bis 63 oder 255)		
			Adresse	SizeOfStatus	Anz. gelesener Vorschaltgeräte
			0-63	1	1 Vorschaltgerät
			255	64	64 Vorschaltgeräte
			255	Anz. Vorschaltgeräte von 1 bis 64	Anz. Vorschaltgeräte in Actual-Level
	SizeOfActualLevel	UINT	Größe d. ActualLevel-Array (1 bis 64)		
	ActualLevelAdr	POINTER_TO_BYTE	Adresse d. ActualLevel-Array		
			Adresswert	Größe d. ActualLevel-Array	
			0 bis 63	1 Byte	
255			Anz. Vorschaltgeräte (1 bis 64)		
VAR_OUTPUT	Done	BOOL	Berechnung beendet		
	Active	BOOL	Berechnung findet statt		
	Error	BOOL	Zeigt einen Fehler an		
	ErrorID	ERROR_CODE	Fehlerklasse		
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Dali, Seite 809		
VAR_IN_OUT	IOstrToken	IH_INLN_DALI_TOKEN IH_INLN_DALI_TOKEN, Seite 806	Struktur zum Umgang mit Daten zwischen IH_Dali-ReadWrite und anderen Dali-Funktionsbausteinen. IH_INLN_DALI_TOKEN, Seite 806		

Tab. 19-27: Schnittstellenvariablen von Funktionsbaustein IH_DaliQueryActualLevel

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
Adress	BYTE	0	255	0	Steigende Flanke von Execute
SizeOfActualLevel	UINT	1	64	1	Steigende Flanke von Execute

Tab. 19-28: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_DaliQueryActualLevel

Funktionsbeschreibung

Mit diesem Funktionsbaustein wird die tatsächliche Lampenleistung der an den Dali-Bus angeschlossenen Vorschaltgeräte ermittelt. Liegt der eingegebene Adresswert zwischen 0 und 63 wird "ActualLevel" des jeweiligen Vorschaltgeräts gelesen. Beträgt der Adresswert 255, liest der Funktionsbaustein die tatsächliche Lampenleistung der in SizeOfActualLevel angegebenen Anzahl Vorschaltgeräte. Die Informationen zur tatsächlichen Lampenleistung sind im ActualLevel-Array enthalten.

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Execute" auf TRUE gesetzt wird. Geänderte Eingangswerte werden erst mit einer neuen steigenden Flanke am Eingang "Execute" angewendet. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Der Funktionsbaustein kann mit einer neuen steigenden Flanke am Eingang "Execute" reinitialisiert werden.

Falls während des Betriebs ein Fehler auftritt, wird das Bit "Error" auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Execute" quittiert werden.

Nicht-funktionale Kriterien

Dieser Funktionsbaustein muss zur Kommunikation mit dem Dali-Modul über den Funktionsbaustein **IH_DaliReadWrite** aufgerufen werden.

Kompatibilität

Kompatibel mit Inline-, Profibus-, sercos III- und Profinet-Feldbussen.

Beispiel eines Funktionsbausteins

Zunächst muss der Funktionsbaustein IH_DaliReadWrite instanziiert werden, bevor der Funktionsbaustein IH_DaliQueryActualLevel zur Kommunikation mit dem Dali-Modul instanziiert werden kann. Die Eingangs-/Ausgangsvariable stDaliToken tauscht die Daten zwischen den Funktionsbausteinen IH_DaliReadWrite und IH_DaliQueryActualLevel aus. Deshalb muss diese Variable in beiden Funktionsbausteinen IOstrToken zugewiesen werden.

Beispiel 1

Im Beispiel wird der Funktionsbaustein IH_DaliQueryActualLevel zum Auslesen der tatsächlichen Lampenleistung eines einzelnen Vorschaltgeräts verwendet.

Der Funktionsbaustein liest die tatsächliche Lampenleistung von Vorschaltgerät 2 und speichert sie in der Variable "byActualLevel".

Variablendeklaration

```
VAR
    fbIH_DaliQueryActualLevel: IH_DaliQueryActualLevel;
    bExecute: BOOL;
```

```

byActualLevel: BYTE;
stDaliToken: IH_INLN_DALI_TOKEN;
bDone: BOOL;
bActive: BOOL;
bError: BOOL;
enErrorID: ERROR CODE;
stErrorIdent: ERROR_STRUCT;

END_VAR

```

Implementation

```

fbIH_DaliQueryActualLevel(
  Execute:= bExecute,
  Address:= BYTE#2,
  SizeOfActualLevel:= SIZEOF(byActualLevel),
  ActualLevelAdr:= ADR(byActualLevel),
  IOstrToken:= stDaliToken,
  Done=> bDone,
  Active=> bActive,
  Error=> bError,
  ErrorID=> enErrorID,
  ErrorIdent=> stErrorIdent);

```

Beispiel 2

In diesem Beispiel wird der Funktionsbaustein IH_DaliQueryActualLevel zum Auslesen der tatsächlichen Lampenleistung aller 63 Vorschaltgeräte verwendet.

Der Funktionsbaustein liest die tatsächliche Lampenleistung von 63 Vorschaltgeräten und speichert sie im Array "arActualLevel".

Es muss ein Array mit 63 Bytes erstellt werden und die Adresse des Arrays muss ActualLevelAdr zugeordnet werden.

Die Größe des Arrays wird inSizeOfActualLevel festgelegt. Ist das tatsächliche Array kleiner als der Wert in der Variable SizeOfActualLevel, können Daten in eine andere Adresse geschrieben und so Fehlfunktionen in der Anwendung verursacht werden.

Variablendeklaration

```

VAR
  fbIH_DaliQueryActualLevel: IH_DaliQueryActualLevel;
  bExecute: BOOL;
  stDaliToken: IH_INLN_DALI_TOKEN;
  bDone: BOOL;
  bActive: BOOL;
  bError: BOOL;
  enErrorID: ERROR CODE;
  stErrorIdent: ERROR_STRUCT;
  arActualLevel: ARRAY [0..63] OF BYTE;
END_VAR

```

Implementation

```

fbIH_DaliQueryActualLevel(
  Execute:= bExecute,
  Address:= BYTE#16#FF,
  SizeOfActualLevel:= SIZEOF(arActualLevel),
  ActualLevelAdr:= ADR(arActualLevel),
  IOstrToken:= stDaliToken,
  Done=> bDone,
  Active=> bActive,
  Error=> bError,
  ErrorID=> enErrorID,
  ErrorIdent=> stErrorIdent);

```

19.11 IH_DaliAddToGroup

Kurzbeschreibung Mit dem Funktionsbaustein **IH_DaliAddToGroup** wird ein Vorschaltgerät in die angegebene Gruppe eingefügt. Der Funktionsbaustein muss mit dem Funktionsbaustein **IH_DaliReadWrite**, [Seite 763](#) aufgerufen werden.

Zielsystem	Bibliothek/Ordner
MLC	RIH_Dali/Basic
MTX	RIH_Dali/Basic
MLD (IL 2G)	RIH_Dali/Basic

Tab. 19-29: Referenztabelle

Schnittstellenbeschreibung

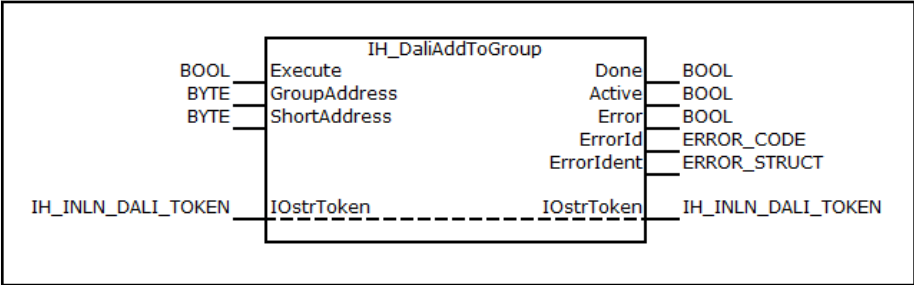


Abb. 19-14: Funktionsbaustein IH_DaliAddToGroup

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Steigende Flanke startet Berechnung
	GroupAddress	BYTE	Adresse der Gruppe (0 bis 15)
	ShortAddress	BYTE	Adresse des Vorschaltgeräts (0 bis 63)
VAR_OUTPUT	Done	BOOL	Berechnung beendet
	Active	BOOL	Berechnung findet statt
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Dali, Seite 809
VAR_IN_OUT	IOstrToken	IH_INLN_DALI_TOKEN IH_INLN_DALI_TOKEN, Seite 806	Struktur zum Umgang mit Daten zwischen IH_Dali-ReadWrite und anderen Dali-Funktionsbausteinen. IH_INLN_DALI_TOKEN, Seite 806

Tab. 19-30: Schnittstellenvariablen von Funktionsbaustein IH_DaliAddToGroup

Min-/Max- und Standard-Werte In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
GroupAddress	BYTE	0	15	0	Steigende Flanke von Execute
ShortAddress	BYTE	0	63	0	Steigende Flanke von Execute

Tab. 19-31: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_DaliAddToGroup

Funktionsbeschreibung

Mit dem Funktionsbaustein IH_DaliAddToGroup können ein oder mehrere Vorschaltgeräte in eine Gruppe eingefügt werden. Es können bis zu 16 Gruppen angelegt werden. Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Execute" auf TRUE gesetzt wird. Geänderte Eingangswerte werden erst mit einer neuen steigenden Flanke am Eingang "Execute" angewendet. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Der Funktionsbaustein kann mit einer neuen steigenden Flanke am Eingang "Execute" reinitialisiert werden. Falls während des Betriebs ein Fehler auftritt, wird das Bit "Error" auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Execute" quittiert werden.

Nicht-funktionale Kriterien

Dieser Funktionsbaustein muss zur Kommunikation mit dem Dali-Modul über den Funktionsbaustein IH_DaliReadWrite aufgerufen werden.

Kompatibilität

Kompatibel mit Inline-, Profibus- und Profinet-Feldbussen.

Beispiel eines Funktionsbausteins

Zunächst muss der Funktionsbaustein IH_DaliReadWrite instanziiert werden, bevor der Funktionsbaustein IH_DaliAddToGroup zur Kommunikation mit dem Dali-Modul instanziiert werden kann. Die Eingangs-/AusgangsvARIABLE stDaliToken tauscht die Daten zwischen den Funktionsbausteinen IH_DaliReadWrite und IH_DaliAddToGroup aus. Deshalb muss diese Variable in beiden Funktionsbausteinen IOstrToken zugewiesen werden.

Im Beispiel ist der Funktionsbaustein IH_DaliAddToGroup so konfiguriert, dass er das Vorschaltgerät mit Adresse 2 zu Gruppe Nummer 10 hinzufügt. Bei einer neuen steigenden Flanke am Eingang "bExecute" wird das Vorschaltgerät zur angegebenen Gruppe hinzugefügt.

Variablendeklaration

```
VAR
    fbDaliAddToGroup      : IH_DaliAddToGroup;
    bExecute              : BOOL;
    stDaliToken           : IH_INLN_DALI_TOKEN;
    bDone                 : BOOL;
    bActive               : BOOL;
    bError                : BOOL;
    enErrorID             : ERROR_CODE;
    stErrorIdent          : ERROR_STRUCT;
END_VAR
```

Beispiel

```
fbDaliAddToGroup (
    Execute:= bExecute,
    GroupAddress:= BYTE#10,
```

```
ShortAddress:= BYTE#2,  
IOstrToken:= stDaliToken,  
Done=> bDone,  
Active=> bActive,  
Error=> bError,  
ErrorID=> enErrorID,  
ErrorIdent=> stErrorIdent);
```

19.12 IH_DaliAddToScene

Kurzbeschreibung Mit dem Funktionsbaustein **IH_DaliAddToScene** wird ein Vorschaltgerät zu der angegebenen Szene hinzugefügt. Der Funktionsbaustein muss mit dem Funktionsbaustein [IH_DaliReadWrite](#), [Seite 763](#) aufgerufen werden.

Zielsystem	Bibliothek/Ordner
MLC	RIH_Dali/Basic
MTX	RIH_Dali/Basic
MLD (IL 2G)	RIH_Dali/Basic

Tab. 19-32: Referenztabelle

Schnittstellenbeschreibung

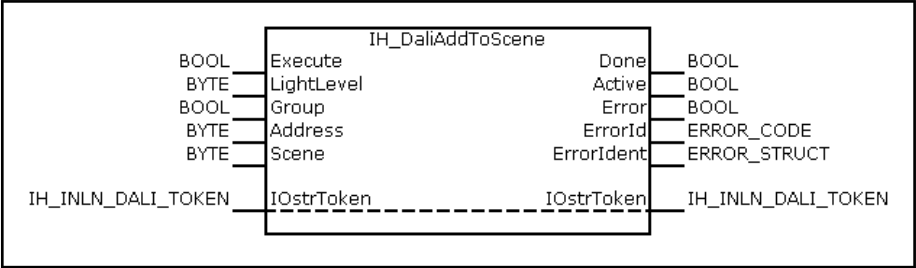


Abb. 19-15: Funktionsbaustein IH_DaliAddToScene

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Steigende Flanke startet Berechnung
	LightLevel	BYTE	Lichtintensität (0 bis 254)
	Group	BOOL	TRUE = Adresse ist Gruppenadresse FALSE = Adresse ist Vorschaltgeräteadresse
	Adress	BYTE	Adresse des Vorschaltgeräts/der Gruppe (hängt vom Eingang „Group“ ab) Für Gruppen sind Werte von 0 bis 15 und für Vorschaltgeräte von 0 bis 63 möglich.
	Scene	BYTE	Szenennummer (0 bis 15)

E/A-Typ	Name	Datentyp	Beschreibung
VAR_OUTPUT	Done	BOOL	Berechnung beendet
	Active	BOOL	Berechnung findet statt
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Dali, Seite 809
VAR_IN_OUT	IOstrToken	IH_INLN_DALI_TOKEN IH_INLN_DALI_TOKEN, Seite 806	Struktur zum Umgang mit Daten zwischen IH_Dali-ReadWrite und anderen Dali-Funktionsbausteinen. IH_INLN_DALI_TOKEN, Seite 806

Tab. 19-33: Schnittstellenvariablen von Funktionsbaustein IH_DaliAddToScene

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
Light Level	BYTE	0	254	254	Steigende Flanke von Execute
Group	BOOL	FALSE	TRUE	FALSE	Steigende Flanke von Execute
Adress	BYTE	0	Gruppe = 15 Vorschaltgerät = 63	0	Steigende Flanke von Execute
Scene	BYTE	0	15	0	Steigende Flanke von Execute

Tab. 19-34: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_DaliAddToScene

Funktionsbeschreibung

Mit diesem Funktionsbaustein wird einer Szene mit einer bestimmten Helligkeit ein Vorschaltgerät oder eine Gruppe hinzugefügt. Die Geräte einer Szene können unterschiedliche Helligkeiten haben. Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Execute" auf TRUE gesetzt wird. Geänderte Eingangswerte werden erst mit einer neuen steigenden Flanke am Eingang "Execute" angewendet. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Der Funktionsbaustein kann mit einer neuen steigenden Flanke am Eingang "Execute" reinitialisiert werden. Falls während des Betriebs ein Fehler auftritt, wird das Bit "Error" auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Execute" quittiert werden.

Nicht-funktionale Kriterien

Dieser Funktionsbaustein muss zur Kommunikation mit dem Dali-Modul über den Funktionsbaustein **IH_DaliReadWrite** aufgerufen werden.

Kompatibilität

Kompatibel mit Inline-, Profibus- und Profinet-Feldbussen.

Beispiel eines Funktionsbausteins

Zunächst muss der Funktionsbaustein IH_DaliReadWrite instanziiert werden, bevor der Funktionsbaustein IH_DaliAddToScene zur Kommunikation mit dem Dali-Modul instanziiert werden kann. Die Eingangs-/Ausgangsvariable stDaliToken tauscht die Daten zwischen den Funktionsbausteinen IH_DaliReadWrite und IH_DaliAddToScene aus. Deshalb muss diese Variable in beiden Funktionsbausteinen IOstrToken zugewiesen werden.

Im Beispiel fügt der Funktionsbaustein DaliAddToScene ein Vorschaltgerät oder eine Gruppe von Vorschaltgeräten mit Adresse 4 zu Szene Nr. 10 mit einer festgelegten Helligkeit von 100 hinzu. Der Adresstyp (Kurzadresse oder Gruppenadresse) hängt vom Status der Variable "Gruppe" ab (TRUE = Adresse ist Gruppenadresse FALSE = Adresse ist Vorschaltgeräteadresse). Bei einer neuen steigenden Flanke am Eingang "bExecute" wird der Szene mit der bestimmten Helligkeit ein Vorschaltgerät oder eine Gruppe hinzugefügt.

Variablendeklaration

```
VAR
    fbDaliAddToScene      : IH_DaliAddToScene;
    bExecute              : BOOL;
    bGroup                : BOOL;
    stDaliToken           : IH_INLN_DALI_TOKEN;
    bDone                 : BOOL;
    bActive               : BOOL;
    bError                : BOOL;
    enErrorID             : ERROR_CODE;
    stErrorIdent          : ERROR_STRUCT;
END_VAR
```

Beispiel

```
fbDaliAddToScene(
    Execute:= bExecute,
    LightLevel:= BYTE#100,
    Group:= bGroup,
    Address:= BYTE#4,
    Scene:= BYTE#12,
    IOstrToken:= stDaliToken,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent);
```

19.13 IH_DaliOff

Kurzbeschreibung

Mit dem Funktionsbaustein **IH_DaliOff** können ein einzelnes Vorschaltgerät, eine Gruppe Vorschaltgeräte oder alle Vorschaltgeräte ausgeschaltet werden. Der Funktionsbaustein muss mit dem Funktionsbaustein [IH_DaliReadWrite](#), [Seite 763](#) aufgerufen werden.

Zielsystem	Bibliothek/Ordner
MLC	RIH_Dali/Basic
MTX	RIH_Dali/Basic
MLD (IL 2G)	RIH_Dali/Basic

Tab. 19-35: Referenztablelle

Schnittstellenbeschreibung

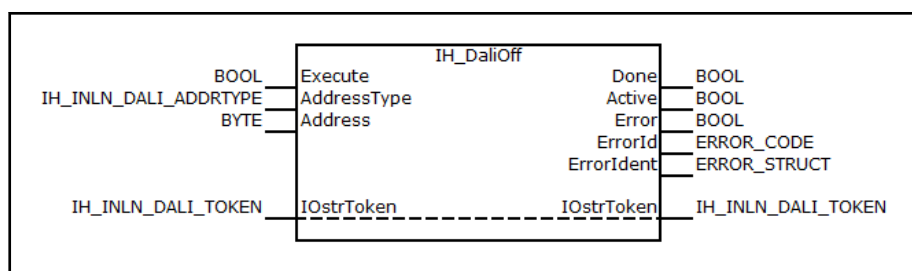


Abb. 19-16: Funktionsbaustein IH_DaliOff

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Steigende Flanke startet Berechnung
	AddressType	IH_INLN_DALI_ADDR- TYPE IH_INLN_DALI_ADDR- TYPE, Seite 806	Aufzählung zur Auswahl des Adresstyps. IH_INLN_DALI_ADDR- TYPE, Seite 806
	Adress	BYTE	Die Adresse hängt vom ausgewählten Typ unter Add- ressType ab. Für Kurzadressen sind Werte von 0 bis 63 und für Gruppenadressen von 0 bis 15 möglich.
VAR_OUTPUT	Done	BOOL	Berechnung beendet
	Active	BOOL	Berechnung findet statt
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Dali, Seite 809
VAR_IN_OUT	IOstrToken	IH_INLN_DALI_TO- KEN IH_INLN_DALI_TO- KEN, Seite 806	Struktur zum Umgang mit Daten zwischen IH_Dali- ReadWrite und anderen Dali-Funktionsbausteinen. IH_INLN_DALI_TOKEN, Seite 806

Tab. 19-36: Schnittstellenvariablen von Funktionsbaustein IH_DaliOff

Min-/Max- und Standard-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
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
AddressType	IH_INLN_DA- LI_ADDR- TYPE	SHORTADDR	BROADCAST	SHORTADDR	Steigende Flanke von Execute
Adress	BYTE	0	Gruppe = 15 Vorschaltgerät = 0 63	0	Steigende Flanke von Execute

Tab. 19-37: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Baus-
teineingänge von IH_DaliOff

Funktionsbeschreibung

Mit dem Funktionsbaustein **IH_DaliOff** können ein einzelnes Vorschaltgerät, eine Gruppe Vorschaltgeräte oder alle Vorschaltgeräte ausgeschaltet werden. Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Execute" auf TRUE gesetzt wird. Geänderte Eingangswerte

te werden erst mit einer neuen steigenden Flanke am Eingang "Execute" angewendet. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Der Funktionsbaustein kann mit einer neuen steigenden Flanke am Eingang "Execute" reinitialisiert werden. Falls während des Betriebs ein Fehler auftritt, wird das Bit "Error" auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Execute" quittiert werden.

Nicht-funktionale Kriterien

Dieser Funktionsbaustein muss zur Kommunikation mit dem Dali-Modul über den Funktionsbaustein **IH_DaliReadWrite** aufgerufen werden.

Kompatibilität

Kompatibel mit Inline-, Profibus- und Profinet-Feldbussen.

Beispiel eines Funktionsbausteins

Zunächst muss der Funktionsbaustein IH_DaliReadWrite instanziiert werden, bevor der Funktionsbaustein IH_DaliOff zur Kommunikation mit dem Dali-Modul instanziiert werden kann. Die Eingangs-/Ausgangsvariable stDaliToken tauscht die Daten zwischen den Funktionsbausteinen IH_DaliReadWrite und IH_DaliOff aus. Deshalb muss diese Variable in beiden Funktionsbausteinen IOstrToken zugewiesen werden.

Im Beispiel ist der Funktionsbaustein IH_DaliOff so konfiguriert, dass er das Vorschaltgerät mit Adresse 4 ausschaltet, da der Adresstyp SHORTADDR ist. Bei einer neuen steigenden Flanke am Eingang "bExecute" werden je nach dem unter AddressType ausgewählten Typ ein Vorschaltgerät, eine Gruppe Vorschaltgeräte oder alle Vorschaltgeräte abgeschaltet.

Variablendeklaration

```
VAR
    fbDaliOff          : IH_DaliOff;
    bExecute           : BOOL;
    stDaliToken         : IH_INLN_DALI_TOKEN;
    bDone              : BOOL;
    bActive            : BOOL;
    bError              : BOOL;
    enErrorID           : ERROR_CODE;
    stErrorIdent        : ERROR_STRUCT;

END_VAR
```

Beispiel

```
fbDaliOff(
    Execute:= bExecute,
    AddressType:= IH_INLN_DALI_ADDRTYPE.SHORTADDR,
    Address:= BYTE#4,
    IOstrToken:= stDaliToken,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent);
```

19.14 IH_DaliOn

Kurzbeschreibung

Mit dem Funktionsbaustein **IH_DaliOn** können ein einzelnes Vorschaltgerät, eine Gruppe Vorschaltgeräte oder alle Vorschaltgeräte eingeschaltet werden. Außerdem kann die Helligkeit auf Min, Mid oder Max gestellt werden. Der Funktionsbaustein muss mit dem Funktionsbaustein **IH_DaliReadWrite**, [Seite 763](#) aufgerufen werden.

Zielsystem	Bibliothek/Ordner
MLC	RIH_Dali/Basic
MTX	RIH_Dali/Basic
MLD (IL 2G)	RIH_Dali/Basic

Tab. 19-38: Referenztabelle

Schnittstellenbeschreibung

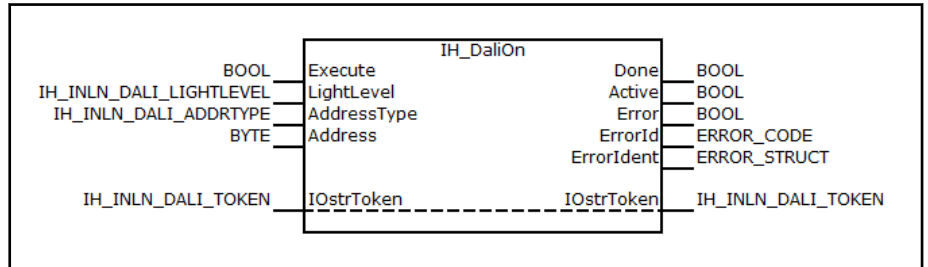


Abb. 19-17: Funktionsbaustein IH_DaliOn

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Steigende Flanke startet Berechnung
	LightLevel	IH_INLN_DALI_LIGHTLEVEL	Der Wert gibt die Helligkeit an. Weitere Informationen enthält der Abschnitt zu Strukturen und Aufzählungen IH_INLN_DALI_LIGHTLEVEL, Seite 807
	AddressType	IH_INLN_DALI_ADDRTYPE	Aufzählung zur Auswahl des Adresstyps. IH_INLN_DALI_ADDRTYPE, Seite 806
	Adress	BYTE	Die Adresse hängt vom ausgewählten Typ unter AddressType ab. Für Kurzadressen sind Werte von 0 bis 63 und für Gruppenadressen von 0 bis 15 möglich.
VAR_OUTPUT	Done	BOOL	Berechnung beendet
	Active	BOOL	Berechnung findet statt
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Dali, Seite 809
VAR_IN_OUT	IOstrToken	IH_INLN_DALI_TOKEN	Struktur zum Umgang mit Daten zwischen IH_Dali-ReadWrite und anderen Dali-Funktionsbausteinen. IH_INLN_DALI_TOKEN, Seite 806

Tab. 19-39: Schnittstellenvariablen von Funktionsbaustein IH_DaliOn

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
LightLevel	IH_INLN_DALI_LIGHTLEVEL	MIN_LEVEL	MAX_LEVEL	MAX_LEVEL	Steigende Flanke von Execute
AddressType	IH_INLN_DALI_ADDRTYPE	SHORTADDR	BROADCAST	SHORTADDR	Steigende Flanke von Execute
Adress	BYTE	0	Gruppe = 15 Vorschaltgerät = 63	0	Steigende Flanke von Execute

Tab. 19-40: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_DaliOn

Funktionsbeschreibung

Mit dem Funktionsbaustein können ein einzelnes Vorschaltgerät, eine Gruppe Vorschaltgeräte oder alle Vorschaltgeräte eingeschaltet werden. Außerdem kann die Helligkeit über den Eingang „LightLevel“ auf Min, Mid oder Max gestellt werden. Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang „Execute“ auf TRUE gesetzt wird. Geänderte Eingangswerte werden erst mit einer neuen steigenden Flanke am Eingang „Execute“ angewendet. Während des Betriebs wird der Ausgang „Active“ auf TRUE gesetzt. Bei Abschluss wird das Bit „Done“ auf TRUE gesetzt. Der Funktionsbaustein kann mit einer neuen steigenden Flanke am Eingang „Execute“ reinitialisiert werden. Falls während des Betriebs ein Fehler auftritt, wird das Bit „Error“ auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang „Execute“ quittiert werden.

Nicht-funktionale Kriterien

Dieser Funktionsbaustein muss zur Kommunikation mit dem Dali-Modul über den Funktionsbaustein **IH_DaliReadWrite** aufgerufen werden.

Kompatibilität

Kompatibel mit Inline-, Profibus- und Profinet-Feldbussen.

Beispiel eines Funktionsbausteins

Zunächst muss der Funktionsbaustein IH_DaliReadWrite instanziiert werden, bevor der Funktionsbaustein IH_DaliOn zur Kommunikation mit dem Dali-Modul instanziiert werden kann. Die Eingangs-/Ausgangsvariable stDaliToken tauscht die Daten zwischen den Funktionsbausteinen IH_DaliReadWrite und IH_DaliOn aus. Deshalb muss diese Variable in beiden Funktionsbausteinen IOstrToken zugewiesen werden.

Im Beispiel schaltet der Funktionsbaustein IH_DaliOn das Vorschaltgerät mit Kurzadresse 2 ein und setzt seine Helligkeit auf Max. Bei einer neuen steigenden Flanke am Eingang „bExecute“ werden ein Vorschaltgerät, eine Gruppe Vorschaltgeräte oder alle Vorschaltgeräte eingeschaltet und je nach Auswahl am Eingang „LightLevel“ ihre Helligkeit auf Min, Mid oder Max gesetzt.

Variablendeklaration

```
VAR
    fbDaliOn          : IH_DaliOn;
    bExecute          : BOOL;
    stDaliToken        : IH_INLN_DALI_TOKEN;
    bDone              : BOOL;
    bActive            : BOOL;
    bError             : BOOL;
```

```

enErrorID          : ERROR_CODE;
stErrorIdent       : ERROR_STRUCT;
END_VAR

```

Beispiel

```

fbDaliOn(
  Execute:= bExecute,
  LightLevel:= IH_INLN_DALI_LIGHTLEVEL.MAX_LEVEL,
  AddressType:= IH_INLN_DALI_ADDRTYPE.SHORTADDR,
  Address:= BYTE#2,
  IOstrToken:= stDaliToken,
  Done=> bDone,
  Active=> bActive,
  Error=> bError,
  ErrorID=> enErrorID,
  ErrorIdent=> stErrorIdent);

```

19.15 IH_DaliSearchAddress

Kurzbeschreibung Der Funktionsbaustein **IH_DaliSearchAddress** speichert die vom Nutzer angegebene Suchadresse. Der Funktionsbaustein muss mit dem Funktionsbaustein **IH_DaliReadWrite**, Seite 763 aufgerufen werden.

Zielsystem	Bibliothek/Ordner
MLC	RIH_Dali/Advanced
MTX	RIH_Dali/Advanced
MLD (IL 2G)	RIH_Dali/Advanced

Tab. 19-41: Referenztable

Schnittstellenbeschreibung

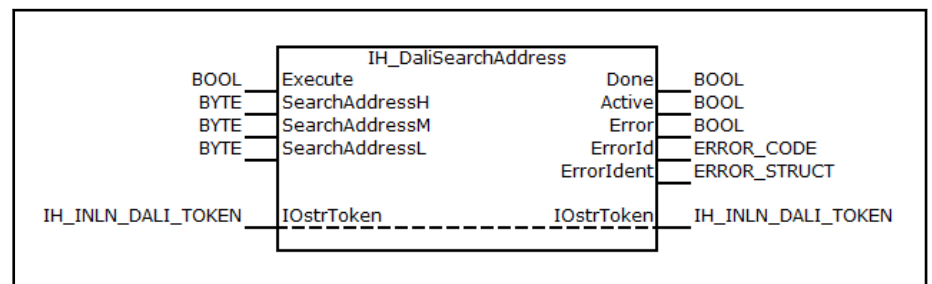


Abb. 19-18: Funktionsbaustein IH_DaliSearchAddress

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Steigende Flanke startet Berechnung
	SearchAddressH	BYTE	Höherwertiges Byte der Suchadresse
	SearchAddressM	BYTE	Mittleres Byte der Suchadresse
	SearchAddressL	BYTE	Niederwertiges Byte der Suchadresse

E/A-Typ	Name	Datentyp	Beschreibung
VAR_OUTPUT	Done	BOOL	Berechnung beendet
	Active	BOOL	Berechnung findet statt
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Dali, Seite 809
VAR_IN_OUT	IOstrToken	IH_INLN_DALI_TOKEN IH_INLN_DALI_TOKEN, Seite 806	Struktur zum Umgang mit Daten zwischen IH_Dali-ReadWrite und anderen Dali-Funktionsbausteinen. IH_INLN_DALI_TOKEN, Seite 806

Tab. 19-42: Schnittstellenvariablen von Funktionsbaustein IH_DaliSearchAddress

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
SearchAddressH	BYTE	0	255	0	Steigende Flanke von Execute
SearchAddressM	BYTE	0	255	0	Steigende Flanke von Execute
SearchAddressL	BYTE	0	255	0	Steigende Flanke von Execute

Tab. 19-43: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_DaliSearchAddress

Funktionsbeschreibung

Der Funktionsbaustein speichert die vom Nutzer angegebene Suchadresse. Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Execute" auf TRUE gesetzt wird. Geänderte Eingangswerte werden erst mit einer neuen steigenden Flanke am Eingang "Execute" angewendet. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Der Funktionsbaustein kann mit einer neuen steigenden Flanke am Eingang "Execute" reinitialisiert werden. Falls während des Betriebs ein Fehler auftritt, wird das Bit "Error" auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an.

Der Fehler kann dann mit einer steigenden Flanke am Eingang "Execute" quittiert werden.

Nicht-funktionale Kriterien

Dieser Funktionsbaustein muss zur Kommunikation mit dem Dali-Modul über den Funktionsbaustein **IH_DaliReadWrite** aufgerufen werden.

Kompatibilität

Kompatibel mit Inline-, Profibus-, Profinet- und Sercos-III-Feldbussen.

Beispiel eines Funktionsbausteins

Zunächst muss der Funktionsbaustein IH_DaliReadWrite instanziiert werden, bevor der Funktionsbaustein IH_DaliSearchAddress zur Kommunikation mit

dem Dali-Modul instanziiert werden kann. Die Eingangs-/Ausgangsvariable stDaliToken tauscht die Daten zwischen den Funktionsbausteinen IH_DaliReadWrite und IH_DaliSearchAddress aus. Deshalb muss diese Variable in beiden Funktionsbausteinen IOstrToken zugewiesen werden.

Im Beispiel ist der Funktionsbaustein IH_DaliSearchAddress so konfiguriert, dass er die Werte von bySearchAddressH,

bySearchAddressM und bySearchAddressL zur Suchadresse des Vorschaltgeräts verschiebt. Bei einer neuen steigenden Flanke am Eingang "bExecute" wird die Nutzereingabe als neue Suchadresse des Vorschaltgeräts zugewiesen.

Variablendeklaration

```
VAR
    fbDaliSearchAddress: IH_DaliSearchAddress;
    bExecute: BOOL;
    bySearchAddressH: BYTE;
    bySearchAddressM: BYTE;
    bySearchAddressL: BYTE;
    stDaliToken: IH_INLN_DALI_TOKEN;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
END_VAR
```

Beispiel

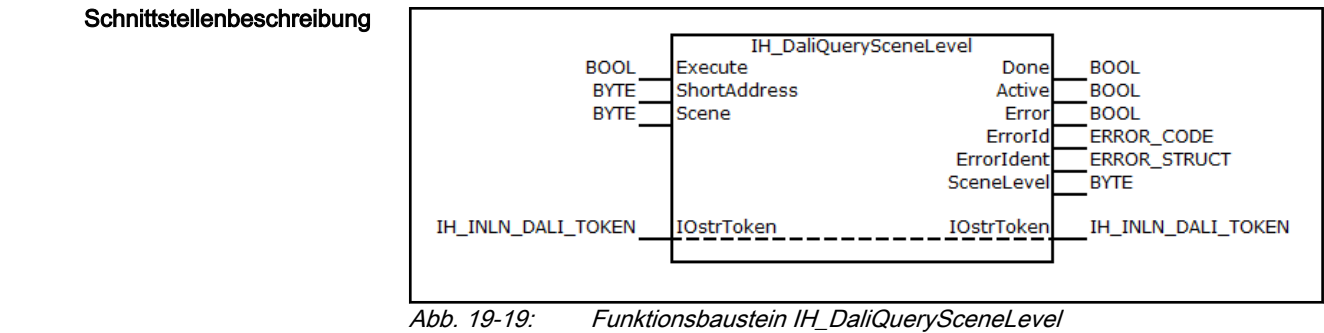
```
fbDaliSearchAddress(
    Execute:= bExecute,
    SearchAddressH:= bySearchAddressH,
    SearchAddressM:= bySearchAddressM,
    SearchAddressL:= bySearchAddressL,
    IOstrToken:= stDaliToken,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent);
```

19.16 IH_DaliQuerySceneLevel

Kurzbeschreibung Der Funktionsbaustein IH_DaliQuerySceneLevel dient zum Auslesen der Szenenebene des Vorschaltgeräts. Der Funktionsbaustein muss mit dem Funktionsbaustein [IH_DaliReadWrite, Seite 763](#) aufgerufen werden.

Zielsystem	Bibliothek/Ordner
MLC	RIH_Dali/Advanced
MTX	RIH_Dali/Advanced
MLD (IL 2G)	RIH_Dali/Advanced

Tab. 19-44: Referenztable



E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Steigende Flanke startet Berechnung
	ShortAddress	BYTE	Adresse des Vorschaltgeräts (0 bis 63)
	Scene	BYTE	Szenennummer, Wertebereich von 0 bis 15
	Done	BOOL	Berechnung beendet
VAR_OUTPUT	Active	BOOL	Berechnung findet statt
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Dali, Seite 809
	SceneLevel	BYTE	Szenenebene des Vorschaltgeräts
VAR_IN_OUT	IOstrToken	IH_INLN_DALI_TOKEN IH_INLN_DALI_TOKEN, Seite 806	Struktur zum Umgang mit Daten zwischen IH_Dali-ReadWrite und anderen Dali-Funktionsbausteinen. IH_INLN_DALI_TOKEN, Seite 806

Tab. 19-45: Schnittstellenvariablen von Funktionsbaustein IH_DaliQuerySceneLevel

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
ShortAddress	BYTE	0	63	0	Steigende Flanke von Execute
Scene	BYTE	0	15	0	Steigende Flanke an Execute

Tab. 19-46: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_DaliQuerySceneLevel

Funktionsbeschreibung

Der Funktionsbaustein dient zum Auslesen der Szenenebene des Vorschaltgeräts. Die Szenenebene hat einen Wert zwischen 0 und 15. Beträgt der Wert von "SceneLevel" 255, ist der Wert ungültig oder die Szene ist nicht konfiguriert. Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Execute" auf TRUE gesetzt wird. Geänderte Eingangswerte werden erst mit einer neuen steigenden Flanke am Eingang "Execute" angewendet. Während des Betriebs wird der Ausgang "Active" auf

TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Der Funktionsbaustein kann mit einer neuen steigenden Flanke am Eingang "Execute" reinitialisiert werden. Falls während des Betriebs ein Fehler auftritt, wird das Bit "Error" auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Execute" quittiert werden.

Nicht-funktionale Kriterien

Dieser Funktionsbaustein muss zur Kommunikation mit dem Dali-Modul über den Funktionsbaustein IH_DaliReadWrite aufgerufen werden.

Kompatibilität

Kompatibel mit Inline-, Profibus-, Profinet- und sercos-III-Feldbussen.

Beispiel eines Funktionsbausteins

Zunächst muss der Funktionsbaustein IH_DaliReadWrite instanziiert werden, bevor der Funktionsbaustein IH_DaliQuerySceneLevel zur Kommunikation mit dem Dali-Modul instanziiert werden kann. Die Eingangs-/Ausgangsvariable stDaliToken tauscht die Daten zwischen den Funktionsbausteinen IH_DaliReadWrite und IH_DaliQuerySceneLevel aus. Deshalb muss diese Variable in beiden Funktionsbausteinen IOstrToken zugewiesen werden.

Im Beispiel ist der Funktionsbaustein IH_DaliQuerySceneLevel so konfiguriert, dass er die Szenenebene von Szene Nummer 8 von Vorschaltgeräteadresse 2 abfragt. Eine neue steigende Flanke am Eingang "bExecute" fragt die Szenenebene eines bestimmten Vorschaltgeräts ab.

Variablendeklaration

```
VAR
    fbDaliQuerySceneLevel: IH_DaliQuerySceneLevel;
    bExecute: BOOL;
    stDaliToken: IH_INLN_DALI_TOKEN;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
    bySceneLevel: BYTE;
END_VAR
```

Beispiel

```
fbDaliQuerySceneLevel(
    Execute:= bExecute,
    ShortAddress:= BYTE#2,
    Scene:= BYTE#8,
    IOstrToken:= stDaliToken,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent,
    SceneLevel=> bySceneLevel);
```

19.17 IH_DaliProgramShortAddress

Kurzbeschreibung

Der Funktionsbaustein **IH_DaliProgramShortAddress** weist Vorschaltgeräten eine Kurzadresse durch zufällige Adressvergabe oder physikalische Auswahl während der Initialisierung zu. Der Funktionsbaustein muss mit dem Funktionsbaustein [IH_DaliReadWrite](#), [Seite 763](#) aufgerufen werden.

Zielsystem	Bibliothek/Ordner
MLC	RIH_Dali/Advanced
MTX	RIH_Dali/Advanced
MLD (IL 2G)	RIH_Dali/Advanced

Tab. 19-47: Referenztabelle

Schnittstellenbeschreibung

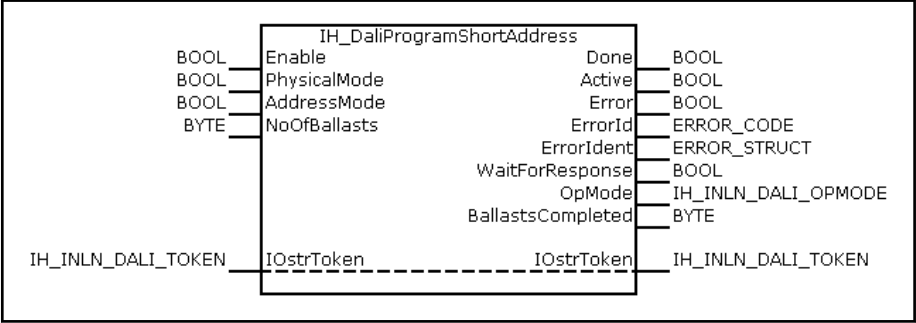


Abb. 19-20: Funktionsbaustein IH_DaliProgramShortAddress

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Funktionsbaustein aktiv, falls TRUE
	PhysicalMode	BOOL	Falls TRUE ist die physikalische Auswahl eingestellt Falls FALSE ist die Zufallsvergabe eingestellt
	AddressMode	BOOL	Falls TRUE sind alle Vorschaltgeräte ausgewählt Falls FALSE sind die Vorschaltgeräte ohne Adresse ausgewählt
	NoOfBallasts	BYTE	Anzahl der Vorschaltgeräte die der Kurzadresse zugeordnet werden sollen

E/A-Typ	Name	Datentyp	Beschreibung
VAR_OUTPUT	Done	BOOL	Berechnung beendet
	Active	BOOL	Berechnung findet statt
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Dali, Seite 809
	WaitForResponse	BOOL	Bei PhysicalMode, Warten auf physikalische Antwort der Vorschaltgeräte
	OpMode	IH_INLN_DALI_OP-MODE IH_INLN_DALI_OP-MODE, Seite 807	Aktuelle Betriebsart IH_INLN_DALI_OPMODE, Seite 807
	BallastsCompleted	BYTE	Gesamtzahl zu Vorschaltgeräten zugeordneter Adressen
VAR_IN_OUT	IOstrToken	IH_INLN_DALI_TOKEN IH_INLN_DALI_TOKEN, Seite 806	Struktur zum Umgang mit Daten zwischen IH_Dali-ReadWrite und anderen Dali-Funktionsbausteinen. IH_INLN_DALI_TOKEN, Seite 806

Tab. 19-48: Schnittstellenvariablen von Funktionsbaustein IH_DaliProgramShortAddress

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
PhysicalMode	BOOL	FALSE	TRUE	FALSE	Steigende Flanke an Enable
AddressMode	BOOL	FALSE	TRUE	FALSE	Steigende Flanke an Enable
NoOfBallasts	BYTE	1	64	0	Steigende Flanke an Enable

Tab. 19-49: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_DaliProgramShortAddress

Funktionsbeschreibung

Der Funktionsbaustein weist Vorschaltgeräten eine Kurzadresse durch zufällige Adressvergabe oder physikalische Auswahl während der Initialisierung zu.

Ist "PhysicalMode" TRUE, wird das Vorschaltgerät im physikalischen Auswahlmodus adressiert. Andernfalls wird die Zufallsvergabe angewendet.

Im Folgenden werden unterschiedliche Adressierungsmodi aufgeführt und geben einen Überblick über das interne Verhalten des Funktionsbausteins.

Zufallsvergabe: (während der Ausführung im Funktionsbaustein stattfindende Verfahren)

- Der Algorithmus startet mit dem Befehl "INITIALIZE" für alle Vorschaltgeräte oder für Vorschaltgeräte ohne Adresse

- Sendebefehl: "RANDOMIZE": alle Vorschaltgeräte erhalten eine zufällige Binärzahl (BRN)
- Die Suchadresse wird ebenso wie die Zufallsadresse bestimmt
- Das Steuerteil sucht mit dem Algorithmus "COMPARE" das Vorschaltgerät mit der niedrigsten BRN. Das Vorschaltgerät mit der BRN wird gefunden
- Diese BRN wird mit IH_DaliSearchAddress in die Suchadresse verschoben.
- Das gefundene Vorschaltgerät erhält eine 6-Bit-Kurzadresse (StartAdresse) durch Ausführen des Befehls "PROGRAM SHORT ADDRESS"
- Die programmierte Kurzadresse wird mit dem Befehl "VERIFY SHORT ADDRESS" verifiziert
- Das gefundene Vorschaltgerät wird aus der Suche entfernt, indem der Befehl "WITHDRAW" ausgeführt wird
- Der Algorithmus wird ab Schritt 3 wiederholt, bis der Zähler den Wert von "NoOfBallasts" erreicht
- Erreicht der Zähler den Wert von "NoOfBallasts", stoppt das Verfahren mit dem Befehl "TERMINATE"

Physikalische Adressierung: (während der Ausführung im Funktionsbaustein stattfindende Verfahren)

- Das Steuerteil sendet den Befehl "INITIALIZE" an alle Vorschaltgeräte oder Vorschaltgeräte ohne Adresse
- Das Steuerteil sendet den Befehl "Physical selection" zur Aktivierung der physikalischen Adressauswahl
- Das Steuerteil wiederholt den Befehl "QUERY SHORT ADDRESS" regelmäßig, bis ein Vorschaltgerät antwortet. Das Vorschaltgerät antwortet, indem es seine Lampe trennt



Damit ein Vorschaltgerät ausgewählt wird, muss der Nutzer dessen Lampe trennen und wieder anschließen. (Das Wiederanschließen der Lampe muss innerhalb von 5 Sekunden erfolgen.)

- Das ausgewählte Vorschaltgerät erhält eine 6-Bit-Kurzadresse durch Ausführen des Befehls "PROGRAM SHORT ADDRESS"
- Die wieder an das Vorschaltgerät angeschlossene und verfügbare Lampe wird mit IH_DaliQueryStatus gefunden
- Das Steuerteil sendet "RECALL MIN LEVEL" und "RECALL MAX LEVEL" an die Kurzadresse, um 5 Sekunden lang visuelles Feedback zu erhalten. Dadurch wird bestätigt, dass die Kurzadresse erfolgreich zugewiesen wurde.
- Der Algorithmus wird ab Schritt 3 wiederholt, bis der Zähler den Wert von "NoOfBallasts" erreicht
- Erreicht der Zähler den Wert von "NoOfBallasts", stoppt das Verfahren mit dem Befehl "TERMINATE"

Ist "AddressMode" TRUE werden alle Vorschaltgeräte ausgewählt. Bei FALSE werden Vorschaltgeräte ohne Kurzadresse für die Adressierung ausgewählt. Während der Funktionsbaustein auf eine physikalische Antwort vom Vorschaltgerät wartet, ist "WaitForResponse" TRUE. "ballastsCompleted" zählt die Anzahl komplett adressierter Vorschaltgeräte. "OpMode" legt den aktuellen Betriebsmodus des Funktionsbausteins fest.

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Enable" auf TRUE gesetzt wird. Geänderte Eingabewerte werden erst mit einer neuen steigenden Flanke am Eingang "Enable" angewendet. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Der Funktionsbaustein kann mit einer neuen steigenden Flanke am Eingang "Execute" reinitialisiert werden. Falls während des Betriebs ein Fehler auftritt, wird das Bit "Error" auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an.

Der Fehler kann dann mit einer steigenden Flanke am Eingang "Enable" quittiert werden.

Besonderheiten bei der Handhabung

- Bei der Adressierung aller Vorschaltgeräte muss die Lampe eingeschaltet (ON) sein
- Ist die Adressierung mit IH_DaliProgramShortAddress abgeschlossen, muss für eine Neuadressierung das Dali-System aus- und wieder eingeschaltet werden

Nicht-funktionale Kriterien

Dieser Funktionsbaustein muss zur Kommunikation mit dem Dali-Modul über den Funktionsbaustein **IH_DaliReadWrite** aufgerufen werden.

Kompatibilität

Kompatibel mit Inline-, Profibus-, Profinet- und sercos-III-Feldbussen.

Beispiel eines Funktionsbausteins

Zunächst muss der Funktionsbaustein IH_DaliReadWrite instanziiert werden, bevor der Funktionsbaustein IH_DaliProgramShortAddress zur Kommunikation mit dem Dali-Modul instanziiert werden kann. Die Eingangs-/Ausgangsvariable stDaliToken tauscht die Daten zwischen den Funktionsbausteinen IH_DaliReadWrite und IH_DaliProgramShortAddress aus. Deshalb muss diese Variable in beiden Funktionsbausteinen IOstrToken zugewiesen werden.

In den Beispielen weist der Funktionsbaustein IH_DaliProgramShortAddress Vorschaltgeräten mit der Zufallsvergabe bzw. der physikalischen Auswahl Adressen zu. Eine neue steigende Flanke am Eingang „bExecute“ ordnet den Vorschaltgeräten die Adressen zu. Hier ist „NoOfballasts“ BYTE#20.

Variablendeklaration

```
VAR
    fbDaliProgramShortAddress: IH_DaliProgramShortAddress;
    bEnable: BOOL;
    stDaliToken: IH_INLN_DALI_TOKEN;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
    bWaitForResponse: BOOL;
    enOpMode: IH_INLN_DALI_OPMODE;
    byBallastsCompleted: BYTE;
END_VAR
```

Dieser Funktionsbaustein verwendet für die Adressierung aller verfügbaren Vorschaltgeräte die Zufallsvergabe. Die Adressierung beginnt bei Null und wird inkrementiert, bis keine weiteren Vorschaltgeräte gefunden werden.

Zufällige Adressvergabe für alle Vorschaltgeräte

```
fbDaliProgramShortAddress (
    Enable:= bEnable,
    PhysicalMode:= FALSE,
    AddressMode:= TRUE,
    NoOfBallasts:= BYTE#20,
    IOstrToken:= stDaliToken,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent,
    WaitForResponse=> bWaitForResponse,
    OpMode=> enOpMode,
    BallastsCompleted=> byBallastsCompleted);
```

Dieser Funktionsbaustein verwendet die physikalische Auswahl nur für Vorschaltgeräte ohne Kurzadresse. Er durchsucht die im Netzwerk verfügbaren Adressen und weist sie den physikalisch ausgewählten Vorschaltgeräten zu.

Physikalische Auswahl für Vorschaltgeräte ohne Kurzadresse

```
fbDaliProgramShortAddress (
    Enable:= bEnable,
    PhysicalMode:= TRUE,
    AddressMode:= FALSE,
    NoOfBallasts:= BYTE#20,
    IOstrToken:= stDaliToken,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent,
    WaitForResponse=> bWaitForResponse,
    OpMode=> enOpMode,
    BallastsCompleted=> byBallastsCompleted);
```

19.18 IH_DaliStoreParameter

Kurzbeschreibung Der Funktionsbaustein **IH_DaliStoreParameter** speichert den Inhalt des DTR in die entsprechenden Parameter des Vorschaltgeräts, wie Fade Rate, Fade Time, Max Level, Min Level, Power On Level, System Failure Level. Der Funktionsbaustein muss mit dem Funktionsbaustein **IH_DaliReadWrite**, [Seite 763](#) aufgerufen werden.

Zielsystem	Bibliothek/Ordner
MLC	RIH_Dali/Advanced
MTX	RIH_Dali/Advanced
MLD (IL 2G)	RIH_Dali/Advanced

Tab. 19-50: Referenztablelle

Schnittstellenbeschreibung

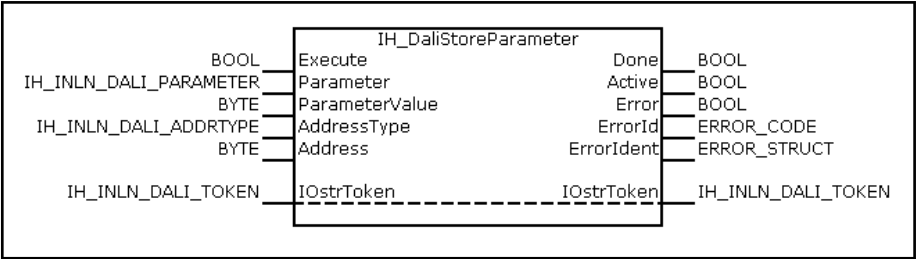


Abb. 19-21: Funktionsbaustein IH_DaliStoreParameter

E/A-Art	Name	Type	Description
VAR_INPUT	Execute	BOOL	Steigende Flanke startet Berechnung
	Parameter	IH_INLN_DALI_PARAMETER IH_INLN_DALI_PARAMETER, Seite 808	Aufzählung zur Parameterauswahl IH_INLN_DALI_PARAMETER, Seite 808
	ParameterValue	BYTE	Der Parameterwert hängt von der Auswahl in Store-Parameter ab
	AddressType	IH_INLN_DALI_ADDR- TYPE IH_INLN_DALI_ADDR- TYPE, Seite 806	Aufzählung zur Auswahl des Adresstyps IH_INLN_DALI_ADDR- TYPE, Seite 806
	Adress	BYTE	Die Adresse hängt vom ausgewählten Typ unter Ad- ressType ab. Für Kurzadressen sind Werte von 0 bis 63 und für Gruppenadressen von 0 bis 15 möglich.
VAR_OUTPUT	Done	BOOL	Berechnung beendet
	Active	BOOL	Berechnung findet statt
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Dali, Seite 809
VAR_IN_OUT	IOstrToken	IH_INLN_DALI_TO- KEN IH_INLN_DALI_TO- KEN, Seite 806	Struktur zum Umgang mit Daten zwischen IH_Dali- ReadWrite und anderen Dali-Funktionsbausteinen. IH_INLN_DALI_TOKEN, Seite 806

Tab. 19-51: Schnittstellenvariablen von Funktionsbaustein *IH_DaliStoreParameter*

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
Parameter	IH_INLN_DALI_PARAMETER	FADE_RATE	SYSTEM_FAILU- RE_LEVEL	FADE_RATE	Steigende Flanke von Execute
ParameterValue	BYTE	-	-	-	Steigende Flanke von Execute
AddressType	IH_INLN_DA- LI_ADDR- TYPE	SHORTADDR	BROADCAST	SHORTADDR	Steigende Flanke von Execute
Adress	BYTE	0	Kurzadresse = 63 Gruppenadresse = 15	0	Steigende Flanke von Execute

Tab. 19-52: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von *IH_DaliStoreParameter*

Funktionsbeschreibung

Der Funktionsbaustein **IH_DaliStoreParameter** speichert den Inhalt des DTR in die entsprechenden Parameter, wie Fade Rate, Fade Time, Max Level, Min Level, Power On Level, System Failure Level, einzelner Vorschaltgeräte, einer Gruppe Vorschaltgeräte oder aller Vorschaltgeräte, je nach dem unter AddressType ausgewählten Typ. Parameter können in der Parameterauflistung ausgewählt werden. Mit der Eingangsvariable ParameterValue lassen sich alle Parameterwerte eingeben.

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Execute" auf TRUE gesetzt wird. Geänderte Eingangswerte werden erst mit einer neuen steigenden Flanke am Eingang "Execute" angewendet. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Der Funktionsbaustein kann mit einer neuen steigenden Flanke am Eingang "Execute" reinitialisiert werden. Falls während des Betriebs ein Fehler auftritt, wird das Bit "Error" auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Execute" quittiert werden.

Nicht-funktionale Kriterien

Dieser Funktionsbaustein muss zur Kommunikation mit dem Dali-Modul über den Funktionsbaustein **IH_DaliReadWrite** aufgerufen werden.

Kompatibilität

Kompatibel mit Inline-, Profibus-, Profinet- und Sercos-III-Feldbussen.

Beispiel eines Funktionsbausteins

Zunächst muss der Funktionsbaustein IH_DaliReadWrite instanziiert werden, bevor der Funktionsbaustein IH_DaliStoreParameter zur Kommunikation mit dem Dali-Modul instanziiert werden kann. Die Eingangs-/Ausgangsvariable stDaliToken tauscht die Daten zwischen den Funktionsbausteinen IH_DaliReadWrite und IH_DaliStoreParameter aus. Deshalb muss diese Variable in beiden Funktionsbausteinen IOstrToken zugewiesen werden.

Im Beispiel speichert der Funktionsbaustein IH_DaliStoreParameter den Wert 120 in den Parameter "Max Level" des Vorschaltgeräts mit Kurzadresse 1.

Variablendeklaration

```

VAR
    fbDaliStoreParameter          : IH_DaliStoreParameter;
    bExecute                     : BOOL;
    stDaliToken                   : IH_INLN_DALI_TOKEN;
    bDone                         : BOOL;
    bActive                       : BOOL;
    bError                        : BOOL;
    enErrorID                     : ERROR_CODE;
    stErrorIdent                  : ERROR_STRUCT;
END_VAR

```

Beispiel

```

fbDaliStoreParameter (
    Execute:= bExecute,
    Parameter := IH_INLN_DALI_PARAMETER.MAX_LEVEL,
    ParameterValue := BYTE#120,
    AddressType:= IH_INLN_DALI_ADDRTYPE.SHORTADDR,
    Address:= BYTE#1,
    IOstrToken:= stDaliToken,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent);

```

19.19 IH_DaliReset

Kurzbeschreibung Der Funktionsbaustein **IH_DaliReset** setzt alle Werte im nichtflüchtigen Speicher des Vorschaltgeräts zurück. Der Funktionsbaustein muss mit dem Funktionsbaustein **IH_DaliReadWrite**, Seite 763 aufgerufen werden.

Zielsystem	Bibliothek/Ordner
MLC	RIH_Dali/Advanced
MTX	RIH_Dali/Advanced
MLD (IL 2G)	RIH_Dali/Advanced

Tab. 19-53: Referenztabelle

Schnittstellenbeschreibung

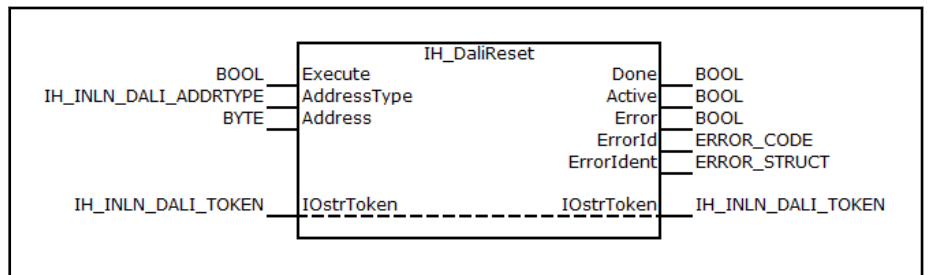


Abb. 19-22: Funktionsbaustein IH_DaliReset

E/A-Art	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Steigende Flanke startet Berechnung
	AddressType	IH_INLN_DALI_ADDR- TYPE IH_INLN_DALI_ADDR- TYPE, Seite 806	Aufzählung zur Auswahl des Adresstyps. IH_INLN_DALI_ADDRTYPE, Seite 806
	Adress	BYTE	Die Adresse hängt vom ausgewählten Typ unter Add- ressType ab. Für Kurzadressen sind Werte von 0 bis 63 und für Gruppenadressen von 0 bis 15 möglich.
VAR_OUTPUT	Done	BOOL	Berechnung beendet
	Active	BOOL	Berechnung findet statt
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Dali, Seite 809
VAR_IN_OUT	IOstrToken	IH_INLN_DALI_TO- KEN IH_INLN_DALI_TO- KEN, Seite 806	Struktur zum Umgang mit Daten zwischen IH_Dali- ReadWrite und anderen Dali-Funktionsbausteinen. IH_INLN_DALI_TOKEN, Seite 806

Tab. 19-54: Schnittstellenvariablen von Funktionsbaustein IH_DaliReset

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
AddressType	IH_INLN_DALI_ADDRTYPE	SHORTADDR	BROADCAST	SHORTADDR	Steigende Flanke an Execute
Adress	BYTE	0	Gruppe = 15 Vorschaltgerät = 63	0	Steigende Flanke an Execute

Tab. 19-55: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_DaliReset

Funktionsbeschreibung

Der Funktionsbaustein **IH_DaliReset** setzt alle Werte im nichtflüchtigen Speicher des Vorschaltgeräts zurück. Die folgende Tabelle enthält die Standardwerte für das Zurücksetzen. Über "AdressType" werden ein einzelnes Vorschaltgerät, eine Gruppe Vorschaltgeräte oder alle verbundenen Vorschaltgeräte ausgewählt. Während der Reset-Befehl ausgeführt wird, kann das Vorschaltgerät ggf. in den nächsten 300 ms keine weiteren Befehle empfangen.

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Execute" auf TRUE gesetzt wird. Geänderte Eingangswerte werden erst mit einer neuen steigenden Flanke am Eingang "Execute" angewendet. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Der Funktionsbaustein kann mit einer neuen steigenden Flanke am Eingang "Execute" reinitialisiert werden. Falls während des Betriebs ein Fehler auftritt, wird das Bit "Error" auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Execute" quittiert werden.



Die unten aufgeführten Rücksetzwerte sind ggf. herstellerabhängig, z. B. Physical Min Level.

Tabelle:

Variable	Reset-Wert
ACTUAL DIM LEVEL	254
POWER ON LEVEL	254
SYSTEM FAILURE LEVEL	254
MIN LEVEL	PHYSICAL MIN LEVEL
MAX LEVEL	254
FADE RATE	7 (45 Schritte/s)
FADE TIME	0 (KEIN ABBLENDEN)
SHORT ADDRESS	KEINE ÄNDERUNG
SEARCH ADDRESS	FF FF FF
RANDOM ADDRESS	FF FF FF
GROUP 0-7	0000 0000 (KEINE GRUPPE)
GROUP 8-15	0000 0000 (KEINE GRUPPE)
SCENE 0-15	255 (MASKE)

Variable	Reset-Wert
STATUS INFORMATION	0?100???
	? : undefiniert
VERSION NUMBER	KEINE ÄNDERUNG
DEVICE TYPE	0
PHYSICAL MIN LEVEL	WERKSEINSTELLUNG

Nicht-funktionale Kriterien

Dieser Funktionsbaustein muss zur Kommunikation mit dem Dali-Modul über den Funktionsbaustein **IH_DaliReadWrite** aufgerufen werden.

Kompatibilität

Kompatibel mit Inline-, Profibus-, Profinet- und sercos-III-Feldbussen.

Beispiel eines Funktionsbausteins

Zunächst muss der Funktionsbaustein IH_DaliReadWrite instanziiert werden, bevor der Funktionsbaustein IH_DaliReset zur Kommunikation mit dem Dali-Modul instanziiert werden kann. Die Eingangs-/Ausgangsvariable stDaliToken tauscht die Daten zwischen den Funktionsbausteinen IH_DaliReadWrite und IH_DaliReset aus. Deshalb muss diese Variable in beiden Funktionsbausteinen IOstrToken zugewiesen werden.

Im Beispiel setzt der Funktionsbaustein IH_DaliReset alle Werte im nicht-flüchtigen Speicher des Vorschaltgeräts zurück. Im ersten Beispiel wird das Vorschaltgerät mit Adresse 1 zurückgesetzt. Im zweiten Beispiel werden alle mit dem Dali-Bus verbundenen Vorschaltgeräte mit dem Sammelbefehl zurückgesetzt.

Variablendeklaration

```

VAR
    fbDaliReset           : IH_DaliReset;
    bExecute              : BOOL;
    stDaliToken            : IH_INLN_DALI_TOKEN;
    bDone                  : BOOL;
    bActive                : BOOL;
    bError                 : BOOL;
    enErrorID              : ERROR_CODE;
    stErrorIdent           : ERROR_STRUCT;

END_VAR

```

Programm mit AddressType = Short Address (Kurzadresse)

```

fbDaliReset(
    Execute:= bExecute,
    AddressType:= IH_INLN_DALI_ADDRTYPE.SHORTADDR,
    Address:= BYTE#1,
    IOstrToken:= stDaliToken,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent);

```

Programm mit AddressType = Broadcast (Sammelbefehl)

```

fbDaliReset(
    Execute:= bExecute,
    AddressType:= IH_INLN_DALI_ADDRTYPE.BROADCAST,
    Address:= ,
    IOstrToken:= stDaliToken,
    Done=> bDone,

```

```
Active=> bActive,  
Error=> bError,  
ErrorID=> enErrorID,  
ErrorIdent=> stErrorIdent);
```

19.20 IH_DaliQueryDTR

Kurzbeschreibung Der Funktionsbaustein **IH_DaliQueryDTR** dient zum Auslesen des Inhalts des Data Transfer Register (DTR) der an den Dali-Bus angeschlossenen Vorschaltgeräte. Dieser Funktionsbaustein muss mit dem Funktionsbaustein [IH_DaliReadWrite, Seite 763](#) aufgerufen werden.

Zielsystem	Bibliothek/Ordner
MLC	RIH_Dali/Advanced
MTX	RIH_Dali/Advanced
MLD (IL 2G)	RIH_Dali/Advanced

Tab. 19-56: Referenztabelle

Schnittstellenbeschreibung

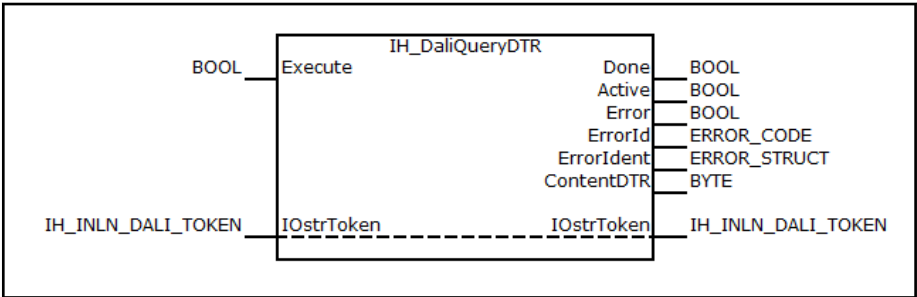


Abb. 19-23: Funktionsbaustein IH_DaliQueryDTR

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Steigende Flanke startet Berechnung
VAR_OUTPUT	Done	BOOL	Berechnung beendet
	Active	BOOL	Berechnung findet statt
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Dali, Seite 809
	ContentDTR	BYTE	DTR-Inhalt
VAR_IN_OUT	IOstrToken	IH_INLN_DALI_TOKEN IH_INLN_DALI_TOKEN, Seite 806	Struktur zum Umgang mit Daten zwischen IH_Dali-ReadWrite und anderen Dali-Funktionsbausteinen. IH_INLN_DALI_TOKEN, Seite 806

Tab. 19-57: Schnittstellenvariablen von Funktionsbaustein IH_DaliQueryDTR

Min-/Max- und Standard-Werte In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich

Tab. 19-58: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_DaliQueryDTR

Funktionsbeschreibung

Der Funktionsbaustein **IH_DaliQueryDTR** dient zum Auslesen des Inhalts des DTR (Data Transfer Register) der an den Dali-Bus angeschlossenen Vorschaltgeräte. Wenn "Execute" TRUE ist, wird das DTR aller angeschlossenen Vorschaltgeräte gelesen und der DTR-Inhaltswert wird in der Ausgabevariable "ContentDTR" gespeichert.

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Execute" auf TRUE gesetzt wird. Geänderte Eingangswerte werden erst mit einer neuen steigenden Flanke am Eingang "Execute" angewendet. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Der Funktionsbaustein kann mit einer neuen steigenden Flanke am Eingang "Execute" reinitialisiert werden. Falls während des Betriebs ein Fehler auftritt, wird das Bit "Error" auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Execute" quittiert werden.

Nicht-funktionale Kriterien

Dieser Funktionsbaustein muss zur Kommunikation mit dem Dali-Modul über den Funktionsbaustein **IH_DaliReadWrite** aufgerufen werden.

Kompatibilität

Kompatibel mit Inline-, Profibus-, Profinet- und sercos-III-Feldbussen.

Beispiel eines Funktionsbausteins

Zunächst muss der Funktionsbaustein **IH_DaliReadWrite** instanziiert werden, bevor der Funktionsbaustein **IH_DaliQueryDTR** zur Kommunikation mit dem Dali-Modul instanziiert werden kann. Die Eingangs-/Ausgangsvariable **stDaliToken** tauscht die Daten zwischen den Funktionsbausteinen **IH_DaliReadWrite** und **IH_DaliQueryDTR** aus. Deshalb muss diese Variable in beiden Funktionsbausteinen **IOstrToken** zugewiesen werden.

Im Beispiel liest der Funktionsbaustein **IH_DaliQueryDTR** den Inhalt des DTR (Data Transfer Register) der an den Dali-Bus angeschlossenen Vorschaltgeräte. Wenn "Execute" TRUE ist, wird das DTR aller angeschlossenen Vorschaltgeräte gelesen und der DTR-Inhaltswert wird in der Ausgabevariable "ContentDTR" gespeichert.

Variablendeklaration

VAR

```
fbDaliQueryDTR: IH_DaliQueryDTR;
bExecute: BOOL;
stDaliToken: IH_INLN_DALI_TOKEN;
bDone: BOOL;
bActive: BOOL;
bError: BOOL;
enErrorID: ERROR_CODE;
stErrorIdent: ERROR_STRUCT;
byContentDTR: BYTE;
```

END_VAR

Beispiel

```
fbDaliQueryDTR(  
    Execute:= bExecute,  
    IOstrToken:= stDaliToken,  
    Done=> bDone,  
    Active=> bActive,  
    Error=> bError,  
    ErrorID=> enErrorID,  
    ErrorIdent=> stErrorIdent,  
    ContentDTR=> byContentDTR);
```

19.21 IH_DaliQueryParameter

Kurzbeschreibung Der Funktionsbaustein IH_DaliQueryParameter liest die Parameter Fade Rate, Fade Time, Min Level, Max Level, Power On Level, Physical Min Level und System Failure Level des Vorschaltgeräts Der Funktionsbaustein muss mit dem Funktionsbaustein [IH_DaliReadWrite](#), [Seite 763](#) aufgerufen werden.

Zielsystem	Bibliothek/Ordner
MLC	RIH_Dali/Advanced
MTX	RIH_Dali/Advanced
MLD (IL 2G)	RIH_Dali/Advanced

Tab. 19-59: Referenztabelle

Schnittstellenbeschreibung

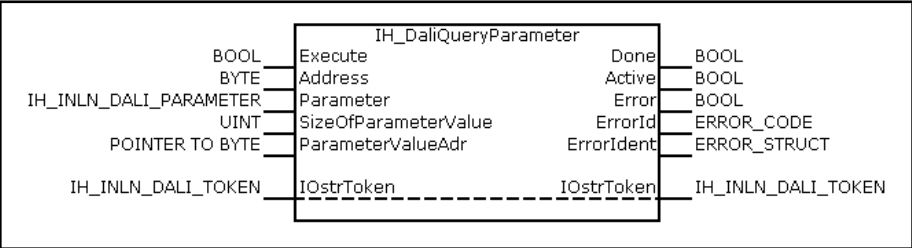


Abb. 19-24: Funktionsbaustein IH_DaliQueryParameter

E/A-Typ	Name	Datentyp	Description		
VAR_INPUT	Execute	BOOL	Steigende Flanke startet Berechnung		
	Adress	BYTE	Adresse des Vorschaltgeräts (0 bis 63 oder 255)		
			Adresse	SizeOf ParameterValue	Anz. zu lesender Vorschaltgeräte
			0-63	1	1 Vorschaltgerät
			255	64	64 Vorschaltgeräte
			255	Anz. Vorschaltgeräte von 1 bis 64	Anz. Vorschaltgeräte in SizeOf-ParameterValue
	Parameter	IH_INLN_DALI_PARAMETER IH_INLN_DALI_PARAMETER, Seite 808	Aufzählung zur Parameterauswahl Weitere Informationen enthält der Abschnitt zu Strukturen und Aufzählungen IH_INLN_DALI_PARAMETER, Seite 808		
	SizeOfParameterValue	UINT	Größe des Parameterwerts (1 bis 64)		
	ParameterValueAdr	POINTER_TO_BYTE	Adresse von ParameterValue. (1 Byte oder Array von 1 bis 64)		
			Adresswert	Größe von ParameterValue-Array	
			0 bis 63	1 Byte	
255			Anz. Vorschaltgeräte (1 bis 64)		
VAR_OUTPUT	Done	BOOL	Berechnung beendet		
	Active	BOOL	Berechnung findet statt		
	Error	BOOL	Zeigt einen Fehler an		
	ErrorID	ERROR_CODE	Fehlerklasse		
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Dali, Seite 809		
VAR_IN_OUT	IOstrToken	IH_INLN_DALI_TOKEN IH_INLN_DALI_TOKEN, Seite 806	Struktur zum Umgang mit Daten zwischen IH_Dali-ReadWrite und anderen Dali-Funktionsbausteinen. IH_INLN_DALI_TOKEN, Seite 806		

Tab. 19-60: Schnittstellenvariablen von Funktionsbaustein *IH_DaliQueryParameter*

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
Adress	BYTE	0	255	0	Steigende Flanke von Execute

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Parameter	IH_INLN_DALI_PARAMETER	MAX_LEVEL	PHYSICAL_MIN_LEVEL	MAX_LEVEL	Steigende Flanke von Execute
SizeOfParameter-Value	UINT	1	64	1	Steigende Flanke von Execute

Tab. 19-61: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_DaliQueryParameter

Funktionsbeschreibung

Der Funktionsbaustein IH_DaliQueryParameter liest die Parameter eines Vorschaltgeräts. Der Funktionsbaustein liest die Parameter, wie Fade Rate, Fade Time, Min Level, Max Level, Power On Level, Physical Min Level und System Failure Level, die in der Aufzählung von IH_INLN_DALI_PARAMETER ausgewählt wurden. Die Antwort hängt von der Auswahl in "Address" und "SizeOfParameterValue" ab.

- Ist die Adresse 0-63 und SizeOfParameterValue = 1, ist die Antwort 1 Byte in ParameterValue
- Ist die Adresse 255 und SizeOfParameterValue = 10, ist die Antwort 1 Byte pro Vorschaltgerät (Adresse 0-9), also insgesamt 10 Byte in ParameterValue
- Ist die Adresse 255 und SizeOfParameterValue = 64, ist die Antwort 1 Byte pro Vorschaltgerät (Adresse 0-63), also insgesamt 64 Byte in ParameterValue

Bei FADE_RATE_AND_FADE_TIME ist die Befehlsantwort 1 Byte und beinhaltet XXXX YYYY, wobei XXXX für FADE TIME und YYYY für FADE RATE steht. Gibt das Vorschaltgerät keine Antwort, wird von 16#FF in ParameterValue ausgegangen. Beträgt der Wert der eingegebenen Adresse 0 bis 63, wird das entsprechende Vorschaltgerät ausgelesen. Beträgt der Adresswert 255, liest der Funktionsbaustein die in SizeOfParameterLevel angegebene Anzahl Vorschaltgeräte aus.

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Execute" auf TRUE gesetzt wird. Geänderte Eingangswerte werden erst mit einer neuen steigenden Flanke am Eingang "Execute" angewendet. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Der Funktionsbaustein kann mit einer neuen steigenden Flanke am Eingang "Execute" reinitialisiert werden. Falls während des Betriebs ein Fehler auftritt, wird das Bit "Error" auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Execute" quittiert werden.

Nicht-funktionale Kriterien

Dieser Funktionsbaustein muss zur Kommunikation mit dem Dali-Modul über den Funktionsbaustein IH_DaliReadWrite aufgerufen werden.

Kompatibilität

Kompatibel mit Inline-, Profibus-, Profinet- und sercos-III-Feldbussen.

Beispiel eines Funktionsbausteins

Zunächst muss der Funktionsbaustein IH_DaliReadWrite instanziiert werden, bevor der Funktionsbaustein IH_DaliQueryParameter zur Kommunikation mit dem Dali-Modul instanziiert werden kann. Die Eingangs-/Ausgangsvariable stDaliToken tauscht die Daten zwischen den Funktionsbausteinen IH_DaliReadWrite und IH_DaliQueryParameter aus. Deshalb muss diese Variable in beiden Funktionsbausteinen IOstrToken zugewiesen werden.

Beispiel 1

Im Beispiel wird der Funktionsbaustein IH_DaliQueryParameter zum Auslesen von "Max Level" eines einzelnen Vorschaltgeräts verwendet.

Der Funktionsbaustein liest den Parameter "Max Level" des Vorschaltgeräts mit Adresse 2 und speichert ihn der Variable byParameterValue.

Variablendeklaration

```
VAR
    fbDaliQueryParameter: IH_DaliQueryParameter;
    bExecute: BOOL;
    byParameterValue: byte;
    stDaliToken: IH_INLN_DALI_TOKEN;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
END_VAR
```

Implementation

```
fbDaliQueryParameter(
    Execute:= bExecute,
    Address:= BYTE#2,
    Parameter:= IH_INLN_DALI_PARAMETER.MAX_LEVEL,
    SizeOfParameterValue:= SIZEOF(byParameterValue),
    ParameterValueAdr:= ADR(byParameterValue),
    IOstrToken:= stDaliToken,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent);
```

Beispiel 2

In diesem Beispiel wird der Funktionsbaustein IH_DaliQueryActualLevel zum Auslesen von "Min Level" aller 63 Vorschaltgeräte verwendet.

Der Funktionsbaustein liest den Parameter "Min Level" der 63 Vorschaltgeräte und speichert sie im Array "arParameterValue".

Es muss ein Array mit 63 Bytes erstellt werden und die Adresse des Arrays muss ParameterValueAdr zugeordnet werden.

Dabei muss die Größe des Datenarrays der Anzahl der Vorschaltgeräte entsprechen. Andernfalls können schwere Fehlfunktionen in der SPS-Anwendung auftreten.

Variablendeklaration

```
VAR
    fbDaliQueryParameter: IH_DaliQueryParameter;
    bExecute: BOOL;
    stDaliToken: IH_INLN_DALI_TOKEN;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
    arParameterValue: ARRAY[0..63] OF BYTE;
END_VAR
```

Implementation

```
fbDaliQueryParameter(
    Execute:= bExecute,
    Address:= BYTE#16#FF,
    Parameter:= IH_INLN_DALI_PARAMETER.PHYSICAL_MIN_LEVEL,
    SizeOfParameterValue:= SIZEOF(arParameterValue),
    ParameterValueAdr:= ADR(arParameterValue),
```

```
IOstrToken:= stDaliToken,  
Done=> bDone,  
Active=> bActive,  
Error=> bError,  
ErrorID=> enErrorID,  
ErrorIdent=> stErrorIdent);
```

19.22 IH_DaliQueryGroup

Kurzbeschreibung Der Funktionsbaustein **IH_DaliQueryGroup** identifiziert die Gruppe eines Vorschaltgeräts. Der Funktionsbaustein muss mit dem Funktionsbaustein **IH_DaliReadWrite**, [Seite 763](#) aufgerufen werden.

Zielsystem	Bibliothek/Ordner
MLC	RIH_Dali/Advanced
MTX	RIH_Dali/Advanced
MLD (IL 2G)	RIH_Dali/Advanced

Tab. 19-62: Referenztabelle

Schnittstellenbeschreibung

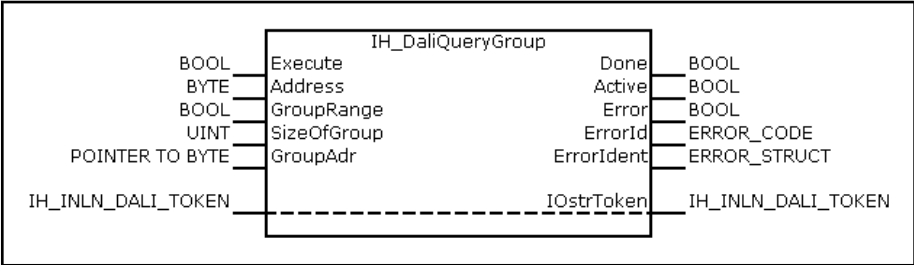


Abb. 19-25: Funktionsbaustein IH_DaliQueryGroup

E/A-Typ	Name	Datentyp	Beschreibung		
VAR_INPUT	Execute	BOOL	Steigende Flanke startet Berechnung		
	Adress	BYTE	Adresse des Vorschaltgeräts (0 bis 63 oder 255)		
			Adresse	SizeOf Group	Anz. gelesener Vorschaltgeräte
			0-63	1	1 Vorschaltgerät
			255	64	64 Vorschaltgeräte
			255	Anz. Vorschaltgeräte von 1 bis 64	Anz. Vorschaltgeräte in SizeOf-Group
	GroupRange	BOOL	FALSE = Gruppe 0 bis 7 abfragen TRUE = Gruppe 8 bis 15 abfragen		
	SizeOfGroup	UINT	Größe der Gruppe (1 bis 64).		
	GroupAdr	POINTER_TO_BYTE	Adresse der Gruppe (1 Byte oder Array von 1 bis 64)		
			Adresswert	Größe der Gruppe	
			0 bis 63	1 BYTE	
			255	Anz. Vorschaltgeräte (1 bis 64)	

E/A-Typ	Name	Datentyp	Beschreibung
VAR_OUTPUT	Done	BOOL	Berechnung beendet
	Active	BOOL	Berechnung findet statt
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Dali, Seite 809
VAR_IN_OUT	IOstrToken	IH_INLN_DALI_TOKEN IH_INLN_DALI_TOKEN, Seite 806	Struktur zum Umgang mit Daten zwischen IH_Dali-ReadWrite und anderen Dali-Funktionsbausteinen. IH_INLN_DALI_TOKEN, Seite 806

Tab. 19-63: Schnittstellenvariablen von Funktionsbaustein IH_DaliQueryGroup

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
Adress	BYTE	0	255	0	Steigende Flanke von Execute
GroupRange	BOOL	FALSE	TRUE	FALSE	Steigende Flanke von Execute
SizeOfGroup	UINT	1	64	1	Steigende Flanke von Execute

Tab. 19-64: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_DaliQueryGroup

Funktionsbeschreibung

Der Funktionsbaustein **IH_DaliQueryGroup** identifiziert die Gruppe eines Vorschaltgeräts.

- Ist die Adresse 0-63 und SizeOfGroup = 1, ist die Antwort 1 Byte in GroupAdr
- Ist die Adresse 255 und SizeOfGroup = 10, ist die Antwort 1 Byte pro Vorschaltgerät (Adresse 0-9), also insgesamt 10 Byte in GroupAdr
- Ist die Adresse 255 und SizeOfGroup = 64, ist die Antwort 1 Byte pro Vorschaltgerät (Adresse 0-63), also insgesamt 64 Byte in GroupAdr

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Execute" auf TRUE gesetzt wird. Geänderte Eingangswerte werden erst mit einer neuen steigenden Flanke am Eingang "Execute" angewendet. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Der Funktionsbaustein kann mit einer neuen steigenden Flanke am Eingang "Execute" reinitialisiert werden. Falls während des Betriebs ein Fehler auftritt, wird das Bit "Error" auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Execute" quittiert werden.

Nicht-funktionale Kriterien

Dieser Funktionsbaustein muss zur Kommunikation mit dem Dali-Modul über den Funktionsbaustein IH_DaliReadWrite aufgerufen werden.

Kompatibilität

Kompatibel mit Inline-, Profibus-, Profinet- und sercos-III-Feldbussen.

Beispiel eines Funktionsbausteins

Zunächst muss der Funktionsbaustein IH_DaliReadWrite instanziiert werden, bevor der Funktionsbaustein IH_DaliQueryGroup zur Kommunikation mit dem Dali-Modul instanziiert werden kann. Die Eingangs-/Ausgangsvariable stDali-Token tauscht die Daten zwischen den Funktionsbausteinen IH_DaliReadWrite und IH_DaliQueryGroup aus. Deshalb muss diese Variable in beiden Funktionsbausteinen IOstrToken zugewiesen werden.

Beispiel 1

Im Beispiel wird der Funktionsbaustein IH_DaliQueryGroup zum Auslesen von „Max Level“ eines einzelnen Vorschaltgeräts verwendet.

Der Funktionsbaustein identifiziert den Gruppenbereich (0 bis 7) von Vorschaltgerät 2 und speichert ihn in der Variable byQueryGroup.

Variablendeklaration

```
VAR
    fbDaliQueryGroup: IH_DaliQueryGroup;
    bExecute: BOOL;
    byQueryGroup: BYTE;
    stDaliToken: IH_INLN_DALI_TOKEN;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
END_VAR
```

Implementation

```
fbDaliQueryGroup(
    Execute:= bExecute,
    Address:= BYTE#2,
    GroupRange:= FALSE,
    SizeOfGroup:= SIZEOF(byQueryGroup),
    GroupAdr:= ADR(byQueryGroup),
    IOstrToken:= stDaliToken,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent);
```

Beispiel 2

In diesem Beispiel identifiziert der Funktionsbaustein IH_DaliQueryGroup den Gruppenbereich (8 bis 15) aller 63 Vorschaltgeräte.

Der Funktionsbaustein identifiziert den Gruppenbereich (8 bis 15) aller 63 Vorschaltgeräte und speichert ihn im Array arQueryGroup.

Es muss ein Array mit 63 Bytes erstellt werden und die Adresse des Arrays muss GroupAdr zugeordnet werden.

Die Größe des Arrays wird SizeOfGroup festgelegt. Ist das tatsächliche Array kleiner als der Wert in der Variable SizeOfGroup, können Daten in eine andere Adresse geschrieben und so Fehlfunktionen in der Anwendung verursacht werden.

Variablendeklaration

```
VAR
    fbDaliQueryGroup: IH_DaliQueryGroup;
    bExecute: BOOL;
```

```

stDaliToken: IH_INLN_DALI_TOKEN;
bDone: BOOL;
bActive: BOOL;
bError: BOOL;
enErrorID: ERROR_CODE;
stErrorIdent: ERROR_STRUCT;
arQueryGroup: ARRAY[0..63] OF BYTE;
END_VAR

```

Implementation

```

fbDaliQueryGroup(
  Execute:= bExecute,
  Address:= BYTE#16#FF,
  GroupRange:= TRUE,
  SizeOfGroup:= SIZEOF(arQueryGroup),
  GroupAdr:= ADR(arQueryGroup),
  IOstrToken:= stDaliToken,
  Done=> bDone,
  Active=> bActive,
  Error=> bError,
  ErrorID=> enErrorID,
  ErrorIdent=> stErrorIdent);

```

19.23 IH_DaliQueryRandomAddress

Kurzbeschreibung Der Funktionsbaustein **IH_DaliQueryRandomAddress** dient zum Auslesen der Zufallsadresse des Vorschaltgeräts. Der Funktionsbaustein muss mit dem Funktionsbaustein **IH_DaliReadWrite**, Seite 763 aufgerufen werden.

Zielsystem	Bibliothek/Ordner
MLC	RIH_Dali/Advanced
MTX	RIH_Dali/Advanced
MLD (IL 2G)	RIH_Dali/Advanced

Tab. 19-65: Referenztablelle

Schnittstellenbeschreibung

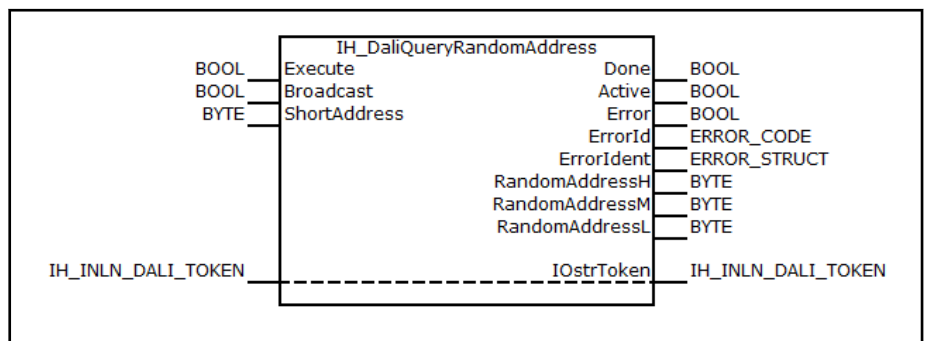


Abb. 19-26: Funktionsbaustein IH_DaliQueryRandomAddress

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Steigende Flanke startet Berechnung
	Broadcast	BOOL	TRUE = Zufallsadresse im Sammelmodus abfragen FALSE = Zufallsadresse des in ShortAddress angegebenen Vorschaltgeräts abfragen
	ShortAddress	BYTE	Kurzadresse des Vorschaltgeräts (0 bis 63)
	Done	BOOL	Berechnung beendet
	Active	BOOL	Berechnung findet statt

E/A-Typ	Name	Datentyp	Beschreibung
VAR_OUTPUT	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Dali, Seite 809
	RandomAddressH	BYTE	Höherwertiges Byte der Zufallsadresse
	RandomAddressM	BYTE	Mittleres Byte der Zufallsadresse
	RandomAddressL	BYTE	Niederwertiges Byte der Zufallsadresse
VAR_IN_OUT	IOstrToken	IH_INLN_DALI_TOKEN IH_INLN_DALI_TOKEN, Seite 806	Struktur zum Umgang mit Daten zwischen IH_Dali-ReadWrite und anderen Dali-Funktionsbausteinen. IH_INLN_DALI_TOKEN, Seite 806

Tab. 19-66: Schnittstellenvariablen von Funktionsbaustein IH_DaliQueryRandomAddress

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
Broadcast	BOOL	FALSE	TRUE	FALSE	Steigende Flanke von Execute
ShortAddress	BYTE	0	63	0	Steigende Flanke von Execute

Tab. 19-67: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_DaliQueryRandomAddress

Funktionsbeschreibung

Der Funktionsbaustein **IH_DaliQueryRandomAddress** dient zum Auslesen der Zufallsadresse des Vorschaltgeräts. Der Funktionsbaustein liest die 8 höherwertigen, die 8 mittleren und die 8 niederwertigen Bits einer Zufallsadresse. Das Ergebnis kann über die Ausgabevariablen RandomAddressH, RandomAddressM und RandomAddressL abgerufen werden.

Ist Broadcast TRUE, wird die Zufallsadresse mittels Sammelbefehl abgefragt. Im Broadcast-Modus müssen einzelne Vorschaltgeräte verbunden sein. Sonst können bei der Abfrage Gerätefehler auftreten. Ist Broadcast FALSE, wird die Zufallsadresse des in ShortAddress angegebenen Vorschaltgeräts abgefragt.

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Execute" auf TRUE gesetzt wird. Geänderte Eingangswerte werden erst mit einer neuen steigenden Flanke am Eingang "Execute" angewendet. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Der Funktionsbaustein kann mit einer neuen steigenden Flanke am Eingang "Execute" reinitialisiert werden. Falls während des Betriebs ein Fehler auftritt, wird das Bit "Error" auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Execute" quittiert werden.

Nicht-funktionale Kriterien

Dieser Funktionsbaustein muss zur Kommunikation mit dem Dali-Modul über den Funktionsbaustein IH_DaliReadWrite aufgerufen werden.

Kompatibilität

Kompatibel mit Inline-, Profibus-, Profinet- und sercos-III-Feldbussen.

Beispiel eines Funktionsbausteins

Zunächst muss der Funktionsbaustein IH_DaliReadWrite instanziiert werden, bevor der Funktionsbaustein IH_DaliQueryRandomAddress zur Kommunikation mit dem Dali-Modul instanziiert werden kann. Die Eingangs-/Ausgangsvariable stDaliToken tauscht die Daten zwischen den Funktionsbausteinen IH_DaliReadWrite und IH_DaliQueryRandomAddress aus. Deshalb muss diese Variable in beiden Funktionsbausteinen IOstrToken zugewiesen werden.

Beispiel 1

Im Beispiel wird der Funktionsbaustein IH_DaliQueryRandomAddress zum Abfragen der Zufallsadresse eines einzelnen Vorschaltgeräts verwendet.

Der Funktionsbaustein liest die Zufallsadresse von Vorschaltgerät 2 und speichert sie in den Variablen byRandomAddressH, byRandomAddressM und byRandomAddressL.

Variablendeklaration

```
VAR
    fbDaliQueryRandomAddress: IH_DaliQueryRandomAddress;
    bExecute: BOOL;
    stDaliToken: IH_INLN_DALI_TOKEN;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
    byRandomAddressH: BYTE;
    byRandomAddressM: BYTE;
    byRandomAddressL: BYTE;
END_VAR
```

Implementation

```
fbDaliQueryRandomAddress (
    Execute:= bExecute,
    Broadcast:= FALSE,
    ShortAddress:= BYTE#2,
    IOstrToken:= stDaliToken,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent,
    RandomAddressH=> byRandomAddressH,
    RandomAddressM=> byRandomAddressM,
    RandomAddressL=> byRandomAddressL);
```

Beispiel 2

Der Funktionsbaustein liest die Zufallsadressen aller Vorschaltgeräte und speichert sie in den Variablen byRandomAddressH, byRandomAddressM und byRandomAddressL. Weicht im Broadcast-Modus die Zufallsadresse eines Vorschaltgeräts von der Zufallsadresse eines anderen Vorschaltgeräts ab, gibt das Dali-System den Fehler ungültige Antwort aus.

Implementation

```
fbDaliQueryRandomAddress (
    Execute:= bExecute,
    Broadcast:= TRUE,
    ShortAddress:= ,
    IOstrToken:= stDaliToken,
```

```

Done=> bDone,
Active=> bActive,
Error=> bError,
ErrorID=> enErrorID,
ErrorIdent=> stErrorIdent,
RandomAddressH=> byRandomAddressH,
RandomAddressM=> byRandomAddressM,
RandomAddressL=> byRandomAddressL;

```

19.24 Datentypen der Bibliothek RIH_Dali

19.24.1 IH_INLN_DALI_TOKEN

This structure is used internally to exchange the command and data between the [IH_DaliReadWrite](#), [page 763](#), and other Dali function blocks.

Library	Folder
RIH_Dali	TYPE

Tab. 19-68: Reference table

Name	Type	Description	Min. value	Max. value	Default value	Application
byRespErr	BYTE	Device error during Communication.	16#0000	16#0F	16#0000	Continuous
byRespSucc	BYTE	Device response during communication	16#0000	16#FF	16#0000	Continuous
byToken	BYTE	Token used by other Dali blocks to communicate with IH_DaliReadWrite FB.	16#0000	16#FF	16#0000	Continuous
bRepeat	BOOL	Repeat Dali command again (50ms)	FALSE	TRUE	FALSE	Continuous
wData2	WORD	Command or data processing variable	-----	-----	-----	Continuous

Tab. 19-69: IH_INLN_DALI_TOKEN

19.24.2 IH_INLN_DALI_ADDRTYPE

Hiermit wird die Aufzählung der Adresstypen beschrieben.

Bibliothek	Ordner
RIH_Dali	TYPE

Tab. 19-70: Bibliothekszuordnung

ENUM	Beschreibung
SHORTADDR	„Address“ ist die Adresse eines Vorschaltgeräts
GROUPADDR	„Address“ ist eine Gruppenadresse
BROADCAST	Sammelbefehl. Befehl wird für alle Vorschaltgeräte ausgeführt.

Tab. 19-71: IH_INLN_DALI_ADDRTYPE

19.24.3 IH_INLN_DALI_DIMMODE

Beschreibt die Helligkeitsänderungen.

Bibliothek	Ordner
RIH_Dali	TYPE

Tab. 19-72: Bibliothekszuordnung

ENUM	Beschreibung
UP	Die Lampe wird über einen Zeitraum von 200 ms heller. Die Abblendrate in diesen 200 ms wird über die Variable FADE RATE festgelegt.
STEPUP	Die Lampenleistung wird um einen Schritt erhöht.
ON_AND_STEPUP	Falls erforderlich wird die Lampe eingeschaltet und die Lampenleistung um einen Schritt erhöht.
DOWN	Die Lampe wird über einen Zeitraum von 200 ms dunkler. Die Abblendrate in diesen 200 ms wird über die Variable FADE RATE festgelegt.
STEPDOWN	Die Lampenleistung wird um einen Schritt gesenkt.
STEPDOWN_AND_OFF	Die Lampenleistung wird um einen Schritt gesenkt. Ist die Istleistung auf dem Minimum (Min Level) wird die Lampe ausgeschaltet.

Tab. 19-73: IH_INLN_DALI_DIMMODE

19.24.4 IH_INLN_DALI_LIGHTLEVEL

Beschreibt die Helligkeitsstufen.

Bibliothek	Ordner
RIH_Dali	TYPE

Tab. 19-74: Bibliothekszuordnung

ENUM	Beschreibung
MIN_LEVEL	Die Helligkeit ist auf der niedrigsten Stufe
MID_LEVEL	Die Helligkeit ist auf 50 Prozent
MAX_LEVEL	Die Helligkeit ist auf der höchsten Stufe

Tab. 19-75: IH_INLN_DALI_LIGHTLEVEL

19.24.5 IH_INLN_DALI_OPMODE

Zeigt den aktuellen Betriebsmodus des Systems an.

Bibliothek	Ordner
RIH_Dali	TYPE

Tab. 19-76: Bibliothekszuordnung

Name	Wert	Kommentar
VALUE_1_TO_DTR	0	Überträgt Wert 1 ins DTR.
VALUE255_TO_DTR	1	Befehl: zum Übertragen der Werte ins Data Transfer Register.
MASKING_ALL_ADDRESS	2	Maskiert die Adressen aller Vorschaltgeräte. (Löscht die Adressen.)
COMMAND_INITIALIZE	3	Befehl: Initialisierung.
COMMAND_RANDOMISE	4	Befehl: Zufallszahl ermitteln.
SET_SEARCHADDRESS	5	Suchadresse mit IH_DaliSearchAddress festlegen.
COMMAND_COMPARE	6	Befehl: Vergleich.
DECREMENTING_SEAR- CHADDR	7	Dekrementiert die Suchadresse wenn Compare = TRUE.

Name	Wert	Kommentar
INCREMENTING_SEAR-CHADDR	8	Inkrementiert die Suchadresse wenn Compare = FALSE.
ENABLE_PHYSICAL_SELECTION	9	Befehl: Aktivierung physikalische Auswahl.
FIN-DING_AVAILABLE_ADDRESS	10	Sucht mit IH_DaliQueryStatus die nächsten verfügbaren Adressen.
QUERY_SHORT_ADDR_IN_PHYSICAL	11	Befehl: Suche der Kurzadresse bei physikalischer Auswahl.
PROGRAM_SHORT_ADDR	12	Befehl: Kurzadresse Programm.
VERIFY_SHORT_ADDR	13	Befehl: Überprüfung Kurzadresse.
VERIFY_LAMP_ON	14	Mit IH_DaliQueryStatus im ausgewählten Vorschaltgerät prüfen, ob die Lampe eingeschaltet ist.
RECALL_MINLEVEL	15	Befehl: MinLevel in Vorschaltgerät mit der Adresse aufrufen.
RECALL_MAXLEVEL	16	Befehl: MaxLevel in Vorschaltgerät mit der Adresse aufrufen.
WITHDRAWAL	17	Befehl: entfernen.
TERMINATE	18	Befehl: beenden.
COMMAND_INITIALIZE1	19	Befehl: Erstinitialisierung.
WITHDRAWAL1	20	Zum ersten Mal entfernen.
TERMINATE1	21	Befehl: Zum ersten Mal beenden.
SEARCH_ADDR_TO_ZERO	22	Befehl: Suchadresse auf Null setzen.
RESET_ALL BALLASTS	23	

Tab. 19-77: ENUM IH_INLN_DALI_OPMODE

19.24.6 IH_INLN_DALI_PARAMETER

Aufzählung zur Parameterauswahl

Bibliothek	Ordner
RIH_Dali	TYPE

Tab. 19-78: Bibliothekszuordnung

Name	Value	Kommentar
FADE_RATE	0	FADE RATE legt die Geschwindigkeit (in Schritten pro Sekunde) fest, in der Änderungen der gemäß der Formel berechneten Lampenleistung vorgenommen werden. Zulässige Grenzwerte (1-15).
FADE_TIME	1	FADE TIME legt die zulässige Zeit für die Änderung der aktuellen Lampenleistung auf den gemäß der Formel berechneten und geforderten Wert fest. Ist die Lampe ausgeschaltet, werden Vorwärm- und Zündzeit nicht in FADE TIME berücksichtigt. Zulässige Grenzwerte (0-15).
MAX_LEVEL	2	Das Vorschaltgerät verwendet die Variable MAX LEVEL, um die maximal an die Lampe abgegebene Leistung zu beschränken. Zulässige Grenzwerte (Min Level - 254).

Name	Value	Kommentar
MIN_LEVEL	3	Das Vorschaltgerät verwendet die Variable MIN LEVEL, um die mindestens an die Lampe abgegebene Leistung festzulegen. Zulässige Grenzwerte (Physical Min Level - Max Level).
POWER_ON_LEVEL	4	Wird das Vorschaltgerät mit Strom versorgt, wird die Lampe auf die in der Variable POWER ON LEVEL festgelegte Leistung gebracht. Zulässige Grenzwerte (1 - 254).
SYSTEM_FAILURE_LEVEL	5	Tritt im DALI-Bus eine Störung auf (die Blindspannung bleibt länger als 500 ms unter dem festgelegten Niveau), wird die Lampe auf die in der Variable SYSTEM FAILURE LEVEL festgelegte Leistung gebracht. Zulässige Grenzwerte (0-255).
FADE_RATE_AND_FADE_TIME	6	Abfragebefehl für FADE RATE und FADE TIME. Die Antwort ist XXXX YYYY wobei XXXX mit FADE RATE und YYYY mit FADE TIME übereinstimmt.
PHYSICAL_MIN_LEVEL	7	

Tab. 19-79: ENUM IH_INLN_DALI_PARAMETER

19.25 Fehlerbehandlung mit der Bibliothek RIH_Dali

Die Funktionsbausteine verwenden die Fehlertabelle **INLINEIO_TABLE**, 16#0190.

In Additional1/Additional2 können folgende Fehlermeldungen erzeugt werden:

ErrorID	Additional1	Additional2	Description
INPUT_INVALID_ERROR	16#3017	16#0000	Die Eingabe in „OldAddress“ oder „NewAddress“ ist ungültig (<0 oder >63)
INPUT_INVALID_ERROR	16#3017	16#0001	„GroupAddress“ ist ungültig (0< GroupAddress >15)
INPUT_INVALID_ERROR	16#3017	16#0002	Die Eingabe in „ShortAddress“ ist ungültig (0< ShortAddress >63)
INPUT_INVALID_ERROR	16#3017	16#0003	Die Eingabe in „LightLevel“ ist ungültig (0 oder IH_INLN_DALI_LIGHTLEVEL.MIN_LEVEL < LightLevel >254 oder IH_INLN_DALI_LIGHTLEVEL.MAX_LEVEL)
INPUT_INVALID_ERROR	16#3017	16#0004	Die Eingabe in „Address“ ist ungültig Für Gruppe (0< Address >15) Für Vorschaltgerät (0< Address >63) Für Abfrage (0 < Address >63 und <= 255)
INPUT_INVALID_ERROR	16#3017	16#0005	Die Eingabe in „DimMode“ ist ungültig
INPUT_INVALID_ERROR	16#3017	16#0006	Die Eingabe in „AddressType“ ist ungültig
INPUT_INVALID_ERROR	16#3017	16#0007	Die Eingabe in „Scene“ ist ungültig
INPUT_INVALID_ERROR	16#3017	16#0008	SizeOfActualLevel ist ungültig (1< SizeOfActualLevel > 64)
INPUT_INVALID_ERROR	16#3017	16#0009	SizeOfStatus ist ungültig (1<SizeOfStatus> 64)
DEVICE_ERROR	16#3021	16#0000	Zeitfehler. Das Gerät reagiert nicht.
DEVICE_ERROR	16#3021	16#0001	Vom Vorschaltgerät wurde eine ungültige Antwort empfangen (Fehler Dali-Bus)
DEVICE_ERROR	16#3021	16#0004	Über Inline ging ein nicht unterstützter Befehl ein (keine Aktion auf Dali-Bus)

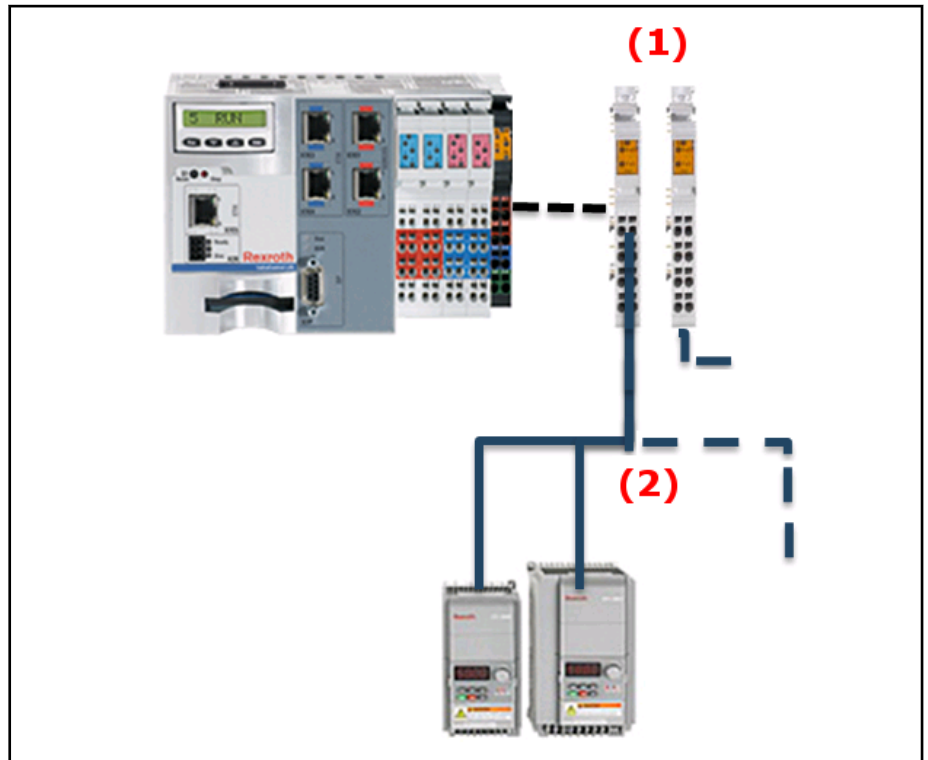
ErrorID	Additional1	Additonal2	Description
DEVICE_ERROR	16#3021	16#0008	E/A-Fehler (DALI-Ausfall)
DEVICE_ERROR	16#3021	16#000F	Nicht definierter Fehler
RESOURCE_ERROR	16#3018	16#0000	Nullzeiger, Variable nicht zugewiesen
OTHER_ERROR	16#3031	16#0000	Ungültiges Token oder Token nicht zwischen IH_DaliReadWrite und anderen Dali-Funktionsbausteinen verfügbar

Tab. 19-80: RIH_Dali-Fehlercodes

20 RIH_Modbus.library

20.1 RIH_Modbus, Allgemeines

Die Bibliothek **RIH_Modbus** ist mit den Inline-Modulen **R-IB IL RS232 PRO**, **R-IB IL RS485/422 PRO** oder **R-IB IL RS-Uni** (Inline-Module für serielle Datenübertragung) zu verwenden. Die Inline-Module können an die Steuerung oder einen Inline-Buskoppler angeschlossen werden (Profibus, Profinet, Sercos).



- (1) 1..n serielle Kommunikationsmodule
 (2) 1..x Geräte

Abb. 20-1: Beispiel eines Modbus-RTU-Netzwerks mit Lx5-Steuerung, R-IB IL RS485/422-PRO und SEFC3600-Frequenzumrichter

Die Bibliothek **RIH_Modbus** stellt Funktionsbausteine zur Kommunikation mit dem Modbus-RTU-Protokoll als Master oder Client bereit.

Zum Starten des SPS-Programms mit RIH_Modbus muss für den Client der Funktionsbaustein **IH_MBRTUSlave** und für den Master der Funktionsbaustein **IH_MBRTUMaster** instanziiert werden.

Der Funktionsbaustein

- **IH_MBRTUMaster** stellt den Kommunikation-Handle für alle anderen Funktionsbausteine in der Bibliothek RIH_Modbus bereit, z. B. IH_MBReadCoils, IH_MBWriteSingleCoil.
- **IH_MBRTUSlave** gewährleistet die Kommunikation mit einem Modbus-RTU-Master.

Bei den Inline-Modulen **R-IB IL RS232 PRO**, **R-IB IL RS485/422 PRO** oder **R-IB IL RS-Uni**, ist mit Unterstützung von **IH_MBRTUMaster** und **IH_MBRTUSlave** ein direkter Datenaustausch zwischen 2 Lx5-Steuerungen von Bosch Rexroth möglich.

Jeder Funktionsbaustein mit Modbus-Funktionscode kann in einer einzelnen Anfrage oder im Polling-Modus verwendet werden. Im Polling-Modus kann das Polling-Intervall festgelegt werden, und der Funktionsbaustein sendet die Anfragen automatisch zur entsprechenden Zeit.



Achtung, wenn Modbus RTU mit EFC-Frequenzumrichter verwendet wird.

Derzeit benötigt 1 Modbus-RTU-Verbindung zum Senden und Empfangen ca. 100 ms. Für 10 Anfragen wird also 1 Sekunde benötigt.

Bsp.: 5 Modbus-RTU-Geräte, 2 Anfragen an jedes Gerät = 10 Anfragen

Werden hohe Aktualisierungsraten erwartet oder viele Modbus-RTU-Geräte eingesetzt, sind weitere serielle Inline-Kommunikationsmodule erforderlich.

Andere Modbus-RTU-Anwendungen benötigen ggf. weniger als 100 ms.

Funktionsbausteine der Bibliothek RIH_Modbus

Typ	Beschreibung	Funktions-code
IH_MBGetCommEventCounter, Seite 813	Gibt das Statuswort und den Ereigniszähler für die Kommunikation zurück	11
IH_MBGetCommEventLog, Seite 816	Gibt die Struktur des Ereignisprotokolls aus, einschließlich Statuswort, Kommunikationsereigniszähler, Meldungszähler und Feld von Ereignisbytes	12
IH_MBMaskWriteRegister, Seite 820	Verändert den Inhalt eines bestimmten Holding-Registers mit einer AND-Maske, OR-Maske und dem aktuellen Inhalt des Registers	22
IH_MBReadCoils, Seite 823	Liest den ON- oder OFF-Wert von 1 bis 2000 Coils, d. h. fortlaufend aufeinanderfolgende Ausgangsbits	1
IH_MBReadDeviceIdentification, Seite 827	Liest die ID und zusätzliche Informationen bezüglich der physikalischen und funktionalen Beschreibung eines Remote-Gerätes aus dem Client (Master)	43
IH_MBReadDiagnostics, Seite 834	Stellt Tests für die Kommunikation zwischen Master und Slave oder nur Master-intern bereit	8
IH_MBReadDiscreteInputs, Seite 837	Liest 1 bis 2000 benachbarte Status von getrennten Eingängen (Discrete Inputs), Bitdatenzugriff	2
IH_MBReadExceptionStatus, Seite 841	Liest das Statusbyte für die Diagnose (8 Bits) des Modbus-RTU-Slave	7
IH_MBReadFIFOQueue, Seite 844	Liest den Inhalt einer First-In-First-Out-Registerschlange (FIFO) eines entfernten Geräts	24
IH_MBReadFileRecord, Seite 847	Liest die angegebenen Dateiinformationen, Unteranfragen möglich	20
IH_MBReadHoldingRegisters, Seite 851	Liest den Inhalt eines benachbarten Holding-Register-Blocks, Wortdatenzugriff	3
IH_MBReadInputRegisters, Seite 855	Liest 1 bis 125 benachbarte Input-Register, Wortdatenzugriff	4

Typ	Beschreibung	Funktionscode
IH_MBReadWriteMultipleRegisters, Seite 859	Führt in einer einzigen Modbus-Transaktion eine Lese- und eine Schreiboperation durch. Die Schreiboperation erfolgt vor dem Lesen	23
IH_MBRTUMaster, Seite 863	Stellt den Kommunikation-Handle für alle anderen Funktionsbausteine in der Bibliothek RIH_Modbus bereit, z. B. IH_MBReadCoils, IH_MBWriteSingleCoil.	
IH_MBRTUSlave, Seite 868	Gewährleistet die Kommunikation mit einem Modbus-RTU-Master, kann z. B. zum Datenaustausch mit RIH_MBRTU-Master verwendet werden	
IH_MBWriteFileRecord, Seite 875	Schreibt die angegebenen Dateiinformationen, Unteranfragen sind möglich	21
IH_MBWriteMultipleCoils, Seite 879	Schreibt mehrere ON- oder OFF-Werte von 1 bis 1968 Coils, d. h. fortlaufend aufeinanderfolgende Ausgangsbits	15
IH_MBWriteMultipleRegisters, Seite 883	Schreibt mehrere Register (1 bis 123 Register), Wortdaten-zugriff	16
IH_MBWriteSingleCoil, Seite 888	Schreibt den ON- oder OFF-Wert von 1 bis 2000 Coils, d. h. einzelnes Ausgangsbit	5
IH_MBWriteSingleRegister, Seite 891	Schreibt ein einziges Holding-Register, Wortdaten-zugriff	6

Tab. 20-1: Funktionsbausteine der Bibliothek RIH_Modbus

20.2 IH_MBGetCommEventCounter

Funktionscode	FC 11
Kurzbeschreibung	Der Funktionsbaustein IH_MBGetCommEventCounter ruft ein Statuswort und die Ereignisanzahl vom Kommunikationsereigniszähler eines Slave-Geräts ab. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein IH_MBRTUMaster, Seite 863 aufgerufen werden.

Zielsystem	Bibliothek / Ordner
MLC	RIH_Modbus / FB
MTX	RIH_Modbus / FB
MLD (IL 2G)	RIH_Modbus / FB

Tab. 20-2: Referenztabelle

Interfacebeschreibung

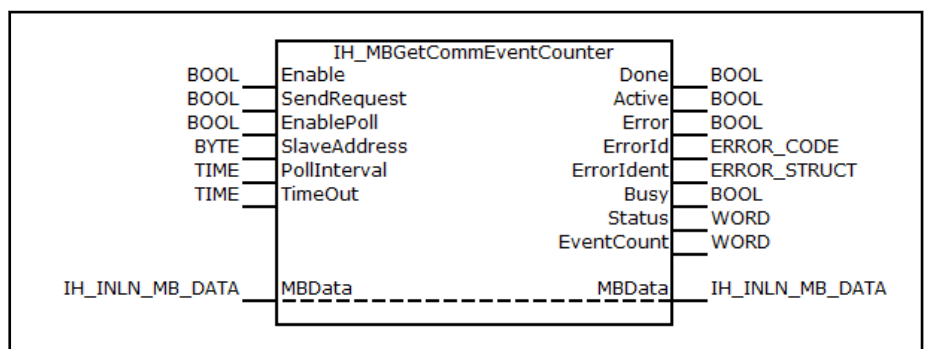


Abb. 20-2: Funktionsbaustein IH_MBGetCommEventCounter

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Funktionsbaustein aktiv, falls TRUE
	SendRequest	BOOL	Bei steigender Flanke wird eine Anfrage gesendet
	EnablePoll	BOOL	Polling aktiviert. SendRequest muss nicht in jedem Intervall ausgelöst werden. Der Baustein sendet Anfragen automatisch
	SlaveAddress	BYTE	Slave-Adresse
	PollInterval	TIME	Polling-Intervall. Anfrage wird kontinuierlich in Intervallen gesendet
	TimeOut	TIME	Timeout-Intervall für Antwort von Slaves
VAR_OUTPUT	Done	BOOL	Berechnung beendet und beginnt im nächsten Zyklus neu
	Active	BOOL	Funktionsbaustein ist aktiv
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Modbus, Seite 898
	Busy	BOOL	Wahr, wenn der Baustein mit dem Master kommuniziert
	Status	WORD	Statuswort
	EventCount	WORD	Anzahl Kommunikationsereignisse
VAR_IN_OUT	MBData	IH_INLN_MB_DATA	Struktur handhabt Daten zwischen Master und anderen Modbus-Funktionsbausteinen der Funktionscodes IH_INLN_MB_DATA, Seite 894

Tab. 20-3: Schnittstellenvariablen von Funktionsbaustein IH_MBGetCommEventCounter

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
SendRequest	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
EnablePoll	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
SlaveAddress	BYTE	0	247	0	steigende Flanke an Enable
PollInterval	TIME	T#1s	-----	T#1s	Kontinuierlich bei Enable
TimeOut	TIME	T#1s	-----	T#1s	steigende Flanke an Enable

Tab. 20-4: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_MBGetCommEventCounter

Funktionsbeschreibung Der Funktionsbaustein **IH_MBGetCommEventCounter** (FC 11) ruft ein Statuswort und die Ereignisanzahl vom Kommunikationsereigniszähler eines Slave-Geräts ab. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein aufgerufen werden. "MBData" muss mit dem Master-Funktionsbaustein im Programm verknüpft sein, er dient zur Kommunikation und zum Datenaustausch.

Der Ereigniszähler des Geräts wird für jede erfolgreich abgeschlossene Meldung um eins erhöht. Er wird nicht bei Ausnahmeantworten, Polling-Befehlen oder Befehlen an den Ereigniszähler erhöht.

Die normale Antwort enthält ein 2-Byte-Statuswort und eine 2-Byte-Ereignisanzahl. Das Statuswort besteht nur aus Einsen (FF FF hex), wenn ein zuvor erteilter Programmbefehl noch vom entfernten Gerät ausgeführt wird (das Gerät besetzt ist). Andernfalls besteht das Statuswort komplett aus Nullen.

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Enable" auf TRUE gesetzt wird. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit **Done** auf TRUE gesetzt. Falls während des Betriebs ein Fehler auftritt, wird das Bit „Error“ auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Enable" quittiert werden.

Beispiel eines Funktionsbausteins

Der Funktionsbaustein IH_MBGetCommEventCounter muss zur Kommunikation mit dem Modbus-Slave zusammen mit IH_MBRTUMaster instanziiert werden. Für MBData der Funktionsbausteine IH_MBGetCommEventCounter und IH_MBRTUMaster ist für den Datenaustausch mit dem Modbus-Slave eine gemeinsame Variable zu verwenden.

In diesem Beispiel ruft der Funktionsbaustein IH_MBGetCommEventCounter das Statuswort und die Ereignisanzahl des Slave mit Adresse 1 ab.

Die beiden Funktionsbausteine IH_MBRTUMaster und IH_MBGetCommEventCounter werden mit steigender Flanke am Eingang bSendRequest aktiviert. Der Funktionsbaustein liest das Statuswort und die Ereignisanzahl von Slave-Adresse 1 und das Ergebnis wird in wStatus und wEventCount gespeichert.

Wenn die Variable bEnablePoll auf TRUE und Pollinterval auf mehr als 1 Sekunde gestellt werden, werden die Daten in regelmäßigen Intervallen aus dem Register gelesen.

Variablendeklaration

```

VAR
    fbIH_MBGetCommEventCounter: IH_MBGetCommEventCounter;
    bEnable: BOOL;
    bEnablePoll: BOOL;
    bSendRequest: BOOL;
    stMBData: IH_INLN_MB_DATA;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
    bBusy: BOOL;
    wStatus: WORD;
    wEventCount: WORD;
END_VAR

```

Implementation

```

fbIH_MBGetCommEventCounter(
    Enable:= bEnable,
    SendRequest:= bSendRequest,

```

```
EnablePoll:= bEnablePoll,  
SlaveAddress:= BYTE#1,  
PollInterval:= T#1S,  
TimeOut:= T#1S,  
MBData:= stMBData,  
Done=> bDone,  
Active=> bActive,  
Error=> bError,  
ErrorID=> enErrorID,  
ErrorIdent=> stErrorIdent,  
Busy=> bBusy,  
Status=> wStatus,  
EventCount=> wEventCount);
```

20.3 IH_MBGetCommEventLog

Funktionscode FC 12

Kurzbeschreibung Der Funktionsbaustein **IH_MBGetCommEventLog** ruft ein Statuswort, Ereignisanzahl, Meldungsanzahl und das Feld der Ereignis-Bytes im Kurzprotokoll der Ereigniskommunikation aus dem Slave-Modul ab. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein [IH_MBRTUMaster](#), [Seite 863](#) aufgerufen werden.

Zielsystem	Bibliothek /Ordner
MLC	RIH_Modbus / FB
MTX	RIH_Modbus / FB
MLD (IL 2G)	RIH_Modbus / FB

Tab. 20-5: Referenztabelle

Interfacebeschreibung

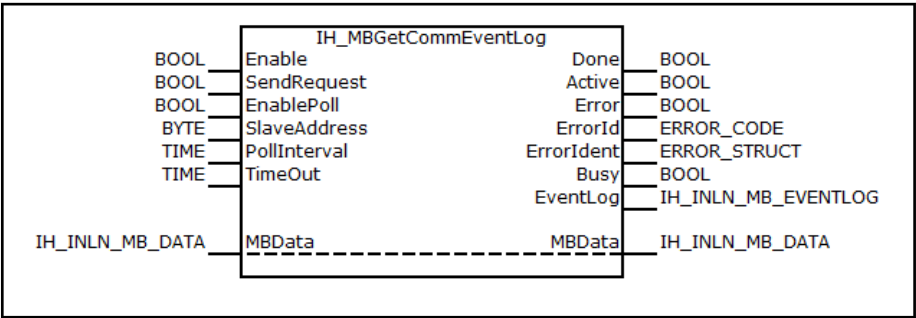


Abb. 20-3: Funktionsbaustein IH_MBGetCommEventLog

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Funktionsbaustein aktiv, falls TRUE
	SendRequest	BOOL	Bei steigender Flanke wird eine Anfrage gesendet
	EnablePoll	BOOL	Polling aktiviert. SendRequest muss nicht in jedem Intervall ausgelöst werden. Der Baustein sendet Anfragen automatisch
	SlaveAddress	BYTE	Slave-Adresse
	PollInterval	TIME	Polling-Intervall. Anfrage wird kontinuierlich in Intervallen gesendet
	TimeOut	TIME	Timeout-Intervall für Antwort von Slaves

E/A-Typ	Name	Datentyp	Beschreibung
VAR_OUTPUT	Done	BOOL	Berechnung beendet und beginnt im nächsten Zyklus neu
	Active	BOOL	Funktionsbaustein ist aktiv
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Modbus, Seite 898
	Busy	BOOL	Wahr, wenn der Baustein mit dem Master kommuniziert
	EventLog	IH_INLN_MB_EVENTLOG	Die Struktur beinhaltet zusätzliche Informationen zu Status, EventCount, MessageCount und Events IH_INLN_MB_EVENTLOG, Seite 897
VAR_IN_OUT	MBData	IH_INLN_MB_DATA	Struktur handhabt Daten zwischen Master und anderen Modbus-Funktionsbausteinen IH_INLN_MB_DATA, Seite 894

Tab. 20-6: Schnittstellenvariablen von Funktionsbaustein IH_MBGetCommEventLog

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
SendRequest	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
EnablePoll	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
SlaveAddress	BYTE	0	247	0	steigende Flanke an Enable
PollInterval	TIME	T#1s	-----	T#1s	Kontinuierlich bei Enable
TimeOut	TIME	T#1s	-----	T#1s	steigende Flanke an Enable

Tab. 20-7: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_MBGetCommEventLog

Funktionsbeschreibung

Der Funktionsbaustein **IH_MBGetCommEventLog** (FC 12) ruft ein Statuswort, Ereignisanzahl, Meldungsanzahl und das Feld der Ereignis-Bytes im Kurzprotokoll der Ereigniskommunikation aus dem Slave-Modul ab. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein aufgerufen werden. "MBData" muss mit dem Master-Funktionsbaustein im Programm verknüpft sein, er dient zur Kommunikation und zum Datenaustausch. Statuswort und Ereignisanzahl sind identisch mit den von der Funktion „Get Communications Event Counter“ ausgegebenen Daten (11, 16#0B). Der Meldungsähler enthält die Anzahl der vom entfernten Gerät verarbeiteten Meldungen seit das Gerät zuletzt neu gestartet, eingeschaltet oder der

Zähler gelöscht wurde. Die Anzahl ist identisch mit der durch die Diagnosefunktion bereitgestellten Anzahl (Code 08), Unterfunktion „Return Bus Message Count“ (Code 11, 0B hex).

Das Ereignis-Bytes-Feld enthält 0-64 Bytes, wobei jedes Byte dem Status einer MODBUS-Operation zum Senden oder Empfangen durch das entfernte Gerät entspricht. Das entfernte Gerät trägt die Ereignisse in chronologischer Reihenfolge in das Feld ein. Byte 0 enthält das zuletzt erfolgte Ereignis. Ein neues Byte überschreibt stets das älteste Byte im Feld.

Inhalt Ereignis-Byte

Die Funktion „Get Communications Event Log“ gibt vier Typen von Ereignis-Bytes zurück. Der Typ wird von Bit 7 (höherwertiges Bit) in jedem Byte definiert. Er kann außerdem von Bit 6 definiert werden. Dies wird nachstehend erläutert.

MODBUS-Empfangsereignis in entfernten Gerät

Das entfernte Gerät speichert diesen Typ Ereignis-Byte, wenn es eine Meldung mit einer Abfrage empfängt. Er wird gespeichert, bevor das entfernte Gerät die Meldung verarbeitet. Das Ereignis wird definiert, indem Bit 7 = "1". Die übrigen Bits sind "1", wenn die zugehörige Bedingung wahr (TRUE) ist. Die Bits sind wie folgt aufgeteilt:

Bit-Inhalt

Bit	Beschreibung
0	Nicht verwendet
1	Kommunikationsfehler
2	Nicht verwendet
3	Nicht verwendet
4	Zeichenüberlauf
5	Derzeit im Modus Listen Only
6	Empfangene Sammelbefehle
7	1

MODBUS-Sendeereignis in entfernten Gerät

Das entfernte Gerät speichert diesen Typ Ereignis-Byte, wenn es eine Meldung mit einer Abfrage verarbeitet hat. Er wird gespeichert, wenn das entfernte Gerät eine normale Antwort, eine Ausnahmeantwort oder keine Antwort gibt. Das Ereignis ist definiert durch Bit 7 = "0" und Bit 6 = "1". Die übrigen Bits sind "1", wenn die zugehörige Bedingung wahr (TRUE) ist. Die Bits sind wie folgt aufgeteilt:

Bit-Inhalt

Bit	Beschreibung
0	Leseausnahme gesendet (Ausnahmecodes 1-3)
1	Ausnahme Abbruch Slave gesendet (Ausnahmecode 4)
2	Ausnahme Slave besetzt gesendet (Ausnahmecodes 5-6)
3	Ausnahme Slave-Programm NAK gesendet (Ausnahmecode 7)
4	Timeout-Fehler beim Schreiben aufgetreten

Bit	Beschreibung
5	Derzeit im Modus Listen Only
6	1
7	0

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Enable" auf TRUE gesetzt wird. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Falls während des Betriebs ein Fehler auftritt, wird das Bit „Error“ auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Enable" quittiert werden.

Beispiel eines Funktionsbausteins

Der Funktionsbaustein **IH_MBGetCommEventLog** muss zur Kommunikation mit dem Modbus-Slave zusammen mit **IH_MBRTUMaster** instanziiert werden. Für **MBData** der Funktionsbausteine **IH_MBGetCommEventLog** und **IH_MBRTUMaster** ist für den Datenaustausch mit dem Modbus-Slave eine gemeinsame Variable zu verwenden.

In diesem Beispiel ruft der Funktionsbaustein **IH_MBGetCommEventLog** das Protokoll des Slave mit Adresse 1 ab.

Die beiden Funktionsbausteine **IH_MBRTUMaster** und **IH_MBGetCommEventLog** werden mit steigender Flanke am Eingang **bSendRequest** aktiviert. Der Funktionsbaustein liest das Statuswort und die Ereignisanzahl von Slave-Adresse 1 und das Ergebnis wird in **stEventLog** gespeichert.

Wenn die Variable **bEnablePoll** auf TRUE und **PollInterval** auf mehr als 1 Sekunde gestellt werden, werden die Daten in regelmäßigen Intervallen aus dem Register gelesen.

Nach jedem erfolgreichen Abschluss der Leseoperation wird das Done-Bit für einen SPS-Zyklus auf TRUE gesetzt.

Variablendeklaration

```
VAR
    fbIH_MBGetCommEventLog: IH_MBGetCommEventLog;
    bEnable: BOOL;
    bSendRequest: BOOL;
    bEnablePoll: BOOL;
    stMBData: IH_INLN_MB_DATA;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
    bBusy: BOOL;
    stEventLog: IH_INLN_MB_EVENTLOG;
END_VAR
```

Implementation

```
fbIH_MBGetCommEventLog(
    Enable:= bEnable,
    SendRequest:= bSendRequest,
    EnablePoll:= bEnablePoll,
    SlaveAddress:= BYTE#1,
    PollInterval:= T#1S,
    Timeout:= T#1S,
    MBData:= stMBData,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
```

```
ErrorIdent=> stErrorIdent,  
Busy=> bBusy,  
Eventlog=> stEventLog);
```

Antwort von Ereignisprotokoll

In diesem Beispiel ist das Statuswort 16#0000, das zeigt an, dass das entfernte Gerät keine Programmfunktion verarbeitet. Der Ereigniszähler zeigt an, dass 264 (16#0108) vom entfernten Gerät gezählt wurden. Der Meldungszähler zeigt an, dass 289 (16#0121) Meldungen verarbeitet wurden.

Das zuletzt erfolgte Kommunikationsereignis wird im Ereignis-Byte 0 angezeigt. Sein Inhalt ist 32 (16#20), das bedeutet, dass das entfernte Gerät zuletzt in den Modus Listen Only übergegangen ist.

Das vorhergehende Kommunikationsereignis wird im Ereignis-Byte 1 angezeigt. Sein Inhalt (00 hex) zeigt an, dass die Kommunikation des entfernten Geräts neu gestartet wurde.

Expression	Type	Value
stEventLog	IH_INLN_MB_EVENTLOG	
Status	WORD	0
EventCount	WORD	264
MessageCount	WORD	289
Events	ARRAY [0..63] OF BYTE	
Events[0]	BYTE	32
Events[1]	BYTE	0

Abb. 20-4: stEventLog

20.4 IH_MBMaskWriteRegister

Funktionscode FC 22

Kurzbeschreibung Der Funktionsbaustein **IH_MBMaskWriteRegister** dient zum Ändern eines bestimmten Holding-Registers mit einer AND-Maske, einer OR-Maske und den Registern des Slave-Moduls. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein **IH_MBRTUMaster**, [Seite863](#) aufgerufen werden.

Zielsystem	Bibliothek /Ordner
MLC	RIH_Modbus / FB
MTX	RIH_Modbus / FB
MLD (IL 2G)	RIH_Modbus / FB

Tab. 20-8: Referenztablelle

Interfacebeschreibung

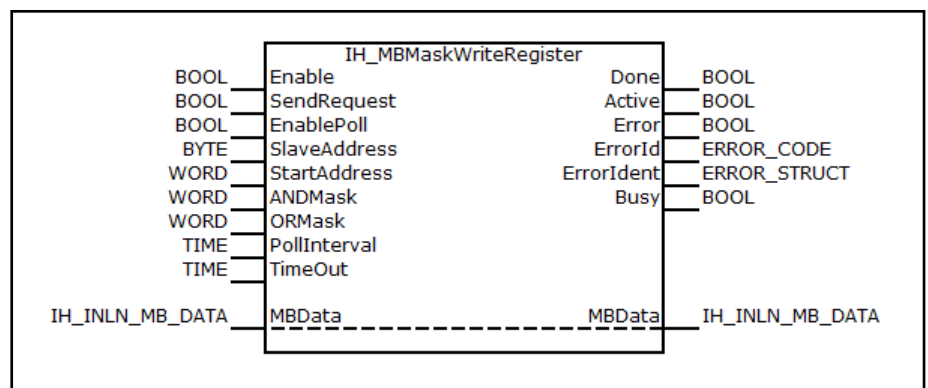


Abb. 20-5: Funktionsbaustein IH_MBMaskWriteRegister

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Funktionsbaustein aktiv, falls TRUE
	SendRequest	BOOL	Bei steigender Flanke wird eine Anfrage gesendet
	EnablePoll	BOOL	Polling aktiviert. SendRequest muss nicht in jedem Intervall ausgelöst werden. Der Baustein sendet Anfragen automatisch
	SlaveAddress	BYTE	Slave-Adresse
	StartAddress	WORD	Startadresse des Holding-Registers
	ANDMask	WORD	AND-Maskenwert
	ORMask	WORD	OR-Maskenwert
	PollInterval	TIME	Polling-Intervall. Anfrage wird kontinuierlich in Intervallen gesendet
	TimeOut	TIME	Timeout-Intervall für Antwort von Slaves
VAR_OUTPUT	Done	BOOL	Berechnung beendet und beginnt im nächsten Zyklus neu
	Active	BOOL	Funktionsbaustein ist aktiv
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Modbus, Seite 898
	Busy	BOOL	Wahr, wenn der Baustein mit dem Master kommuniziert
VAR_IN_OUT	MBData	IH_INLN_MB_DATA	Struktur handhabt Daten zwischen Master und anderen Modbus-Funktionsbausteinen der Funktionscodes IH_INLN_MB_DATA, Seite 894

Tab. 20-9: Schnittstellenvariablen von Funktionsbaustein IH_MBMaskWriteRegister

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
SendRequest	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
EnablePoll	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
SlaveAddress	BYTE	0	247	0	steigende Flanke an Enable
StartAddress	WORD	0	65535	0	steigende Flanke an Enable
ANDMask	WORD	0	65535	0	steigende Flanke an Enable
ORMask	WORD	0	65535	0	steigende Flanke an Enable
PollInterval	TIME	T#1s	-----	T#1s	Kontinuierlich bei Enable
TimeOut	TIME	T#1s	-----	T#1s	steigende Flanke an Enable

Tab. 20-10: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_MBMaskWriteRegister

Funktionsbeschreibung

Der Funktionsbaustein **IH_MBMaskWriteRegister** (FC22) dient zum Ändern eines bestimmten Holding-Registers mit einer AND-Maske, einer OR-Maske und den Registern des Slave-Moduls. Mit der Funktion können einzelne Bits im Register eingestellt oder gelöscht werden. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein aufgerufen werden. "MBData" muss mit dem Master-Funktionsbaustein verknüpft sein, um mit dem Master zu kommunizieren und Daten auszutauschen.

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Enable" auf TRUE gesetzt wird. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Falls während des Betriebs ein Fehler auftritt, wird das Bit „Error“ auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Enable" quittiert werden.

Beispiel eines Funktionsbausteins

Der Funktionsbaustein IH_MBMaskWriteRegister muss zur Kommunikation mit dem Modbus-Slave zusammen mit IH_MBRTUMaster instanziiert werden. Die Eingangs-/Ausgangsvariable stMBData tauscht die Daten zwischen den Funktionsbausteinen IH_MBRTUMaster und IH_MBMaskWriteRegister aus. Deshalb muss diese Variable in beiden Funktionsbausteinen MBData zugewiesen werden.

Im Beispiel wendet der Funktionsbaustein IH_MBMaskWriteRegister eine Maske auf ein Register des Slave-Geräts an.

Register mit SendRequest-Bedingung maskieren

IH_MBMaskWriteRegister dient zum Maskieren eines Registers der Adresse 100 in Slave-Adresse 2 mit "bSendRequest" als BOOL-Variable zum Auslösen der Registermaskierung.

Variablendeklaration

```
VAR
    fbIH_MBMaskWriteRegister: IH_MBMaskWriteRegister;
    bEnable: BOOL;
    bSendRequest: BOOL;
    stMBData: IH_INLN_MB_DATA;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
    bBusy: BOOL;
END_VAR
```

Implementation

```
fbIH_MBMaskWriteRegister(
    Enable:= bEnable,
    SendRequest:= bSendRequest,
    EnablePoll:= FALSE,
    SlaveAddress:= BYTE#2,
    StartAddress:= WORD#100,
    ANDMask:= WORD#16#F2,
    ORMask:= WORD#16#25,
    PollInterval:= T#2S,
    TimeOut:= T#2S,
    MBData:= stMBData,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent,
    Busy=> bBusy);
```

Expression	Type	Value
HoldingRegisters[99]	WORD	0
HoldingRegisters[100]	WORD	18 → Old data before Execution.
HoldingRegisters[101]	WORD	0

Expression	Type	Value
HoldingRegisters[99]	WORD	0
HoldingRegisters[100]	WORD	23 → New data After Execution.
HoldingRegisters[101]	WORD	0

Abb. 20-6: Zeigt die Änderung nach Ausführung im Slave-Gerät

20.5 IH_MBReadCoils

Funktionscode FC 1

Kurzbeschreibung Der Funktionsbaustein **IH_MBReadCoils** erkennt, ob Coils den Status ON oder OFF haben, liest also die Ausgangsbits vom Slave-Modul. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein **IH_MBRTU-Master**, Seite 863 aufgerufen werden.

Zielsystem	Bibliothek / Ordner
MLC	RIH_Modbus / FB
MTX	RIH_Modbus / FB
MLD (IL 2G)	RIH_Modbus / FB

Tab. 20-11: Referenztablelle

Interfacebeschreibung

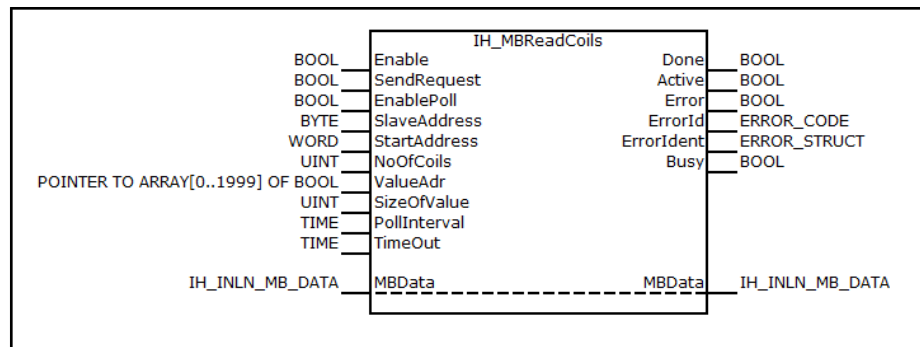


Abb. 20-7: Funktionsbaustein IH_MBReadCoils

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Funktionsbaustein aktiv, falls TRUE
	SendRequest	BOOL	Bei steigender Flanke wird eine Anfrage gesendet
	EnablePoll	BOOL	Polling aktiviert. SendRequest muss nicht in jedem Intervall ausgelöst werden. Der Baustein sendet Anfragen automatisch
	SlaveAddress	BYTE	Slave-Adresse
	StartAddress	WORD	Startadresse der Ausgangsbits
	NoOfCoils	UINT	Anzahl der Coils (getrennte Ausgänge)
	ValueAdr	POINTER TO ARRAY[0..1999] OF BOOL	Adresse des Arrays zum Speichern des Lesewerts der Ausgangsbits. Operator ADR () verwenden
	SizeOfValue	UINT	Größe des Value-Arrays. Operator SIZEOF () verwenden
	PollInterval	TIME	Polling-Intervall. Anfrage wird kontinuierlich in Intervallen gesendet
	TimeOut	TIME	Timeout-Intervall für Antwort von Slaves
VAR_OUTPUT	Done	BOOL	Berechnung beendet und beginnt im nächsten Zyklus neu
	Active	BOOL	Funktionsbaustein ist aktiv
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Modbus, Seite 898
	Busy	BOOL	Wahr, wenn der Baustein mit dem Master kommuniziert
VAR_IN_OUT	MBData	IH_INLN_MB_DATA	Struktur handhabt Daten zwischen Master und anderen Modbus-Funktionsbausteinen IH_INLN_MB_DATA, Seite 894

Tab. 20-12: Schnittstellenvariablen von Funktionsbaustein IH_MBReadCoils

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
SendRequest	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
EnablePoll	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
SlaveAddress	BYTE	0	247	0	steigende Flanke an Enable
StartAddress	WORD	0	65535	0	steigende Flanke an Enable
NoOfCoils	UINT	1	2000	1	steigende Flanke an Enable
SizeOfValue	UINT	1	2000	-----	steigende Flanke an Enable
PollInterval	TIME	T#1s	-----	T#1s	Kontinuierlich bei Enable
TimeOut	TIME	T#1s	-----	T#1s	steigende Flanke an Enable

Tab. 20-13: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_MBReadCoils

Funktionsbeschreibung

Der Funktionsbaustein **IH_MBReadCoils** (FC 1) liest den Status von 1 bis 2000 Coils, d. h. fortlaufend aufeinanderfolgende Ausgangsbits des Slave-Moduls. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein aufgerufen werden. „MBData“ muss mit dem Master-Funktionsbaustein im Programm verknüpft sein, er dient zur Kommunikation und zum Datenaustausch.

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Enable" auf TRUE gesetzt wird. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Falls während des Betriebs ein Fehler auftritt, wird das Bit „Error“ auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Enable" quittiert werden.

Beispiel eines Funktionsbausteins

Der Funktionsbaustein IH_MBReadCoils muss zur Kommunikation mit dem Modbus-Slave zusammen mit dem Funktionsbaustein IH_MBRTUMaster instanziiert werden. Die Eingangs-/Ausgangsvariable stMBData tauscht die Daten zwischen den Funktionsbausteinen IH_MBRTUMaster und IH_MBReadCoils aus. Deshalb muss diese Variable in beiden Funktionsbausteinen MBData zugewiesen werden.

Im Beispiel wird der Funktionsbaustein IH_MBReadCoils zum Auslesen von Coils des Slave-Geräts verwendet.

Coils mit SendRequest-Bedingung lesen

IH_MBReadCoils dient zum Lesen von 8 Coils der Startadresse 0x1000 von Slave-Adresse 2 mit "bSendRequest" als BOOL-Variable zum Auslösen des Lesens der Coils.

Variablendeklaration

```

VAR
    fbIH_MBReadCoils: IH_MBReadCoils;
    bEnable: BOOL;
    bSendRequest: BOOL;
    arReadCoils: ARRAY [0..20] OF BOOL;
    stMBData: IH_INLN_MB_DATA;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
    bBusy: BOOL;
END_VAR

```

Implementation

```

fbIH_MBReadCoils(
    Enable:= bEnable,
    SendRequest:= bSendRequest,
    EnablePoll:= FALSE,
    SlaveAddress:= BYTE#2,
    StartAddress:= WORD#1000,
    NoOfCoils:= UINT#8,
    ValueAdr:= ADR(arReadCoils),
    SizeOfValue:= SIZEOF(arReadCoils),
    PollInterval:= T#2S,
    Timeout:= T#2S,
    MBData:= stMBData,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent,
    Busy=> bBusy);

```

Expression	Type	Value
arReadCoils	ARRAY [0..20] OF BOOL	
arReadCoils[0]	BOOL	TRUE → Data value of Coil(1000)
arReadCoils[1]	BOOL	FALSE → Data value of Coil(1001)
arReadCoils[2]	BOOL	TRUE
arReadCoils[3]	BOOL	FALSE
arReadCoils[4]	BOOL	TRUE
arReadCoils[5]	BOOL	FALSE
arReadCoils[6]	BOOL	TRUE
arReadCoils[7]	BOOL	FALSE → Data value of Coil(1007)

Abb. 20-8: arReadCoils' für Programm 1

Coils mit PollIntervall-Bedingung lesen

IH_MBReadCoils dient zum Lesen von 5 Coils der Startadresse 0x0500 von Slave-Adresse 5 mit "PollInterval" (3 Sekunden) zum Auslösen des Lesens der Coils.

Variablendeklaration

```

VAR
    fbIH_MBReadCoils2: IH_MBReadCoils;
    bEnable2: BOOL;
    arReadCoils2: ARRAY [0..4] OF BOOL;
    stMBData: IH_INLN_MB_DATA;
    bDone2: BOOL;
    bActive2: BOOL;
    bError2: BOOL;
    enErrorID2: ERROR_CODE;
    stErrorIdent2: ERROR_STRUCT;
    bBusy2: BOOL;
END_VAR

```

Implementation

```
fbIH_MBReadCoils2(  
  Enable:= bEnable2,  
  SendRequest:= FALSE,  
  EnablePoll:= TRUE,  
  SlaveAddress:= BYTE#5,  
  StartAddress:= WORD#500,  
  NoOfCoils:= UINT#5,  
  ValueAdr:= ADR(arReadCoils2),  
  SizeOfValue:= SIZEOF(arReadCoils2),  
  PollInterval:= T#3S,  
  Timeout:= T#2S,  
  MBData:= stMBData,  
  Done=> bDone2,  
  Active=> bActive2,  
  Error=> bError2,  
  ErrorID=> enErrorID2,  
  ErrorIdent=> stErrorIdent2,  
  Busy=> bBusy2);
```

Expression	Type	Value
arReadCoils2	ARRAY [0..4] OF BOOL	
arReadCoils2[0]	BOOL	TRUE → Data value of Coil(500)
arReadCoils2[1]	BOOL	FALSE → Data value of Coil(501)
arReadCoils2[2]	BOOL	TRUE
arReadCoils2[3]	BOOL	FALSE
arReadCoils2[4]	BOOL	TRUE → Data value of Coil(504)

Abb. 20-9: arReadCoils2' für Programm 2

20.6 IH_MBReadDeviceIdentification

Funktionscode FC 43

Kurzbeschreibung Der **IH_MBReadDeviceIdentification** Funktionsbaustein nutzt FC 43(0x2B) mit dem MEI (MODBUS Encapsulated Interface) Typ 14 (0x0E), um die ID und zusätzliche Informationen bezüglich der physikalischen und funktionalen Beschreibung eines Remote-Gerätes aus dem Client (Master) auszulesen. Es wird empfohlen, den Funktionsbaustein mit dem Modbus Master Funktionsbaustein aufzurufen [IH_MBRTUMaster](#), Seite 863.

Zielsystem	Bibliothek/Datei
MLC	RIH_Modbus/FB
MTX	RIH_Modbus/FB
MLD (IL 2G)	RIH_Modbus/FB

Tab. 20-14: Referenztablelle

Schnittstellenbeschreibung

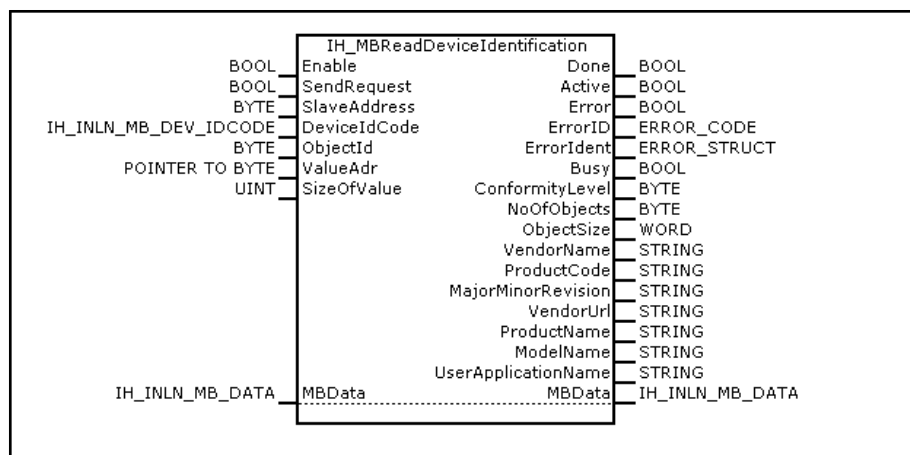


Abb. 20-10: IH_MBReadDeviceIdentification Funktionbaustein

I/O-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Funktionsbaustein läuft wenn TRUE
	SendRequest	BOOL	Eine steigende Flanke sendet eine Anfrage
	SlaveAddress	BYTE	Slave-Adresse
	DeviceIdCode	IH_INLN_MB_DEV_ID CODE	Parameter für Definition von Zugriffstypen der verfügbaren Objekt IDs (Auslesen des ID-Codes des Geräts) IH_INLN_MB_DEV_IDCODE, Seite 895
	ObjectId	BYTE	Datenobjektidentifikation (Objekt ID)
	ValueAdr	POINTER TO BYTE	Objektadresse
	SizeOfValue	UINT	Objektgröße
	Timeout	TIME	Timeout-Intervall in dem der Slave antwortet
VAR_OUTPUT	Done	BOOL	Berechnung abgeschlossen; wird im nächsten Zyklus erneut gestartet
	Active	BOOL	Funktionsbaustein ist aktiv
	Error	BOOL	Zeigt Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detaillierte Fehlerinformationen Fehlerbehandlung mit der Bibliothek RIH_Modbus, Seite 898
	Busy	BOOL	TRUE wenn der Baustein mit dem Master kommuniziert
	ConformityLevel	BYTE	Identifikationskonformitätslevel des Geräts und Typ des unterstützten Zugriffs
	NoOfObjects	BYTE	Anzahl der Identifikationsobjekte die in der Antwort zurückgegeben werden
	ObjectSize	WORD	Gesamte Datengröße der angeforderten 'DeviceIdCode' oder 'ObjectId' Elemente in Bytes ObjectSize = (ObjectId (1 Byte)+Objektlänge (1 Byte)+ Wert in Objectlength) * NoOfObjects

I/O-Typ	Name	Datentyp	Beschreibung
	VendorName	STRING	'VendorName' der von der Objekt ID zurückgegeben wird: 0x00
	ProductCode	STRING	'ProductCode' der von der Objekt ID zurückgegeben wird: 0x01
	MajorMinorRevision	STRING	'MajorMinorRevision' der von der Objekt ID zurückgegeben wird: 0x02
	VendorUrl	STRING	'VendorUrl' der von der Objekt ID zurückgegeben wird: 0x03
	ProductName	STRING	'ProductName' der von der Objekt ID zurückgegeben wird: 0x04
	ModelName	STRING	'ModelName' der von der Objekt ID zurückgegeben wird: 0x05
	UserApplicationName	STRING	'UserApplicationName' der von der Objekt ID zurückgegeben wird: 0x06
VAR_IN_OUT	MBData	IH_INLN_MB_DATA	Struktur, die verwendet wird, um Daten zwischen dem Master, anderen Modbus Funktionsbausteinen und Funktionscode zu verwalten IH_INLN_MB_DATA, Seite 894

Tab. 20-15: Schnittstellenvariablen des IH_MBReadDeviceIdentification Funktionsbausteins

Min./max. und Standardwerte

Die folgende Tabelle gibt die min./max. Werte, die Standardwerte sowie das Übernahmeverhalten der Funktionsbausteineingänge an:

Name	Typ	Min. Wert	Max. Wert	Standardwert	Anwendung
Enable	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
SendRequest	BOOL	FALSE	TRUE	FALSE	Kontinuierlich mit "Enable"
SlaveAddress	BYTE	0	247	0	Steigende Flanke bei "Enable"
DeviceIdCode	IH_INLN_MB_DEV_ID CODE	1	4	1	Steigende Flanke bei "Enable"
ObjectId	BYTE	16#00	16#FF	16#00	Steigende Flanke bei "Enable"
ValueAdr	POINTER TO BYTE	nicht definiert	nicht definiert	0	Kontinuierlich mit "Enable"
SizeOfValue	UINT	0	Abhängig von der Geräteobjektgröße des Slaves	0	Steigende Flanke bei "Enable"
Timeout	TIME	T#0ms	T#3600s	T#1s	Steigende Flanke bei "Enable"

Tab. 20-16: Min., max. Werte und Standardwerte sowie das Übernahmeverhalten der Eingänge des IH_MBReadDeviceIdentification Funktionsbausteins

Funktionsbeschreibung

Der **IH_MBReadDeviceIdentification** Funktionsbaustein nutzt FC 43(0x2B) mit dem MEI (MODBUS Encapsulated Interface) Typ 14 (0x0E), um die ID und zusätzliche Informationen bezüglich der physikalischen und funktionalen Beschreibung eines Remote-Gerätes aus dem Client (Master) auszulesen.

Adressierbare Datenobjekte lesen die Geräteidentifikation aus. Die Datenobjekte werden mittels der Objekt-ID adressiert. Mithilfe des Funktionsbausteins wird auf die folgenden Datenobjektkategorien zugegriffen:

- **Basisgeräteidentifikation:** Alle Objekte dieser Kategorie müssen vorhanden sein: VendorName, ProductCode und Änderungsversion
- **Allgemeine Geräteidentifikation:** Alle Objekte dieser Kategorien sind im Standard definiert; die Implementierung ist optional.
- **Erweiterte Geräteidentifikation:** Stellt die zusätzliche und optionale Identifikation sowie die privaten Beschreibungsdaten über das Gerät selbst bereit. Alle Geräte sind abhängig und optional.

Die unterstützten Datenobjekte mit der Objekt-ID und den Kategorien werden in der folgenden Tabelle angezeigt.

Objekt-ID	Objektname/Beschreibung	Typ	M/O	Kategorie
16#00	VendorName	STRING	Mussfeld	Basis
16#01	ProductCode	STRING	Mussfeld	
16#02	MajorMinorRevision	STRING	Mussfeld	
16#03	VendorUrl	STRING	Optional	Regulär
16#04	ProductName	STRING	Optional	
16#05	ModelName	STRING	Optional	
16#06	UserApplicationName	STRING	Optional	
16#07 ... 16#7F	Reserviert		Optional	
16#80 ... 16#FF	Private Objekte können optional definiert werden Der Bereich [16#80 - 16#FF] ist produktabhängig	Geräteabhängig	Optional	Erweitert

Objektkategoriezugangsebenen werden im "DeviceIdCode" Eingang angegeben. Der Funktionsbaustein unterstützt die folgenden Read Device ID-Code-Zugangstypen:

- 01: **IH_INLN_MB_READ01** - fordert die Basisgeräteidentifikation an (Zugang über Stream)
- 02: **IH_INLN_MB_READ02** - fordert die reguläre Geräteidentifikation an (Zugang über Stream)
- 03: **IH_INLN_MB_READ03** - fordert die erweiterte Geräteidentifikation an (Zugang über Stream)
- 04: **IH_INLN_MB_READ04** - fordert ein spezifisches Identifikationsobjekt an (benutzerdefinierter Zugang)

Beim benutzerdefinierten Objekt-ID Zugang "DeviceIdCode" als **IH_INLN_MB_READ04**, geben Sie die gewünschte Objekt-ID im "ObjectId" Eingang an. Für die restlichen ID-Codes der Geräte sollte die "ObjectId" 0 sein.

Objekte mit definiertem Datentyp in den Basis- und Regulär-Kategorien werden als direkte Funktionbausteinausgänge bearbeitet, z.B. "VendorName", "ProductCode" ... "UserApplicationName". Alle anderen geräteabhängigen Objekte werden standardmäßig als Byte-Datentyp per "ValueAdr" verarbeitet.

Für ObjectIDs größer 7, sind die gelesenen Daten in "ValueAdr" verfügbar. "ValueAdr" enthält die ObjectID, ObjectSize und den Objektwert.

Gibt es mehrere Objekte, werden die Daten im folgenden Format gespeichert und die gültige Datengröße kann der "ObjectSize" entnommen werden.

◆ arValueAdr[0]	BYTE	144	→	ObjectID 1	
◆ arValueAdr[1]	BYTE	2	→	ObjectSize 1	
◆ arValueAdr[2]	BYTE	42	}	Object 1	
◆ arValueAdr[3]	BYTE	42			
◆ arValueAdr[4]	BYTE	145	→	ObjectID 2	
◆ arValueAdr[5]	BYTE	1	→	ObjectSize 2	
◆ arValueAdr[6]	BYTE	42	→	Object 2	
◆ arValueAdr[7]	BYTE	0			

Abb. 20-11: Mehrere Objekte

Der "ConformityLevel" Ausgang gibt das tatsächliche Identifikationskonformitätslevel des Geräts sowie den unterstützten Zugangstyp der angeforderten Eingänge "objectId" und "DeviceIdCode" an. Die folgenden Information wird in "ConformityLevel" zurückgeliefert:

- 0x01: Basisidentifikation (Zugang nur über Stream)
- 0x02: Reguläre Identifikation (Zugang nur über Stream)
- 0x03: Erweiterte Identifikation (Zugang nur über Stream)
- 0x81: Basisidentifikation (Zugang über Stream und benutzerdefinierter Zugang)
- 0x82: Reguläre Identifikation (Zugang über Stream und benutzerdefinierter Zugang)
- 0x83: Erweiterte Identifikation (Zugang über Stream und benutzerdefinierter Zugang)

Der "NoOfObjects" Ausgang gibt die Anzahl der zurückgelieferten Identifikationsobjekte bei angefordertem "DeviceIdCode" an.

Bei DeviceIdCode = IH_INLN_MB_READ04, ist die "NoOfObjects" immer 1.

Die Größe der Objekt-IDs 0x07...0xFF müssen im Gerätedatenblatt identifiziert werden können und die gewünschte Länge muss zugewiesen werden um die Daten zu vervollständigen.

Dieser Funktionsbaustein wird mit dem MODBUS Master-Funktionsbaustein aufgerufen. "MBData" verknüpft den Master-Funktionsbaustein im Programm. MBData ist eine allgemeine Struktur, um zwischen diesen 2 Funktionsbausteinen zu kommunizieren und Daten auszutauschen.

Die steigende Flanke an den Enable-Eingängen startet den Vorgang. Wird der Vorgang erfolgreich beendet, wird der "Done" Ausgang auf TRUE gesetzt. Tritt ein Fehler auf, wird das Error-Bit gesetzt und detaillierte Informationen über den Fehler werden in ErrorID und ErrorIdent bereitgestellt.

Besonderheiten

Dieser Funktionsbaustein wird nur mit einem CIMS-Sensor getestet und nicht mit anderen Geräten, die den Funktionscode 43 unterstützen.

Obwohl der Funktionsbaustein das Auslesen des Geräte-ID Codes aus IH_INLN_MB_READ01 bis IH_INLN_MB_READ04 unterstützt, unterstützt der CIMS-Sensor nur IH_INLN_MB_READ04. Für andere Geräte-ID Codes wurden keine Tests durchgeführt.

Beispiel 1: Kommunikation mit dem Bosch Rexroth Sensor

Deklaration

```

VAR
  fbIH_MBReadDeviceIdentification: IH_MBReadDeviceIdentification;
  bEnable: BOOL;
  bDone: BOOL;
  bActive: BOOL;
  bError: BOOL;
  enErrorID: ERROR_CODE;
  stErrorIdent: ERROR_STRUCT;
  bSendRequest: BOOL;
  bySlaveAddress: BYTE := 162;
  enDeviceIDCode: IH_INLN_MB_DEV_IDCODE := IH_INLN_MB_DEV_IDCODE.IH_INLN_MB_READ_DEVID_04;
  byObjectID: BYTE := 0;
  arValueAdr: array[0..30] of byte;
  bBusy: BOOL;
  byConformityLevel: BYTE;
  byNoOfObjects: BYTE;
  wObjectSize: WORD;
  strVendorName: STRING;
  strProductCode: STRING;
  strMajorMinorRevision: STRING;
  strVendorUrl: STRING;
  strProductName: STRING;
  strModelName: STRING;
  strUserApplicationName: STRING;
END_VAR

```

Implementierung

```

fbIH_MBReadDeviceIdentification(
  Enable:= bEnable,
  Done=> bDone,
  Active=> bActive,
  Error=> bError,
  ErrorID=> enErrorID,
  ErrorIdent=> stErrorIdent,
  SendRequest:= bSendRequest,
  SlaveAddress:= bySlaveAddress,
  DeviceIDCode:= enDeviceIDCode,
  ObjectID:= byObjectID,
  ValueAdr:= ADR(arValueAdr),
  SizeofValue:= SIZEOF(arValueAdr),
  Timeout:= T#2S,
  Busy=> bBusy,
  ConformityLevel=> byConformityLevel,
  NoOfObjects=> byNoOfObjects,
  ObjectSize=> wObjectSize,
  VendorName=> strVendorName,
  ProductCode=> strProductCode,
  MajorMinorRevision=> strMajorMinorRevision,
  VendorUrl=> strVendorUrl,
  ProductName=> strProductName,
  ModelName=> strModelName,
  UserApplicationName=> strUserApplicationName,
  MBData:= stMBData);

```

Expression	Type	Value
* fIH_MBReadDeviceIdentification	IH_MBReadDeviceIdentifica...	
bEnable	BOOL	TRUE
bDone	BOOL	FALSE
bActive	BOOL	TRUE
bError	BOOL	FALSE
enErrorID	ERROR_CODE	NONE_ERROR
* stErrorIdent	ERROR_STRUCT	
bSendRequest	BOOL	TRUE
bySlaveAddress	BYTE	162
enDeviceIDCode	IH_INLN_MB_DEV_IDCODE	IH_INLN_MB_READ_DEVID_04
byObjectID	BYTE	0
* arValueAdr	ARRAY [0..30] OF BYTE	
bBusy	BOOL	FALSE
byConformityLevel	BYTE	131
byNoOfObjects	BYTE	1
wObjectSize	WORD	18
strVendorName	STRING	'Bosch Rexroth B.V.'
strProductCode	STRING	"
strMajorMinorRevision	STRING	"
strVendorUrl	STRING	"
strProductName	STRING	"
strModelName	STRING	"
strUserApplicationName	STRING	"

Abb. 20-12: Ergebnis 1

Beispiel 2:*Deklaration*

```

VAR
byObjectID: BYTE := 16#90; // Read order number in CIMS Sensor.
END_VAR

```

Expression	Type	Value
bDone	BOOL	FALSE
bActive	BOOL	TRUE
bError	BOOL	FALSE
enErrorID	ERROR_CODE	NONE_ERROR
* stErrorIdent	ERROR_STRUCT	
bSendRequest	BOOL	TRUE
bySlaveAddress	BYTE	162
enDeviceIDCode	IH_INLN_MB_DEV_IDCODE	IH_INLN_MB_READ_DEVID_04
byObjectID	BYTE	144
bBusy	BOOL	FALSE
byConformityLevel	BYTE	131
byNoOfObjects	BYTE	1
wObjectSize	WORD	13
strVendorName	STRING	"
* arValueAdr	ARRAY [0..30] OF BYTE	
arValueAdr[0]	BYTE	144
arValueAdr[1]	BYTE	13
arValueAdr[2]	BYTE	70
arValueAdr[3]	BYTE	32
arValueAdr[4]	BYTE	48
arValueAdr[5]	BYTE	51
arValueAdr[6]	BYTE	55
arValueAdr[7]	BYTE	32
arValueAdr[8]	BYTE	66
arValueAdr[9]	BYTE	48
arValueAdr[10]	BYTE	48
arValueAdr[11]	BYTE	32
arValueAdr[12]	BYTE	48
arValueAdr[13]	BYTE	48
arValueAdr[14]	BYTE	49
arValueAdr[15]	BYTE	0

ObjectID
ObjectSize

Object

Abb. 20-13: Ergebnis 2

20.7 IH_MBReadDiagnostics

Funktionscode FC 8

Kurzbeschreibung Der Funktionsbaustein **IH_MBReadDiagnostics** stellt eine Reihe Tests zum Prüfen des Kommunikationssystems zwischen einem Client (Master) und einem Server (Slave) oder zum Prüfen verschiedener interner Fehlerbedingungen eines Servers im RTU-Modus bereit. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein [IH_MBRTUMaster](#), [Seite863](#) aufgerufen werden.

Zielsystem	Bibliothek /Ordner
MLC	RIH_Modbus / FB
MTX	RIH_Modbus / FB
MLD (IL 2G)	RIH_Modbus / FB

Tab. 20-17: Referenztabelle

Interfacebeschreibung

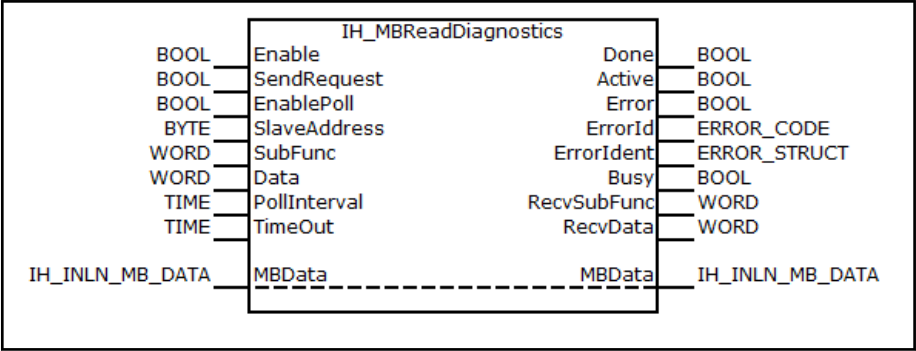


Abb. 20-14: Funktionsbaustein IH_MBReadDiagnostics

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Funktionsbaustein aktiv, falls TRUE
	SendRequest	BOOL	Bei steigender Flanke wird eine Anfrage gesendet
	EnablePoll	BOOL	Polling aktiviert. SendRequest muss nicht in jedem Intervall ausgelöst werden. Der Baustein sendet Anfragen automatisch
	SlaveAddress	BYTE	Slave-Adresse
	SubFunc	WORD	Unterfunktionscode
	Daten	WORD	Datenfeld
	PollInterval	TIME	Polling-Intervall. Anfrage wird kontinuierlich in Intervallen gesendet
	TimeOut	TIME	Timeout-Intervall für Antwort von Slaves

E/A-Typ	Name	Datentyp	Beschreibung
VAR_OUTPUT	Done	BOOL	Berechnung beendet und beginnt im nächsten Zyklus neu
	Active	BOOL	Funktionsbaustein ist aktiv
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Modbus, Seite 898
	Busy	BOOL	Wahr, wenn der Baustein mit dem Master kommuniziert
	RecvSubFunc	WORD	Vom Gerät empfangener Unterfunktionscode
	RecvData	WORD	Antwortdaten vom Gerät
VAR_IN_OUT	MBData	IH_INLN_MB_DATA	Struktur handhabt Daten zwischen Master und anderen Modbus-Funktionsbausteinen IH_INLN_MB_DATA, Seite 894

Tab. 20-18: Schnittstellenvariablen von Funktionsbaustein *IH_MBReadDiagnostics*

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
SendRequest	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
EnablePoll	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
SlaveAddress	BYTE	0	247	0	steigende Flanke an Enable
PollInterval	TIME	T#1s	-----	T#1s	Kontinuierlich bei Enable
TimeOut	TIME	T#1s	-----	T#1s	steigende Flanke an Enable

Tab. 20-19: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von *IH_MBReadDiagnostics*

Funktionsbeschreibung

Der Funktionsbaustein **IH_MBReadDiagnostics** stellt eine Reihe Tests zum Prüfen des Kommunikationssystems zwischen einem Client (Master) und einem Server (Slave) oder zum Prüfen verschiedener interner Fehlerbedingungen eines Servers bereit. Die Funktion verwendet in der Anfrage ein Feld für einen 2-Byte-Unterfunktionscode zur Definition des durchzuführenden Tests. Der Server wiederholt sowohl den Funktionscode als auch den Unterfunktionscode in einer normalen Antwort. Bei einigen Diagnosen werden Daten vom entfernten Gerät im Datenfeld einer Normalen Antwort zurückgegeben. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein aufgerufen werden. "MBData" muss mit dem Master-Funktionsbaustein im Programm verknüpft sein, er dient zur Kommunikation und zum Datenaustausch. Folgende Unterfunktionscodes werden von seriellen Geräten unterstützt:

Diagnosenname	Unterfunktionscode	Daten (angefordert)	Daten (Antwort)
Return Query Data	16#0000	ANY	Anfragedaten wiederholen
Restart Communications Option	16#0001	16#0000 (oder) 16#FF00	Anfragedaten wiederholen
Return Diagnostic Register	16#0002	16#0000	Inhalt des Diagnoseregisters
Change ASCII Input Delimiter	16#0003	CHAR 00	Anfragedaten wiederholen
Force Listen Only Mode	16#0004	16#0000	Keine Antwort
RESERVED	16#05 ... 16#09	-----	-----
Clear Counters and Diagnostic Register	16#000A	16#0000	Anfragedaten wiederholen
Return Bus Message Count	16#000B	16#0000	Gesamtanzahl Meldungen
Return Bus Communication Error Count	16#000C	16#0000	CRC-Fehleranzahl
Return Bus Exception Error Count	16#000D	16#0000	Anzahl Ausnahmefehler
Return Slave Message Count	16#000E	16#0000	Anzahl Slave-Meldungen
Return Slave No Response Count	16#000F	16#0000	Anzahl keine Antwort vom Slave
Return Slave NAK Count	16#0010	16#0000	Anzahl NAK vom Slave
Return Slave Busy Count	16#0011	16#0000	Anzahl Slave-Gerät ausgelastet
Return Bus Character Overrun Count	16#0012	16#0000	Anzahl Zeichenüberlauf Slave
RESERVED	16# 0013	-----	-----
Clear Overrun Counter and Flag	16#0014	16#0000	Anfragedaten wiederholen
RESERVED	keine Angabe	-----	-----

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Enable" auf TRUE gesetzt wird. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Falls während des Betriebs ein Fehler auftritt, wird das Bit „Error“ auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang „Enable“ quittiert werden.

Beispiel eines Funktionsbausteins

Der Funktionsbaustein IH_MBReadDiagnostics muss zur Kommunikation mit dem Modbus-Slave zusammen mit dem Funktionsbaustein IH_MBRTUMaster instanziiert werden. Die Eingangs-/Ausgangsvariable stMBData tauscht die Daten zwischen den Funktionsbausteinen IH_MBRTUMaster und IH_MBReadDiagnostics aus. Deshalb muss diese Variable in beiden Funktionsbausteinen MBData zugewiesen werden.

Im Beispiel wird der Funktionsbaustein IH_MBReadDiagnostics zum Auslesen des Diagnosestatus des Slave-Geräts verwendet.

Diagnose mit SendRequest-Bedingung lesen

IH_MBReadDiagnostics wird zum Lesen von (Return Diagnostic Register) von Slave-Adresse 2 verwendet.

Variablendeklaration

```
VAR
    fbIH_MBReadDiagnostics: IH_MBReadDiagnostics;
    bEnable: BOOL;
    bSendRequest: BOOL;
    wSubFunc : word;
    wData : word;
    stMBData: IH_INLN_MB_DATA;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
    bBusy: BOOL;
    wRecvSubFunc: WORD;
    wRecvdata: WORD;
END_VAR
```

Implementation

```
fbIH_MBReadDiagnostics(
    Enable:= bEnable,
    SendRequest:= bSendRequest,
    EnablePoll:= FALSE,
    SlaveAddress:= BYTE#2,
    SubFunc:= wSubFunc,
    Data:= wData,
    PollInterval:= T#2S,
    TimeOut:= T#2S,
    MBData:= stMBData,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent,
    Busy=> bBusy,
    RecvSubFunc:= wRecvSubFunc,
    RecvData=> wRecvdata);
```

Expression	Type	Value
wSubFunc	WORD	0
wData	WORD	4660
wRecvSubFunc	WORD	0
wRecvdata	WORD	4660

Inputs: wSubFunc, wData
Outputs: wRecvSubFunc, wRecvdata

Abb. 20-15: Antwort für das Programm

20.8 IH_MBReadDiscreteInputs

Funktionscode FC 2

Kurzbeschreibung Der Funktionsbaustein **IH_MBReadDiscreteInputs** dient zum Lesen von Eingangsbits im RTU-Modus. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein **IH_MBRTUMaster**, [Seite 863](#) aufgerufen werden.

Zielsystem	Bibliothek /Ordner
MLC	RIH_Modbus / FB
MTX	RIH_Modbus / FB
MLD (IL 2G)	RIH_Modbus / FB

Tab. 20-20: Referenztablelle

Interfacebeschreibung

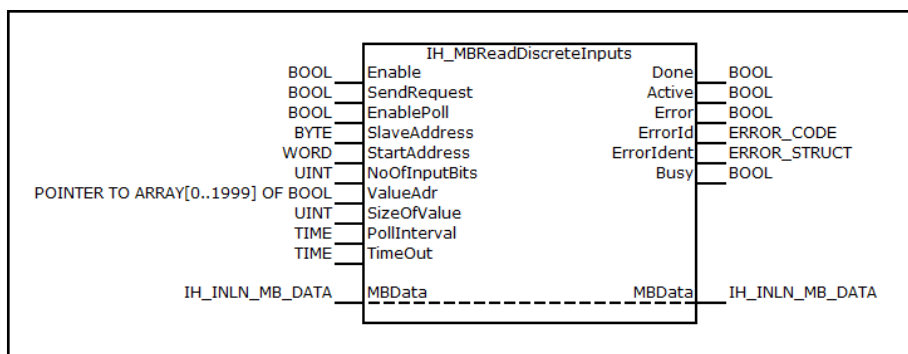


Abb. 20-16: Funktionsbaustein IH_MBReadDiscreteInputs

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Funktionsbaustein aktiv, falls TRUE
	SendRequest	BOOL	Bei steigender Flanke wird eine Anfrage gesendet
	EnablePoll	BOOL	Polling aktiviert. SendRequest muss nicht in jedem Intervall ausgelöst werden. Der Baustein sendet Anfragen automatisch
	SlaveAddress	BYTE	Slave-Adresse
	StartAddress	WORD	Startadresse der Eingangsbits
	NoOfInputBits	UINT	Anzahl zu lesender Eingangsbits
	ValueAdr	POINTER TO ARRAY[0..1999] OF BOOL	Adresse des Arrays zum Speichern der Eingangsbits Operator ADR () verwenden
	SizeOfValue	UINT	Größe des Value-Arrays. Operator SIZEOF () verwenden
	PollInterval	TIME	Polling-Intervall. Anfrage wird kontinuierlich in Intervallen gesendet
	TimeOut	TIME	Timeout-Intervall für Antwort von Slaves
VAR_OUTPUT	Done	BOOL	Berechnung beendet und beginnt im nächsten Zyklus neu
	Active	BOOL	Funktionsbaustein ist aktiv
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Modbus, Seite 898
	Busy	BOOL	Wahr, wenn der Baustein mit dem Master kommuniziert
VAR_IN_OUT	MBData	IH_INLN_MB_DATA	Struktur handhabt Daten zwischen Master und anderen Modbus-Funktionsbausteinen IH_INLN_MB_DATA, Seite 894

Tab. 20-21: Schnittstellenvariablen von Funktionsbaustein IH_MBReadDiscreteInputs

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Type	Min Value	Max Value	Defaultwert	Wirksam
Enable	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
SendRequest	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
EnablePoll	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
SlaveAddress	BYTE	0	247	0	steigende Flanke an Enable
StartAddress	WORD	0	16#FFFF	0	steigende Flanke an Enable
NoOfInputBits	UINT	1	2000	1	steigende Flanke an Enable
SizeOfValue	UINT	1	2000	-----	steigende Flanke an Enable
PollInterval	TIME	T#1s	-----	T#1s	Kontinuierlich bei Enable
TimeOut	TIME	T#1s	-----	T#1s	steigende Flanke an Enable

Tab. 20-22: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_MBReadDiscreteInputs

Funktionsbeschreibung

Der Funktionsbaustein **IH_MBReadDiscreteInputs** (FC 2) dient zum Lesen von Eingangsbits im RTU-Modus. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein **IH_MBRТУMaster** aufgerufen werden. "MBData" muss mit dem Master-Funktionsbaustein im Programm verknüpft sein, er dient zur Kommunikation und zum Datenaustausch.

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Enable" auf TRUE gesetzt wird. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Falls während des Betriebs ein Fehler auftritt, wird das Bit „Error“ auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Enable" quittiert werden.

Beispiel eines Funktionsbausteins

Der Funktionsbaustein **IH_MBReadDiscreteInputs** muss zur Kommunikation mit dem Modbus-Slave zusammen mit dem Funktionsbaustein **IH_MBRТУMaster** instanziiert werden. Die Eingangs-/Ausgangsvariable **stMBData** tauscht die Daten zwischen den Funktionsbausteinen **IH_MBRТУMaster** und **IH_MBReadDiscreteInputs** aus. Deshalb muss diese Variable in beiden Funktionsbausteinen **MBData** zugewiesen werden.

Im Beispiel wird der Funktionsbaustein **IH_MBReadDiscreteInputs** zum Auslesen von getrennten Eingängen (Discrete Inputs) des Slave-Geräts verwendet.

Getrennte Eingänge mit SendRequest-Bedingung lesen

IH_MBReadDiscreteInputs dient zum Lesen von 10 getrennten Eingängen der Startadresse 0x1000 von Slave-Adresse 2 mit "bSendRequest" als BOOL-Variable zum Auslösen des Lesens der Eingänge.

Variablendeklaration

```

VAR
    fbIH_MBReadDiscreteInputs: IH_MBReadDiscreteInputs;
    bEnable: BOOL;
    bSendRequest: BOOL;
    arDiscreteInputs: ARRAY [0..2000] OF BOOL;
    stMBData: IH_INLN_MB_DATA;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
    bBusy: BOOL;
END_VAR

```

Implementation

```

fbIH_MBReadDiscreteInputs (
    Enable:= bEnable,
    SendRequest:= bSendRequest,
    EnablePoll:= FALSE,
    SlaveAddress:= BYTE#2,
    StartAddress:= WORD#1000,
    ValueAdr:= ADR(arDiscreteInputs),
    NoOfInputBits:= UINT#10,
    SizeOfValue:= SIZEOF(arDiscreteInputs),
    PollInterval:= T#2S,
    TimeOut:= T#2S,
    MBData:= stMBData,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent,
    Busy=> bBusy);

```

Expression	Type	Value
arDiscreteInputs	ARRAY [0..2000] OF BOOL	
arDiscreteInputs[0]	BOOL	TRUE → Data of Address (1000)
arDiscreteInputs[1]	BOOL	FALSE → Data of Address (1001)
arDiscreteInputs[2]	BOOL	TRUE → Data of Address (1002)
arDiscreteInputs[3]	BOOL	FALSE
arDiscreteInputs[4]	BOOL	TRUE
arDiscreteInputs[5]	BOOL	FALSE
arDiscreteInputs[6]	BOOL	TRUE
arDiscreteInputs[7]	BOOL	FALSE
arDiscreteInputs[8]	BOOL	TRUE
arDiscreteInputs[9]	BOOL	FALSE → Data of Address (1009)

Abb. 20-17: *arReadDiscreteInputs* für Programm 1

Getrennte Eingänge mit PollInterval-Bedingung lesen

IH_MBReadDiscreteInputs dient zum Lesen von 5 getrennten Eingängen der Startadresse 0x0500 von Slave-Adresse 5 mit Polling-Intervall (3 Sekunden) zum Auslösen des Lesens der Eingänge.

Variablendeklaration

```

VAR
    fbIH_MBReadDiscreteInputs2: IH_MBReadDiscreteInputs;
    bEnable2: BOOL;
    arDiscreteInputs2: ARRAY [0..10] OF BOOL;
    stMBData: IH_INLN_MB_DATA;
    bDone2: BOOL;
    bActive2: BOOL;
    bError2: BOOL;
    enErrorID2: ERROR_CODE;
    stErrorIdent2: ERROR_STRUCT;
    bBusy2: BOOL;
END_VAR

```

Implementation

```
fbIH_MBReadDiscreteInputs2(  
  Enable:= bEnable2,  
  SendRequest:= FALSE,  
  EnablePoll:= TRUE,  
  SlaveAddress:= BYTE#5,  
  StartAddress:= WORD#500,  
  ValueAdr:= ADR(arDiscreteInputs2),  
  NoOfInputBits:= UINT#5,  
  SizeOfValue:= SIZEOF(arDiscreteInputs2),  
  PollInterval:= T#3S,  
  TimeOut:= T#2S,  
  MBData:= stMBData,  
  Done=> bDone2,  
  Active=> bActive2,  
  Error=> bError2,  
  ErrorID=> enErrorID2,  
  ErrorIdent=> stErrorIdent2,  
  Busy=> bBusy2);
```

Expression	Type	Value
arDiscreteInputs2	ARRAY [0..10] OF BOOL	
arDiscreteInputs2[0]	BOOL	TRUE → Data of Address (500)
arDiscreteInputs2[1]	BOOL	FALSE → Data of Address (501)
arDiscreteInputs2[2]	BOOL	TRUE →
arDiscreteInputs2[3]	BOOL	FALSE →
arDiscreteInputs2[4]	BOOL	TRUE → Data of Address (504)

Abb. 20-18: arDiscreteInputs2 für Programm 2

20.9 IH_MBReadExceptionStatus

Funktionscode FC 7

Kurzbeschreibung Der Funktionsbaustein **IH_MBReadExceptionStatus** dient zum Lesen des Status der Exception-Ausgangsbits eines Slave-Geräts mit RTU-Modus. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein **IH_MBRRTUMaster**, [Seite863](#) aufgerufen werden.

Zielsystem	Bibliothek /Ordner
MLC	RIH_Modbus / FB
MTX	RIH_Modbus / FB
MLD (IL 2G)	RIH_Modbus / FB

Tab. 20-23: Referenztabelle

Interfacebeschreibung

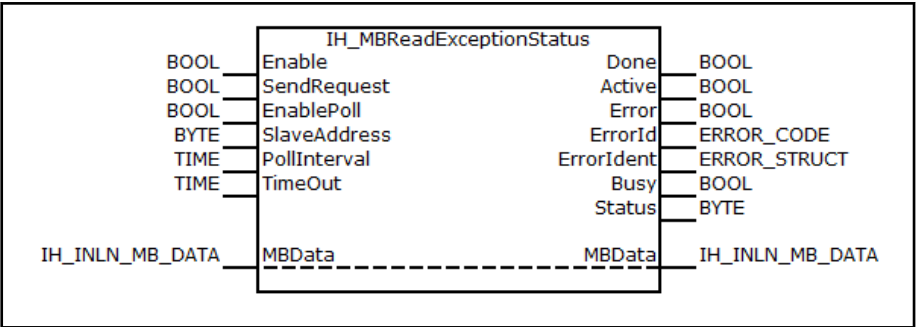


Abb. 20-19: Funktionsbaustein IH_MBReadExceptionStatus

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Funktionsbaustein aktiv, falls TRUE
	SendRequest	BOOL	Bei steigender Flanke wird eine Anfrage gesendet
	EnablePoll	BOOL	Polling aktiviert. SendRequest muss nicht in jedem Intervall ausgelöst werden. Der Baustein sendet Anfragen automatisch
	SlaveAddress	BYTE	Slave-Adresse
	PollInterval	TIME	Polling-Intervall. Anfrage wird kontinuierlich in Intervallen gesendet
	TimeOut	TIME	Timeout-Intervall für Antwort von Slaves
VAR_OUTPUT	Done	BOOL	Berechnung beendet und beginnt im nächsten Zyklus neu
	Active	BOOL	Funktionsbaustein ist aktiv
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Modbus, Seite 898
	Busy	BOOL	Wahr, wenn der Baustein mit dem Master kommuniziert
	Status	BYTE	Enthält acht Ausnahmezustände in den Ausgangsbits des entfernten Geräts
VAR_IN_OUT	MBData	IH_INLN_MB_DATA	Struktur handhabt Daten zwischen Master und anderen Modbus-Funktionsbausteinen IH_INLN_MB_DATA, Seite 894

Tab. 20-24: Schnittstellenvariablen von Funktionsbaustein *IH_MBReadExceptionStatus*

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
SendRequest	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
EnablePoll	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
SlaveAddress	BYTE	0	247	0	steigende Flanke an Enable
PollInterval	TIME	T#1s	-----	T#1s	Kontinuierlich bei Enable
TimeOut	TIME	T#1s	-----	T#1s	steigende Flanke an Enable

Tab. 20-25: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von *IH_MBReadExceptionStatus*

Funktionsbeschreibung

Der Funktionsbaustein **IH_MBReadExceptionStatus** (FC 7) dient zum Lesen des Status der Exception-Ausgangsbits eines Slave-Geräts mit RTU-Modus. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein aufgerufen werden. "MBData" muss mit dem Master-Funktionsbaustein im Programm verknüpft sein, er dient zur Kommunikation und zum Datenaustausch.

Er liest den Inhalt von acht Exception-Status-Coils im Slave-Regler. Einige Coils haben vordefinierte Zuweisungen in den verschiedenen Reglern. Andere Coils können vom Nutzer so programmiert werden, dass sie Informationen zum Reglerstatus enthalten, wie "machine ON/OFF" (Maschine ein/aus), "heads retracted" (Köpfe zurückgezogen), "safeties satisfied" (Sicherheit erfüllt), "error conditions exist" (Fehlerbedingungen vorhanden) oder andere benutzerdefinierte Flags.

Zuweisungen zu Exception-Coils sind vordefiniert und hängen vom verwendeten Reglermodell ab.

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Enable" auf TRUE gesetzt wird. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Falls während des Betriebs ein Fehler auftritt, wird das Bit „Error“ auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Enable" quittiert werden.

Beispiel eines Funktionsbausteins:

Der Funktionsbaustein IH_MBReadExceptionStatus muss zur Kommunikation mit dem Modbus-Slave zusammen mit dem Funktionsbaustein IH_MBRTUMaster instanziiert werden. Die Eingangs-/Ausgangsvariable stMBData tauscht die Daten zwischen den Funktionsbausteinen IH_MBRTUMaster und IH_MBReadExceptionStatus aus. Deshalb muss diese Variable in beiden Funktionsbausteinen MBData zugewiesen werden.

Im Beispiel wird der Funktionsbaustein IH_MBReadExceptionStatus zum Auslesen des Exception-Status des Slave-Geräts verwendet.

Statusbyte mit SendRequest-Bedingung lesen

IH_MBReadExceptionStatus wird zum Lesen des Statusbyte von Slave-Adresse 2 verwendet, dabei löst "bSendRequest" als BOOL-Variable das Lesen des Exception-Status aus.

Variablendeklaration

```
VAR
    fbIH_MBReadExceptionStatus: IH_MBReadExceptionStatus;
    bEnable: BOOL;
    bSendRequest: BOOL;
    stMBData: IH_INLN_MB_DATA;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
    bBusy: BOOL;
    byStatus: BYTE;
END_VAR
```

Implementation

```
fbIH_MBReadExceptionStatus(
    Enable:= bEnable,
    SendRequest:= bSendRequest,
    EnablePoll:= FALSE,
    SlaveAddress:= BYTE#2,
    PollInterval:= T#2S,
    Timeout:= T#2S,
```

```
MBData:= stMBData,  
Done=> bDone,  
Active=> bActive,  
Error=> bError,  
ErrorID=> enErrorID,  
ErrorIdent=> stErrorIdent,  
Busy=> bBusy,  
Status=> byStatus);
```

20.10 IH_MBReadFIFOQue

Funktionscode FC 24

Kurzbeschreibung Der Funktionsbaustein **IH_MBReadFIFOQue** liest die First-In-First-Out-Registerschlange (FIFO) eines Slave-Moduls Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein **IH_MBRТУMaster**, [Seite863](#) aufgerufen werden.

Zielsystem	Bibliothek /Ordner
MLC	RIH_Modbus / FB
MTX	RIH_Modbus / FB
MLD (IL 2G)	RIH_Modbus / FB

Tab. 20-26: Referenztabelle

Interfacebeschreibung

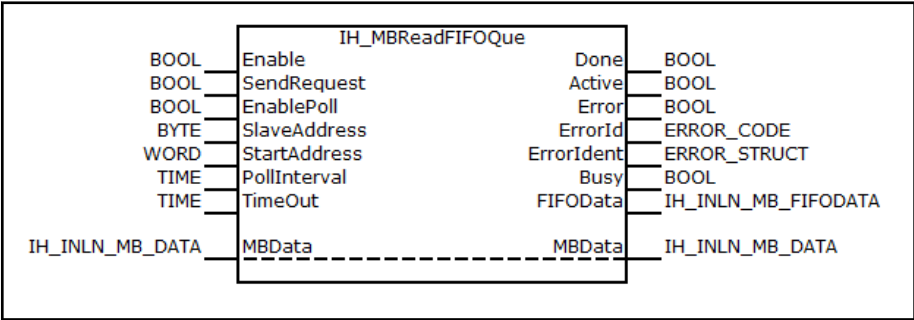


Abb. 20-20: Funktionsbaustein IH_MBReadFIFOQue

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Funktionsbaustein aktiv, falls TRUE
	SendRequest	BOOL	Bei steigender Flanke wird eine Anfrage gesendet
	EnablePoll	BOOL	Polling aktiviert. SendRequest muss nicht in jedem Intervall ausgelöst werden. Der Baustein sendet Anfragen automatisch
	SlaveAddress	BYTE	Slave-Adresse
	StartAddress	WORD	Startadresse der Ausgangsbits
	PollInterval	TIME	Polling-Intervall. Anfrage wird kontinuierlich in Intervallen gesendet
	TimeOut	TIME	Timeout-Intervall für Antwort von Slaves

E/A-Typ	Name	Datentyp	Beschreibung
VAR_OUTPUT	Done	BOOL	Berechnung beendet und beginnt im nächsten Zyklus neu
	Active	BOOL	Funktionsbaustein ist aktiv
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Modbus, Seite 898
	Busy	BOOL	Wahr, wenn der Baustein mit dem Master kommuniziert
	FIFOData	IH_INLN_MB_FIFODATA	Die Struktur beinhaltet zusätzliche Informationen zu Byte-Anzahl, FIFO-Anzahl und FIFO-Registerwerten IH_INLN_MB_FIFODATA, Seite 895
VAR_IN_OUT	MBData	IH_INLN_MB_DATA	Struktur handhabt Daten zwischen Master und anderen Modbus-Funktionsbausteinen IH_INLN_MB_DATA, Seite 894

Tab. 20-27: Schnittstellenvariablen von Funktionsbaustein IH_MBReadFIFOQue

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
SendRequest	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
EnablePoll	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
SlaveAddress	BYTE	0	247	0	steigende Flanke an Enable
StartAddress	WORD	0	65535	0	steigende Flanke an Enable
PollInterval	TIME	T#1s	-----	T#1s	Kontinuierlich bei Enable
TimeOut	TIME	T#1s	-----	T#1s	steigende Flanke an Enable

Tab. 20-28: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_MBReadFIFOQue

Funktionsbeschreibung

Der Funktionsbaustein **IH_MBReadFIFOQue** (FC 24) liest die First-In-First-Out-Registerschlange (FIFO) eines Slave-Moduls. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein aufgerufen werden. "MBData" muss mit dem Master-Funktionsbaustein im Programm verknüpft sein, er dient zur Kommunikation und zum Datenaustausch.

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Enable" auf TRUE gesetzt wird. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Falls während des Betriebs ein Fehler auftritt, wird das Bit

„Error“ auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Enable" quittiert werden.

Beispiel eines Funktionsbausteins

Der Funktionsbaustein **IH_MBReadFIFOQue** muss zur Kommunikation mit dem Modbus-Slave zusammen mit **IH_MBRUMaster** instanziiert werden. Für MBData der Funktionsbausteine **IH_MBReadFIFOQue** und **IH_MBRUMaster** ist für den Datenaustausch mit dem Modbus-Slave eine gemeinsame Variable zu verwenden.

In diesem Beispiel liest der Funktionsbaustein **IH_MBReadFIFOQue** die FIFO-Schlange des Slave mit Adresse 1 und StartAddress 2550.

Die beiden Funktionsbausteine **IH_MBRUMaster** und **IH_MBReadFIFOQue** werden mit steigender Flanke am Eingang bSendRequest aktiviert. Der Funktionsbaustein liest die FIFO-Schlange des Slave mit Adresse 1 und das Ergebnis wird in stFIFOData gespeichert.

Wenn die Variable bEnablePoll auf TRUE und Pollinterval auf mehr als 1 Sekunde gestellt werden, werden die Daten in regelmäßigen Intervallen aus dem Register gelesen.

Nach jedem erfolgreichen Abschluss der Leseoperation wird das Done-Bit für einen SPS-Zyklus auf TRUE gesetzt.

Variablendeklaration

```
VAR
    fbIH_MBReadFIFOQue: IH_MBReadFIFOQue;
    bEnable: BOOL;
    bSendRequest: BOOL;
    bEnablePoll: BOOL;
    stMBData: IH_INLN_MB_DATA;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
    bBusy: BOOL;
    stFIFOData: IH_INLN_MB_FIFODATA;
END_VAR
```

Implementation

```
fbIH_MBReadFIFOQue (
    Enable:= bEnable,
    SendRequest:= bSendRequest,
    EnablePoll:= bEnablePoll,
    SlaveAddress:= BYTE#1,
    StartAddress:= WORD#2550,
    PollInterval:= T#1S,
    Timeout:= T#1S,
    MBData:= stMBData,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent,
    Busy=> bBusy,
    FIFOData=> stFIFOData);
```

Antwort

Expression	Type	Value
stFIFOData	IH_INLN_MB_FIFODATA	
ByteCount	UINT	6
FIFOCount	UINT	2
FIFORegValues	ARRAY [1..31] OF WORD	
FIFORegValues[1]	WORD	100
FIFORegValues[2]	WORD	200

Abb. 20-21: stFIFOData

20.11 IH_MBReadFileRecord

Funktionscode FC 20

Kurzbeschreibung Der Funktionsbaustein **IH_MBReadFileRecord** dient zum Lesen von Datei-protokollen im RTU-Modus. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein **IH_MBRTUMaster**, [Seite 863](#) aufgerufen werden.

Zielsystem	Bibliothek /Ordner
MLC	RIH_Modbus / FB
MTX	RIH_Modbus / FB
MLD (IL 2G)	RIH_Modbus / FB

Tab. 20-29: Referenztabelle

Interfacebeschreibung

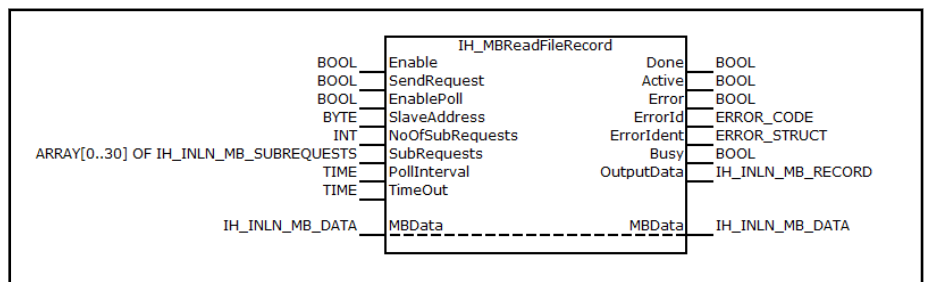


Abb. 20-22: Funktionsbaustein IH_MBReadFileRecord

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Funktionsbaustein aktiv, falls TRUE
	SendRequest	BOOL	Bei steigender Flanke wird eine Anfrage gesendet
	EnablePoll	BOOL	Polling aktiviert. SendRequest muss nicht in jedem Intervall ausgelöst werden. Der Baustein sendet Anfragen automatisch
	SlaveAddress	BYTE	Slave-Adresse
	NoOfSubRequests	INT	Anzahl Unteranfragen
	SubRequests	ARRAY[1..30] OF IH_INLN_MB_SUBREQUESTS	Liste der Unteranfragen. Siehe hierzu: Strukturen und Aufzählungen
	PollInterval	TIME	Polling-Intervall. Anfrage wird kontinuierlich in Intervallen gesendet
	TimeOut	TIME	Timeout-Intervall für Antwort von Slaves

E/A-Typ	Name	Datentyp	Beschreibung
VAR_OUTPUT	Done	BOOL	Berechnung beendet und beginnt im nächsten Zyklus neu
	Active	BOOL	Funktionsbaustein ist aktiv
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Modbus, Seite 898
	Busy	BOOL	Wahr, wenn der Baustein mit dem Master kommuniziert
	OutputData	IH_INLN_MB_RECORD	Enthält die angefragte Datenlänge und unterangefragte Daten Siehe hierzu: Strukturen und Aufzählungen IH_INLN_MB_RECORD, Seite 897
VAR_IN_OUT	MBData	IH_INLN_MB_DATA	Struktur handhabt Daten zwischen Master und anderen Modbus-Funktionsbausteinen IH_INLN_MB_DATA, Seite 894

Tab. 20-30: Schnittstellenvariablen von Funktionsbaustein IH_MBReadFileRecord

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Value	Max.-Value	Defaultwert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
SendRequest	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
EnablePoll	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
SlaveAddress	BYTE	0	247	0	steigende Flanke an Enable
NoOfSubRequests	INT	1	30	1	Kontinuierlich bei Enable
PollInterval	TIME	T#1s	-----	T#1s	Kontinuierlich bei Enable
TimeOut	TIME	T#1s	-----	T#1s	steigende Flanke an Enable

Tab. 20-31: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_MBReadFileRecord

Funktionsbeschreibung

Der Funktionsbaustein **IH_MBReadFileRecord** (FC 20) dient zum Lesen von Dateiprotokollen im RTU-Modus. Die angefragten Datenlängen in Struktur (IH_INLN_MB_SUBREQUESTS) werden in Bytes angegeben, Protokollängen werden als Register angegeben. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein aufgerufen werden. "MBData" muss mit dem Master-Funktionsbaustein im Programm verknüpft sein, er dient zur Kommunikation und zum Datenaustausch.

Die Funktion kann mehrere Gruppen von Referenzen auslesen. Es kann sich dabei um getrennte (nicht benachbarte) Gruppen handeln,

aber die Referenzen in den einzelnen Gruppen müssen fortlaufend sein.

- Jede Gruppe wird in einer separaten Feld für Unteranfragen definiert, das 7 Bytes enthält:
- Referenztyp: 1 Byte (muss als 6 angegeben werden). Info: In diesem FB als Standard betrachtet.
- Dateinummer: (Wort) muss (0x0001 bis 0xFFFF) sein.
- Anfangsprotokollnummer in der Datei. (Wort) muss (0x0000 bis 0x270F) sein.
- Länge des zu lesenden Protokolls (Wort).

Die Anzahl der zu lesenden Register sowie aller anderen Felder in der erwarteten Antwort,

darf die zulässige Länge der MODBUS PDU nicht überschreiten: 253 Bytes.

Demnach muss die Gesamtlänge des Protokolls aller Unteranfragen kleiner als (123 – NoOfSubRequests) sein.

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Enable" auf TRUE gesetzt wird. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Falls während des Betriebs ein Fehler auftritt, wird das Bit „Error“ auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Enable" quittiert werden.

Beispiel eines Funktionsbausteins

Der Funktionsbaustein IH_MBReadFileRecord muss zur Kommunikation mit dem Modbus-Slave zusammen mit dem Funktionsbaustein IH_MBRTUMaster instanziiert werden. Die Eingangs-/Ausgangsvariable stMBData tauscht die Daten zwischen den Funktionsbausteinen IH_MBRTUMaster und IH_MBReadFileRecord aus. Deshalb muss diese Variable in beiden Funktionsbausteinen MBData zugewiesen werden.

Im Beispiel wird der Funktionsbaustein IH_MBReadFileRecord zum Auslesen der Dateiprotokolle des Slave-Geräts verwendet.

2 Dateiprotokolle mit SendRequest-Bedingung lesen

IH_MBReadFileRecord dient zum Lesen von 2 Protokollen mit der Länge 4 und 3 von Datei 1 und 2 von Slave-Adresse 2 mit "bSendRequest" als BOOL-Variable zum Auslösen des Lesens.

Variablendeklaration

```
VAR
    fbIH_MBReadFileRecord: IH_MBReadFileRecord;
    bEnable: BOOL;
    bSendRequest: BOOL;
    stSubRequests: ARRAY [1..30] OF IH_INLN_MB_SUBREQUESTS;
    stMBData: IH_INLN_MB_DATA;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
    bBusy: BOOL;
    stOutputData: IH_INLN_MB_RECORD;
END_VAR
```

Implementation

```
stSubRequests[1].FileNumber := WORD#1;
stSubRequests[1].RecordNumber := WORD#0;
stSubRequests[1].RecordLength := WORD#4;
stSubRequests[2].FileNumber := WORD#2;
stSubRequests[2].RecordNumber := WORD#5;
stSubRequests[2].RecordLength := WORD#3;
```

```
fbIH_MBReadFileRecord(
  Enable:= bEnable,
  SendRequest:= bSendRequest,
  EnablePoll:= FALSE,
  SlaveAddress:= BYTE#2,
  NoOfSubRequests:= INT#2,
  SubRequests:= stSubRequests,
  PollInterval:= T#2S,
  Timeout:= T#2S,
  MBData:= stMBData,
  Done=> bDone,
  Active=> bActive,
  Error=> bError,
  ErrorID=> enErrorID,
  ErrorIdent=> stErrorIdent,
  Busy=> bBusy,
  OutputData=> stOutputData);
```

Expression	Type	Value
stOutputData	IH_INLN_MB_RECORD	
DataLength	UINT	9 → Valid data in SubRequestData
SubRequestData	ARRAY [0..124] OF WORD	
SubRequestData[0]	WORD	4 → SubRequest-1 data length
SubRequestData[1]	WORD	100
SubRequestData[2]	WORD	200
SubRequestData[3]	WORD	300
SubRequestData[4]	WORD	400
SubRequestData[5]	WORD	3 → SubRequest-2 data length
SubRequestData[6]	WORD	6000
SubRequestData[7]	WORD	7000
SubRequestData[8]	WORD	8000 → SubRequest-2 Data

Abb. 20-23: stOutputData für Programm 1

1 Dateiprotokoll mit PollInterval-Bedingung lesen

IH_MBReadFileRecord dient zum Lesen von 1 Protokoll mit der Länge 6 von Slave-Adresse 5 mit PollInterval (3 Sekunden) zum Auslösen des Lesens.

Variablendeklaration

```
VAR
  fbIH_MBReadFileRecord2: IH_MBReadFileRecord;
  bEnable2: BOOL;
  stSubRequests2: ARRAY [1..30] OF IH_INLN_MB_SUBREQUESTS;
  stMBData: IH_INLN_MB_DATA;
  bDone2: BOOL;
  bActive2: BOOL;
  bError2: BOOL;
  enErrorID2: ERROR_CODE;
  stErrorIdent2: ERROR_STRUCT;
  bBusy2: BOOL;
  stOutputdata2: IH_INLN_MB_RECORD;
END_VAR
```

Implementation

```
stSubRequests2[1].FileNumber := WORD#1;
stSubRequests2[1].RecordNumber := WORD#0;
stSubRequests2[1].RecordLength := WORD#6;
```

```
fbIH_MBReadFileRecord2(
  Enable:= bEnable2,
  SendRequest:= FALSE,
  EnablePoll:= TRUE,
```

```

SlaveAddress:= BYTE#5,
NoOfSubRequests:= INT#1,
SubRequests:= stSubRequests2,
PollInterval:= T#3S,
TimeOut:= T#2S,
MBData:= stMBData,
Done=> bDone2,
Active=> bActive2,
Error=> bError2,
ErrorID=> enErrorID2,
ErrorIdent=> stErrorIdent2,
Busy=> bBusy2,
OutputData=> stOutputdata2);

```

Expression	Type	Value
stOutputdata2	IH_INLN_MB_RECORD	
DataLength	UINT	7 → Valid Data in SubRequestData
SubRequestData	ARRAY [0..124] OF WORD	
SubRequestData[0]	WORD	6 → SubRequest1 - Data length
SubRequestData[1]	WORD	100
SubRequestData[2]	WORD	200
SubRequestData[3]	WORD	300
SubRequestData[4]	WORD	400
SubRequestData[5]	WORD	500
SubRequestData[6]	WORD	600

Abb. 20-24: stOutputData2 für Programm 2

20.12 IH_MBReadHoldingRegisters

Funktionscode FC 3

Kurzbeschreibung Der Funktionsbaustein **IH_MBReadHoldingRegisters** dient zum Lesen eines Holding-Registers des Modbus-Slave-Moduls. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein **IH_MBRTUMaster**, [Seite 863](#) aufgerufen werden.

Zielsystem	Bibliothek /Ordner
MLC	RIH_Modbus / FB
MTX	RIH_Modbus / FB
MLD (IL 2G)	RIH_Modbus / FB

Tab. 20-32: Referenztafel

Interfacebeschreibung

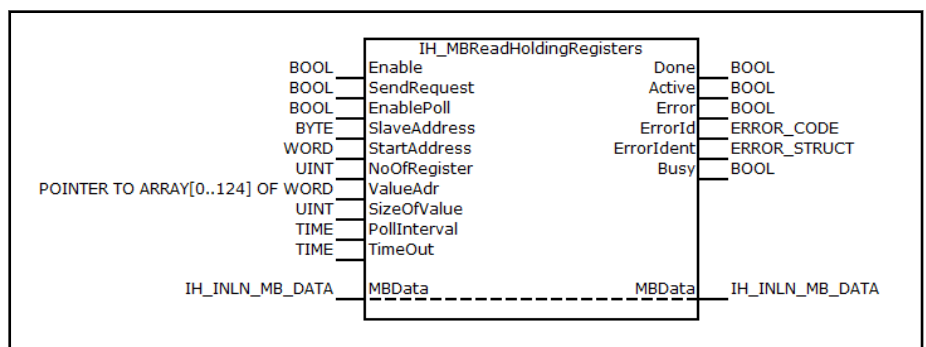


Abb. 20-25: Funktionsbaustein IH_MBReadHoldingRegisters

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Funktionsbaustein aktiv, falls TRUE
	SendRequest	BOOL	Bei steigender Flanke wird eine Anfrage gesendet
	EnablePoll	BOOL	Polling aktiviert. SendRequest muss nicht in jedem Intervall ausgelöst werden. Der Baustein sendet Anfragen automatisch
	SlaveAddress	BYTE	Slave-Adresse
	StartAddress	WORD	Startadresse des Holding-Registers
	NoOfRegisters	UINT	Anzahl Register
	ValueAdr	POINTER TO ARRAY[0..124] OF WORD	Adresse des Arrays zum Speichern der Holding-Register. Operator ADR () verwenden
	SizeOfValue	UINT	Größe des Value-Arrays. Operator SIZEOF () verwenden
	PollInterval	TIME	Polling-Intervall. Anfrage wird kontinuierlich in Intervallen gesendet
	TimeOut	TIME	Timeout-Intervall für Antwort von Slaves
VAR_OUTPUT	Done	BOOL	Berechnung beendet und beginnt im nächsten Zyklus neu
	Active	BOOL	Funktionsbaustein ist aktiv
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Modbus, Seite 898
	Busy	BOOL	Wahr, wenn der Baustein mit dem Master kommuniziert
VAR_OUTPUT	MBData	IH_INLN_MB_DATA	Struktur handhabt Daten zwischen Master und anderen Modbus-Funktionsbausteinen der Funktionscodes IH_INLN_MB_DATA, Seite 894

Tab. 20-33: Schnittstellenvariablen von Funktionsbaustein *IH_MBReadHoldingRegisters*

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
SendRequest	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
EnablePoll	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
SlaveAddress	BYTE	0	247	0	steigende Flanke an Enable

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
StartAddress	WORD	0	65535	0	steigende Flanke an Enable
NoOfRegisters	UINT	1	125	1	steigende Flanke an Enable
SizeOfValue	UINT	1	-----	-----	steigende Flanke an Enable
PollInterval	TIME	T#1s	-----	T#1s	Kontinuierlich bei Enable
TimeOut	TIME	T#1s	-----	T#1s	steigende Flanke an Enable

Tab. 20-34: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_MBReadHoldingRegisters

Funktionsbeschreibung

Der Funktionsbaustein **IH_MBReadHoldingRegisters** (FC 3) liest die Holding-Register eines in SlaveAddress festgelegten Slave-Moduls. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein aufgerufen werden. "MBData" muss mit dem Master-Funktionsbaustein verknüpft sein, um mit dem Master zu kommunizieren und Daten auszutauschen.

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Enable" auf TRUE gesetzt wird. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Falls während des Betriebs ein Fehler auftritt, wird das Bit „Error“ auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Enable" quittiert werden.

Beispiel eines Funktionsbausteins

Der Funktionsbaustein IH_MBReadHoldingRegisters muss zur Kommunikation mit dem Modbus-Slave zusammen mit [IH_MBRTUMaster](#), Seite 863, instanziiert werden. Für MBData der Funktionsbausteine IH_MBReadHoldingRegisters und IH_MBRTUMaster ist für den Datenaustausch mit dem Modbus-Slave eine gemeinsame Variable zu verwenden.

Beispiel 1

In diesem Beispiel liest der Funktionsbaustein IH_MBReadHoldingRegisters das Holding-Register von Slave-Adresse 1 mit StartAddress 5 und NoOfRegisters 5.

Die Funktionsbausteine IH_MBRTUMaster und IH_MBReadHoldingRegisters werden mit einer steigenden Flanke am Eingang bSendRequest aktiviert. Der Funktionsbaustein liest die Daten von Slave-Adresse 1 und nach erfolgreicher Beendigung der Leseoperation wird das Done-Bit für einen SPS-Zyklus auf TRUE gesetzt.

Variablendeklaration

```
VAR
    fbIH_MBReadHoldingRegisters: IH_MBReadHoldingRegisters;
    bEnable: BOOL;
    bSendRequest: BOOL;
    bEnablePoll: BOOL;
    arHoldingRegister: ARRAY [0..4] OF WORD;
    stMBData: IH_INLN_MB_DATA;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
```

```

    stErrorIdent: ERROR_STRUCT;
    bBusy: BOOL;
END_VAR

```

Implementation

```

fbIH_MBReadHoldingRegisters(
    Enable:= bEnable,
    SendRequest:= bSendRequest,
    EnablePoll:= bEnablePoll,
    SlaveAddress:= BYTE#1,
    StartAddress:= WORD#5,
    NoOfRegisters:= UINT#5,
    ValueAdr:= ADR(arHoldingRegister),
    SizeOfValue:= SIZEOF(arHoldingRegister),
    PollInterval:= T#1S,
    Timeout:= T#1S,
    MBData:= stMBData,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent,
    Busy=> bBusy);

```

Antwort

Die gelesenen Daten stehen im Array arHoldingRegister bereit.

Expression	Type	Value
arHoldingRegister	ARRAY [0..4] OF WO...	
arHoldingRegister[0]	WORD	100 Data value of register 5
arHoldingRegister[1]	WORD	200
arHoldingRegister[2]	WORD	300
arHoldingRegister[3]	WORD	400
arHoldingRegister[4]	WORD	500 Data value of register 9

Abb. 20-26: Array arHoldingRegister

Beispiel 2

In diesem Beispiel liest der Funktionsbaustein IH_MBReadHoldingRegisters das Holding-Register von Slave-Adresse 2 mit StartAddress 100 und NoOf-Registers 10 mit einem Polling-Intervall von 1 Sekunde.

Die Funktionsbausteine IH_MBRTUMaster und IH_MBReadHoldingRegisters werden aktiviert und EnablePoll mit einem PollInterval von 1 Sekunde auf TRUE gesetzt. Daraufhin liest der Funktionsbaustein die Daten aus Slave-Adresse 2 in regelmäßigen Intervallen von 1 Sekunde. Nach jedem erfolgreichen Abschluss der Leseoperation wird das Done-Bit für einen SPS-Zyklus auf TRUE gesetzt.

Variablendeklaration

```

VAR
    fbIH_MBReadHoldingRegisters_2: IH_MBReadHoldingRegisters;
    bEnable2: BOOL;
    bSendRequest2: BOOL;
    arHoldingRegister2: ARRAY [0..9] OF WORD;
    bDone2: BOOL;
    bActive2: BOOL;
    bError2: BOOL;
    enErrorID2: ERROR_CODE;
    stErrorIdent2: ERROR_STRUCT;
    bBusy2: BOOL;
    stMBData: IH_INLN_MB_DATA;
END_VAR

```

Implementation

```
fbIH_MBReadHoldingRegisters_2(  
  Enable:= bEnable2,  
  SendRequest:= bSendRequest2,  
  EnablePoll:= TRUE,  
  SlaveAddress:= BYTE#2,  
  StartAddress:= WORD#100,  
  NoOfRegisters:= UINT#10,  
  ValueAdr:= ADR(arHoldingRegister2),  
  SizeOfValue:= SIZEOF(arHoldingRegister2),  
  PollInterval:= T#1S,  
  Timeout:= T#1S,  
  MBData:= stMBData,  
  Done=> bDone2,  
  Active=> bActive2,  
  Error=> bError2,  
  ErrorID=> enErrorID2,  
  ErrorIdent=> stErrorIdent2,  
  Busy=> bBusy2);
```

Antwort

Die gelesenen Daten stehen im Array arHoldingRegister2 bereit.

Expression	Type	Value
arHoldingRegister2	ARRAY [0..9] OF WORD	
arHoldingRegister2[0]	WORD	600 Data value of register 100
arHoldingRegister2[1]	WORD	700 Data value of register 101
arHoldingRegister2[2]	WORD	800
arHoldingRegister2[3]	WORD	900
arHoldingRegister2[4]	WORD	1000
arHoldingRegister2[5]	WORD	1100
arHoldingRegister2[6]	WORD	1200
arHoldingRegister2[7]	WORD	1300
arHoldingRegister2[8]	WORD	1400 Data value of register 108
arHoldingRegister2[9]	WORD	1500 Data value of register 109

Abb. 20-27: Array arHoldingRegister2

20.13 IH_MBReadInputRegisters

Funktionscode FC 4

Kurzbeschreibung Der Funktionsbaustein **IH_MBReadInputRegisters** dient zum Lesen von Eingangsregistern im RTU-Modus. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein **IH_MBRTUMaster**, Seite 863 aufgerufen werden.

Zielsystem	Bibliothek /Ordner
MLC	RIH_Modbus / FB
MTX	RIH_Modbus / FB
MLD (IL 2G)	RIH_Modbus / FB

Tab. 20-35: Referenztablelle

Interfacebeschreibung

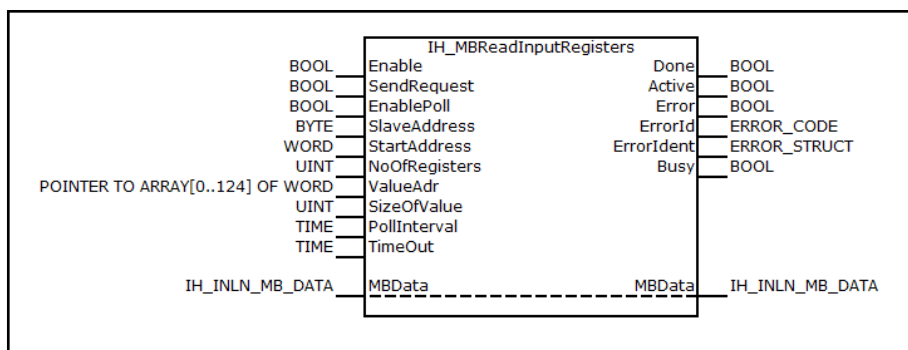


Abb. 20-28: Funktionsbaustein IH_MBReadInputRegisters

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Funktionsbaustein aktiv, falls TRUE
	SendRequest	BOOL	Bei steigender Flanke wird eine Anfrage gesendet
	EnablePoll	BOOL	Polling aktiviert. SendRequest muss nicht in jedem Intervall ausgelöst werden. Der Baustein sendet Anfragen automatisch
	SlaveAddress	BYTE	Slave-Adresse
	StartAddress	WORD	Startadresse der Register
	NoOfRegisters	UINT	Anzahl zu lesender Register
	ValueAdr	POINTER TO ARRAY[0..124] OF WORD	Adresse des Arrays zum Speichern der Eingangsregister Operator ADR () verwenden
	SizeOfValue	UINT	Größe des Value-Arrays. Operator SIZEOF () verwenden
	PollInterval	TIME	Polling-Intervall. Anfrage wird kontinuierlich in Intervallen gesendet
	TimeOut	TIME	Timeout-Intervall für Antwort von Slaves
VAR_OUTPUT	Done	BOOL	Berechnung beendet und beginnt im nächsten Zyklus neu
	Active	BOOL	Funktionsbaustein ist aktiv
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Modbus, Seite 898
	Busy	BOOL	Wahr, wenn der Baustein mit dem Master kommuniziert
VAR_IN_OUT	MBData	IH_INLN_MB_DATA	Struktur handhabt Daten zwischen Master und anderen Modbus-Funktionsbausteinen IH_INLN_MB_DATA, Seite 894

Tab. 20-36: Schnittstellenvariablen von Funktionsbaustein IH_MBReadInputRegisters

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
SendRequest	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
EnablePoll	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
SlaveAddress	BYTE	0	247	0	steigende Flanke an Enable
StartAddress	WORD	0	16#FFFF	0	steigende Flanke an Enable
NoOfRegisters	UINT	1	125	1	steigende Flanke an Enable
SizeOfValue	UINT	1	-----	-----	steigende Flanke an Enable
PollInterval	TIME	T#1s	-----	T#1s	Kontinuierlich bei Enable
TimeOut	TIME	T#1s	-----	T#1s	steigende Flanke an Enable

Tab. 20-37: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_MBReadInputRegisters

Funktionsbeschreibung

Der Funktionsbaustein **IH_MBReadInputRegisters** (FC 4) dient zum Lesen von Eingangsregistern im RTU-Modus. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein aufgerufen werden. "MBData" muss mit dem Master-Funktionsbaustein im Programm verknüpft sein, er dient zur Kommunikation und zum Datenaustausch.

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Enable" auf TRUE gesetzt wird. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Falls während des Betriebs ein Fehler auftritt, wird das Bit „Error“ auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Enable" quittiert werden.

Beispiel eines Funktionsbausteins

Der Funktionsbaustein IH_MBReadInputRegisters muss zur Kommunikation mit dem Modbus-Slave zusammen mit dem Funktionsbaustein IH_MBRTU-Master instanziiert werden. Die Eingangs-/Ausgangsvariable stMBData tauscht die Daten zwischen den Funktionsbausteinen IH_MBRTUMaster und IH_MBReadInputRegisters aus. Deshalb muss diese Variable in beiden Funktionsbausteinen MBData zugewiesen werden.

Im Beispiel wird der Funktionsbaustein IH_MBReadInputRegisters zum Auslesen von Eingangsregistern des Slave-Geräts verwendet.

Eingangsregister mit SendRequest-Bedingung lesen

IH_MBReadInputRegisters dient zum Lesen von 10 Eingangsregistern der Startadresse 0x1000 von Slave-Adresse 2 mit "bSendRequest" als BOOL-Variable zum Auslösen der Leseeingänge.

Variablendeklaration

```

VAR
    fbIH_MBReadInputRegisters: IH_MBReadInputRegisters;
    bEnable: BOOL;
    bSendrequest: BOOL;
    arInputRegisters: ARRAY [0..124] OF WORD;
    stMBData: IH_INLN_MB_DATA;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
    bBusy: BOOL;
END_VAR

```

Implementation

```

fbIH_MBReadInputRegisters (
    Enable:= bEnable,
    SendRequest:= bSendrequest,
    EnablePoll:= FALSE,
    SlaveAddress:= BYTE#2,
    StartAddress:= WORD#1000,
    ValueAdr:= ADR(arInputRegisters),
    NoOfRegisters:= UINT#10,
    SizeOfValue:= SIZEOF(arInputRegisters),
    PollInterval:= T#2S,
    Timeout:= T#2S,
    MBData:= stMBData,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent,
    Busy=> bBusy);

```

Expression	Type	Value
arInputRegisters	ARRAY [0..124] OF WORD	
arInputRegisters[0]	WORD	50 → Data value of Register(1000)
arInputRegisters[1]	WORD	100 → Data value of Register(1001)
arInputRegisters[2]	WORD	50 → Data value of Register(1002)
arInputRegisters[3]	WORD	100
arInputRegisters[4]	WORD	50
arInputRegisters[5]	WORD	100
arInputRegisters[6]	WORD	50
arInputRegisters[7]	WORD	100
arInputRegisters[8]	WORD	50
arInputRegisters[9]	WORD	100 → Data value of Register(1009)

Abb. 20-29: arInputRegisters für Programm 1

Eingangsregister mit PollInterval-Bedingung lesen

IH_MBReadInputRegisters dient zum Lesen von 5 Eingangsregistern der Startadresse 0x0500 von Slave-Adresse 5 mit "PollInterval" (3 Sekunden) zum Auslösen der Leseeingänge.

Variablendeklaration

```

VAR
    fbIH_MBReadInputRegisters2: IH_MBReadInputRegisters;
    bEnable2: BOOL;
    arInputRegisters2: ARRAY [0..7] OF WORD;
    stMBData: IH_INLN_MB_DATA;
    bDone2: BOOL;
    bActive2: BOOL;
    bError2: BOOL;
    enErrorID2: ERROR_CODE;
    stErrorIdent2: ERROR_STRUCT;
    bBusy2: BOOL;
END_VAR

```

Implementation

```

fbIH_MBReadInputRegisters2(
    Enable:= bEnable2,
    SendRequest:= FALSE,
    EnablePoll:= TRUE,
    SlaveAddress:= BYTE#5,
    StartAddress:= WORD#500,
    ValueAdr:= ADR(arInputRegisters2),
    NoOfRegisters:= UINT#5,
    SizeOfValue:= SIZEOF(arInputRegisters2),
    PollInterval:= T#3S,
    Timeout:= T#2S,
    MBData:= stMBData,
    Done=> bDone2,
    Active=> bActive2,
    Error=> bError2,
    ErrorID=> enErrorID2,
    ErrorIdent=> stErrorIdent2,
    Busy=> bBusy2);

```

Expression	Type	Value
arInputRegisters2	ARRAY [0..7] OF WORD	
arInputRegisters2[0]	WORD	50 → Data value of Register(500)
arInputRegisters2[1]	WORD	100 → Data value of Register(501)
arInputRegisters2[2]	WORD	50
arInputRegisters2[3]	WORD	100
arInputRegisters2[4]	WORD	50 → Data value of Register(504)

Abb. 20-30: arInputRegisters2 für Programm 2

20.14 IH_MBReadWriteMultipleRegisters

Funktionscode FC 23

Kurzbeschreibung Der Funktionsbaustein **IH_MBReadWriteMultipleRegisters** führt in einer einzigen Modbus-Transaktion eine Lese- und eine Schreiboperation durch. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein **IH_MBRUMaster**, [Seite 863](#) aufgerufen werden.

Zielsystem	Bibliothek /Ordner
MLC	RIH_Modbus / FB
MTX	RIH_Modbus / FB
MLD (IL 2G)	RIH_Modbus / FB

Tab. 20-38: Referenztablelle

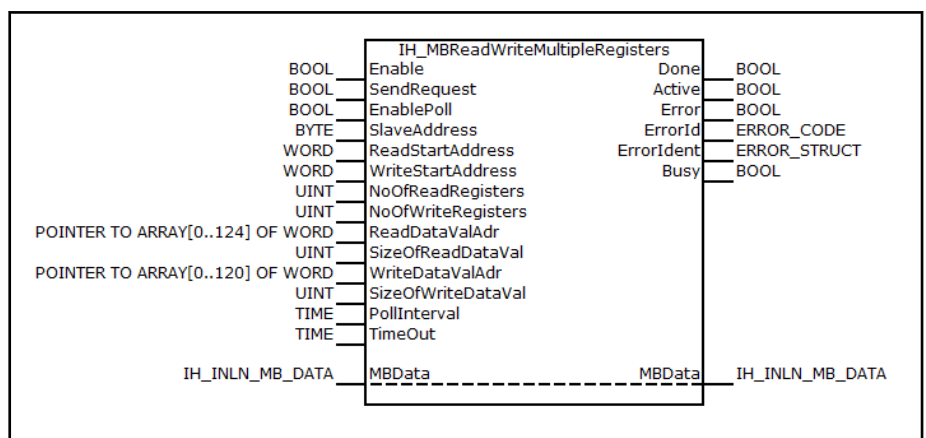
Interfacebeschreibung

Abb. 20-31: Funktionsbaustein IH_MBReadWriteMultipleRegisters

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Funktionsbaustein aktiv, falls TRUE
	SendRequest	BOOL	Bei steigender Flanke wird eine Anfrage gesendet
	EnablePoll	BOOL	Polling aktiviert. SendRequest muss nicht in jedem Intervall ausgelöst werden. Der Baustein sendet Anfragen automatisch
	SlaveAddress	BYTE	Slave-Adresse
	ReadStartAddress	WORD	Startadresse des Holding-Registers gilt für Leseoperation
	WriteStartAddress	WORD	Startadresse des Holding-Registers gilt für Schreiboperation
	NoOfReadRegisters	UINT	Anzahl zu lesender Register
	NoOfWriteRegisters	UINT	Anzahl zu schreibender Register
	ReadDataValAdr	POINTER TO ARRAY[0..124] WORD	Adresse des Arrays zum Speichern des Lesewerts der Register. Operator ADR () verwenden
	SizeOfReadDataVal	UINT	Größe des ReadDataVal-Arrays. Operator SIZEOF () verwenden
	WriteDataValAdr	POINTER TO ARRAY[0..120] WORD	Die Adresse des Arrays enthält den Schreibwert der Register. Operator ADR () verwenden
	SizeOfWriteDataVal	UINT	Größe des WriteDataVal-Arrays. Operator SIZEOF () verwenden
	PollInterval	TIME	Polling-Intervall. Anfrage wird kontinuierlich in Intervallen gesendet
	TimeOut	TIME	Timeout-Intervall für Antwort von Slaves
VAR_OUTPUT	Done	BOOL	Berechnung beendet und beginnt im nächsten Zyklus neu
	Active	BOOL	Funktionsbaustein ist aktiv
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Modbus, Seite 898
	Busy	BOOL	Wahr, wenn der Baustein mit dem Master kommuniziert
VAR_IN_OUT	MBData	IH_INLN_MB_DATA	Struktur handhabt Daten zwischen Master und anderen Modbus-Funktionsbausteinen IH_INLN_MB_DATA, Seite 894

Tab. 20-39: Schnittstellenvariablen von Funktionsbaustein IH_MBReadWriteMultipleRegisters

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Typ	Min.-Wert	Max.-Wert	Defaultwert	Übernahme
Enable	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
SendRequest	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
EnablePoll	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
SlaveAddress	BYTE	0	247	0	steigende Flanke an Enable
ReadStartAddress	WORD	0	65535	0	steigende Flanke an Enable
WriteStartAddress	WORD	0	65535	0	steigende Flanke an Enable
NoOfReadRegisters	UINT	1	125	1	steigende Flanke an Enable
NoOfWriteRegisters	UINT	1	121	1	steigende Flanke an Enable
SizeOfReadDataVal	UINT	1	-----	-----	steigende Flanke an Enable
SizeOfWriteDataVal	UINT	1	-----	-----	steigende Flanke an Enable
PollInterval	TIME	T#1s	-----	T#1s	Kontinuierlich bei Enable
TimeOut	TIME	T#1s	-----	T#1s	steigende Flanke an Enable

Tab. 20-40: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_MBReadWriteMultipleRegisters

Funktionsbeschreibung

Der Funktionsbaustein **IH_MBReadWriteMultipleRegisters** (FC 23) führt in einer einzigen Modbus-Transaktion eine Lese- und eine Schreiboperation durch. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein aufgerufen werden. "MBData" muss mit dem Master-Funktionsbaustein verknüpft sein, um mit dem Master zu kommunizieren und Daten auszutauschen.

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Enable" auf TRUE gesetzt wird. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Falls während des Betriebs ein Fehler auftritt, wird das Bit „Error“ auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Enable" quittiert werden.

Beispiel eines Funktionsbausteins:

Der Funktionsbaustein **IH_MBReadWriteMultipleRegisters** muss zur Kommunikation mit dem Modbus-Slave zusammen mit dem Funktionsbaustein **IH_MBRTUMaster** instanziiert werden. Die Eingangs-/Ausgangsvariable stMBData tauscht die Daten zwischen den Funktionsbausteinen **IH_MBRTUMaster** und **IH_MBReadWriteMultipleRegisters** aus. Deshalb muss diese Variable in beiden Funktionsbausteinen MBData zugewiesen werden.

Im Beispiel wird der Funktionsbaustein **IH_MBWriteMultipleCoils** zum Lesen und Schreiben von Registern auf das Slave-Gerät verwendet.

Lesen und Schreiben von Registern mit SendRequest-Bedingung

Im Beispiel werden 5 Register der Startadresse 0x4000 eines Modbus-Slave mit Adresse 1 geschrieben und gelesen. Eine steigende Flanke am "bSend-Request"-Eingang löst die Lese- und Schreiboperation aus.

Variablendeklaration

```
VAR
  fbIH_MBReadWriteMultipleRegisters: IH_MBReadWriteMultipleRegisters;
  bEnable: BOOL;
  bSendRequest: BOOL;
  arReadData: ARRAY [0..4] OF WORD;
  arWriteData: ARRAY [0..4] OF WORD;
  stMBData: IH_INLN_MB_DATA;
  bDone: BOOL;
  bActive: BOOL;
  bError: BOOL;
  enErrorID: ERROR_CODE;
  stErrorIdent: ERROR_STRUCT;
  bBusy: BOOL;
END_VAR
```

Implementation

```
fbIH_MBReadWriteMultipleRegisters(
  Enable:= bEnable,
  SendRequest:= bSendRequest,
  EnablePoll:= FALSE,
  SlaveAddress:= BYTE#1,
  ReadStartAddress:= WORD#4000,
  WriteStartAddress:= WORD#4000,
  NoOfReadRegisters:= UINT#5,
  NoOfWriteRegisters:= UINT#5,
  ReadDataValAdr:= ADR(arReadData),
  SizeOfReadDataVal:= SIZEOF(arReadData),
  WriteDataValAdr:= ADR(arWriteData),
  SizeOfWriteDataVal:= SIZEOF(arWriteData),
  PollInterval:= T#1S,
  Timeout:= T#1S,
  MBData:= stMBData,
  Done=> bDone,
  Active=> bActive,
  Error=> bError,
  ErrorID=> enErrorID,
  ErrorIdent=> stErrorIdent,
  Busy=> bBusy);
```

Expression	Type	Value
arWriteData	ARRAY [0..4] OF WORD	
arWriteData[0]	WORD	100 → Data written to Register addr 4000
arWriteData[1]	WORD	200
arWriteData[2]	WORD	300
arWriteData[3]	WORD	400
arWriteData[4]	WORD	500 → Data written to Register addr 4004
arReadData	ARRAY [0..4] OF WORD	
arReadData[0]	WORD	100 → Data Read from Register addr 4000
arReadData[1]	WORD	200
arReadData[2]	WORD	300
arReadData[3]	WORD	400
arReadData[4]	WORD	500 → Data Read from Register addr 4004

Abb. 20-32: Daten in Slave-Gerät in Programm 1 lesen und schreiben

Lesen und Schreiben von Registern mit SendRequest-Bedingung

Im Beispiel werden 10 Register der Startadresse 0x4200 geschrieben und 5 Register aus Startadresse 0x4100 eines Modbus-RTU-Slave mit Adresse 1 gelesen.

"PollInterval" ist auf 3 Sekunden eingestellt und "EnablePoll" ist TRUE.

Eine steigende Flanke am "bSendRequest"-Eingang löst die Lese- und Schreiboperation aus.

Variablendeklaration

```
VAR
    fbIH_MBReadWriteMultipleRegisters2: IH_MBReadWriteMultipleRegisters;
    bEnable2: BOOL;
    arReadData2: ARRAY [0..4] OF WORD;
    arWriteData2: ARRAY [0..9] OF WORD;
    stMBData: IH_INLN_MB_DATA;
    bDone2: BOOL;
    bActive2: BOOL;
    bError2: BOOL;
    enErrorID2: ERROR_CODE;
    stErrorIdent2: ERROR_STRUCT;
    bBusy2: BOOL;
END_VAR
```

Implementation

```
fbIH_MBReadWriteMultipleRegisters2(
    Enable:= bEnable2,
    SendRequest:= FALSE,
    EnablePoll:= TRUE,
    SlaveAddress:= BYTE#1,
    ReadStartAddress:= WORD#4100,
    WriteStartAddress:= WORD#4200,
    NoOfReadRegisters:= UINT#5,
    NoOfWriteRegisters:= UINT#10,
    ReadDataValAdr:= ADR(arReadData2),
    SizeOfReadDataVal:= SIZEOF(arReadData2),
    WriteDataValAdr:= ADR(arWriteData2),
    SizeOfWriteDataVal:= SIZEOF(arWriteData2),
    PollInterval:= T#3S,
    Timeout:= T#1S,
    MBData:= stMBData,
    Done=> bDone2,
    Active=> bActive2,
    Error=> bError2,
    ErrorID=> enErrorID2,
    ErrorIdent=> stErrorIdent2,
    Busy=> bBusy2);
```

Expression	Type	Value
arWriteData2	ARRAY [0..9] OF WORD	
arWriteData2[0]	WORD	100 → Data written to Register addr 4200
arWriteData2[1]	WORD	200 → Data written to Register addr 4201
arWriteData2[2]	WORD	300
arWriteData2[3]	WORD	400
arWriteData2[4]	WORD	500
arWriteData2[5]	WORD	600
arWriteData2[6]	WORD	700
arWriteData2[7]	WORD	800
arWriteData2[8]	WORD	900
arWriteData2[9]	WORD	1000 → Data written to Register addr 4209
arReadData2	ARRAY [0..4] OF WORD	
arReadData2[0]	WORD	10 → Data Read from Register addr 4100
arReadData2[1]	WORD	20
arReadData2[2]	WORD	30
arReadData2[3]	WORD	40
arReadData2[4]	WORD	50 → Data Read from Register addr 4104

Abb. 20-33: Daten in Slave-Gerät in Programm 2 lesen und schreiben

20.15 IH_MBRTUMaster

Kurzbeschreibung

Der Funktionsbaustein **IH_MBRTUMaster** gewährleistet die Kommunikation zwischen dem Controller und dem physikalischen Liniennetz (Slaves) im RTU-Modus. Er empfängt Anfragen von anderen Funktionsbausteinen von Funktionscodes und sendet sie über die Module **R-IB IL RS 485_422 PRO/R-**

IB IL RS232 Pro/R-IB IL RS UNI an die Slaves. Er empfängt von allen Slaves die Antwort und sendet sie zurück an den Baustein, der ursprünglich die Anfrage gesendet hat. Für alle Terminals muss der Funktionsbaustein einmal instanziiert werden. Er kommuniziert mit maximal 33 Funktionsbausteinen von Funktionscodes gleichzeitig.

Zielsystem	Bibliothek /Ordner
MLC	RIH_Modbus / FB
MTX	RIH_Modbus / FB
MLD (IL 2G)	RIH_Modbus / FB

Tab. 20-41: Referenztabelle

Interfacebeschreibung

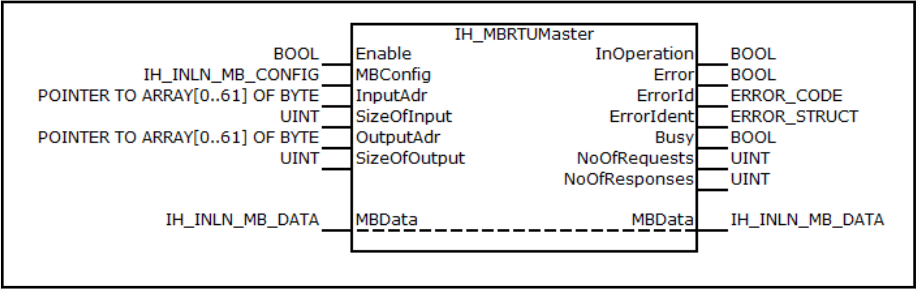


Abb. 20-34: Funktionsbaustein IH_MBRTUMaster

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Funktionsbaustein aktiv, falls TRUE
	MBConfig	IH_INLN_MB_CONFIG	Dient zur Modbus-Konfiguration. IH_INLN_MB_CONFIG, Seite 895
	InputAdr	POINTER TO AR- RAY[0..61] OF BYTE	Adresse des Arrays für Eingangsprozessdaten. Ope- rator ADR () verwenden
	SizeOfInput	UINT	Größe des InputAdr-Arrays. Operator SIZEOF () ver- wenden
	OutputAdr	POINTER TO AR- RAY[0..61] OF BYTE	Adresse des Arrays für Ausgangsprozessdaten. Ope- rator ADR () verwenden
	SizeOfOutput	UINT	Größe des OutputAdr-Arrays. Operator SIZEOF () ver- wenden

E/A-Typ	Name	Datentyp	Beschreibung
VAR_OUTPUT	InOperation	BOOL	Der Funktionsbaustein arbeitet an seiner eigentlichen Aufgabe
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Modbus, Seite 898
	Busy	BOOL	Wahr, wenn der Funktionsbaustein mit Funktionsbausteinen von Funktionscode kommuniziert
	NoOfRequests	UINT	Anzahl gesendeter Anfragen
	NoOfResponses	UINT	Anzahl empfangener Antworten
VAR_IN_OUT	MBData	IH_INLN_MB_DATA	Struktur handhabt Daten zwischen IH_MBRTUMaster und anderen Modbus-Funktionsbausteinen von Funktionscodes IH_INLN_MB_DATA, Seite 894

Tab. 20-42: Schnittstellenvariablen von Funktionsbaustein IH_MBRTUMaster

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Type	Min Value	Max Value	Defaultwert	Wirksam
Enable	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
SizeOfInput	UINT	12	62	-----	steigende Flanke an Enable
SizeOfOutput	UINT	12	62	-----	steigende Flanke an Enable

Tab. 20-43: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_MBRTUMaster

Funktionsbeschreibung

Der Funktionsbaustein **IH_MBRTUMaster** gewährleistet die Kommunikation zwischen dem Controller und dem physikalischen Liniennetz (Slaves) im RTU-Modus. Er empfängt Anfragen von anderen Funktionsbausteinen von Funktionscodes und sendet sie über die Module **R-IB IL RS 485_422 PRO/R-IB IL RS232 Pro/R-IB IL RS UNI** an die Slaves. Er empfängt von einem Slave eine Antwort und sendet sie zurück an den Funktionsbaustein, der die Anfrage gesendet hat. Der Funktionsbaustein gewährleistet die Kommunikation mit dem Terminal. Die "InputAdr" und "OutputAdr" müssen im Terminal abgebildet werden. "MBData" muss mit anderen Funktionsbausteinen von Funktionscodes im Programm verknüpft sein, die Struktur dient zur Kommunikation und zum Datenaustausch mit den anderen Funktionsbausteinen. Der Funktionsbaustein kommuniziert mit maximal 33 Funktionsbausteinen von Funktionscodes

gleichzeitig.

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Enable" auf TRUE gesetzt wird. Während des Betriebs wird der Ausgang "InOperation" auf TRUE gesetzt. Werden in der Warteschlange Funktionscodes erkannt, beginnt der Funktionsbaustein mit der Verarbeitung, bis das Bit "Enable" auf TRUE gesetzt wird. Falls während des Betriebs ein Fehler auftritt, wird das Bit „Error“ auf TRUE gesetzt. ErrorID und ErrorIdent

zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Enable" quittiert werden.

Handhaben

1. Ist für die Kommunikation eine Datenbreite von 7 Bits ausgewählt, können der MODBUS-Master oder -Slave einen Fehler verursachen. Für die Kommunikation ist eine Datenbreite von 8 Bits zu bevorzugen.
2. Werden Master- und Slave-Funktionsbausteine in MLD verwendet, werden Datenbytes und die empfangenen Daten ausgetauscht.

Beispiel eines Funktionsbausteins:

Der Funktionsbaustein IH_MBRTUMaster muss zur Kommunikation mit dem Modbus-Slave zusammen mit anderen Funktionsbausteinen instanziiert werden. Die Eingangs-/Ausgangsvariable stMBData tauscht die Daten zwischen den Funktionsbausteinen IH_MBRTUMaster und anderen Funktionsbausteinen aus. Deshalb muss diese Variable in beiden Funktionsbausteinen MBData zugewiesen werden.

Im Beispiel ist der Funktionsbaustein IH_MBRTUMaster so konfiguriert, dass er Daten mit der Modbus-Einheit und anderen Modbus-Funktionsbausteinen austauscht. Die Variablen "Input processData" und "Output processData" müssen der Modbus-Einheit zur Kommunikation zugewiesen werden.

Kommunikation mit dem RS232Pro-Modul

IH_MBRTUMaster wird zur Kommunikation mit den Modbus-Slaves über das RS232Pro-Modul verwendet.

Variablendeklaration

```
VAR
    fbIH_MBRTUMaster: IH_MBRTUMaster;
    bEnable: BOOL;
    stMBConfig: IH_INLN_MB_CONFIG;
    arOutprocData_RS232: ARRAY [0..11] OF BYTE;
    arInprocData_RS232: ARRAY [0..11] OF BYTE;
    stMBData: IH_INLN_MB_DATA;
    bInOperation: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
    bBusy: BOOL;
    uiNoOfRequests: UINT;
    uiNoOfResponses: UINT;
```

END_VAR

Implementation

```
stMBConfig.Baudrate := IL_INLN_RS_BAUDRATE.BAUDRATE_19200;
stMBConfig.DataWidth := IL_INLN_RS_DATAWIDTH.DATABIT8_EVEN_STOPBIT2;
stMBConfig.InterfaceType := IL_INLN_RS_INTERFACE.RS232;
stMBConfig.ModuleType := IL_INLN_RS_MODULETYPE.RS232PRO;
```

```
fbIH_MBRTUMaster(
    Enable:= bEnable,
    MBConfig:= stMBConfig,
    InputAdr:= ADR(arInprocData_RS232),
    SizeOfInput:= SIZEOF(arInprocData_RS232),
    OutputAdr:= ADR(arOutprocData_RS232),
    SizeOfOutput:= SIZEOF(arOutprocData_RS232),
    MBData:= stMBData,
    InOperation=> bInOperation,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent,
    Busy=> bBusy,
    NoOfRequests=> uiNoOfRequests,
    NoOfResponses=> uiNoOfResponses);
```

Abbildung von Prozessdaten

Es gibt zwei Möglichkeiten zur Prozessdatenabbildung. Sie hängen vom Abbild des verwendeten Moduls ab. Wir schlagen die Verwendung von **Alternative I** vor, da sie einfach und kompakt ist.

Alternative I

Channels			
Variable	Mapping	Channel	Address
Application.fbIH_MBRTUMaster.arInprocData_RS232		Input0	%IB90
Application.fbIH_MBRTUMaster.arOutprocData_RS232		Output0	%QB90

Abb. 20-35: Abbildung von Prozessdaten von IH_MBRTUMaster

Alternative II

Channels				
Variable	Mapping	Channel	Address	Type
Application.fbIH_MBRTUMaster.arInprocData_RS232[0]		Byte0	%IB1	BYTE
Application.fbIH_MBRTUMaster.arInprocData_RS232[1]		Byte1	%IB2	BYTE
Application.fbIH_MBRTUMaster.arInprocData_RS232[2]		Byte2	%IB3	BYTE
Application.fbIH_MBRTUMaster.arInprocData_RS232[3]		Byte3	%IB4	BYTE
Application.fbIH_MBRTUMaster.arInprocData_RS232[4]		Byte4	%IB5	BYTE
Application.fbIH_MBRTUMaster.arInprocData_RS232[5]		Byte5	%IB6	BYTE
Application.fbIH_MBRTUMaster.arInprocData_RS232[6]		Byte6	%IB7	BYTE
Application.fbIH_MBRTUMaster.arInprocData_RS232[7]		Byte7	%IB8	BYTE
Application.fbIH_MBRTUMaster.arInprocData_RS232[8]		Byte8	%IB9	BYTE
Application.fbIH_MBRTUMaster.arInprocData_RS232[9]		Byte9	%IB10	BYTE
Application.fbIH_MBRTUMaster.arInprocData_RS232[10]		Byte10	%IB11	BYTE
Application.fbIH_MBRTUMaster.arInprocData_RS232[11]		Byte11	%IB12	BYTE
Application.fbIH_MBRTUMaster.arOutprocData_RS232[0]		Byte0	%QB1	BYTE
Application.fbIH_MBRTUMaster.arOutprocData_RS232[1]		Byte1	%QB2	BYTE
Application.fbIH_MBRTUMaster.arOutprocData_RS232[2]		Byte2	%QB3	BYTE
Application.fbIH_MBRTUMaster.arOutprocData_RS232[3]		Byte3	%QB4	BYTE
Application.fbIH_MBRTUMaster.arOutprocData_RS232[4]		Byte4	%QB5	BYTE
Application.fbIH_MBRTUMaster.arOutprocData_RS232[5]		Byte5	%QB6	BYTE
Application.fbIH_MBRTUMaster.arOutprocData_RS232[6]		Byte6	%QB7	BYTE
Application.fbIH_MBRTUMaster.arOutprocData_RS232[7]		Byte7	%QB8	BYTE
Application.fbIH_MBRTUMaster.arOutprocData_RS232[8]		Byte8	%QB9	BYTE
Application.fbIH_MBRTUMaster.arOutprocData_RS232[9]		Byte9	%QB10	BYTE
Application.fbIH_MBRTUMaster.arOutprocData_RS232[10]		Byte10	%QB11	BYTE
Application.fbIH_MBRTUMaster.arOutprocData_RS232[11]		Byte11	%QB12	BYTE

Abb. 20-36: Abbildung von Prozessdaten von IH_MBRTUMaster

Kommunikation mit dem RSUni-Modul (RS485 mit 15 Wörtern)

IH_MBRTUMaster wird zur Kommunikation mit den Modbus-Slaves über das RSUni-Modul mit 15 Wörtern verwendet. Mit BaudRate: 19200, DataWidth: DATABIT8_EVEN_STOPBIT1.

Variablendeklaration

```
VAR
fbIH_MBRTUMasterRSUni: IH_MBRTUMaster;
bEnableRSUni: BOOL;
stMBCConfigRSUni: IH_INLN_MB_CONFIG;
arInProcData_RSUNI_15WORDS: ARRAY [0..29] OF BYTE;
arOutprocData_RSUNI_15WORDS: ARRAY [0..29] OF BYTE;
bInOperationRSUni: BOOL;
bErrorRSUni: BOOL;
enErrorIDRSUni: ERROR_CODE;
stErrorIdentRSUni: ERROR_STRUCT;
bBusyRSUni: BOOL;
```

```
    uiNoOfRequestsRSUni: UINT;  
    uiNoOfResponsesRSUni: UINT;  
END_VAR
```

Implementation

```
stMBConfigRSUni.Baudrate := IL_INLN_RS_BAUDRATE.BAUDRATE_19200;  
stMBConfigRSUni.DataWidth := IL_INLN_RS_DATAWIDTH.DATABIT8_EVEN_STOPBIT1;  
stMBConfigRSUni.InterfaceType := IL_INLN_RS_INTERFACE.RS485;  
stMBConfigRSUni.ModuleType := IL_INLN_RS_MODULETYPE.RSUNI_15WORDS;  
  
fbIH_MBRTUMasterRSUni(  
    Enable:= bEnableRSUni,  
    MBConfig:= stMBConfigRSUni,  
    InputAdr:= ADR(arInProcData_RSUNI_15WORDS),  
    SizeOfInput:= SIZEOF(arInProcData_RSUNI_15WORDS),  
    OutputAdr:= ADR(arOutProcData_RSUNI_15WORDS),  
    SizeOfOutput:= SIZEOF(arOutProcData_RSUNI_15WORDS),  
    MBData:= stMBData,  
    InOperation=> bInOperationRSUni,  
    Error=> bErrorRSUni,  
    ErrorID=> enErrorIDRSUni,  
    ErrorIdent=> stErrorIdentRSUni,  
    Busy=> bBusyRSUni,  
    NoOfRequests=> uiNoOfRequestsRSUni,  
    NoOfResponses=> uiNoOfResponsesRSUni);
```

20.16 IH_MBRTUSlave

- Funktionscode

FC 1, 2, 3, 4, 5, 6, 15, 16, 22 und 23
- Kurzbeschreibung

Der Funktionsbaustein **IH_MBRTUSlave** gewährleistet die Kommunikation zwischen dem Controller und dem physikalischen Liniennetz (Master) im RTU-Modus. Durch ihn kann der Controller (Module **R-IB IL RS 485_422 PRO / R-IB IL RS232 Pro / R-IB IL RS UNI**) wie ein Slave mit der Adresse am SlaveAddress-Eingang agieren. Folgende Funktionscodes werden unterstützt: 1, 2, 3, 4, 5, 6, 15, 16, 22 und 23.

Für alle Terminals muss der Funktionsbaustein einmal instanziiert werden.

Zielsystem	Bibliothek /Ordner
MLC	RIH_Modbus / FB
MTX	RIH_Modbus / FB
MLD (IL 2G)	RIH_Modbus / FB

Tab. 20-44: Referenztabelle

Interfacebeschreibung

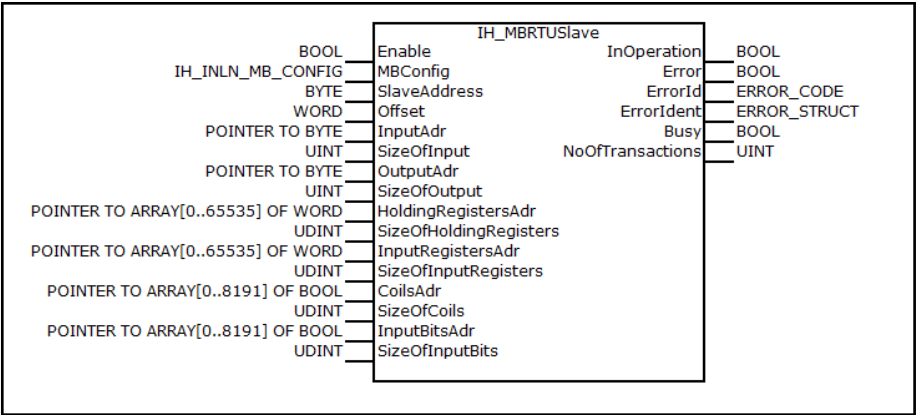


Abb. 20-37: Funktionsbaustein IH_MBRTUSlave

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Funktionsbaustein aktiv, falls TRUE
	MBConfig	IH_INLN_MB_CONFIG	Dient zur Konfiguration. IH_INLN_MB_CONFIG , Seite 895
	SlaveAddress	BYTE	Slave-Adresse
	Offset	WORD	Adressoffset. Bei einem Offset von 2000 beispielsweise beträgt die Adresse des Holding-Registers mit Adressnr. 3 im Array HoldingRegisters 2003 (2000 + 3) und nicht 3.
	InputAdr	POINTER TO ARRAY[0..61] OF BYTE	Adresse des Arrays zum Speichern der Eingangsprozessdaten. Operator ADR () verwenden
	SizeOfInput	UINT	Größe des InputAdr-Arrays. Operator SIZEOF () verwenden
	OutputAdr	POINTER TO ARRAY[0..61] OF BYTE	Adresse des Arrays zum Speichern der Ausgangsprozessdaten. Operator ADR () verwenden
	SizeOfOutput	UINT	Größe des OutputAdr-Arrays. Operator SIZEOF () verwenden
	HoldingRegistersAdr	POINTER TO ARRAY[0..65535] OF WORD	Adresse des Arrays zum Speichern von HoldingRegisters Operator ADR () verwenden
	SizeOfHoldingRegisters	UDINT	Größe des HoldingRegisters-Arrays. Operator SIZEOF () verwenden
	InputRegistersAdr	POINTER TO ARRAY[0..65535] OF WORD	Adresse des Arrays zum Speichern von InputRegisters. Operator ADR () verwenden
	SizeOfInputRegisters	UDINT	Größe des InputRegisters-Arrays. Operator SIZEOF () verwenden
	CoilsAdr	POINTER TO ARRAY[0..8191] OF BOOL	Adresse des Arrays zum Speichern der Coils-Daten. Operator ADR () verwenden
	SizeOfCoils	UDINT	Größe des Coils-Arrays. Operator SIZEOF () verwenden
	InputBitsAdr	POINTER TO ARRAY[0..8191] OF BOOL	Adresse des Arrays zum Speichern von InputBits. Operator ADR () verwenden
	SizeOfInputBits	UDINT	Größe des InputBits-Arrays. Operator SIZEOF () verwenden

E/A-Typ	Name	Datentyp	Beschreibung
VAR_OUTPUT	InOperation	BOOL	Der Funktionsbaustein arbeitet an seiner eigentlichen Aufgabe
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Modbus, Seite 898
	Busy	BOOL	Wahr, wenn der Baustein mit dem Master kommuniziert
	NoOfTransactions	UINT	Anzahl der Transaktionen

Tab. 20-45: Schnittstellenvariablen von Funktionsbaustein IH_MBRTUSlave

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Type	Min Value	Max Value	Defaultwert	Wirksam
Enable	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
SlaveAddress	BYTE	0	247	0	steigende Flanke an Enable
Offset	WORD	0	65536	0	steigende Flanke an Enable
SizeOfInput	UINT	12	62	-----	steigende Flanke an Enable
SizeOfOutput	UINT	12	62	-----	steigende Flanke an Enable
SizeOfHoldingRegisters	UDINT	1	65535	-----	steigende Flanke an Enable
SizeOfInputRegisters	UDINT	1	65535	-----	steigende Flanke an Enable
SizeOfCoils	UDINT	1	8192	-----	steigende Flanke an Enable
SizeOfInputBits	UDINT	1	8192	-----	steigende Flanke an Enable

Tab. 20-46: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_MBRTUSlave

Funktionsbeschreibung

durch den Funktionsbaustein **IH_MBRTUSlave** agiert der Controller wie ein Slave im RTU-Modus. Durch ihn kann der Controller (Module **R-IB IL RS 485_422 PRO/R-IB IL RS232 Pro/R-IB IL RS UN**) wie ein Slave mit der Adresse am SlaveAddress-Eingang agieren. Er verfügt über Datenarrays, die Register und Bits wie im Speicher eines Slave-Geräts emulieren.

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Enable" auf TRUE gesetzt wird. Während des Betriebs wird der Ausgang "InOperation" auf TRUE gesetzt. Werden in der Warteschlange Funktionscodes erkannt, beginnt der Funktionsbaustein mit der Verarbeitung, bis das Bit "Enable" auf TRUE gesetzt wird. Falls während des Betriebs ein Fehler auftritt, wird das Bit „Error“ auf TRUE gesetzt. ErrorID und ErrorIdent

zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Enable" quittiert werden.

Handhaben

1. Ist für die Kommunikation eine Datenbreite von 7 Bits ausgewählt, können der MODBUS-Master oder -Slave einen CRC-Fehler verursachen. Für die Kommunikation ist eine Datenbreite von 8 Bits zu bevorzugen.
2. Werden Master- und Slave-Funktionsbausteine in MLD verwendet, werden Datenbytes und die empfangenen Daten ausgetauscht.

Beispiel eines Funktionsbausteins

Der Funktionsbaustein IH_MBRTUSlave muss für jedes Modbus-Modul, das wie ein Slave mit der am SlaveAddress-Eingang angegebenen Adresse agieren soll, einmal instanziiert werden.

Beispiel 1

In diesem Beispiel konfiguriert der Funktionsbaustein das Modul R IB IL RS485 PRO als Modbus-Slave.

Vor Aktivierung des Funktionsbausteins muss ein Array für Holding-Register (arHoldingRegisters), Input-Register (arInputRegisters), Coils (arCoils) und Input-Bits (arInputBits) erstellt werden.

Außerdem muss ein Array für prozessinterne (arInProcessData) und prozesseexterne Daten (arOutProcessData) erstellt sowie die Variablen und Prozessdaten des Moduls R IB IL RS 485 PRO abgebildet werden.

Die Adresse ist am SlaveAddress-Eingang zuzuweisen und der Funktionsbaustein zu aktivieren. Der Funktionsbaustein stellt die Variable InOperation auf TRUE und zeigt so an, dass der Slave betriebsbereit ist.

Variablendeklaration

```
VAR
    fbIH_MBRTUSlave: IH_MBRTUSlave;
    bEnable: BOOL;
    arInProcessData: ARRAY [0..11] OF BYTE;
    arOutProcessData: ARRAY [0..11] OF BYTE;
    arHoldingRegisters: ARRAY [0..65535] OF WORD;
    arInputRegisters: ARRAY [0..65535] OF WORD;
    arCoils: ARRAY [0..8191] OF BOOL;
    arInputBits: ARRAY [0..8191] OF BOOL;
    bInOperation: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
    bBusy: BOOL;
    uiNoOfTransactions: UINT;
END_VAR
```

Implementation

```
fbIH_MBRTUSlave.MBConfig.Baudrate := IL_INLN_RS_BAUDRATE.BAUDRATE_19200;
fbIH_MBRTUSlave.MBConfig.DataWidth := IL_INLN_RS_DATAWIDTH.DATABIT8_EVEN_STOPBIT2;
fbIH_MBRTUSlave.MBConfig.ModuleType := IL_INLN_RS_MODULETYPE.RS485PRO;
fbIH_MBRTUSlave.MBConfig.InterfaceType := IL_INLN_RS_INTERFACE.RS485;

fbIH_MBRTUSlave(
    Enable:= bEnable,
    SlaveAddress:= BYTE#1,
    Offset:= WORD#0,
    InputAdr:= ADR(arInProcessData),
    SizeOfInput:= SIZEOF(arInProcessData),
    OutputAdr:= ADR(arOutProcessData),
    SizeOfOutput:= SIZEOF(arOutProcessData),
    HoldingRegistersAdr:= ADR(arHoldingRegisters),
    SizeOfHoldingRegisters:= SIZEOF(arHoldingRegisters),
    InputRegistersAdr:= ADR(arInputRegisters),
    SizeOfInputRegisters:= SIZEOF(arInputRegisters),
```

RIH_Modbus.library

```

CoilsAdr:= ADR(arCoils),
SizeOfCoils:= SIZEOF(arCoils),
InputBitsAdr:= ADR(arInputBits),
SizeOfInputBits:= SIZEOF(arInputBits),
InOperation=> bInOperation,
Error=> bError,
ErrorID=> enErrorID,
ErrorIdent=> stErrorIdent,
Busy=> bBusy,
NoOfTransactions=> uiNoOfTransactions);

```

Abbildung von Prozessdaten

R_IB_IL_RS_485_422_PRO[IndraMotionMlc1: Inline_I_O]					
Inline-Module I/O Mapping					
Channels					
Variable	Mapping	Channel	Address	Type	
Application.PlcProg.arInProcessData[0]		Input0	%IB13		
Application.PlcProg.arInProcessData[1]		Byte0	%IB13	BYTE	
Application.PlcProg.arInProcessData[2]		Byte1	%IB14	BYTE	
Application.PlcProg.arInProcessData[3]		Byte2	%IB15	BYTE	
Application.PlcProg.arInProcessData[4]		Byte3	%IB16	BYTE	
Application.PlcProg.arInProcessData[5]		Byte4	%IB17	BYTE	
Application.PlcProg.arInProcessData[6]		Byte5	%IB18	BYTE	
Application.PlcProg.arInProcessData[7]		Byte6	%IB19	BYTE	
Application.PlcProg.arInProcessData[8]		Byte7	%IB20	BYTE	
Application.PlcProg.arInProcessData[9]		Byte8	%IB21	BYTE	
Application.PlcProg.arInProcessData[10]		Byte9	%IB22	BYTE	
Application.PlcProg.arInProcessData[11]		Byte10	%IB23	BYTE	
Application.PlcProg.arInProcessData[11]		Byte11	%IB24	BYTE	
Application.PlcProg.arOutProcessData[0]		Output0	%QB13		
Application.PlcProg.arOutProcessData[1]		Byte0	%QB13	BYTE	
Application.PlcProg.arOutProcessData[2]		Byte1	%QB14	BYTE	
Application.PlcProg.arOutProcessData[3]		Byte2	%QB15	BYTE	
Application.PlcProg.arOutProcessData[4]		Byte3	%QB16	BYTE	
Application.PlcProg.arOutProcessData[5]		Byte4	%QB17	BYTE	
Application.PlcProg.arOutProcessData[6]		Byte5	%QB18	BYTE	
Application.PlcProg.arOutProcessData[7]		Byte6	%QB19	BYTE	
Application.PlcProg.arOutProcessData[8]		Byte7	%QB20	BYTE	
Application.PlcProg.arOutProcessData[9]		Byte8	%QB21	BYTE	
Application.PlcProg.arOutProcessData[10]		Byte9	%QB22	BYTE	
Application.PlcProg.arOutProcessData[11]		Byte10	%QB23	BYTE	
Application.PlcProg.arOutProcessData[11]		Byte11	%QB24	BYTE	

Abb. 20-38: Abbildung von Prozessdaten von IH_MBRTUSlave

Beispiel 2

In diesem Beispiel konfiguriert der Funktionsbaustein das Modul R IB IL RSUNI als Modbus-Slave.

Vor Aktivierung des Funktionsbausteins muss ein Array für Holding-Register (arHoldingRegistersUni), Input-Register (arInputRegistersUni), Coils (arCoilsUni) und Input-Bits (arInputBitsUni) erstellt werden.

Außerdem muss ein Array für prozessinterne (arInProcessDataUni) und prozesseexterne Daten (arOutProcessDataUni) erstellt sowie die Variablen und Prozessdaten des Moduls R IB IL RS UNI abgebildet werden.

Die Adresse ist am SlaveAddress-Eingang zuzuweisen und der Funktionsbaustein zu aktivieren. Der Funktionsbaustein stellt die Variable InOperation auf TRUE und zeigt so an, dass der Slave betriebsbereit ist.

Variablendeklaration

```

VAR
    fbIH_MBRTUSlaveUni: IH_MBRTUSlave;
    bEnableUni: BOOL;
    arInProcessDataUni: ARRAY [0..13] OF BYTE;
    arOutProcessDataUni: ARRAY [0..13] OF BYTE;
    arHoldingRegistersUni: ARRAY [0..65535] OF WORD;
    arInputRegistersUni: ARRAY [0..65535] OF WORD;
    arCoilsUni: ARRAY [0..8191] OF BOOL;
    arInputBitsUni: ARRAY [0..8191] OF BOOL;
    bInOperationUni: BOOL;
    bErrorUni: BOOL;
    enErrorIDUni: ERROR_CODE;
    stErrorIdentUni: ERROR_STRUCT;
    bBusyUni: BOOL;
    uiNoOfTransactionsUni: UINT;
END_VAR

```

Implementation

```

fbIH_MBRTUSlaveUni.MBConfig.Baudrate := IL_INLN_RS_BAUDRATE.BAUDRATE_19200;
fbIH_MBRTUSlaveUni.MBConfig.DataWidth := IL_INLN_RS_DATAWIDTH.DATABIT8_EVEN_STOPBIT2;
fbIH_MBRTUSlaveUni.MBConfig.ModuleType := IL_INLN_RS_MODULETYPE.RSUNI_7WORDS;
fbIH_MBRTUSlaveUni.MBConfig.InterfaceType := IL_INLN_RS_INTERFACE.RS485;

fbIH_MBRTUSlaveUni(
    Enable:= bEnableUni,
    SlaveAddress:= BYTE#1,
    Offset:= WORD#0,
    InputAdr:= ADR(arInProcessDataUni),
    SizeOfInput:= SIZEOF(arInProcessDataUni),
    OutputAdr:= ADR(arOutProcessDataUni),
    SizeOfOutput:= SIZEOF(arOutProcessDataUni),
    HoldingRegistersAdr:= ADR(arHoldingRegistersUni),
    SizeOfHoldingRegisters:= SIZEOF(arHoldingRegistersUni),
    InputRegistersAdr:= ADR(arInputRegistersUni),
    SizeOfInputRegisters:= SIZEOF(arInputRegistersUni),
    CoilsAdr:= ADR(arCoilsUni),
    SizeOfCoils:= SIZEOF(arCoilsUni),
    InputBitsAdr:= ADR(arInputBitsUni),
    SizeOfInputBits:= SIZEOF(arInputBitsUni),
    InOperation=> bInOperationUni,
    Error=> bErrorUni,
    ErrorID=> enErrorIDUni,
    ErrorIdent=> stErrorIdentUni,
    Busy=> bBusyUni,
    NoOfTransactions=> uiNoOfTransactionsUni);

```

Abbildung von Prozessdaten von RS UNI auf lokalem (Inline) Bus

R_IB_IL_RS_UNI[IndraMotionMlc1: Inline_I_O]					
Inline-Module I/O Mapping					
Channels					
Variable	Mapping	Channel	Address	Type	
Application.PlcProg.arInProcessDataUni		Input0	%IW26		
		Word0	%IW26	WORD	
		Word1	%IW28	WORD	
		Word2	%IW30	WORD	
		Word3	%IW32	WORD	
		Word4	%IW34	WORD	
		Word5	%IW36	WORD	
		Word6	%IW38	WORD	
Application.PlcProg.arOutProcessDataUni		Output0	%QW26		
		Word0	%QW26	WORD	
		Word1	%QW28	WORD	
		Word2	%QW30	WORD	
		Word3	%QW32	WORD	
		Word4	%QW34	WORD	
		Word5	%QW36	WORD	
		Word6	%QW38	WORD	

Abb. 20-39: Abbildung von Prozessdaten von RS UNI auf lokalem (Inline) Bus
Abbildung von Prozessdaten von RS UNI auf sercos-Bus

IB_IL_RS_UNI_PD_112_11_Byte_user_data_[IndraMotionMlc1: Sercos_III_Master: R_IL_S3_BK_DI8_DO4_PAC]					
sercos Module Function Groups User Parameter sercos Module I/O Mapping Information					
Channels					
Variable	Mapping	Channel	Address	Type	
		DiagOut	%QB41		
Application.prgIH_MBRTUSlave.arOutProcessDataUni[0]		Byte 0	%QB41	BYTE	
Application.prgIH_MBRTUSlave.arOutProcessDataUni[1]		Byte 1	%QB42	BYTE	
		Complex Protocol	%QB43		
Application.prgIH_MBRTUSlave.arOutProcessDataUni[2]		IL RS UNI - No of Bytes	%QB43	BYTE	
Application.prgIH_MBRTUSlave.arOutProcessDataUni[3]		IL RS UNI - Byte out 0	%QB44	BYTE	
Application.prgIH_MBRTUSlave.arOutProcessDataUni[4]		IL RS UNI - Byte out 1	%QB45	BYTE	
Application.prgIH_MBRTUSlave.arOutProcessDataUni[5]		IL RS UNI - Byte out 2	%QB46	BYTE	
Application.prgIH_MBRTUSlave.arOutProcessDataUni[6]		IL RS UNI - Byte out 3	%QB47	BYTE	
Application.prgIH_MBRTUSlave.arOutProcessDataUni[7]		IL RS UNI - Byte out 4	%QB48	BYTE	
Application.prgIH_MBRTUSlave.arOutProcessDataUni[8]		IL RS UNI - Byte out 5	%QB49	BYTE	
Application.prgIH_MBRTUSlave.arOutProcessDataUni[9]		IL RS UNI - Byte out 6	%QB50	BYTE	
Application.prgIH_MBRTUSlave.arOutProcessDataUni[10]		IL RS UNI - Byte out 7	%QB51	BYTE	
Application.prgIH_MBRTUSlave.arOutProcessDataUni[11]		IL RS UNI - Byte out 8	%QB52	BYTE	
Application.prgIH_MBRTUSlave.arOutProcessDataUni[12]		IL RS UNI - Byte out 9	%QB53	BYTE	
Application.prgIH_MBRTUSlave.arOutProcessDataUni[13]		IL RS UNI - Byte out 10	%QB54	BYTE	
		DiagIn	%IB42		
Application.prgIH_MBRTUSlave.arInProcessDataUni[0]		Byte 0	%IB42	BYTE	
Application.prgIH_MBRTUSlave.arInProcessDataUni[1]		Byte 1	%IB43	BYTE	
		Complex Protocol	%IB44		
Application.prgIH_MBRTUSlave.arInProcessDataUni[2]		IL RS UNI - No of Bytes	%IB44	BYTE	
Application.prgIH_MBRTUSlave.arInProcessDataUni[3]		IL RS UNI - Byte in 0	%IB45	BYTE	
Application.prgIH_MBRTUSlave.arInProcessDataUni[4]		IL RS UNI - Byte in 1	%IB46	BYTE	
Application.prgIH_MBRTUSlave.arInProcessDataUni[5]		IL RS UNI - Byte in 2	%IB47	BYTE	
Application.prgIH_MBRTUSlave.arInProcessDataUni[6]		IL RS UNI - Byte in 3	%IB48	BYTE	
Application.prgIH_MBRTUSlave.arInProcessDataUni[7]		IL RS UNI - Byte in 4	%IB49	BYTE	
Application.prgIH_MBRTUSlave.arInProcessDataUni[8]		IL RS UNI - Byte in 5	%IB50	BYTE	
Application.prgIH_MBRTUSlave.arInProcessDataUni[9]		IL RS UNI - Byte in 6	%IB51	BYTE	
Application.prgIH_MBRTUSlave.arInProcessDataUni[10]		IL RS UNI - Byte in 7	%IB52	BYTE	
Application.prgIH_MBRTUSlave.arInProcessDataUni[11]		IL RS UNI - Byte in 8	%IB53	BYTE	
Application.prgIH_MBRTUSlave.arInProcessDataUni[12]		IL RS UNI - Byte in 9	%IB54	BYTE	
Application.prgIH_MBRTUSlave.arInProcessDataUni[13]		IL RS UNI - Byte in 10	%IB55	BYTE	

Abb. 20-40: Abbildung von Prozessdaten von RS UNI auf sercos-Bus

20.17 IH_MBWriteFileRecord

Funktionscode FC 21

Kurzbeschreibung Der Funktionsbaustein **IH_MBWriteFileRecord** schreibt ein Dateiprotokoll im RTU-Modus. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein **IH_MBRTUMaster**, [Seite 863](#) aufgerufen werden.

Zielsystem	Bibliothek /Ordner
MLC	RIH_Modbus / FB
MTX	RIH_Modbus / FB
MLD (IL 2G)	RIH_Modbus / FB

Tab. 20-47: Referenztable

Interfacebeschreibung

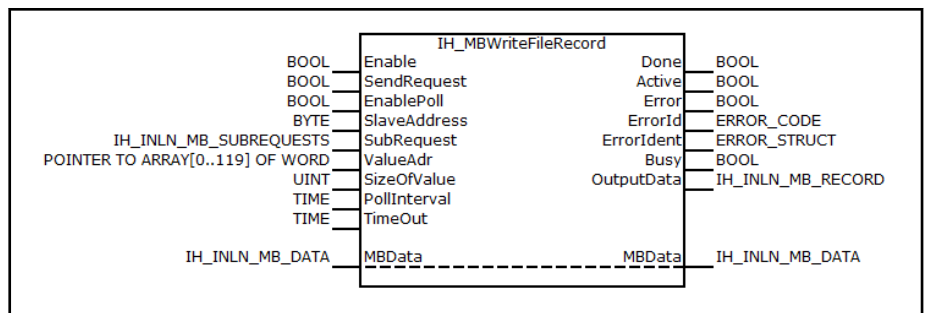


Abb. 20-41: Funktionsbaustein IH_MBWriteFileRecord

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Funktionsbaustein aktiv, falls TRUE
	SendRequest	BOOL	Bei steigender Flanke wird eine Anfrage gesendet
	EnablePoll	BOOL	Polling aktiviert. SendRequest muss nicht in jedem Intervall ausgelöst werden. Der Baustein sendet Anfragen automatisch
	SlaveAddress	BYTE	Slave-Adresse
	SubRequest	IH_INLN_MB_SUBREQUESTS	Unteranfrage. IH_INLN_MB_SUBREQUESTS, Seite 897
	ValueAdr	POINTER TO ARRAY[0..119] OF WORD	Adresse des Protokolldaten-Arrays. Operator ADR () verwenden
	SizeOfValue	UINT	Größe des ValueAdr-Arrays. Operator SIZEOF () verwenden
	PollInterval	TIME	Polling-Intervall. Anfrage wird kontinuierlich in Intervallen gesendet
	TimeOut	TIME	Timeout-Intervall für Antwort von Slaves

E/A-Typ	Name	Datentyp	Beschreibung
VAR_OUTPUT	Done	BOOL	Berechnung beendet und beginnt im nächsten Zyklus neu
	aktiv	BOOL	Funktionsbaustein ist aktiv
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Modbus, Seite 898
	Busy	BOOL	Wahr, wenn der Baustein mit dem Master kommuniziert
	OutputData	IH_INLN_MB_RECORD	Es enthält die Datenlänge von Request sowie unterangefragte Daten. Siehe hierzu: Strukturen und Aufzählungen IH_INLN_MB_RECORD, Seite 897
VAR_IN_OUT	MBData	IH_INLN_MB_DATA	Struktur handhabt Daten zwischen dem Master und anderen Modbus-Funktionsbausteinen der Funktionscodes IH_INLN_MB_DATA, Seite 894

Tab. 20-48: Schnittstellenvariablen Funktionsbaustein IH_MBWriteFileRecord

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Type	Min Value	Max Value	Defaultwert	Wirksam
Enable	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
SendRequest	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
EnablePoll	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
SlaveAddress	BYTE	0	247	0	steigende Flanke an Enable
SizeOfValue	UINT	1	-----	-----	Kontinuierlich bei Enable
PollInterval	TIME	T#1s	-----	T#1s	Kontinuierlich bei Enable
TimeOut	TIME	T#1s	-----	T#1s	steigende Flanke an Enable

Tab. 20-49: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_MBWriteFileRecord

Funktionsbeschreibung

Der Funktionsbaustein **IH_MBWriteFileRecord** (FC 21) dient zum Schreiben von Dateiprotokollen. Die angefragten Datenlängen in Struktur (IH_INLN_MB_SUBREQUESTS) werden in Bytes angegeben, Protokolllängen werden in Worten angegeben. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein aufgerufen werden. "MBData" muss mit dem Master-Funktionsbaustein im Programm verknüpft sein, er dient zur Kommunikation und zum Datenaustausch.

Die Funktion schreibt eine einzige Referenzgruppe. Die Gruppen können getrennt (nicht kontinuierlich) sein,

aber die Referenzen in den einzelnen Gruppen müssen fortlaufend sein.

Jede Gruppe wird in einem separaten Feld für "Unteranfragen" definiert, das 7 Bytes enthält:

- Referenztyp: 1 Byte (muss als 6 angegeben werden). HINWEIS: In diesem FB als Standard betrachtet
- Dateinummer: (Wort) muss (0x0001 bis 0xFFFF) sein
- Anfangsprotokollnummer in der Datei. (Wort) muss (0x0000 bis 0x270F) sein
- Länge des zu lesenden Protokolls (Wort) muss (1 bis 120) sein

Die Anzahl der zu lesenden Register sowie aller anderen Felder in der erwarteten Antwort,

darf die zulässige Länge der MODBUS PDU nicht überschreiten: 253 Bytes.

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Enable" auf TRUE gesetzt wird. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Falls während des Betriebs ein Fehler auftritt, wird das Bit „Error“ auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Enable" quittiert werden.

Beispiel eines Funktionsbausteins

Der Funktionsbaustein IH_MBWriteFileRecord muss zur Kommunikation mit dem Modbus-Slave zusammen mit dem Funktionsbaustein IH_MBRTUMaster instanziiert werden. Die Eingangs-/Ausgangsvariable stMBData tauscht die Daten zwischen den Funktionsbausteinen IH_MBRTUMaster und IH_MBWriteFileRecord aus. Deshalb muss diese Variable in beiden Funktionsbausteinen MBData zugewiesen werden.

Im Beispiel wird der Funktionsbaustein IH_MBWriteFileRecord zum Schreiben des Dateiprotokolls auf das Slave-Gerät verwendet.

Dateiprotokoll mit SendRequest-Bedingung schreiben

IH_MBWriteFileRecord wird verwendet, um ein Protokoll mit Protokolllänge 10 aus Protokoll Nr. 4 in Datei Nr. 1 von Slave-Adresse 2 zu schreiben, mit "bSendRequest" als BOOL-Variable zum Auslösen des Schreibens.

Variablendeklaration

```
VAR
    fbIH_MBWriteFileRecord: IH_MBWriteFileRecord;
    bEnable: BOOL;
    bSendRequest: BOOL;
    stSubRequest: IH_INLN_MB_SUBREQUESTS;
    arWriteRecordData: ARRAY[0..119] OF WORD;
    stMBData: IH_INLN_MB_DATA;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
    bBusy: BOOL;
    stOutputData: IH_INLN_MB_RECORD;
END_VAR
```

Implementation

```
stSubRequest.FileNumber := WORD#1;
stSubRequest.RecordNumber := WORD#4;
```

```
stSubRequest.RecordLength := WORD#10;
```

```
fbIH_MBWriteFileRecord(
  Enable:= bEnable,
  SendRequest:= bSendRequest,
  EnablePoll:= FALSE,
  SlaveAddress:= BYTE#2,
  SubRequest:= stSubRequest,
  ValueAdr:= ADR(arWriteRecordData),
  SizeOfValue:= SIZEOF(arWriteRecordData),
  PollInterval:= T#2S,
  Timeout:= T#2S,
  MBData:= stMBData,
  Done=> bDone,
  Active=> bActive,
  Error=> bError,
  ErrorID=> enErrorID,
  ErrorIdent=> stErrorIdent,
  Busy=> bBusy,
  OutputData=> stOutputData);
```

Expression	Type	Value
arWriteRecordData	ARRAY [0..119] OF WORD	
arWriteRecordData[0]	WORD	100 → Data to write in Record Number 4
arWriteRecordData[1]	WORD	200 → Data to write in Record Number 5
arWriteRecordData[2]	WORD	300
arWriteRecordData[3]	WORD	400
arWriteRecordData[4]	WORD	500
arWriteRecordData[5]	WORD	600
arWriteRecordData[6]	WORD	700
arWriteRecordData[7]	WORD	800
arWriteRecordData[8]	WORD	900
arWriteRecordData[9]	WORD	1000 → Data to write in Record Number 13

Abb. 20-42: arWriteRecordData für Programm 1

Expression	Type	Value
stOutputData	IH_INLN_MB_RECORD	
DataLength	UINT	13 → Valid data in SubRequestData
SubRequestData	ARRAY [0..124] OF WORD	
SubRequestData[0]	WORD	1 → File Number
SubRequestData[1]	WORD	4 → Record Number
SubRequestData[2]	WORD	10 → Record Length
SubRequestData[3]	WORD	100
SubRequestData[4]	WORD	200
SubRequestData[5]	WORD	300
SubRequestData[6]	WORD	400
SubRequestData[7]	WORD	500 → Data written
SubRequestData[8]	WORD	600
SubRequestData[9]	WORD	700
SubRequestData[10]	WORD	800
SubRequestData[11]	WORD	900
SubRequestData[12]	WORD	1000

Abb. 20-43: stOutputData für Programm 1

Dateiprotokoll mit PollInterval-Bedingung schreiben

IH_MBWriteFileRecord wird verwendet, um ein Protokoll mit Protokolllänge 5 aus Protokoll Nr. 1 in Datei Nr. 2 von Slave-Adresse 5 zu schreiben, mit Poll-Interval (3 Sekunden) zum Auslösen des Schreibens.

Variablendeklaration

```
VAR
  fbIH_MBWriteFileRecord2: IH_MBWriteFileRecord;
  bEnable2: BOOL;
  stSubRequest2: IH_INLN_MB_SUBREQUESTS;
  stMBData: IH_INLN_MB_DATA;
  arWriteRecordData2: ARRAY [0..4] OF WORD;
  bDone2: BOOL;
  bActive2: BOOL;
```

```

bError2: BOOL;
enErrorID2: ERROR_CODE;
stErrorIdent2: ERROR_STRUCT;
bBusy2: BOOL;
stOutputData2: IH_INLN_MB_RECORD;
END_VAR

```

Implementation

```

stSubRequest2.FileNumber := WORD#2;
stSubRequest2.RecordNumber := WORD#1;
stSubRequest2.RecordLength := WORD#5;

```

```

fbIH_MBWriteFileRecord2(
  Enable:= bEnable2,
  SendRequest:= FALSE,
  EnablePoll:= TRUE,
  SlaveAddress:= BYTE#5,
  SubRequest:= stSubRequest2,
  ValueAdr:= ADR(arWriteRecordData2),
  SizeOfValue:= SIZEOF(arWriteRecordData2),
  PollInterval:= T#3S,
  TimeOut:= T#2S,
  MBData:= stMBData,
  Done=> bDone2,
  Active=> bActive2,
  Error=> bError2,
  ErrorID=> enErrorID2,
  ErrorIdent=> stErrorIdent2,
  Busy=> bBusy2,
  OutputData=> stOutputData2);

```

Expression	Type	Value
arWriteRecordData2	ARRAY [0..4] OF WORD	
arWriteRecordData2[0]	WORD	100 → Data to write in Record Number 1
arWriteRecordData2[1]	WORD	200 → Data to write in Record Number 2
arWriteRecordData2[2]	WORD	300
arWriteRecordData2[3]	WORD	400
arWriteRecordData2[4]	WORD	500 → Data to write in Record Number 5

Abb. 20-44: arWriteRecordData2 für Programm 2

Expression	Type	Value
stOutputData2	IH_INLN_MB_RECORD	
DataLength	UINT	8 → Valid data in SubRequestData
SubRequestData	ARRAY [0..124] OF WORD	
SubRequestData[0]	WORD	2 → File Number
SubRequestData[1]	WORD	1 → Record Number
SubRequestData[2]	WORD	5 → Record Length
SubRequestData[3]	WORD	100
SubRequestData[4]	WORD	200
SubRequestData[5]	WORD	300 → Data Written
SubRequestData[6]	WORD	400
SubRequestData[7]	WORD	500

Abb. 20-45: stOutputData2 für Programm 2

20.18 IH_MBWriteMultipleCoils

Funktionscode FC 15

Kurzbeschreibung Der Funktionsbaustein **IH_MBWriteMultipleCoils** dient zum Erzwingen mehrerer Coils, d. h. Ausgangsbits ON oder OFF des Slave-Moduls. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein **IH_MBRTUMaster**, Seite 863 aufgerufen werden.

Zielsystem	Bibliothek /Ordner
MLC	RIH_Modbus / FB
MTX	RIH_Modbus / FB
MLD (IL 2G)	RIH_Modbus / FB

Tab. 20-50: Referenztabelle

Interfacebeschreibung

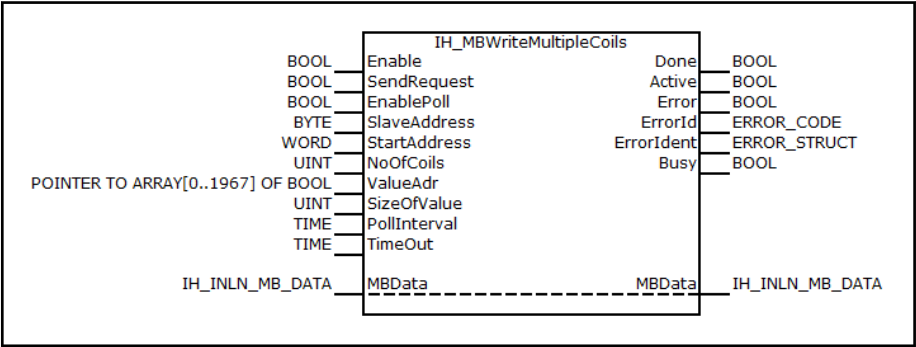


Abb. 20-46: Funktionsbaustein IH_MBWriteMultipleCoils

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Funktionsbaustein aktiv, falls TRUE
	SendRequest	BOOL	Bei steigender Flanke wird eine Anfrage gesendet
	EnablePoll	BOOL	Polling aktiviert. SendRequest muss nicht in jedem Intervall ausgelöst werden. Der Baustein sendet Anfragen automatisch
	SlaveAddress	BYTE	Slave-Adresse
	StartAddress	WORD	Startadresse der Ausgangsbits.
	NoOfCoils	UINT	Anzahl der Coils (getrennte Ausgänge)
	ValueAdr	POINTER TO ARRAY[0..1967] OF BOOL	Adresse des Arrays zum Speichern des Schreibwerts der Ausgangsbits. Operator ADR () verwenden
	SizeOfValue	UINT	Größe des WriteDataVal-Arrays. Operator SIZEOF () verwenden
	PollInterval	TIME	Polling-Intervall. Anfrage wird kontinuierlich in Intervallen gesendet
	TimeOut	TIME	Timeout-Intervall für Antwort von Slaves

E/A-Typ	Name	Datentyp	Beschreibung
VAR_OUTPUT	Done	BOOL	Berechnung beendet und beginnt im nächsten Zyklus neu
	aktiv	BOOL	Funktionsbaustein ist aktiv
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Modbus, Seite 898
	Busy	BOOL	Wahr, wenn der Baustein mit dem Master kommuniziert
VAR_IN_OUT	MBData	IH_INLN_MB_DATA	Struktur handhabt Daten zwischen Master und anderen Modbus-Funktionsbausteinen IH_INLN_MB_DATA, Seite 894

Tab. 20-51: Schnittstellenvariablen von Funktionsbaustein *IH_MBWriteMultipleCoils*

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Type	Min Value	Max Value	Defaultwert	Wirksam
Enable	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
SendRequest	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
EnablePoll	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
SlaveAddress	BYTE	0	247	0	steigende Flanke an Enable
StartAddress	WORD	0	FFFF	0	steigende Flanke an Enable
NoOfCoils	UINT	1	1968	1	steigende Flanke an Enable
SizeOfValue	UINT	1	1968	-----	steigende Flanke an Enable
PollInterval	TIME	T#1s	-----	T#1s	Kontinuierlich bei Enable
TimeOut	TIME	T#1s	-----	T#1s	steigende Flanke an Enable

Tab. 20-52: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von *IH_MBWriteMultipleCoils*

Funktionsbeschreibung

Der Funktionsbaustein **IH_MBWriteMultipleCoils** (FC 15) dient zum Erzwingen mehrerer Coils, d. h. Ausgangsbits ON oder OFF des Slave-Moduls. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein aufgerufen werden. "MBData" muss mit dem Master-Funktionsbaustein im Programm verknüpft sein, er dient zur Kommunikation und zum Datenaustausch.

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Enable" auf TRUE gesetzt wird. Während des Betriebs wird der

Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Falls während des Betriebs ein Fehler auftritt, wird das Bit „Error“ auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Enable" quittiert werden.

Beispiel eines Funktionsbausteins

Der Funktionsbaustein IH_MBWriteMultipleCoils muss zur Kommunikation mit dem Modbus-Slave zusammen mit IH_MBRTUMaster instanziiert werden. Die Eingangs-/Ausgangsvariable stMBData tauscht die Daten zwischen den Funktionsbausteinen IH_MBRTUMaster und IH_MBWriteMultipleCoils aus. Deshalb muss diese Variable in beiden Funktionsbausteinen MBData zugewiesen werden.

Im Beispiel wird der Funktionsbaustein IH_MBWriteMultipleCoils zum Schreiben von Coils auf das Slave-Gerät verwendet.

Coils mit SendRequest-Bedingung schreiben

IH_MBWriteMultipleCoils dient zum Schreiben von 7 Coils der Startadresse 0x1000 in Slave-Adresse 2 mit "bSendRequest" als BOOL-Variable zum Auslösen des Schreibens der Coils.

Variablendeklaration

```
VAR
    fbIH_MBWriteMultipleCoils: IH_MBWriteMultipleCoils;
    bEnable: BOOL;
    bSendRequest: BOOL;
    arWriteMultipleCoils: ARRAY [0..6] OF BOOL;
    stMBData: IH_INLN_MB_DATA;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
    bBusy: BOOL;
END_VAR
```

Implementation

```
fbIH_MBWriteMultipleCoils(
    Enable:= bEnable,
    SendRequest:= bSendRequest,
    EnablePoll:= FALSE,
    SlaveAddress:= BYTE#2,
    StartAddress:= WORD#1000,
    NoOfCoils:= UINT#7,
    ValueAdr:= ADR(arWriteMultipleCoils),
    SizeOfValue:= SIZEOF(arWriteMultipleCoils),
    PollInterval:= T#2S,
    TimeOut:= T#2S,
    MBData:= stMBData,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent,
    Busy=> bBusy);
```

Expression	Type	Value
arWriteMultipleCoils	ARRAY [0..6] OF BOOL	
arWriteMultipleCoils[0]	BOOL	TRUE → Data to write in coil addr 1000
arWriteMultipleCoils[1]	BOOL	FALSE → Data to write in coil addr 1001
arWriteMultipleCoils[2]	BOOL	
arWriteMultipleCoils[3]	BOOL	FALSE
arWriteMultipleCoils[4]	BOOL	TRUE
arWriteMultipleCoils[5]	BOOL	FALSE
arWriteMultipleCoils[6]	BOOL	TRUE → Data to write in coil addr 1006

Abb. 20-47: Daten, die in Programm 1 an das Slave-Gerät übertragen werden müssen

Coils mit PollIntervall-Bedingung schreiben

IH_MBWriteMultipleCoils dient zum Schreiben von 4 Coils der Startadresse 0x0500 in Slave-Adresse 5 mit "PollInterval" (3 Sekunden) zum Auslösen des Schreibens der Coils.

Variablendeklaration

```
VAR
    fbIH_MBWriteMultipleCoils2: IH_MBWriteMultipleCoils;
    bEnable2: BOOL;
    arWriteMultipleCoils2: ARRAY [0..3] OF BOOL;
    stMBData: IH_INLN_MB_DATA;
    bDone2: BOOL;
    bActive2: BOOL;
    bError2: BOOL;
    stErrorIdent2: ERROR_STRUCT;
    enErrorID2: ERROR_CODE;
    bBusy2: BOOL;
END_VAR
```

Implementation

```
fbIH_MBWriteMultipleCoils2(
    Enable:= bEnable2,
    SendRequest:= FALSE,
    EnablePoll:= TRUE,
    SlaveAddress:= BYTE#5,
    StartAddress:= WORD#500,
    NoOfCoils:= UINT#4,
    ValueAdr:= ADR(arWriteMultipleCoils2),
    SizeOfValue:= SIZEOF(arWriteMultipleCoils2),
    PollInterval:= T#3S,
    Timeout:= T#2S,
    MBData:= stMBData,
    Done=> bDone2,
    Active=> bActive2,
    Error=> bError2,
    ErrorID=> enErrorID2,
    ErrorIdent=> stErrorIdent2,
    Busy=> bBusy2);
```

Expression	Type	Value
arWriteMultipleCoils2	ARRAY [0..3] OF BOOL	
arWriteMultipleCoils2[0]	BOOL	TRUE → Data to write in coil addr 500
arWriteMultipleCoils2[1]	BOOL	FALSE
arWriteMultipleCoils2[2]	BOOL	TRUE → Data to write in coil addr 503
arWriteMultipleCoils2[3]	BOOL	FALSE

Abb. 20-48: Daten, die in Programm 2 an das Slave-Gerät übertragen werden müssen

20.19 IH_MBWriteMultipleRegisters

Funktionscode FC 16

Kurzbeschreibung Der Funktionsbaustein **IH_MBWriteMultipleRegisters** dient zum Schreiben eines Holding-Registers des Modbus-Slave-Moduls. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein [IH_MBRTUMaster](#), Seite 863 aufgerufen werden.

Zielsystem	Bibliothek /Ordner
MLC	RIH_Modbus / FB
MTX	RIH_Modbus / FB
MLD (IL 2G)	RIH_Modbus / FB

Tab. 20-53: Referenztabelle

Interfacebeschreibung

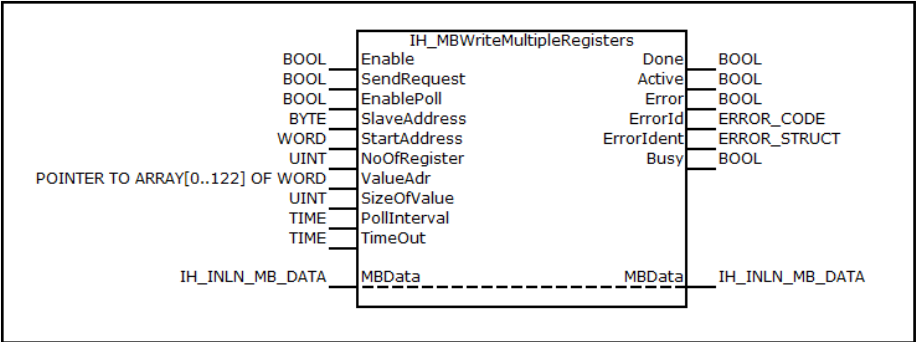


Abb. 20-49: Funktionsbaustein IH_MBWriteMultipleRegisters

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Funktionsbaustein aktiv, falls TRUE
	SendRequest	BOOL	Bei steigender Flanke wird eine Anfrage gesendet
	EnablePoll	BOOL	Polling aktiviert. SendRequest muss nicht in jedem Intervall ausgelöst werden. Der Baustein sendet Anfragen automatisch
	SlaveAddress	BYTE	Slave-Adresse
	StartAddress	WORD	Startadresse des Holding-Registers
	NoOfRegisters	UINT	Anzahl Register
	ValueAdr	POINTER TO ARRAY[0..122] OF WORD	Die Adresse des Arrays enthält den Schreibwert der Register. Operator ADR () verwenden
	SizeOfValue	UINT	Größe des Value-Arrays. Operator SIZEOF () verwenden
	PollInterval	TIME	Polling-Intervall. Anfrage wird kontinuierlich in Intervallen gesendet
	TimeOut	TIME	Timeout-Intervall für Antwort von Slaves

E/A-Typ	Name	Datentyp	Beschreibung
VAR_OUTPUT	Done	BOOL	Berechnung beendet und beginnt im nächsten Zyklus neu
	aktiv	BOOL	Funktionsbaustein ist aktiv
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Modbus, Seite 898
	Busy	BOOL	Wahr, wenn der Baustein mit dem Master kommuniziert
VAR_IN_OUT	MBData	IH_INLN_MB_DATA	Struktur handhabt Daten zwischen Master und anderen Modbus-Funktionsbausteinen IH_INLN_MB_DATA, Seite 894

Tab. 20-54: Schnittstellenvariablen von Funktionsbaustein IH_MBWriteMultipleRegisters

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Type	Min Value	Max Value	Defaultwert	Wirksam
Enable	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
SendRequest	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
EnablePoll	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
SlaveAddress	BYTE	0	247	0	steigende Flanke an Enable
StartAddress	WORD	0	65535	0	steigende Flanke an Enable
NoOfRegisters	UINT	1	123	1	steigende Flanke an Enable
SizeOfValue	UINT	1	-----	-----	steigende Flanke an Enable
PollInterval	TIME	T#1s	-----	T#1s	Kontinuierlich bei Enable
TimeOut	TIME	T#1s	-----	T#1s	steigende Flanke an Enable

Tab. 20-55: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_MBWriteMultipleRegisters

Funktionsbeschreibung

Der Funktionsbaustein **IH_MBWriteMultipleRegisters** (FC 16) schreibt die Holding-Register eines in SlaveAddress festgelegten Slave-Moduls. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein aufgerufen werden. "MBData" muss mit dem Master-Funktionsbaustein verknüpft sein, um mit dem Master zu kommunizieren und Daten auszutauschen.

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Enable" auf TRUE gesetzt wird. Während des Betriebs wird der

Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Falls während des Betriebs ein Fehler auftritt, wird das Bit „Error“ auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Enable" quittiert werden.

Beispiel eines Funktionsbausteins

Der Funktionsbaustein IH_MBWriteMultipleRegisters muss zur Kommunikation mit dem Modbus-Slave zusammen mit IH_MBRTUMaster instanziiert werden. Für MBData der Funktionsbausteine IH_MBWriteMultipleRegisters und IH_MBRTUMaster ist für den Datenaustausch mit dem Modbus-Slave eine gemeinsame Variable zu verwenden.

Beispiel 1

In diesem Beispiel schreibt der Funktionsbaustein IH_MBWriteMultipleRegisters das Holding-Register von Slave-Adresse 1 mit StartAddress 2000 und NoOfRegisters 5.

Die Funktionsbausteine IH_MBRTUMaster und IH_MBWriteMultipleRegisters werden mit einer steigenden Flanke am Eingang bSendRequest aktiviert. Der Funktionsbaustein schreibt die Daten in Slave-Adresse 1 und nach erfolgreicher Beendigung der Schreiboperation wird das Done-Bit für einen SPS-Zyklus auf TRUE gesetzt.

Variablendeklaration

```
VAR
    fbIH_MBWriteMultipleRegisters: IH_MBWriteMultipleRegisters;
    bEnable: BOOL;
    bSendRequest: BOOL;
    arWriteRegisters: ARRAY [0..4] OF WORD;
    stMBData: IH_INLN_MB_DATA;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
    bBusy: BOOL;
END_VAR
```

Implementation

```
fbIH_MBWriteMultipleRegisters(
    Enable:= bEnable,
    SendRequest:= bSendRequest,
    EnablePoll:= FALSE,
    SlaveAddress:= BYTE#1,
    StartAddress:= WORD#2000,
    NoOfRegisters:= UINT#5,
    ValueAdr:= ADDR(arWriteRegisters),
    SizeOfValue:= SIZEOF(arWriteRegisters),
    PollInterval:= T#1S,
    Timeout:= T#1S,
    MBData:= stMBData,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent,
    Busy=> bBusy);
```

Anforderung

Die in das Slave-Modul zu schreibenden Daten müssen in das Array arWriteRegisters eingegeben werden.

Expression	Type	Value
arWriteRegisters	ARRAY [0..4] OF WORD	
arWriteRegisters[0]	WORD	65000 Data value for register 2000
arWriteRegisters[1]	WORD	65100
arWriteRegisters[2]	WORD	65200
arWriteRegisters[3]	WORD	65300
arWriteRegisters[4]	WORD	65400 Data value for register 2004

Abb. 20-50: Array arWriteRegisters

Beispiel 2

In diesem Beispiel schreibt der Funktionsbaustein IH_MBWriteMultipleRegisters das Holding-Register von Slave-Adresse 2 mit StartAddress 3000 und NoOfRegisters 10 mit einem Polling-Intervall von 1 Sekunde. Die Funktionsbausteine IH_MBRTUMaster und IH_MBWriteMultipleRegisters werden aktiviert und EnablePoll mit einem PollInterval von 1 Sekunde auf TRUE gesetzt. Daraufhin schreibt der Funktionsbaustein die Daten aus Slave-Adresse 2 in regelmäßigen Intervallen von 1 Sekunde. Nach jedem erfolgreichen Abschluss der Schreiboperation wird das Done-Bit für einen SPS-Zyklus auf TRUE gesetzt.

Variablendeklaration

```

VAR
    fbIH_MBWriteMultipleRegisters2: IH_MBWriteMultipleRegisters;
    bEnable2: BOOL;
    bSendRequest2: BOOL;
    bEnablePoll: BOOL;
    arWriteRegisters2: ARRAY [0..9] OF WORD;
    stMBData: IH_INLN_MB_DATA;
    bDone2: BOOL;
    bActive2: BOOL;
    bError2: BOOL;
    enErrorID2: ERROR_CODE;
    stErrorIdent2: ERROR_STRUCT;
    bBusy2: BOOL;
END_VAR

```

Implementation

```

fbIH_MBWriteMultipleRegisters2(
    Enable:= bEnable2,
    SendRequest:= bSendRequest2,
    EnablePoll:= bEnablePoll,
    SlaveAddress:= BYTE#2,
    StartAddress:= WORD#3000,
    NoOfRegisters:= UINT#10,
    ValueAdr:= ADR(arWriteRegisters2),
    SizeOfValue:= SIZEOF(arWriteRegisters2),
    PollInterval:= T#1S,
    Timeout:= T#1S,
    MBData:= stMBData,
    Done=> bDone2,
    Active=> bActive2,
    Error=> bError2,
    ErrorID=> enErrorID2,
    ErrorIdent=> stErrorIdent2,
    Busy=> bBusy2);

```

Antwort:

Die in das Slave-Modul zu schreibenden Daten müssen in das Array arWriteRegisters2 eingegeben werden.

Expression	Type	Value
arWriteRegisters2	ARRAY [0..9] OF WORD	
arWriteRegisters2[0]	WORD	2000 Data value for register 3000
arWriteRegisters2[1]	WORD	3000
arWriteRegisters2[2]	WORD	4000
arWriteRegisters2[3]	WORD	4500
arWriteRegisters2[4]	WORD	5000
arWriteRegisters2[5]	WORD	5500
arWriteRegisters2[6]	WORD	6000
arWriteRegisters2[7]	WORD	6500
arWriteRegisters2[8]	WORD	7000
arWriteRegisters2[9]	WORD	7500 Data value for register 3009

Abb. 20-51: Array arWriteRegisters2

20.20 IH_MBWriteSingleCoil

Funktionscode FC 5

Kurzbeschreibung Der Funktionsbaustein **IH_MBWriteSingleCoil** schreibt den ON- oder OFF-Wert in einen einzigen Coil, d. h. ein einziges Ausgangsbit des Slave-Moduls. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein **IH_MBRUMaster**, [Seite863](#) aufgerufen werden.

Zielsystem	Bibliothek /Ordner
MLC	RIH_Modbus / FB
MTX	RIH_Modbus / FB
MLD (IL 2G)	RIH_Modbus / FB

Tab. 20-56: Referenztabelle

Interfacebeschreibung

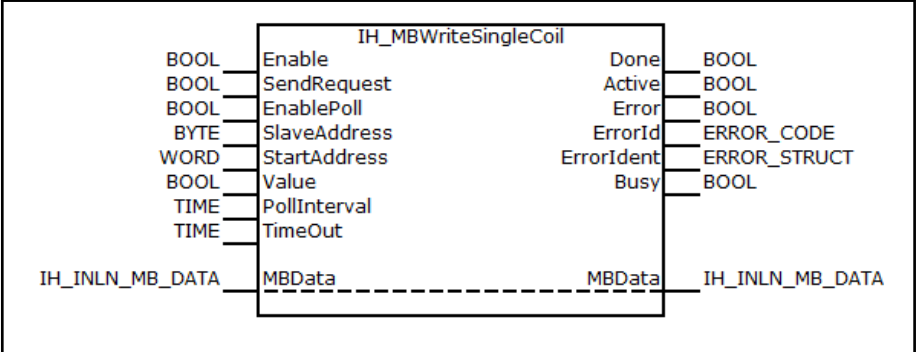


Abb. 20-52: Funktionsbaustein IH_MBWriteSingleCoil

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Funktionsbaustein aktiv, falls TRUE
	SendRequest	BOOL	Bei steigender Flanke wird eine Anfrage gesendet
	EnablePoll	BOOL	Polling aktiviert. SendRequest muss nicht in jedem Intervall ausgelöst werden. Der Baustein sendet Anfragen automatisch
	SlaveAddress	BYTE	Slave-Adresse
	StartAddress	WORD	Startadresse der Eingangsbits
	Value	BOOL	Ausgangsbitwert: TRUE oder FALSE
	PollInterval	TIME	Polling-Intervall. Anfrage wird kontinuierlich in Intervallen gesendet
	TimeOut	TIME	Timeout-Intervall für Antwort von Slaves
VAR_OUTPUT	Done	BOOL	Berechnung beendet und beginnt im nächsten Zyklus neu
	aktiv	BOOL	Funktionsbaustein ist aktiv
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Modbus, Seite 898
	Busy	BOOL	Wahr, wenn der Baustein mit dem Master kommuniziert
VAR_IN_OUT	MBData	IH_INLN_MB_DATA	Struktur handhabt Daten zwischen Master und anderen Modbus-Funktionsbausteinen IH_INLN_MB_DATA, Seite 894

Tab. 20-57: Schnittstellenvariablen von Funktionsbaustein IH_MBWriteSingleCoil

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Type	Min Value	Max Value	Defaultwert	Wirksam
Enable	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
SendRequest	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
EnablePoll	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
SlaveAddress	BYTE	0	247	0	steigende Flanke an Enable
StartAddress	WORD	0	FFFF	0	steigende Flanke an Enable
Value	BOOL	FALSE	TRUE	FALSE	steigende Flanke an Enable

Name	Type	Min Value	Max Value	Defaultwert	Wirksam
PollInterval	TIME	T#1s	-----	T#1s	Kontinuierlich bei Enable
TimeOut	TIME	T#1s	-----	T#1s	steigende Flanke an Enable

Tab. 20-58: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_MBWriteSingleCoil

Funktionsbeschreibung

Der Funktionsbaustein **IH_MBWriteSingleCoil** (FC 5) schreibt den ON- oder OFF-Wert in einen einzigen Coil, d. h. ein einziges Ausgangsbit des Slave-Moduls. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein aufgerufen werden. "MBData" muss mit dem Master-Funktionsbaustein im Programm verknüpft sein, er dient zur Kommunikation und zum Datenaustausch.

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Enable" auf TRUE gesetzt wird. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Falls während des Betriebs ein Fehler auftritt, wird das Bit „Error“ auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Enable" quittiert werden.

Beispiel eines Funktionsbausteins

Der Funktionsbaustein IH_MBWriteSingleCoil muss zur Kommunikation mit dem Modbus-Slave zusammen mit dem Funktionsbaustein IH_MBRTUMaster instanziiert werden. Die Eingangs-/Ausgangsvariable stMBData tauscht die Daten zwischen den Funktionsbausteinen IH_MBRTUMaster und IH_MBWriteSingleCoil aus. Deshalb muss diese Variable in beiden Funktionsbausteinen MBData zugewiesen werden.

Im Beispiel wird der Funktionsbaustein IH_MBWriteSingleCoil zum Schreiben eines einzelnen Coils auf das Slave-Gerät verwendet.

Einzelnen Coil mit SendRequest-Bedingung schreiben

IH_MBWriteSingleCoil dient zum Schreiben eines einzelnen Coils der Startadresse 0x1000 in Slave-Adresse 2 mit "bSendRequest" als BOOL-Variable zum Auslösen des Schreibens des Coils.

Variablendeklaration

```
VAR
    fbIH_MBWriteSingleCoil: IH_MBWriteSingleCoil;
    bEnable: BOOL;
    bSendRequest: BOOL;
    stMBData: IH_INLN_MB_DATA;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
    bBusy: BOOL;
END_VAR
```

Implementation

```
fbIH_MBWriteSingleCoil(
    Enable:= bEnable,
    SendRequest:= bSendRequest,
    EnablePoll:= FALSE,
    SlaveAddress:= BYTE#2,
    StartAddress:= WORD#1000,
    Value:= TRUE,
```

```
PollInterval:= T#2S,  
TimeOut:= T#2S,  
MBData:= stMBData,  
Done=> bDone,  
Active=> bActive,  
Error=> bError,  
ErrorID=> enErrorID,  
ErrorIdent=> stErrorIdent,  
Busy=> bBusy);
```

Expression	Type	Value
coils[999]	BOOL	FALSE
coils[1000]	BOOL	FALSE
coils[1001]	BOOL	FALSE

Old data before Execution.

Expression	Type	Value
coils[999]	BOOL	FALSE
coils[1000]	BOOL	TRUE
coils[1001]	BOOL	FALSE

New data After Execution.

Abb. 20-53: Zeigt die Änderung nach Ausführung

20.21 IH_MBWriteSingleRegister

Funktionscode FC 6

Kurzbeschreibung Der Funktionsbaustein **IH_MBWriteSingleRegister** dient zum Schreiben eines einzigen Holding-Registers des Modbus in das Slave-Modul. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein **IH_MBRTUMaster**, Seite 863 aufgerufen werden.

Zielsystem	Bibliothek /Ordner
MLC	RIH_Modbus / FB
MTX	RIH_Modbus / FB
MLD (IL 2G)	RIH_Modbus / FB

Tab. 20-59: Referenztabelle

Interfacebeschreibung

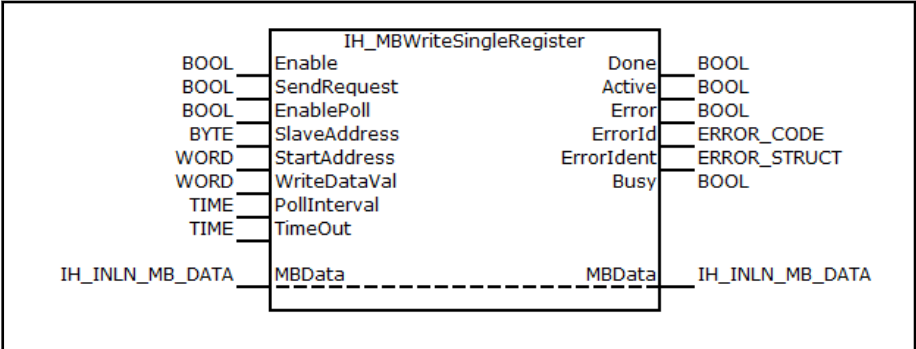


Abb. 20-54: Funktionsbaustein IH_MBWriteSingleRegister

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Funktionsbaustein aktiv, falls TRUE
	SendRequest	BOOL	Bei steigender Flanke wird eine Anfrage gesendet
	EnablePoll	BOOL	Polling aktiviert. SendRequest muss nicht in jedem Intervall ausgelöst werden. Der Baustein sendet Anfragen automatisch
	SlaveAddress	BYTE	Slave-Adresse
	StartAddress	WORD	Startadresse des Holding-Registers
	WriteDataVal	WORD	Wert des Registers schreiben.
	PollInterval	TIME	Polling-Intervall. Anfrage wird kontinuierlich in Intervallen gesendet
	TimeOut	TIME	Timeout-Intervall für Antwort von Slaves
VAR_OUTPUT	Done	BOOL	Berechnung beendet und beginnt im nächsten Zyklus neu
	aktiv	BOOL	Funktionsbaustein ist aktiv
	Error	BOOL	Zeigt einen Fehler an
	ErrorID	ERROR_CODE	Fehlerklasse
	ErrorIdent	ERROR_STRUCT	Detailinformation über den Fehler Fehlerbehandlung mit der Bibliothek RIH_Modbus, Seite 898
	Busy	BOOL	Wahr, wenn der Baustein mit dem Master kommuniziert
VAR_IN_OUT	MBData	IH_INLN_MB_DATA	Struktur handhabt Daten zwischen Master und anderen Modbus-Funktionsbausteinen IH_INLN_MB_DATA, Seite 894

Tab. 20-60: Schnittstellenvariablen von Funktionsbaustein IH_MBWriteSingleRegister

Min-/Max- und Standard-Werte

In der folgenden Tabelle sind die Min-/Max- und Default-Werte sowie das Übernahmeverhalten der Bausteineingänge aufgelistet.

Name	Type	Min Value	Max Value	Defaultwert	Wirksam
Enable	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
SendRequest	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
EnablePoll	BOOL	FALSE	TRUE	FALSE	Kontinuierlich bei Enable
SlaveAddress	BYTE	0	247	0	steigende Flanke an Enable
StartAddress	WORD	0	65535	0	steigende Flanke an Enable
WriteDataVal	WORD	0	65535	0	Kontinuierlich bei Enable

Name	Type	Min Value	Max Value	Defaultwert	Wirksam
PollInterval	TIME	T#1s	-----	T#1s	Kontinuierlich bei Enable
TimeOut	TIME	T#1s	-----	T#1s	steigende Flanke an Enable

Tab. 20-61: Min-, Max- und Defaultwerte sowie Übernahmeverhalten der Bausteineingänge von IH_MBWriteSingleRegister

Funktionsbeschreibung

Der Funktionsbaustein **IH_MBWriteSingleRegister** (FC 6) schreibt ein einziges Holding-Register eines in SlaveAddress festgelegten Modbus-Slave-Moduls. Der Funktionsbaustein muss mit dem Modbus-Master-Funktionsbaustein aufgerufen werden. "MBData" muss mit dem Master-Funktionsbaustein verknüpft sein, um mit dem Master zu kommunizieren und Daten auszutauschen.

Der Funktionsbaustein wird nach dem ersten SPS-Zyklus aktiviert, indem der Eingang "Enable" auf TRUE gesetzt wird. Während des Betriebs wird der Ausgang "Active" auf TRUE gesetzt. Bei Abschluss wird das Bit "Done" auf TRUE gesetzt. Falls während des Betriebs ein Fehler auftritt, wird das Bit „Error“ auf TRUE gesetzt. ErrorID und ErrorIdent zeigen die Fehlerinformationen an. Der Fehler kann dann mit einer steigenden Flanke am Eingang "Enable" quittiert werden.

Beispiel eines Funktionsbausteins

Der Funktionsbaustein IH_MBWriteSingleRegister muss zur Kommunikation mit dem Modbus-Slave zusammen mit IH_MBRTUMaster instanziiert werden. Für MBData der Funktionsbausteine IH_MBWriteSingleRegister und IH_MBRTUMaster ist für den Datenaustausch mit dem Modbus-Slave eine gemeinsame Variable zu verwenden.

In diesem Beispiel schreibt der Funktionsbaustein IH_MBWriteSingleRegister das Holding-Register von Slave-Adresse 1 mit StartAddress 1000 und WriteDataVal 200.

Die Funktionsbausteine IH_MBRTUMaster und IH_MBWriteSingleRegister werden mit einer steigenden Flanke am Eingang bSendRequest aktiviert. Der Funktionsbaustein schreibt die Daten in Slave-Adresse 1.

Wenn die Variable bEnablePoll auf TRUE und Pollinterval auf mehr als 1 Sekunde gestellt werden, werden die Daten in regelmäßigen Intervallen in das Register geschrieben.

Nach jedem erfolgreichen Abschluss der Schreiboperation wird das Done-Bit für einen SPS-Zyklus auf TRUE gesetzt.

Variablendeklaration

```
VAR
    fbIH_MBWriteSingleRegister: IH_MBWriteSingleRegister;
    bEnable: BOOL;
    bSendRequest: BOOL;
    bEnablePoll: BOOL;
    stMBData: IH_INLN_MB_DATA;
    bDone: BOOL;
    bActive: BOOL;
    bError: BOOL;
    enErrorID: ERROR_CODE;
    stErrorIdent: ERROR_STRUCT;
    bBusy: BOOL;
END_VAR
```

Implementation

```

fbIH_MBWriteSingleRegister(
    Enable:= bEnable,
    SendRequest:= bSendRequest,
    EnablePoll:= bEnablePoll,
    SlaveAddress:= BYTE#1,
    StartAddress:= WORD#1000,
    WriteDataVal:= WORD#200,
    PollInterval:= T#1S,
    TimeOut:= T#1S,
    MBData:= stMBData,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> enErrorID,
    ErrorIdent=> stErrorIdent,
    Busy=> bBusy);

```

20.22 Datentypen

20.22.1 IH_INLN_MB_DATA

Die Struktur wird intern zum Austausch von Befehlen und Daten zwischen den Master-Funktionsbausteinen und anderen Funktionsbausteinen der Funktionscodes verwendet.

Bibliothek	Ordner
RIH_Modbus	Type

Tab. 20-62: Bibliothekszuordnung

Name	Type	Beschreibung	Min Value	Max Value	Default-wert	Applikation
Ready	BOOL	Master-FB ist kommunikationsbereit	FALSE	TRUE	FALSE	Kontinuierlich
SendRequest	BOOL	Zeigt an, dass der Master eine Abfrage senden möchte.	FALSE	TRUE	FALSE	Kontinuierlich
SlaveAddress	BYTE	Adresse des betreffenden Slaves.	0	247	0	Kontinuierlich
FunctionCode	INT	Vom Master abgefragter Funktionscode	-----	-----	-----	Kontinuierlich
StartAddress	WORD	Startadresse zur Ausführung des Codes	-----	-----	-----	Kontinuierlich
ArrayData	ARRAY[0..124] OF WORD.	Datenbytes des Telegramms	-----	-----	-----	Kontinuierlich
TimeOut	TIME	Timeout-Intervall	T#1s	-----	T#1s	Kontinuierlich
RespErr	BYTE	Fehlerinformation	-----	-----	-----	Kontinuierlich
Busy	BOOL	Master-Baustein ist beschäftigt	FALSE	TRUE	FALSE	Kontinuierlich
NewDataRecv	BOOL	Neue Daten werden empfangen	FALSE	TRUE	FALSE	Kontinuierlich

Name	Type	Beschreibung	Min Value	Max Value	Default-wert	Applikation
DataCount	UINT	Datenlänge	-----	-----	-----	Kontinuierlich
Done	BOOL	Prozess (Abfrage und Antwort) ist abgeschlossen	FALSE	TRUE	FALSE	Kontinuierlich
Reset	BOOL	Interne Reset-Variable zum Zurücksetzen des Masters verwendet	FALSE	TRUE	FALSE	Kontinuierlich
Token	ARRAY[0..32] OF IH_INLN_MB_TOKEN	Token	-----	-----	-----	Kontinuierlich

Tab. 20-63: IH_INLN_MB_DATA

20.22.2 IH_INLN_MB_DEV_IDCODE

The ENUM **IH_INLN_MB_DEV_IDCODE** gibt die Zugangsebenen der Objektkategorien an. Der Funktionsbaustein [IH_MBReadDeviceIdentification](#) on [page 827](#) unterstützt die folgenden Read Device ID-Code Zugangstypen:

Name	Wert	Beschreibung
IH_INLN_MB_READ01	1	fordert die Basisgeräteidentifikation an (Zugang über Stream)
IH_INLN_MB_READ02	2	fordert die reguläre Geräteidentifikation an (Zugang über Stream)
IH_INLN_MB_READ03	3	fordert die erweiterte Geräteidentifikation an (Zugang über Stream)
IH_INLN_MB_READ04	4	fordert ein spezifisches Identifikationsobjekt an (individueller Zugang)

Für einen benutzerdefinierten Objekt-ID Zugang "DeviceIdCode" als **IH_INLN_MB_READ04**, muss die Objekt-ID im "ObjectId" Eingang angegeben werden. Bei allen anderen Geräte-ID-Codes muss die "ObjectId" 0 sein.

20.22.3 IH_INLN_MB_FIFODATA

Die Struktur gibt die endgültige Antwortausgabe vom Funktionsbaustein **IH_MBReadFIFOQue** ([IH_MBReadFIFOQue](#), [Seite 844](#)) wieder.

Bibliothek	Ordner
RIH_Modbus	Type

Tab. 20-64: Bibliothekszuordnung

Name	Type	Beschreibung
ByteCount	UINT	Byte-Anzahl (Gesamtzahl empfangener Bytes)
FIFOCount	UINT	Anzahl der FIFO-Register
FIFORegValues	ARRAY [1..31] OF WORD	FIFO-Registerwerte

20.22.4 IH_INLN_MB_CONFIG

Die Struktur ist eine Erweiterung von **IL_INLN_RS_CONFIG** und dient zur Konfiguration der Kommunikationseinstellungen des Modbus-Master/Slave-Funktionsbausteins ([RIL_InlineIOModules.library](#), **FB IL_RsCom**, **STRUCT IL_INLN_RS_CONFIG**).

Bibliothek	Ordner
RIH_Modbus	Type

Tab. 20-65: Bibliothekszuordnung

Name	Type	Beschreibung	Min Value	Max Value	Defaultwert	Applikation
Protocol	IL_INLN_RS_PROTOCOL	Bei Modbus nicht zutreffend.				
Baudrate	IL_INLN_RS_BAUDRATE	Baudrate. Für weitere Informationen siehe: IL_INLN_RS_BAUDRATE	BAUDRATE_110	BAUDRATE_250000	BAUDRATE_9600	steigende Flanke an Enable
DataWidth	IL_INLN_RS_DATAWIDTH	Datenbreite	DATA_BIT7_EVEN_STOPBIT1	DATA_BIT6_NO_STOPBIT1	DATA_BIT7_EVEN_STOPBIT1	steigende Flanke an Enable
ErrorPattern	BYTE	Bei Modbus nicht zutreffend				
FirstDelimiter	BYTE	Bei Modbus nicht zutreffend				
SecondDelimiter	BYTE	Bei Modbus nicht zutreffend				
InterfaceType	IL_INLN_RS_INTERFACE	Schnittstellentyp (RS232, RS485 oder RS422). Für weitere Informationen siehe: IL_INLN_RS_INTERFACE	RS232	RS422	RS232	steigende Flanke an Enable
ModuleType	IL_INLN_RS_MODULETYPE	Dient zur Auswahl des angeschlossenen Moduls. Für weitere Informationen siehe: IL_INLN_RS_MODULETYPE	RS232PRO	RSUN_I_31WORDS	RS232PRO	steigende Flanke an Enable

Tab. 20-66: IH_INLN_MB_CONFIG

20.22.5 IH_INLN_MB_TOKEN

Mit dieser Struktur wird die Token-Verfügbarkeit des Master-Funktionsbausteins abgerufen.

Bibliothek	Ordner
RIH_Modbus	Type

Tab. 20-67: Bibliothekszuordnung

Name	Datentyp	Beschreibung	Min Value	Max Value	Defaultwert	Applikation
Occupied	BOOL	Token-Nummer wird bereits verwendet	FALSE	TRUE	FALSE	Kontinuierlich
SendRequest	BOOL	Dem Master wird Token-Anfrage gesendet	FALSE	TRUE	FALSE	Kontinuierlich
RequestGranted	BOOL	Master gewährt Token-Zugriff	FALSE	TRUE	FALSE	Kontinuierlich

Tab. 20-68: IH_INLN_MB_TOKEN

20.22.6 IH_INLN_MB_SUBREQUESTS

Mit der Struktur werden Zugriffsinformationen zu Modbus-Dateiprotokollen abgerufen.

Bibliothek	Ordner
RIH_Modbus	Type

Tab. 20-69: Bibliothekszuordnung

Name	Datentyp	Beschreibung
FileNumber	WORD	Dateinummer
RecordNumber	WORD	Protokollnummer
RecordLength	WORD	Protokolllänge

Tab. 20-70: IH_INLN_MB_SUBREQUESTS

20.22.7 IH_INLN_MB_RECORD

Die Struktur gibt die endgültige Antwortausgabe vom Funktionsbaustein IH_MBReadFileRecord wieder.

Bibliothek	Ordner
RIH_Modbus	Type

Tab. 20-71: Bibliothekszuordnung

Name	Datentyp	Beschreibung
DataLength	UINT	Gesamtmenge empfangener Daten (Ausgabe)
SubRequestData	ARRAY [0.0,124] OF WORD	Abgefragte Protokolldaten

Tab. 20-72: IH_INLN_MB_RECORD

20.22.8 IH_INLN_MB_EVENTLOG

Die Struktur gibt die endgültige Antwortausgabe vom Funktionsbaustein IH_MBGetCommEventLog wieder.

Bibliothek	Ordner
RIH_Modbus	Type

Tab. 20-73: Bibliothekszuordnung

Name	Datentyp	Beschreibung
Status	WORD	Status
EventCount	WORD	Ereignisanzahl
MessageCount	WORD	Meldungsanzahl
Ereignisse	ARRAY [0..63] OF BYTE	Ereignisse

Tab. 20-74: IH_INLN_MB_EVENTLOG

20.23 Fehlerbehandlung mit der Bibliothek RIH_Modbus

Die Funktionsbausteine nutzen die Fehlertabelle **INLINEIO_TABLE**, 16#0190. In Additional1/Additional2 können folgende Fehlermeldungen erzeugt werden:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR	16#3002	16#0001	Die Baudrate ist ungültig
INPUT_INVALID_ERROR	16#3002	16#0002	Datenbreite ist ungültig
INPUT_INVALID_ERROR	16#3002	16#0003	Protokoll ist ungültig
INPUT_INVALID_ERROR	16#3002	16#0004	Schnittstelleneingabe ist ungültig
INPUT_INVALID_ERROR	16#3002	16#0005	Die Größe gesendeter Daten ist ungültig
INPUT_INVALID_ERROR	16#3002	16#0006	Die Größe empfangener Daten ist ungültig
INPUT_INVALID_ERROR	16#3002	16#0007	SizeOfInput oder SizeOfOutput ist ungültig
INPUT_INVALID_ERROR	16#3010	16#0000	Falscher Modultyp angeschlossen
INPUT_INVALID_ERROR	16#3022	16#0000	Der Eingabewert „SlaveAddress“ ist ungültig (0 < SlaveAddress < 247)
INPUT_INVALID_ERROR	16#3022	16#0001	Der Eingabewert „NoOfInputBits/NoOfCoils“ ist ungültig; zum Lesen: (1 < NoOfInputBits/NoOfCoils < 2000) zum Schreiben: (1 < NoOfInputBits/NoOfCoils < 1968)
INPUT_INVALID_ERROR	16#3022	16#0002	Die Eingabe „SizeOfReadDataVal“ ist ungültig (SizeOfReadDataVal < NoOfReadRegisters)
INPUT_INVALID_ERROR	16#3022	16#0003	Die Eingabe „NoOfRegisters“ ist ungültig; zum Lesen: (1 < NoOfRegisters < 125) zum Schreiben: (1 < NoOfRegisters < 123)
INPUT_INVALID_ERROR	16#3022	16#0004	Die Eingabe „NoOfSubRequests“ ist ungültig (1 < NoOfSubRequests < 30)
INPUT_INVALID_ERROR	16#3022	16#0005	Die Eingabe „RecordNumber“ ist ungültig (0 < IH_INLN_MB_SUBREQUESTS.RecordNumber < 9999)
INPUT_INVALID_ERROR	16#3022	16#0006	Die Eingabe „FileNumber“ ist ungültig (FileNumber < 1)
INPUT_INVALID_ERROR	16#3022	16#0007	Die Eingabe „RecordLength“ ist ungültig (1 < IH_INLN_MB_SUBREQUESTS.RecordLength < 120)
INPUT_INVALID_ERROR	16#3022	16#0008	Die Gesamtlänge des Protokolls aller Unteranfragen muss kleiner als (123 – NoOfSubRequests) sein.
INPUT_INVALID_ERROR	16#3022	16#0009	Die Eingabe „SizeOfWriteDataVal“ ist ungültig (SizeOfWriteDataVal < NoOfWriteRegisters)

INPUT_INVALID_ERROR	16#3022	16#000A	Die Eingabe „PollInterval/TimeOut“ ist ungültig (wenn kleiner als 1 Sekunde)
INPUT_INVALID_ERROR	16#3022	16#000C	Die Eingabe „SizeOfValue“ ist ungültig (SizeOfValue < 'NoOfCoils/NoOfRegisters/NoOfInputBits/RecordLength')
INPUT_INVALID_ERROR	16#3022	16#000D	Die Eingabe „NoOfReadRegisters“ ist ungültig (1 < NoOfReadRegisters > 125)
INPUT_INVALID_ERROR	16#3022	16#000E	Die Eingabe „NoOfWriteRegisters“ ist ungültig (1 < NoOfWriteRegisters > 121)
INPUT_INVALID_ERROR	16#3022	16#000F	Die Eingabe „SubFunc“ ist ungültig. Siehe unterstützte Unterfunktionscodes der Lesediagnose
INPUT_INVALID_ERROR	16#3022	16#0010	SizeOfCoils input value ist ungültig (1 < SizeOfCoils > 8191)
INPUT_INVALID_ERROR	16#3022	16#0011	SizeOfInputBits-Eingabewert ist ungültig (1 < SizeOfInputBits > 8191)
INPUT_INVALID_ERROR	16#3022	16#0012	SizeOfHoldingRegisters-Eingabewert ist ungültig (1 < SizeOfHoldingRegisters > 65535)
INPUT_INVALID_ERROR	16#3022	16#0013	SizeOfInputRegisters-Eingabewert ist ungültig (1 < SizeOfInputRegisters > 65535)
DEVICE_ERROR	16#3020	16#0000	Timeout-Fehler. Gerät reagiert nicht
DEVICE_ERROR	16#3023	16#0000	Timeout-Fehler. Gerät reagiert nicht
DEVICE_ERROR	16#3023	16#0001	Modbus-Ausnahmecode 16#01, unzulässige Funktion
DEVICE_ERROR	16#3023	16#0002	Modbus-Ausnahmecode 16#02, unzulässige Datenadresse
DEVICE_ERROR	16#3023	16#0003	Modbus-Ausnahmecode 16#03, unzulässiger Datenwert
DEVICE_ERROR	16#3023	16#0004	Modbus-Ausnahmecode 16#04, Ausfall des Slave-Geräts
DEVICE_ERROR	16#3023	16#0005	Modbus-Ausnahmecode 16#05, quittieren
DEVICE_ERROR	16#3023	16#0006	Modbus-Ausnahmecode 16#06, Slave-Gerät ist beschäftigt
DEVICE_ERROR	16#3023	16#0008	Modbus-Ausnahmecode 16#08, Paritätsfehler im Speicher
DEVICE_ERROR	16#3023	16#000A	Modbus-Ausnahmecode 16#0A, Gateway-Pfad nicht verfügbar
DEVICE_ERROR	16#3023	16#000B	Modbus-Ausnahmecode 16#0B, Gateway-Zielgerät antwortet nicht
DEVICE_ERROR	16#3023	16#000C	Unterschiedliche CRC-Werte für Anfrage und Antwort
DEVICE_ERROR	16#3023	16#000D	Ungültige Antwort, der Fehler kann auftreten, wenn Master und Slave mit unterschiedlicher Geschwindigkeit kommunizieren oder der Slave eine Antwort außerhalb von Modbus schickt
COMMUNICATION_ERROR	16#3011	16#0000	Timeout beim Senden
COMMUNICATION_ERROR	16#3011	16#0001	Fehler beim Senden der Daten
COMMUNICATION_ERROR	16#3011	16#0002	Fehler beim Senden der Daten an den Puffer
COMMUNICATION_ERROR	16#3012	16#0000	Timeout beim Empfangen
RESOURCE_ERROR	16#3013	16#0000	Nullzeiger, Variable nicht zugewiesen

OTHER_ERROR	16#3024	16#0000	Master-Funktionsbaustein ist nicht bereit
INPUT_RANGE_ERROR	16#3025	16#0000	Anzahl der Funktionscode-FBs für Master überschritten. (maximal 33 FBs). Token nicht für die Verarbeitung verfügbar

Tab. 20-75: RIH_Modbus-Fehlercodes

21 RIL_IoLink.library

21.1 Einführung und Übersicht

Die Bibliothek RIL_IoLink enthält feldbusunabhängige Funktionsbausteine zum Ansteuern von IO-Link-Geräten. Feldbusabhängige Funktionsbausteine zum Ansteuern von IO-Link-Geräten sind in der Bibliothek zum jeweiligen Feldbus zu finden, z.B. RIL_ProfinetIO.library.

Zielsysteme Die Bibliothek kann auf folgenden Systemen eingesetzt werden:

Zielsystem	Bibliothek
MLC/MTX	RIL_IoLink
MLD mit IndraLogic 2G ab 20VRS	RIL_IoLink

Anwendungsgebiete und Funktionsbausteine von RIL_IoLink

Anwendungsgebiet (Ordner in der Bibliothek)	
Funktionsbausteine	Beschreibung
IL_SIII_IoLinkProxy, Seite 903	Der Funktionsbaustein implementiert den Proxy-Baustein für die Kommunikation über Sercos zum IO-Link-Master
IL_IoLinkCall, Seite 906	Der Baustein IL_IoLinkCall implementiert den Client-Baustein für die IO-Link-Kommunikation zum IO-Link-Master
IL_IoLinkErrorText, Seite 910	Funktion, die Fehlercodes der IO-Link-Bausteine in lesbaren Text wandelt
IL_IoLinkIdent, Seite 911	Dieser Funktionsbaustein fragt alle Pflicht-IO-Link-Parameter ab, die zur Identifizierung eines Gerätes notwendig sind
RexrothHydraulicValve (Ansteuerung von hydraulischen IO-Link-Geräten)	
IL_IoLinkDirectValveControlType01, Seite 913	Der Funktionsbaustein dient dazu ein IO-Link-Ventil der Firma Bosch Rexroth anzusteuern
Mit Datentypen:	
IL_IOL_OUT_PROC_SPOOL_TYPE01, Seite 917	Prozess-Ausgangsdaten des Funktionsbausteins IL_IoLink-DirectValveControlType01
IL_IOL_IN_PROC_SPOOL_TYPE01, Seite 917	Prozess-Eingangsdaten des Funktionsbausteins IL_IoLink-DirectValveControlType01

Tab. 21-1: RIL_IoLink-Bibliothek

21.2 Allgemeines zu IO-Link

IO-Link ist ein Standard zum Anschluss von Sensoren/Aktoren in der Feldebene. Dabei besteht eine Punkt-zu-Punkt-Verbindung zwischen IO-Link-Master und IO-Link-Device. IO-Link-Master bilden dabei in der Regel eine Gateway-Funktionalität zu bestehenden Feldbussen ab. Neben den zyklischen Prozessdaten können auch azyklische Daten zur Konfiguration und Identifikation gelesen werden. Diese stehen sowohl im IO-Link-Master, als auch in den angeschlossenen IO-Link-Devices bereit. Die nachfolgenden Bausteine dienen dem Lesen und Schreiben dieser Parameter.

Sercos-spezifische IO-Link-Ausprägung

Um IO-Link-Parameter über den Sercos-Feldbus zu lesen/schreiben, wird eine Sercos-Kommunikation zum IO-Link-Master durchgeführt. Der IO-Link-Master ist dabei ein Sercos-Slave. Die IO-Link-Kommunikation wird dabei durch den Sercos getunnelt. Dazu wird das spezifizierte Sercos Sub-Bus-Protokoll verwendet. Dabei werden im Wesentlichen die Parameter S-0-1508.x.65, S-0-1508.x.66 und S-0-1508.x.67 zur Kommunikation mit dem IO-Link-Master verwendet. Da an einem IO-Link-Master in der Regel mehrere IO-Link-Devices angeschlossen sind, werden auch mehrere Kommunikationskanäle benötigt. Da über den Sercos-Feldbus aber nur jeweils eine Anfrage möglich ist, muss verhindert werden, dass mehrere Anfragen gleichzeitig an den IO-Link-Master gestellt werden. Dazu wird ein Proxy-Konzept umgesetzt, das im Folgenden beschrieben wird.

Der Proxy-Baustein übernimmt dabei die (Sercos-spezifische) Kommunikation mit dem IO-Link-Master. Dieser Proxy-Baustein darf je Sercos-Slave nur einmal instanziiert werden. Eine fehlerhafte mehrfache Instanziierung kann dabei nicht verhindert werden. Der IO-Link-Client-Baustein verwendet nur IO-Link spezifische Daten und kommuniziert mit dem Proxy. Dabei können mehrere Instanzen des Clients vorhanden sein. Diese Instanzen müssen sich am Proxy anmelden und können dann unabhängig voneinander Daten mit dem IO-Link-Device austauschen. Der Proxy stellt dabei sicher, dass alle Anfragen der Clients serialisiert an den IO-Link-Master gesendet und die jeweiligen Antworten an den zugehörigen Client zurückgesendet werden. Die Mechanismen, um die Kommunikation zwischen Client und Proxy zu ermöglichen, müssen dabei vom Benutzer nicht verstanden und implementiert werden. Diese Mechanismen erfolgen intern.

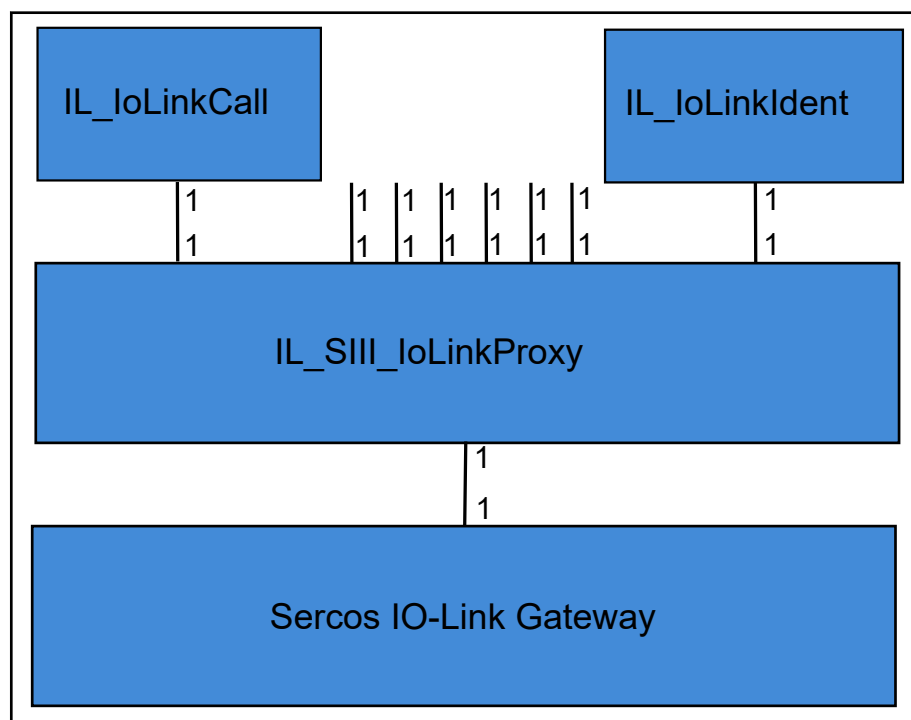


Abb. 21-1: Beziehung zwischen Proxy-Baustein, IO-Link-Client-Bausteinen (*IL_IoLinkCall*, *IL_IoLinkIdent*) und Sercos IO-Link Gateway

Profinet-spezifische IO-Link-Ausprägung

Für Profinet bestehen auch schon mehrere Funktionsbausteine. Diese sind allerdings nicht in dieser Bibliothek vorhanden, sondern in der Bibliothek RIL_ProfinetIO.library. Dort wird aber das Proxy-Konzept noch nicht verwendet. Die vorhandenen Bausteine sind aber in ihrer Funktionalität vergleichbar.

So haben Profinet-Bausteine den Präfix "IL_PNIO" und Baustein aus dieser Bibliothek den Präfix "IL_".

Die Absicht dabei ist, sobald für Profinet auch ein entsprechender Proxy-Baustein vorliegt, die gleichen "IL_" Bausteine aus dieser Bibliothek verwenden zu können.

21.3 IL_SIII_IoLinkProxy

Kurzbeschreibung

Der Baustein IL_SIII_IoLinkProxy implementiert den Proxy-Baustein für die Kommunikation über Sercos zum IO-Link-Master. Proxy ist hier ein "Stellvertreter"-Baustein. Auf der einen Seite ist IO-Link-Feldbus unabhängig, auf der anderen Seite erfolgt die Kommunikation über einen Feldbus. Der Proxy-Baustein implementiert genau dies. Der Baustein hat als Eingänge alle feldbus-abhängigen Werte. In diesem Fall sind dies Sercos-Daten, wie BusMaster, Sercos-Adresse und Slot-Nummer. Alle feldbus-unabhängigen Daten werden dagegen dem Baustein [IL_IoLinkCall](#), [Seite 906](#) übergeben. Somit erfolgt eine Trennung und es kann die IO-Link Funktionalität auf andere Feldbusse erweitert werden, indem einfach der Proxy getauscht wird.

⚠ WARNUNG

Der Baustein darf je IO-Link-Master, bzw. je Sercos-Adresse nur einmal instanziiert werden. Wird er mehrfach instanziiert, so führt dies zu fehlerhaften Anfragen und damit auch zu fehlerhaften Antworten. Dabei gibt der Baustein entweder Fehler aus, oder die erhaltenen Daten sind fehlerhaft.

Es liegt in der Verantwortung des Anwenders jeweils nur eine Proxy-Instanz je Sercos-Adresse anzulegen.

Zielsystem	Bibliothek
MLC/MTX	RIL_IoLink
MLD mit IndraLogic 2G ab 20VRS	RIL_IoLink

Tab. 21-2: Referenztablelle

Schnittstellenbeschreibung

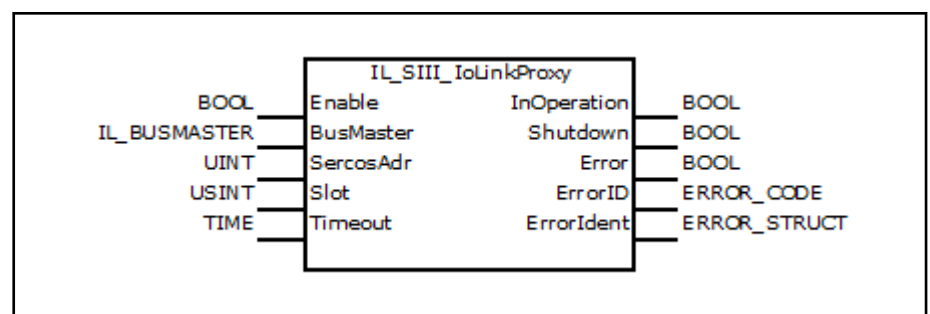


Abb. 21-2: Schnittstelle IL_SIII_IoLinkProxy

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (pegelgesteuert)
	BusMaster	IL_BUSMASTER	Auswahl des Sercos-Busmasters
	SercosAdr	UINT	Sercos Adresse des IO-Link-Masters

E/A-Typ	Name	Datentyp	Beschreibung
	Slot	USINT	Bei modularen Geräten der Slot in dem der IO-Link Master sitzt
	Timeout	TIME	Zeit, die der Baustein maximal benötigen darf, um eine Antwort des IO-Link-Masters zu erhalten
VAR_OUTPUT	InOperation	BOOL	Funktionsbaustein aktiv
	Shutdown	BOOL	Der Baustein führt nach dem Beenden noch notwendige Aufgaben aus
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCT	Detaildiagnose Fehlerbehandlung, Seite 905

Tab. 21-3: Schnittstelle IL_SIII_IoLinkProxy

Name	Datentyp	Min	Max	Default	Übernahme
Enable	BOOL	-	-	FALSE	Kontinuierlich
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Steigende Flanke an Enable
SercosAdr	UINT	0	511	511	Steigende Flanke an Enable
Slot	USINT	0	255	0	Steigende Flanke an Enable
Timeout	TIME	0	49d17h2m47s295ms	t#2s	Steigende Flanke an Enable

Tab. 21-4: Min-/Max- und Default-Werte der Eingänge IL_SIII_IoLinkProxy

Funktionsbeschreibung

Der Baustein stellt die Kommunikation mit einem IO-Link-Master über den Sercos-Feldbus her. Mit dem Setzen des "Enable"-Eingangs nimmt der Baustein seine Arbeit auf. Er ermittelt dabei zyklisch, ob Anfragen von angeschlossenen Clients vorhanden sind, serialisiert diese ggf. und leitet die Anfragen an den IO-Link-Master weiter. Der Baustein muss dazu zyklisch aufgerufen werden. Der Baustein meldet Fehler über seine Ausgänge "Error", "ErrorID" und "ErrorIdent", wobei diese Fehler in der Regel durch fehlerhafte Parametrierung verursacht werden (fehlerhafter Busmaster, Sercos-Adresse, Slot). Fehler im IO-Link-Protokoll, in der IO-Link-Kommunikation oder IO-Link-Daten werden dagegen nicht an diesem Baustein angezeigt, sondern in den jeweiligen Clients. Deshalb arbeitet dieser Baustein in der Regel fehlerfrei und auftretende Fehler müssen an den Clients berücksichtigt werden. Der Eingang "Timeout" hat eine besondere Bedeutung. Die hier eingegebene Zeit wird je Client verwendet. D.h. sind mehrere Clients verbunden, wird die Timeout-Zeit mit der Anzahl der Clients multipliziert. Eine Timeout-Zeit von 0 bedeutet, dass kein Timeout eingestellt ist und der Proxy beliebig lange wartet.

Die Kommunikation zwischen dem Proxy-Baustein und den angeschlossenen Clients (IL_IoLinkCall) erfolgt dabei über interne Interfaces. Darüber werden Daten, Fehler und Status Informationen in beide Richtungen ausgetauscht. Kommunikation wird aber nur intern verwendet und ist dem Anwender nicht zugänglich, um die Handhabung der Bausteine möglichst einfach zu gestalten.

Ein IO-Link-Master kann mehrere "Client Access Points" (CAP) besitzen. Dies sind parallele Wege, um auf Daten des Gerätes zugreifen zu können.

Die Anzahl der CAPs sind je nach Hersteller unterschiedlich. Bei Sercos gibt es, durch die Norm festgelegt, nur einen CAP, beschrieben im Sub-Bus-Protokoll. Über dieses Protokoll können aber gleichzeitig mehrere Anfragen gestellt werden, so dass wieder eine Art Parallelität vorhanden ist. Zudem kann der Proxy bei zu vielen Anfragen diese Anfragen serialisieren. Der Proxy unterstützt bis zu 8 CAPs. Somit können sich bis zu 8 Client-Bausteine mit einem Proxy verbinden.

Eigenschaft pSetLoggerID

Mit dieser Eigenschaft kann die Logger-ID verändert werden. Diese Eigenschaft ist nur für Debug-Zwecke notwendig und wird im laufenden Betrieb nicht verwendet. Diese Eigenschaft ist nur auf Anweisung durch Bosch Rexroth Personal zu ändern.

Fehlerbehandlung

Der Baustein erzeugt nur Fehler, die bei der Sercos-Kommunikation mit dem IO-Link-Master auftreten. Fehler innerhalb der IO-Link-Kommunikation führen dagegen an diesem Baustein zu keinem Fehler. Diese Fehler werden in den Client-Bausteinen erzeugt. Auftretende Fehler stammen dabei überwiegend von den unterlagerten Bausteinen IL_SIIISvcWrite, IL_SIIISvcRead und IL_SIIICommand.

Die am häufigsten vorkommenden Fehler sind hier aufgelistet:

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#00000006	16#00000001	Ungültiger Zustand im Ablauf eines unterlagerten Bausteins
STATE_MACHINE_ERROR	16#00000006	16#00000002	Interner Fehler: Ungültiger Zustand im Ablauf des Bausteins (Betrieb)
STATE_MACHINE_ERROR	16#00000006	16#00000003	Interner Fehler: Ungültiger Zustand im Ablauf des Bausteins (Initialisierung)
INPUT_RANGE_ERROR	16#00000002	16#00000001	Der Eingang "BusMaster" liegt außerhalb des zulässigen Bereiches.
INPUT_RANGE_ERROR	16#00000002	16#00000002	Der Eingang "SercosAdr" liegt außerhalb des zulässigen Bereiches
INPUT_RANGE_ERROR	16#00000002	16#00000007	Der Eingang "Timeout" liegt außerhalb des zulässigen Bereiches
OTHER_ERROR	16#00000015	16#00000001	Der Funktionsbaustein wurde aufgrund einer Zeitüberschreitung beendet
DEVICE_ERROR	16#00002E01	16#00000040	Interner Fehler: Semaphore konnte nicht angelegt werden
DEVICE_ERROR	16#00002E01	16#00000041	Interner Fehler: Semaphore konnte nicht allokiert werden
DEVICE_ERROR	16#00002E01	16#00000042	Interner Fehler: Semaphore konnte nicht frei gegeben werden
DEVICE_ERROR	16#00002E01	16#00000043	Interner Fehler: Semaphore konnte nicht gelöscht werden

Bei den internen Fehlern handelt es sich um Fehler, bei denen der Anwender keine Fehlerbehebung durchführen kann. In diesem Fall ist lediglich ein Neustart des Bausteins vorzunehmen (Eingang "Enable" auf "False" und wieder auf "True" setzen). Tritt der Fehler mehrfach auf, muss der Kundendienst verständigt werden.

Beispielcode

Beispielcodes zur Anwendung des Bausteins sind in der Bibliothek zu finden. Dort gibt es einen Ordner mit der Bezeichnung "_Examples". Der Ordner ent-

hält Beispiele, sowohl für eine einzelne Instanz, die sich mit dem Proxy-Baustein verbindet, als auch ein Beispiel in dem sich mehrere Client-Instanzen mit dem Proxy-Baustein verbinden.

21.4 IL_IoLinkCall

Kurzbeschreibung Der Baustein IL_IoLinkCall implementiert den Client Baustein für die IO-Link-Kommunikation zum IO-Link-Master. Es können dabei mehrere Bausteine an einen Proxy ([IL_IoLinkCall](#), [Seite 906](#)) angeschlossen werden.

Zielsystem	Bibliothek
MLC/MTX	RIL_IoLink
MLD mit IndraLogic 2G ab 20VRS	RIL_IoLink

Tab. 21-5: Referenztablelle

Schnittstellenbeschreibung

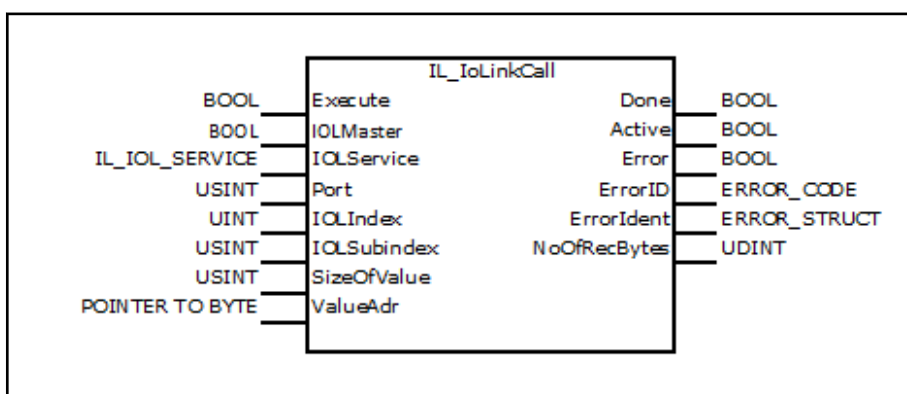


Abb. 21-3: Schnittstelle von IL_IoLinkCall

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (flankengesteuert)
	IOLMaster	BOOL	Auswahl zwischen IO-Link-Device und IO-Link-Master
	IOLService	IL_IOL_SERVICE	Lesen oder Schreiben der Daten
	Port	USINT	IO-Link Port von dem gelesen/geschrieben werden soll (beginnend bei 1)
	IOLIndex	UINT	IO-Link Index
	IOLSubindex	USINT	IO-Link Subindex
	SizeOfValue	USINT	Größe der zu schreibenden Daten oder verfügbarer Speicher (beim Lesen)
	ValueAdr	POINTER TO BYTE	Adresse, an der die Daten liegen (schreiben) oder abgelegt werden (lesen)
VAR_OUTPUT	Done	BOOL	Der Baustein hat die Bearbeitung fehlerfrei durchgeführt
	Active	BOOL	Der Baustein ist noch in Bearbeitung
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall

E/A-Typ	Name	Datentyp	Beschreibung
	ErrorIdent	ERROR_STRUCT	Detailldiagnose Fehlerbehandlung, Seite 908
	NoOfRecBytes	UDINT	Anzahl der erhaltenen Daten in Bytes

Tab. 21-6: Schnittstelle IL_IoLinkCall

Name	Datentyp	Min	Max	Default	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
IOLMaster	BOOL	FALSE	TRUE	IL_FALSE_0	Steigende Flanke an Execute
IOLService	IL_IOL_SERVICE	IL_IOL_SERVICE.WRITE	IL_IOL_SERVICE.READ	IL_IOL_SERVICE.READ	Steigende Flanke an Execute
Port	USINT	0	63	0	Steigende Flanke an Execute
IOLIndex	UINT	0	65535	0	Steigende Flanke an Execute
IOLSubindex	USINT	0	255	0	Steigende Flanke an Execute
SizeOfValue	USINT	0	232	0	kontinuierlich
ValueAdr	POINTER TO BYTE	-	-	0	kontinuierlich

Tab. 21-7: Min-/Max- und Default-Werte der Eingänge IL_IoLinkCall

Funktionsbeschreibung

Der Baustein ermöglicht das Lesen und Schreiben von IO-Link-Parametern, sowohl von IO-Link-Devices als auch vom IO-Link-Master (Auswahl via "IOLMaster"). Daten können sowohl gelesen, als auch geschrieben werden (Auswahl über "IOLService"). Andere Dienste (Cancel und Idle) sind nicht möglich und führen zum Fehler. Zur Adressierung von IO-Link-Device-Parametern werden die Eingänge "Port", "IOLIndex" und "IOLSubindex" verwendet. Bei der Adressierung von IL-Link-Master-Parametern werden nur die Eingänge "Port" und "IOLIndex" verwendet. "IOLSubindex" ist dann ohne Bedeutung. Immer notwendig sind die Eingänge "ValueAdr", die einen Zeiger auf einen Datenbereich darstellt, sowie "SizeOfValue", das die Größe des Speichers angibt. Im Falle eines Schreibzugriffs gibt es die Anzahl der zu schreibenden Bytes an, im Fall von Lesen, den verfügbaren Speicher. Der Baustein arbeitet immer mit einem Proxy-Baustein zusammen. Im Moment gibt es als Proxy-Baustein nur IL_SIII_IoLinkProxy. Die verwendete Proxy-Bausteininstanz muss bei der Deklaration angegeben werden. Sie kann während der Laufzeit nicht geändert werden.

Programm:

```
fbPProxy : IL_SIII_IoLinkProxy; // Declaration of Proxy function block
fbIOLCall : IL_IoLinkCall(fbPProxy); // Declaration of Client function block
// with initialization of proxy instance
```

Der Baustein registriert sich nur am Proxy, solange eine Anfrage aktiv ist. Sobald die Anfrage erfolgreich, oder auch mit Fehler abgebrochen wird, meldet sich der Baustein wieder vom Proxy ab.

Eigenschaft pSetLoggerID

Mit dieser Eigenschaft kann die Logger-ID verändert werden. Diese Eigenschaft ist nur für Debug-Zwecke notwendig und wird im laufenden Betrieb

nicht verwendet. Diese Eigenschaft ist nur auf Anweisung durch Bosch Rexroth Personal zu ändern.

Fehlerbehandlung

Der Baustein erzeugt nur Fehler, die innerhalb der IO-Link Kommunikation auftreten. Fehler bei der Sercos-Kommunikation werden dagegen am Proxy-Baustein (IL_SIII_IoLinkProxy) ausgegeben. Die Fehler werden teilweise im Baustein generiert, oder werden durch den IO-Link Master explizit geliefert und sind im IO-Link Standard spezifiziert (Fehler mit dem Additional1 = 16#00002E03).

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#00000006	16#00000002	Ungültiger Zustand im Ablauf eines unterlagerten Bausteins
INPUT_RANGE_ERROR	16#00000002	16#00000005	Der Eingang "SizeOfValue" liegt außerhalb des zulässigen Bereiches
INPUT_RANGE_ERROR	16#00002E00	16#00000020	Der Eingang "Proxy" ist ungültig
INPUT_RANGE_ERROR	16#00002E00	16#00000021	Der Eingang "IOLService" liegt außerhalb des zulässigen Bereiches
INPUT_INVALID_ERROR	16#00002E01	16#00000022	Interner Fehler (Proxy-Datenadresse oder Call Interface Adresse ungültig)
SYSTEM_ERROR	16#00002E01	16#00000023	Proxy befindet sich im Fehler. Anforderung wird nicht bearbeitet. Proxy Baustein muss zuerst neu gestartet werden
RESOURCE_ERROR	16#00002E02	16#00000024	IL_IOLinkCall() wurde aufgerufen bevor der Proxy "In-Operation" anzeigt.
RESOURCE_ERROR	16#00002E02	16#00000025	Interner Fehler (Baustein hat bereits Anforderung gestellt. Es kann keine neue Anforderung gestellt werden, solange die alte noch in Bearbeitung ist)
RESOURCE_ERROR	16#00002E02	16#00000026	Alle Ressourcen im Proxy sind belegt, z.Zt. keine Anforderung möglich
RESOURCE_ERROR	16#00002E02	16#00000027	Interner Fehler (Baustein wird abgemeldet, war aber nie angemeldet)
RESOURCE_ERROR	16#00002E02	16#00000028	Interner Fehler (neue Anforderung wird gestellt, aber Baustein ist noch nicht angemeldet)
DEVICE_ERROR	16#00002E03	16#00000029	Interner Fehler (Command PDU Größe zu klein)
DEVICE_ERROR	16#00002E03	16#0000002A	Interner Fehler (Port in Anfrage und Antwort sind unterschiedlich)
DEVICE_ERROR	16#00002E03	16#0000002B	Interner Fehler (Command SDU Größe zu klein)
DEVICE_ERROR	16#00002E03	16#0000002C	Interner Fehler (Kommando in Anfrage und Antwort sind unterschiedlich)
DEVICE_ERROR	16#00002E03	16#0000002D	Interner Fehler (Index oder Subindex in Anfrage und Antwort sind unterschiedlich)
DEVICE_ERROR	16#00002E03	16#0000002E	Interner Fehler (Telegramm Größe in Antwort ist ungültig)
DEVICE_ERROR	16#00002E03	16#0000002F	Interner Fehler (Datengröße in Antwort ist ungültig)
DEVICE_ERROR	16#00002E03	16#00000030	Interner Fehler (Telegramm Typ in Antwort ist ungültig)
DEVICE_ERROR	16#00002E03	16#00000031	Interner Fehler (Sub-Bus Telegramm zu klein)

ErrorID	Additional1	Additional2	Beschreibung
DEVICE_ERROR	16#00002E03	16#00000032	Interner Fehler (Sub-Bus PDU Version unbekannt)
DEVICE_ERROR	16#00002E03	16#00000033	Interner Fehler (Sub-Bus Telegramm Typ ist nicht IO-Link)
DEVICE_ERROR	16#00002E03	16#00000034	Interner Fehler (Sub-Bus SDU Version unbekannt)
DEVICE_ERROR	16#00002E03	16#00000035	Interner Fehler (Sub-Bus PDU Größe und Listenlänge stimmen nicht überein)
DEVICE_ERROR	16#00002E03	16#00000036	Proxy wurde zurück gesetzt (durch Fehler oder Enable = False) während noch Anfragen liefen
DEVICE_ERROR	16#00002E05	16#00008000	Ausgewählter Port ist nicht verfügbar
DEVICE_ERROR	16#00002E05	16#00008001	Ausgewählter Parameterindex (Eingang "IOIndex") ist nicht verfügbar
DEVICE_ERROR	16#00002E05	16#00008002	Parameterindex (Eingang "IOIndex") und Daten innerhalb des Telegramms stimmen nicht überein
DEVICE_ERROR	16#00002E05	16#00008003	Control-Feld innerhalb des Anforderungstelegramms ist unbekannt
DEVICE_ERROR	16#00002E05	16#00008004	Die gewählte Port-Zykluszeit wird nicht unterstützt oder ist nicht definiert
DEVICE_ERROR	16#00002E05	16#00008005	Die konfigurierte Zykluszeit wird nicht unterstützt oder ist nicht definiert
DEVICE_ERROR	16#00002E05	16#00008006	Der konfigurierte Speichermodus wird nicht unterstützt oder ist nicht definiert
DEVICE_ERROR	16#00002E06	16#00008001	Der Dienst ist nicht verfügbar (z.B. ist der Port nicht als IO-Link Port konfiguriert)
DEVICE_ERROR	16#00002E06	16#00008002	Port ist durch anderen Dienst belegt
DEVICE_ERROR	16#00002E06	16#00008003	Timeout wenn IO-Link Device oder IO-Link Master Ports belegt sind
DEVICE_ERROR	16#00002E06	16#00008004	Ungültiger Index
DEVICE_ERROR	16#00002E06	16#00008005	Ungültiger Subindex
DEVICE_ERROR	16#00002E06	16#00008007	Port-Funktion wird nicht unterstützt
DEVICE_ERROR	16#00002E06	16#00008008	Länge ungültig
DEVICE_ERROR	16#00002E06	16#00008009	Das IO-Link-Device unterstützt das verwendete Protokoll nicht
DEVICE_ERROR	16#00002E06	16#00008010	Werte im Control-Feld der Anforderung ist nicht definiert
DEVICE_ERROR	16#00002E07	16#00008000	Allgemeiner Fehler
DEVICE_ERROR	16#00002E07	16#00008011	Index nicht vorhanden
DEVICE_ERROR	16#00002E07	16#00008012	Subindex nicht vorhanden
DEVICE_ERROR	16#00002E07	16#00008020	Dienst vorübergehend nicht verfügbar
DEVICE_ERROR	16#00002E07	16#00008021	Dienst vorübergehend nicht verfügbar – lokal in Verwendung
DEVICE_ERROR	16#00002E07	16#00008022	Dienst vorübergehend nicht verfügbar – durch IO-Link Device

ErrorID	Additional1	Additional2	Beschreibung
DEVICE_ERROR	16#00002E07	16#00008023	Zugriff nicht gestattet
DEVICE_ERROR	16#00002E07	16#00008030	Parameter außerhalb des Wertebereichs
DEVICE_ERROR	16#00002E07	16#00008031	Parameter größer als maximaler Wert
DEVICE_ERROR	16#00002E07	16#00008032	Parameter kleiner als minimaler Wert
DEVICE_ERROR	16#00002E07	16#00008033	Parameterlänge zu lang
DEVICE_ERROR	16#00002E07	16#00008034	Parameterlänge zu kurz
DEVICE_ERROR	16#00002E07	16#00008035	Funktion nicht verfügbar
DEVICE_ERROR	16#00002E07	16#00008036	Funktion temporär nicht verfügbar
DEVICE_ERROR	16#00002E07	16#00008040	Ungültiger Parametersatz
DEVICE_ERROR	16#00002E07	16#00008041	Inkonsistenter Parametersatz
DEVICE_ERROR	16#00002E07	16#00008082	Applikation nicht verfügbar
DEVICE_ERROR	16#00002E07	16#00008100 .. 16#0000810FF	Herstellerspezifischer Fehler



Bei den internen Fehlern handelt es sich um Fehler bei denen der Anwender keine Fehlerbehebung durchführen kann. In diesem Fall ist lediglich ein Neustart des Bausteins vorzunehmen (Eingang "Enable" auf "False" und wieder auf "True" setzen). Tritt der Fehler mehrfach auf, muss der Kundendienst verständigt werden.



Die Additional1 Fehler mit den Werten 16#00002E05, 16#00002E06 und 16#00002E07 werden durch den IO-Link-Master generiert. Je nach Hersteller können neben den hier gelisteten Additional2 Fehlern noch weitere Fehler auftreten. Deren Bedeutung ist dann in der jeweiligen Dokumentation des IO-Link-Master nachzulesen.

Beispielcode

Beispielcode zur Anwendung des Bausteins sind in der Bibliothek zu finden. Dort gibt es einen Ordner mit der Bezeichnung "_Examples". Der Ordner enthält Beispiele, sowohl für eine einzelne Instanz, die sich mit dem Proxy-Baustein verbindet, als auch ein Beispiel in dem sich mehrere Client-Instanzen mit dem Proxy-Baustein verbinden.

21.5 IL_IoLinkErrorText

Kurzbeschreibung

Die Funktion IL_IoLinkErrorText wandelt einen Fehlercode der Bausteine [IL_SIII_IoLinkProxy, Seite 903](#), [IL_IoLinkCall, Seite 906](#) und dessen abgeleitete Bausteine in einen Text um.



Die Funktion ist nur für Entwicklungszwecke gedacht. Deshalb erfolgt die Ausgabe auch nur in englischer Sprache.

Zielsystem	Bibliothek
MLC/MTX	RIL_IoLink
MLD mit IndraLogic 2G ab 20VRS	RIL_IoLink

Tab. 21-8: Referenztablelle

Schnittstellenbeschreibung

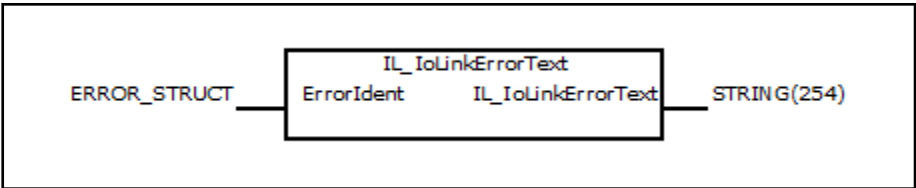


Abb. 21-4: Schnittstelle von IL_IoLinkErrorText

E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	ErrorIdent	ERROR_STRUCT	Error Struktur eines der Bausteine IL_SIII_IoLinkProxy, IL_IoLinkCall oder dessen abgeleitete Bausteine, dessen Text angezeigt werden soll
VAR_OUTPUT	IL_IoLinkError-Text	STRING(254)	Text in englischer Sprache für den Fehlercode

Tab. 21-9: Schnittstelle IL_IoLinkErrorText

Funktionsbeschreibung

Die Funktion IL_IoLinkErrorText wandelt einen Fehlercode der Bausteine IL_SIII_IoLinkProxy, IL_IoLinkCall und dessen abgeleitete Bausteine in einen Text um. Die Fehlercodes werden aus den Konstanten IL_IO_LINK_CONSTANTS_ADD1 und IL_IO_LINK_CONSTANTS_ADD1 verwendet und als Text in englischer Sprache ausgegeben. Andere Sprachen sind nicht verfügbar, da diese Funktion nur für Entwicklerzwecke erstellt wurde.

21.6 IL_IoLinkIdent

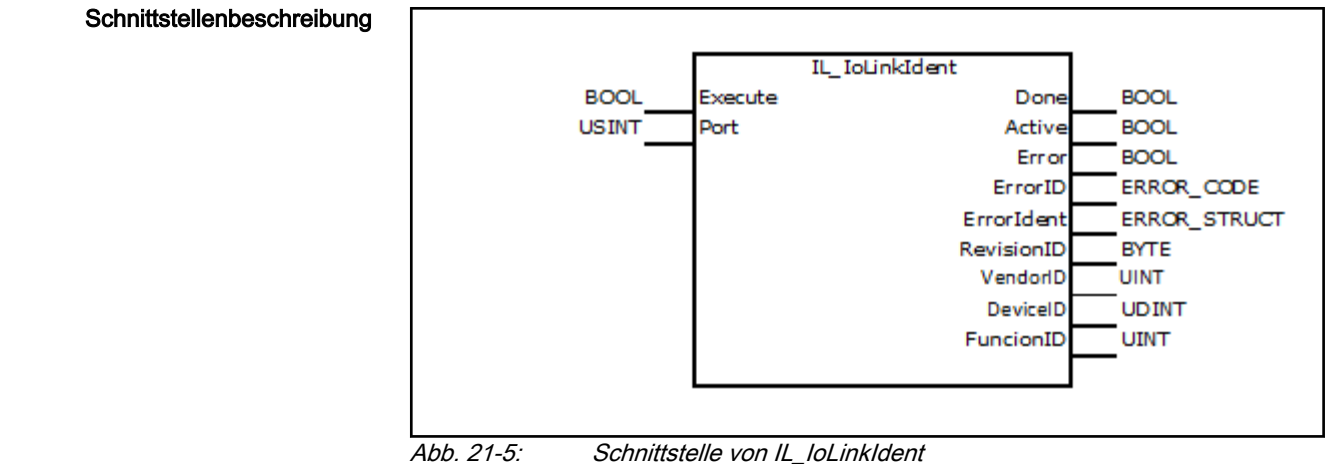
Kurzbeschreibung

Dieser Funktionsbaustein fragt alle Pflicht-IO-Link-Parameter ab, die zur Identifizierung eines Gerätes notwendig sind. Der Baustein IL_IoLinkIdent implementiert einen Client-Baustein für die IO-Link-Kommunikation zum IO-Link-Master. Es können dabei mehrere Bausteine an einen Proxy (IL_SIII_IoLinkProxy, Seite 903) angeschlossen werden. Er basiert auf dem Kap. 21.4 "IL_IoLinkCall" auf Seite 906 Baustein.

Der Baustein liest die "Direct parameter page 1". Daraus extrahiert der Baustein dann die erforderlichen Informationen und stellt sie an den Ausgängen zur Verfügung. Da es sich bei "Direct parameter page 1" um Pflichtparameter handelt, sind auch alle Parameter verfügbar. Bei anderen IO-Link-Parametern ist dies nicht zwingend der Fall.

Zielsystem	Bibliothek
MLC/MTX	RIL_IoLink
MLD mit IndraLogic 2G ab 20VRS	RIL_IoLink

Tab. 21-10: Referenztablelle



E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Execute	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (flankengesteuert)
	Port	USINT	IO-Link Port von dem gelesen/geschrieben werden soll (beginnend bei 1)
VAR_OUTPUT	Done	BOOL	Der Baustein hat die Bearbeitung fehlerfrei durchgeführt
	Active	BOOL	Der Baustein ist noch in Bearbeitung
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCT	Detaildiagnose Fehlerbehandlung, Seite 913
	RevisionID	BYTE	Revision des IO-Link-Gerätes
	VendorID	UINT	Vendor ID des IO-Link-Gerätes
	DeviceID	UDINT	Device ID des IO-Link-Gerätes
	FunctionID	UINT	Function ID des IO-Link-Gerätes

Tab. 21-11: Schnittstelle IL_IoLinkIdent

Name	Datentyp	Min	Max	Default	Übernahme
Execute	BOOL	FALSE	TRUE	FALSE	Kontinuierlich
Port	USINT	0	63	0	Steigende Flanke an Execute

Tab. 21-12: Min-/Max- und Default-Werte der Eingänge IL_IoLinkIdent

Funktionsbeschreibung

Der Baustein liest die zur Identifikation eines IO-Link-Gerätes notwendigen Werte aus. Da es sich dabei um Pflichtparameter handelt, müssen diese Werte auch von jedem Gerät unterstützt werden. Der Baustein arbeitet immer mit einem Proxy-Baustein zusammen. Im Moment gibt es als Proxy-Baustein nur [Kap. 21.3 "IL_SIII_IoLinkProxy" auf Seite 903](#). Die verwendete Proxy-Bausteininstanz muss bei der Deklaration angegeben werden. Sie kann während der Laufzeit nicht geändert werden.

Programm:

```
fbPProxy : IL_SIII_IoLinkProxy; // Declaration of Proxy function block
fbIOLIdent : IL_IL_IoLinkIdent(fbPProxy); // Declaration of Client function block
// with initialization of proxy instance
```

- Eigenschaft pSetLoggerID

Der Baustein registriert sich nur am Proxy, solange eine Anfrage aktiv ist. Sobald die Anfrage erfolgreich oder auch mit Fehler abgebrochen wird, meldet sich der Baustein wieder vom Proxy ab.
- Eigenschaft pSetLoggerID

Mit dieser Eigenschaft kann die Logger-ID verändert werden. Diese Eigenschaft ist nur für Debug-Zwecke notwendig und wird im laufenden Betrieb nicht verwendet. Diese Eigenschaft ist nur auf Anweisung durch Bosch Rexroth Personal zu ändern.
- Fehlerbehandlung

Der Baustein erzeugt nur Fehler, die innerhalb der IO-Link-Kommunikation auftreten. Fehler bei der Sercos-Kommunikation werden dagegen am Proxy-Baustein (IL_SIII_IoLinkProxy) ausgegeben. Die Fehler werden teilweise im Baustein generiert, oder werden durch den IO-Link-Master explizit geliefert und sind im IO-Link-Standard spezifiziert (Fehler mit dem Additional1 = 16#00002E03). Da der Baustein unterlagert auf den IL_IoLinkCall aufbaut, werden auch die gleichen Fehler erzeugt (siehe "Fehlerbehandlung" auf Seite 908).
- Beispielcode

Beispielcode zur Anwendung des Bausteins sind in der Bibliothek zu finden. Dort gibt es einen Ordner mit der Bezeichnung "_Examples". Der Ordner enthält Beispiele, sowohl für eine einzelne Instanz, die sich mit dem Proxy-Baustein verbindet, als auch ein Beispiel in dem sich mehrere Client-Instanzen mit dem Proxy-Baustein verbinden.

21.7 RexrothHydraulicValve

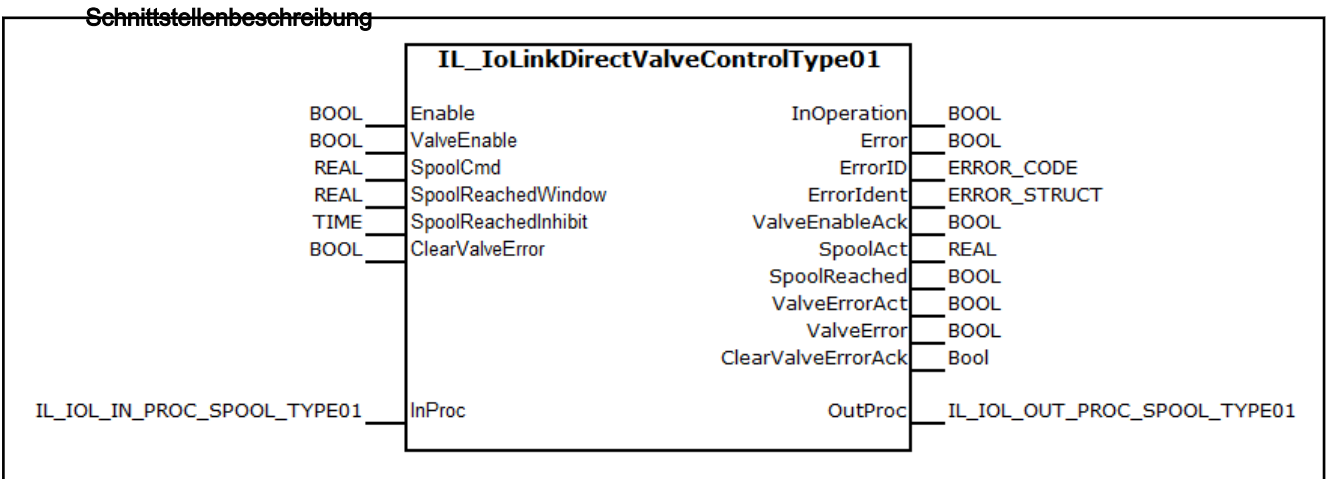
21.7.1 IL_IoLinkDirectValveControlType01

- Kurzbeschreibung

Der Funktionsbaustein IL_IoLinkDirectValveControlType01 erweitert den Funktionsbaustein IL_EnableInOpBase und dient dazu, ein IO-Link-Ventil der Firma Bosch Rexroth anzusteuern.

Zielsystem	Bibliothek
MLC/MTX	RIL_IoLink
MLD mit IndraLogic 2G ab 20VRS	RIL_IoLink

Tab. 21-13: Referenztablelle



E/A-Typ	Name	Datentyp	Beschreibung
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Funktionsbausteins (pegelgesteuert)
	ValveEnable	BOOL	Freigabe des Ventils
	SpoolCmd	REAL	Ventilsollwert. Einheit in %. Der Arbeitsbereich ist +/-100%
	SpoolReachedWindow	REAL	Zielbereich um Sollwert. Einheit in %
	SpoolReachedInhibit	TIME	Zeit in Zielbereich bevor SpoolReached gesetzt wird
	ClearValveError	BOOL	Ventilfehler löschen
	InProc	IL_IOL_IN_PROC_SPOOL_TYPE01	Prozess-Eingangsdaten %I anschließen IL_IOL_IN_PROC_SPOOL_TYPE01, Seite 917
VAR_OUTPUT	InOperation	BOOL	Funktionsbaustein aktiv
	Error	BOOL	Bearbeitung mit Fehler abgeschlossen
	ErrorID	ERROR_CODE	Beschreibung der Diagnose im Fehlerfall
	ErrorIdent	ERROR_STRUCT	Detaildiagnose Fehlerbeschreibung, Seite 916
	ValveEnableAck	BOOL	Ventilfreigabe quittiert
	SpoolAct	REAL	Ventilistwert. Einheit in %
	SpoolReached	BOOL	Ventilsollwert = Ventilistwert +/- SpoolReachedWindow
	ValveErrorAct	BOOL	Ventilfehler liegt aktuell an. Weitere Informationen zur Fehlerursache können aus dem IO-Link-Gerät azyklisch ausgelesen werden. Funktionsbausteine zur azyklischen Kommunikation sind in der Bibliothek zum verwendeten Feldbus zu finden. Bei z.B. Profinet siehe Bibliothek RIL_ProfinetIO, Ordner IO-Link.
	ValveError	BOOL	Ein Ventilfehler wurde erkannt (nach dem letzten Ventilfehler löschen)
	ClearValveErrorAck	BOOL	Ventilfehler löschen quittiert
	OutProc	IL_IOL_OUT_PROC_SPOOL_TYPE01	Prozess-Ausgangsdaten %Q anschließen IL_IOL_OUT_PROC_SPOOL_TYPE01, Seite 917

Tab. 21-14: Schnittstelle IL_IoLinkDirectValveControlType01

Min-/Max und Default-Werte

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
Enable	BOOL			FALSE	Kontinuierlich
ValveEnable	BOOL			FALSE	Kontinuierlich
SpoolCmd	REAL	-110	+110	0%	Kontinuierlich

Name	Typ	Min.-Wert	Max.-Wert	Default-Wert	Übernahme
SpoolReachedWindow	REAL	0	50	3%	Kontinuierlich
SpoolReachedInhibit	TIME			TIME#4 ms	Kontinuierlich
ClearValveError	BOOL			FALSE	Kontinuierlich
InProc	IL_IOL_IN_PROC_SPOOL_TYPE01			0	Kontinuierlich

Funktionsbeschreibung Der Funktionsbaustein sendet und empfängt vom Ventil IL_IOL_IN_PROC_SPOOL_TYPE01 bzw. IL_IOL_OUT_PROC_SPOOL_TYPE01. Darin enthalten ist jeweils ein 14 Bit Soll- bzw. Istwert des Ventils, der in einen Prozentwert umgerechnet wird. Zusätzlich werden Fehlermeldungen des Ventils, der Befehl zum Aktivieren des Ventils, sowie eine Bestätigung dieser Aktivierung übertragen.

Der Eingang "SpoolCmd" gibt den Ventilsollwert an. Der Wert wird zyklisch über den Ausgang "OutProc.PDO" an das Ventil übergeben.

Der Eingang "SpoolReachedWindow" gibt ein prozentuales Fenster um den Sollwert an. Wenn sich der Istwert in diesem Bereich befindet, gilt der Sollwert als erreicht. Der Prozentwert bezieht sich auf +100%.

Der Eingang "SpoolReachedInhibit" gibt eine Zeit an, die der Ventilistwert mindestens im angegebenen Bereich um den Sollwert (siehe "SpoolReachedWindow") sein soll, bevor der Ausgang SpoolReached aktiv wird. Wenn der Ventilistwert den Bereich um den Sollwert verlässt, wird der Ausgang "SpoolReached" sofort deaktiviert, ohne die eingestellte Zeit abzuwarten.

In der folgenden Abbildung ist für "SpoolReachedWindow" ein Wert von 5 % und für "SpoolReachedInhibit" eine Zeit von 4 ms eingestellt.

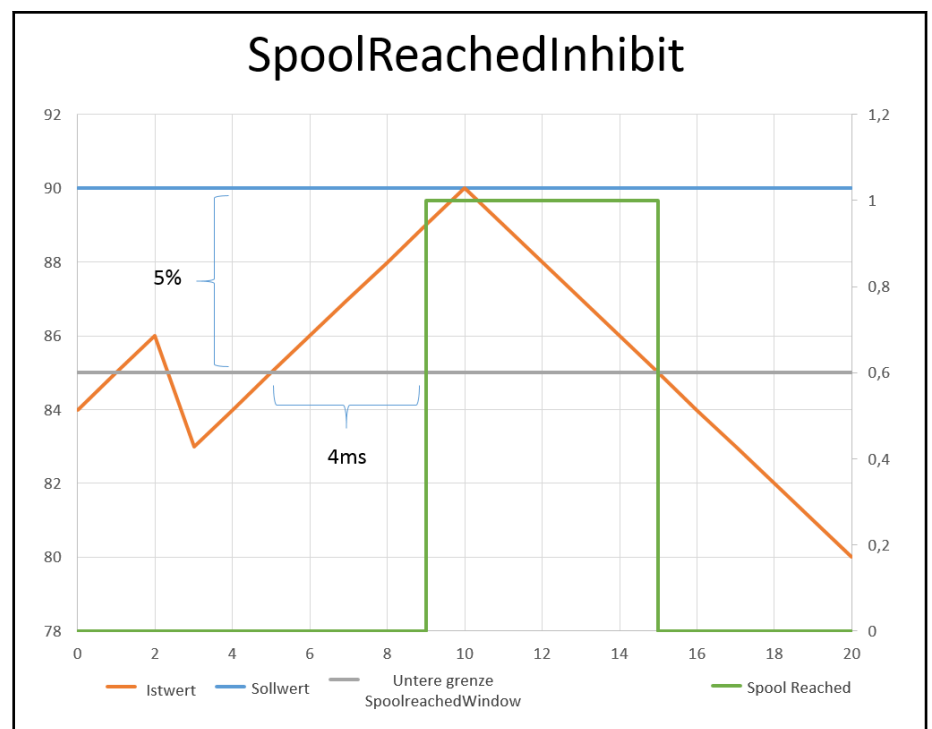


Abb. 21-7: SpoolReachedInhibit

- Der Ausgang "SpoolReached" wird aktiviert, wenn der Ventilistwert den Ventilsollwert erreicht.
- Der Ausgang "SpoolAct" gibt den Ventilistwert aus. Der Wert wird zyklisch über den "InProc.PDI" vom Ventil gelesen.

Konfiguration der zyklischen Daten am Beispiel Profinet:

Um die zyklische Kommunikation des Funktionsbausteins mit dem Ventil aufzubauen, müssen Parameter auf die jeweiligen Ein- und Ausgangsvariablen kopiert werden. Die Konfiguration der zyklisch übertragenen Parameter erfolgt im Projektextplorer unterhalb des PROFINET_IO_Controller-Knotens. Dazu muss unterhalb des gewählten Knotens das entsprechende IO-Link Gateway hinzugefügt werden. Ist das Gateway angeschlossen und die Gerätebeschreibungsdatei installiert, kann dies über Rechtsklick auf den Knoten und "Gerät suchen" bzw. "Buskonfiguration scannen" geschehen (installieren der Gerätebeschreibungsdatei über **Extras ▶ Gerätedatenbank**).

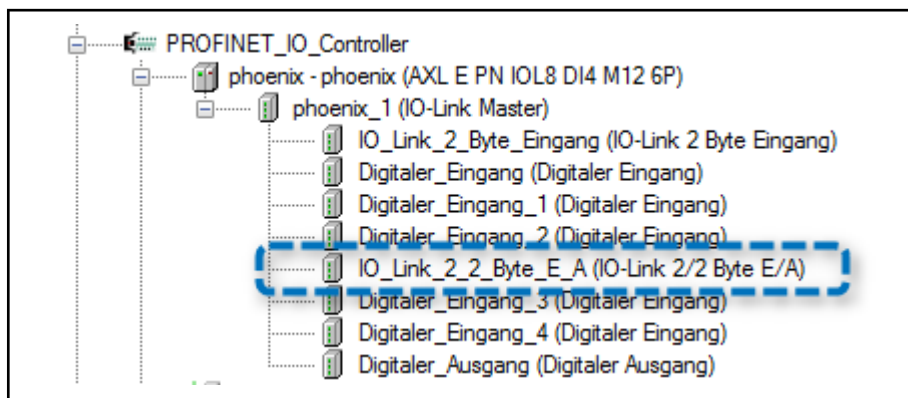


Abb. 21-8: Konfiguration der zyklischen Daten (Beispiel Profinet IO <> IO-Link Gateway)

Anschließend müssen IO-Link-Teilnehmer mit der entsprechenden Datenbreite hinzugefügt werden. Diesen Teilnehmern können im "E/A-Abbild" die Ausgänge "InProc.PDI" und "OutProc.PDO" übergeben werden.

IO_Link_2_2_Byte_E_A					
Allgemein PNIO SubModule Parameter E/A-Abbild Status Information					
Suchen Filter Alle anzeigen					
Variable	Mapping	Kanal	Adresse	Typ	
Application.PlcProg.fb_IoLinkValveControl.InProc.PDI		2 Byte Eingang	%IB7	ARRAY [0..1] OF BYTE	
		2 Byte Eingang[0]	%IB7	BYTE	
		2 Byte Eingang[1]	%IB8	BYTE	
Application.PlcProg.fb_IoLinkValveControl.OutProc.PDO		2 Byte Ausgang	%QB5	ARRAY [0..1] OF BYTE	
		2 Byte Ausgang[0]	%QB5	BYTE	
		2 Byte Ausgang[1]	%QB6	BYTE	

Abb. 21-9: Mappen der Ein- und Ausgangsvariablen (Beispiel Profinet IO <> IO-Link Gateway)

Fehlerbeschreibung Spezifische Fehlercodes des Funktionsbausteins:

ErrorID	Additional1	Additional2	Beschreibung
INPUT_INVALID_ERROR	16#00000002	16#00000001	Eingang "SpoolCmd" außerhalb des zulässigen Bereichs
INPUT_INVALID_ERROR	16#00000002	16#00000002	Eingang "SpoolReachedWindow" außerhalb des zulässigen Bereichs

Zum Löschen eines Fehlers die Fehlerursache beseitigen und den Eingang Enable auf FALSE und danach wieder auf TRUE setzen.

21.7.2 IL_IOL_OUT_PROC_SPOOL_TYPE01

Kurzbeschreibung Die Datenstruktur IL_IOL_OUT_PROC_SPOOL_TYPE01 enthält die Prozess-Ausgangsdaten des Funktionsbausteins [IL_IoLinkDirectValveControlType01](#) auf Seite 913.

Strukturelement	Typ	Default-Wert	Beschreibung
PDO	WORD		Ausgangsvariable

Tab. 21-15: Elemente der Struktur IL_IOL_OUT_PROC_SPOOL_TYPE01

Ausgangsprozessdaten PDO - gesamte Bitlänge = 16

Name	Beschreibung	Datentyp	Bitoffset	Bitlänge	Wertebereich
Ventilsollwert	Aktueller Ventil-sollwert	IntegerT	2	14	-110% = 0x8000 -100% = 0x8BB0 -0,0134% = 0xFFFC 0% = 0 +0,0134% = 0x0004 +100% = 0x7450 +110% = 0x7FFC
nicht genutzt			1	1	
Freigabe	High -> Endstufe aktiv Low -> Endstufe nicht aktiv	BooleanT	0	1	(False) inaktiv (True) aktiv

Tab. 21-16: Aufbau der Ausgangsprozessdaten PDO

21.7.3 IL_IOL_IN_PROC_SPOOL_TYPE01

Kurzbeschreibung Die Datenstruktur IL_IOL_IN_PROC_SPOOL_TYPE01 enthält die Prozess-Eingangsdaten des Funktionsbausteins [IL_IoLinkDirectValveControlType01](#) auf Seite 913.

Strukturelement	Typ	Default-Wert	Beschreibung
PDI	WORD		Eingangsvariable

Tab. 21-17: Elemente der Struktur IL_IOL_IN_PROC_SPOOL_TYPE01

Eingangsprozessdaten PDI - gesamte Bitlänge = 16

Name	Beschreibung	Datentyp	Bitoffset	Bitlänge	Wertebereich
Ventilistwert	Aktueller Ventilistwert	IntegerT	2	14	-110% = 0x8000 -100% = 0x8BB0 -0,0134% = 0xFFFFC 0% = 0 +0,0134% = 0x0004 +100% = 0x7450 +110% = 0x7FFC
Ventil ok	High -> kein Fehler Low -> Fehler	BooleanT	1	1	(False) inaktiv (True) aktiv
Ventil ein	High -> Endstufe aktiv Low -> Endstufe nicht aktiv	BooleanT	0	1	(False) inaktiv (True) aktiv

Tab. 21-18: Aufbau der Eingangsprozessdaten PDI

22 Modbus RTU - Bibliotheken

22.1 IoDrvModbusComPort

Kurzbeschreibung **Bibliothek:** IoDrvModbus

Dieser Baustein wird erzeugt für jeden seriellen Modbus-Master im Gerätebaum.

Schnittstellenbeschreibung

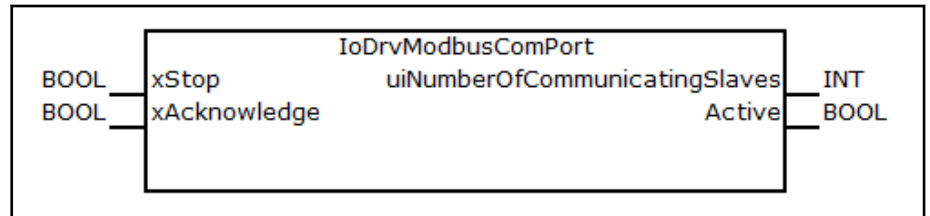


Abb. 22-1: Funktionsbaustein IoDrvModbusComPort

E/A-Typ	Name	Datentyp	Initialwert	Beschreibung
VAR_INPUT	xStop	BOOL		Wenn TRUE, wird jede neue Anfrage zu den Slaves gestoppt. Bei FALSE wird fortgefahren
	xAcknowledge	BOOL		Schließt und Öffnet sofort wieder den COM-Port bei einer steigenden Flanke
VAR_OUTPUT	uiNumberOfCommunica- tingSlaves	INT	0	Bearbeitung ohne Fehler abgeschlossen, Ausgangsdaten sind gültig
	Active	BOOL	FALSE	Bearbeitung läuft, Ausgangsdaten sind ungültig

Tab. 22-1: Schnittstelle IoDrvModbusComPort

22.2 ModbusSlaveComPort

Kurzbeschreibung **Bibliothek:** IoDrvModbus

Dieser Baustein wird erzeugt für jeden seriellen Modbus-Slave im Gerätebaum. Dieser Funktionsbaustein repräsentiert die logische Verbindung zu einem einzelnen Modbus Serial Slave. Im Hintergrund werden die für dieses Gerät definierten Kanäle abgearbeitet. Die Ein-/Ausgänge dieses Bausteins geben dem Anwender dazu Statusinformationen und bieten zum Beispiel die Möglichkeit, aufgetretene Fehler zu bestätigen (um mit der Abarbeitung fortzufahren).

Schnittstellenbeschreibung

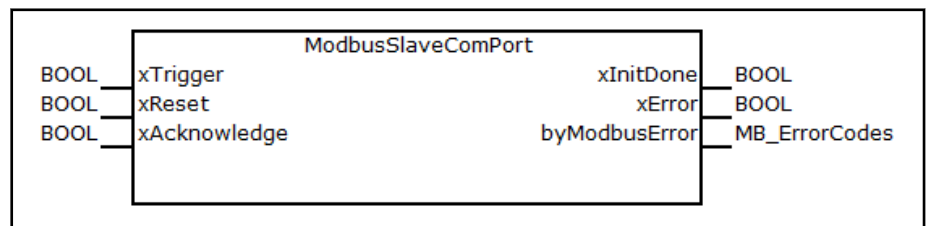


Abb. 22-2: ModbusSlaveComPort

E/A-Typ	Name	Datentyp	Initialwert	Beschreibung
VAR_INPUT	xTrigger	BOOL	FALSE	Triggert bei einer steigender Flanke das Senden aller konfigurierten Modbus-Kanäle
	xReset	BOOL	FALSE	Startet die Kommunikation neu und setzt xError und byModbusError zurück

E/A-Typ	Name	Datentyp	Initialwert	Beschreibung
	xAcknowledge	BOOL	FALSE	Startet die Kommunikation neu und setzt xError und byModbusError nicht zurück
VAR_OUTPUT	xInitDone	BOOL	FALSE	Alle Initialisierungsbefehle wurden zurückgesetzt
	xError	BOOL	FALSE	Ist TRUE, wenn irgend ein Fehler auftritt. Die Kommunikation zu diesem Slave ist unterbrochen
	byModbusError	MB_Error-Codes	0	Aktueller Fehler, definiert in der Enumeration MB_ErrorCodes

Tab. 22-2: ModbusSlaveComPort

22.3 IoDrvModbusTCP

Kurzbeschreibung **Bibliothek:** IoDrvModbusTCP

Dieser Baustein wird erzeugt für jeden Modbus-Master TCP im Gerätebaum.

Schnittstellenbeschreibung

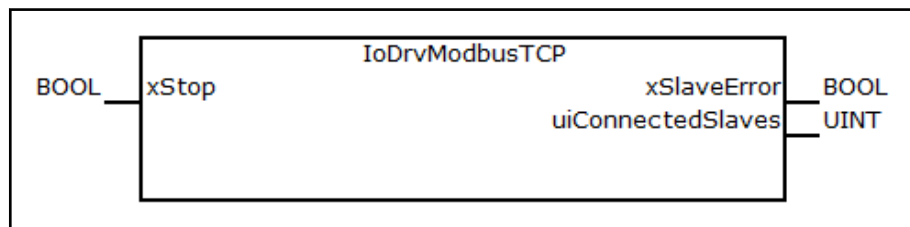


Abb. 22-3: Funktionsbaustein IoDrvModbusTCP

E/A-Typ	Name	Datentyp	Initialwert	Beschreibung
VAR_INPUT	xStop	BOOL	FALSE	Stoppt alle TCP/IP-Kommunikation
VAR_OUTPUT	xSlaveError	BOOL	FALSE	Signalisiert, dass es einen Slave-FB gibt, bei dem ein Fehler aufgetreten ist
	uiConnectedSlaves	UINT	0	Anzahl der Slaves, zu denen eine TCP/IP-Verbindung steht

Tab. 22-3: Schnittstelle IoDrvModbusTCP

22.4 ModbusTCPSlave

Kurzbeschreibung **Bibliothek:** IoDrvModbusTCP

Dieser Baustein wird erzeugt für jeden Modbus-Slave TCP im Gerätebaum. Dieser Funktionsbaustein repräsentiert eine TCP/IP-Verbindung zu einem einzelnen ModbusTCP-Slave. Im Hintergrund werden die für dieses Gerät definierten Kanäle abgearbeitet. Die Ein/Ausgänge dieses Bausteins geben dem Anwender dazu Statusinformationen und bieten zum Beispiel die Möglichkeit, aufgetretene Fehler zu bestätigen (um mit der Abarbeitung fortzufahren). Über die Property xEnable kann die Kommunaktion gestartet oder gestoppt werden.

Schnittstellenbeschreibung

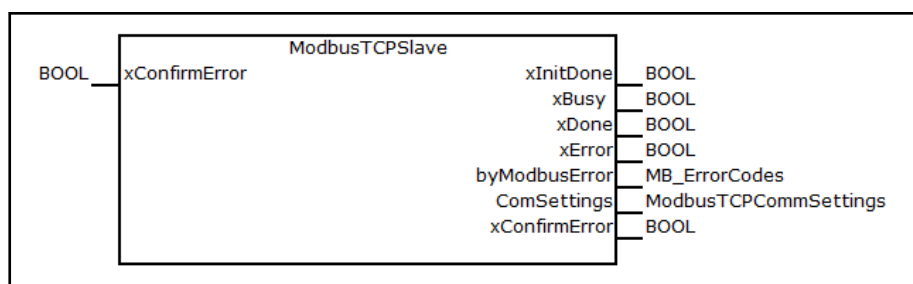


Abb. 22-4: Funktionsbaustein ModbusTCPSlave

E/A-Typ	Name	Datentyp	Initialwert	Beschreibung
VAR_INPUT	xConfirmError	BOOL	FALSE	Der Prozess wird fortgeführt mit einer steigenden Flanke
VAR_OUTPUT	xInitDone	BOOL	FALSE	Zeigt die Anzahl von Kommunikationsslaves
	xBusy	BOOL	FALSE	Ist TRUE, solange eine Anfrage in Bearbeitung ist
	xDone	BOOL	FALSE	Ist TRUE, wenn eine Anfrage erfolgreich beendet wurde
	xError	BOOL	FALSE	Ist TRUE, wenn eine Anfrage mit Fehler beendet wurde
	byModbusError	MB_ErrorCodes		Spezifiziert den aktuellen Fehler, wie in der Enumeration MB_ErrorCodes definiert
	ComSettings	ModbusTCPCommSettings		Aktuell konfigurierte Kommunikationseinstellungen
	xConfirmError	BOOL	FALSE	Der Prozess wird fortgeführt mit einer steigenden Flanke

Tab. 22-4: ModbusTCPSlave

Methode "UpdateCommunicationSettings"

Die Kommunikationsparameter können nur aktualisiert werden, wenn der Slave auf "Stop" steht. Das Stoppen/Starten des Slaves erfolgt über die Eigenschaft xEnable.

E/A-Typ	Name	Datentyp	Initialwert	Beschreibung
VAR_INPUT	ipAdress	ARRAY [0..3] OF BYTE		IP-Adresse des Slaves
	uiPort	UINT		Port des Slaves
VAR_OUTPUT	UpdateCommunicationSettings	UDINT		Führt zu einem Fehler (Rückgabewert != 0), wenn eine TCP-Verbindung besteht

22.5 IoDrvModbusSerialSlave

Kurzbeschreibung

Bibliothek: IoDrvModbusSerialSlave

Dieser Baustein wird erzeugt für jeden seriellen Modbus-Slave-Device im Gerätebaum.

Schnittstellenbeschreibung

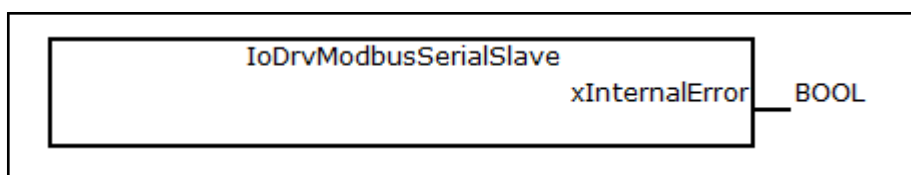


Abb. 22-5: Funktionsbaustein IoDrvModbusSerialSlave

E/A-Typ	Name	Datentyp	Initialwert	Beschreibung
VAR_OUTPUT	xInternalError	BOOL	FALSE	Nicht behebbarer interner Fehler (z.B. wenn mem-alloc fehlschlägt oder keine Lizenz vorhanden ist)

Tab. 22-5: Schnittstelle IoDrvModbusSerialSlave

22.6 IoDrvModbusTCPSlave

Kurzbeschreibung **Bibliothek:** IoDrvModbusTCPSlave

Dieser Baustein wird erzeugt für jedes Modbus TCP Slave Device im Gerätebaum.

Schnittstellenbeschreibung

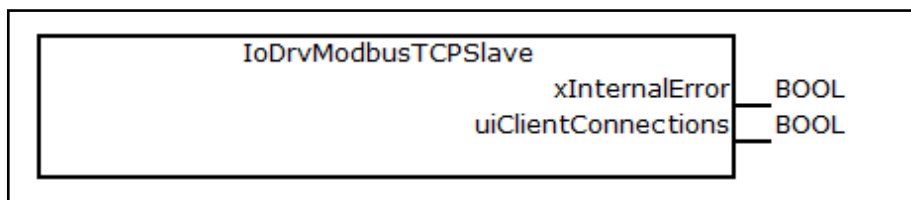


Abb. 22-6: Funktionsbaustein IoDrvModbusTCPSlave

E/A-Typ	Name	Datentyp	Initialwert	Beschreibung
VAR_OUTPUT	xInternalError	BOOL	FALSE	Nicht behebbarer interner Fehler (z.B. wenn mem-alloc fehlschlägt oder keine Lizenz vorhanden ist)
	uiClientConnections	BOOL	FALSE	Anzahl der aktuell aufgebauten Client-Verbindungen (TCP/IP)

Tab. 22-6: Schnittstelle IoDrvModbusTCPSlave

22.7 ModbusServer

Kurzbeschreibung **Bibliothek:** IoDrvModbusTCPSlave

Der Funktionsbaustein verarbeitet Anfragen von Modbus TCP, wodurch bis zu 10 parallele Client-Verbindungen verarbeitet werden können. Modbus-Anfragen arbeiten auf Datenbereichen, die die Input-Register und die Holding-Register abbilden.

Die folgenden Modbus-Funktionscodes werden unterstützt:

- 0x01 Read Coils
- 0x02 Read Discrete Inputs
- 0x03 Read Holding Registers
- 0x04 Read Input Registers
- 0x05 Write Single Coil
- 0x06 Write Single Register
- 0x0F Write Multiple Coils

- 0x10 Write Multiple Registers
- 0x17 Read/Write Multiple Registers

Der ModbusServer-Funktionsbaustein benutzt die Parameter, die in der Registerkarte "Geräteeditor" des Modbus TCP-Slaves konfiguriert wurden.

Schnittstellenbeschreibung

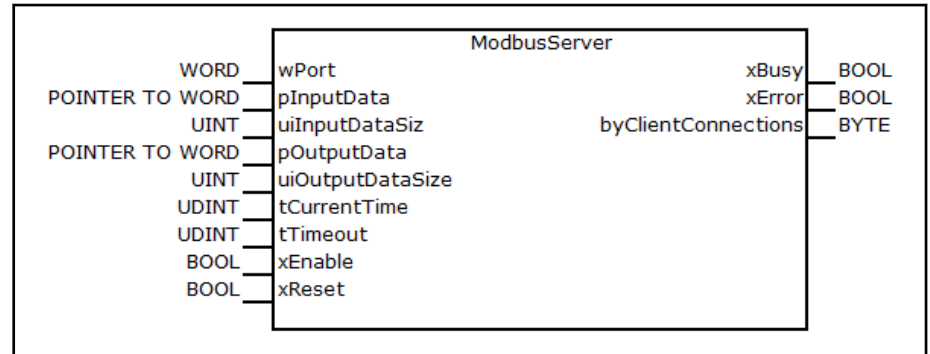


Abb. 22-7: Funktionsbaustein ModbusServer

E/A-Typ	Name	Datentyp	Initialwert	Beschreibung
VAR_INPUT	wPort	WORD	502	Server-Port (TCP/IP-Protokollstack) 502 ist Modbus-Standard
	pInputData	POINTER TO WORD		Pointer auf WORD, zeigt auf das Abbild des Modbus-Eingangsregisters
	uiInputDataSiz	UINT	1024	Länge des Arrays der Eingangsdaten (WORDS pro Register): Standard: 1024
	pOutputData	POINTER TO WORD		Pointer auf WORD, zeigt auf das Abbild des Modbus-Holding-Registers
	uiOutputDataSize	UINT		Länge der Arrays der Ausgangsdaten (WORDS pro Register)
	tCurrentTime	UDINT		Aktuelle SPS-Zeit in ms. Falls ein Timeout konfiguriert wurde, muss sie aktualisiert worden sein, bevor der Funktionsbaustein ausgeführt wird
	tTimeout	UDINT		Zeitüberschreitung in ms. Falls sie nicht 0 ist und kein Modbus-Schreibfehler bis zum Timeout aufgetreten ist, werden die Ausgangsdaten auf 0 gesetzt
	xEnable	BOOL	FALSE	Flag zum Starten der Abarbeitung der Anfrage
	xReset	BOOL	FALSE	Neustart der normalen Kommunikation mit den Slaves und Rücksetzen der Status-Words und des Fehlerbits bei steigender Flanke
VAR_OUTPUT	xBusy	BOOL	FALSE	Ist TRUE, solange die Anfrage in Bearbeitung ist
	xError	BOOL	FALSE	Information über gefundenen Fehler
	byClientConnections	BYTE	0	Anzahl der Verbindungen

Tab. 22-7: ModbusServer

22.8 ModbusRequest

Kurzbeschreibung **Bibliothek:** IoDrvModbusTCP

Mit diesem Funktionsbaustein können Sie, zusätzlich zu den in der Registerkarte "Modbus Slave-Kanal" vordefinierten Modbus-Kommandos, in der Applikation dynamisch weitere Kommandos ausführen. Für diese programmatisch erzeugten Kommandos gibt es allerdings kein I/O-Mapping sondern nur Ein-/Ausgangsdatenbereiche in Form von ARRAY OF BYTE bzw. ARRAY OF WORD.

Schnittstellenbeschreibung

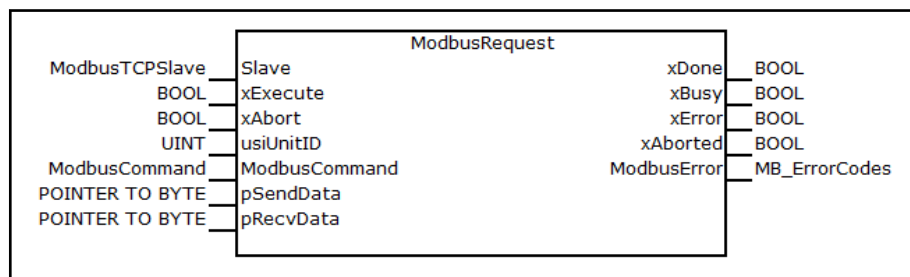


Abb. 22-8: Funktionsbaustein ModbusRequest

E/A-Typ	Name	Datentyp	Initialwert	Beschreibung
VAR_INPUT	Slave	ModbusTCPSlave		
	xExecute	BOOL		Steigende Flanke: Die Aktion startet Fallende Flanke: Die Ausgänge werden zurückgesetzt wenn eine fallende Flanke auftritt, bevor der Funktionsbaustein die Aktion beendet hat. Die Ausgänge verhalten sich wie üblich und werden nur zurückgesetzt, wenn die Aktion beendet ist oder abgebrochen wurde (xAbort) oder im Falle eines Fehlers. In diesem Fall stehen die entsprechenden Ausgabewerte (xDone, xError, xError) für genau einen Zyklus an den Ausgängen an
	xAbort	BOOL		TRUE: Die Aktion wird sofort gestoppt und alle Ausgänge werden auf ihren Initialisierungswert gesetzt
	usiUnitID	UINT	15#FF	Die Unit-ID wird nur für die Weiterleitung (Gateway) auf eine Modbus Serial Bus verwendet. Bei TCP/IP wird der Modbus-Server über seine IP-Adresse adressiert. Modbus Unit-ID wird in diesem Fall nicht verwendet (Wert 0xFF muss verwendet werden). Bei serieller Verbindung werden die Modbus-Geräte mit Unit-IDs von 1 bis 247 adressiert, 0 wird für Broadcast-Adressen verwendet
	ModbusCommand	Modbus-Command		Der Modbus-Befehl, der ausgeführt wird. Funktionsbausteine für Bit- und Word-Zugriff werden unterstützt

E/A-Typ	Name	Datentyp	Initialwert	Beschreibung
	pSendData	POINTER TO BYTE		Pointer auf einen Puffer mit den Daten, die zum Slave gesendet werden. Für den Registerzugriff ist der Puffer vom Typ <code>Array[0..ModbusCommand.uiWriteLen - 1] OF WORD</code> . Für den Bit-Zugriff ist der Puffer vom Typ <code>ARRAY[0..ModbusCommand.uiWriteLen / 8] OF BYTE</code>
	pRecvData	POINTER TO BYTE		Pointer auf einen Puffer, in dem die Daten, die vom Slave empfangen wurden, geschrieben werden. Für den Registerzugriff ist der Puffer vom Typ <code>ARRAY[0..ModbusCommand.uiReadLen - 1] OF WORD</code> . Für den Bit-Zugriff ist der Puffer vom Typ <code>ARRAY[0..ModbusCommand.uiReadLen / 8] OF BYTE</code>
VAR_OUTPUT	xDone	BOOL	FALSE	Ist TRUE, wenn eine Anfrage erfolgreich beendet wurde
	xBusy	BOOL	FALSE	Ist TRUE, solange die Anfrage in Bearbeitung ist
	xError	BOOL	FALSE	Ist TRUE, wenn eine Anfrage mit Fehler beendet wurde
	xAborted	BOOL	FALSE	Ist TRUE, wenn die Anfrage durch den Benutzer abgebrochen wurde
	ModbusError	MB_Error- Codes		Aktueller Fehler, definiert in der Enumeration <code>MB_ErrorCodes</code>

Tab. 22-8: ModbusRequest

23 RIH_IndraControl_S20.library

23.1 Einführung und Überblick

Die Bibliothek **RIH_IndraControl_S20** beschreibt die Zugriffsmöglichkeiten des IEC-Programms auf das IndraControl S20-System mithilfe von Funktionsbausteinen.

Das IndraControl S20-System findet Verwendung bei IndraControl XM-Steuerungen, IndraControl S20-Buskopplern und IndraControl S20-Modulen.

Verfügbare Funktionen für IndraControl XM-Steuerung + IndraControl S20 Buskoppler mit IndraControl S20 Module:

- Zugriff auf PDI-Objekte
- Diagnose
- Typenschild
- Normalisieren von Eingangsdaten/Ausgangsdaten
- Konfiguration

IndraControl S20-Station: Beispiel mit Buskoppler

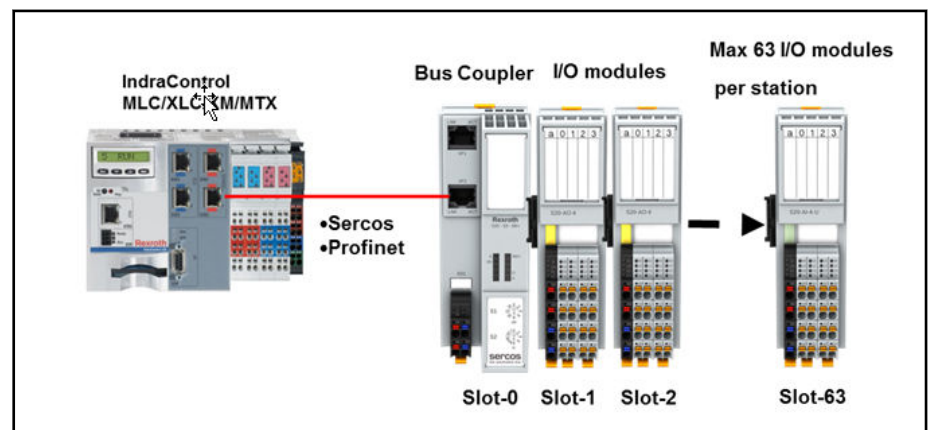


Abb. 23-1: IndraControl S20-Station: Beispiel mit Buskoppler

Die Bibliothek ist nutzbar für folgende Kategorien/Module:

Kategorie/Modul	Funktionsbaustein	Beschreibung
Diagnose (Diagnostics)		
IndraControl XM	IH_S20IOState, Seite 932	Gibt den Zustand des lokalen S20-IO aus
IndraControl XM	IH_S20IOStateDetails, Seite 932	Gibt einen detaillierten Zustand des lokalen S20-IO aus, zum identifizieren der Fehlerstelle und Ursache
IndraControl S20 Module an einer IndraControl XM	IH_S20IOModuleStateDetails, Seite 931	Gibt einen detaillierten Zustand des lokal an einer IndraControl S20 angeschlossenen S20-Moduls aus, zum Identifizieren der Fehlerstelle und Ursache
IndraControl S20 Buskoppler und Module	IH_S20ModuleDiagnostics	Gibt die Diagnose eines S20-Moduls aus, angeschlossen an einen IndraControl S20-Buskoppler (PDI-Objekt 0x18 wird gelesen)
I&M (Identification and Maintenance) (IM Data)		

Kategorie/Modul	Funktionsbaustein	Beschreibung
IndraControl S20 Buskoppler und Module	IH_S20ModuleIdentDetails, Seite 933	Typenschild lesen und schreiben vom "Tag Location"
Azyklische Kommunikation (Acyclic Communication)		
IndraControl S20 Buskoppler und Module	IH_S20ComInterface, Seite 937	Lesender und schreibender Zugriff auf die PDI-Objekte der IndraControl S20-Module; Übersicht der unterstützten PDI-Objekte ist in den Modul-Datenblättern zu finden. Dient als Basis für die anderen Funktionsbausteine.
IndraControl XM mit IndraControl S20 Buskoppler und Module	IH_S20ComConfiguration, Seite 934	Generiert ein Kommunikations- Handle für die azyklischen *Type02 Funktionsbausteine. Je Busmaster (IndraControl XM oder IndraControl S20 Buskoppler) wird eine neue Instanz benötigt, nicht für jedes IndraControl S20-Modul
IndraControl XM mit IndraControl S20 Buskoppler und Module	IH_S20ComInterfaceType02, Seite 935	Lesender und schreibender Zugriff auf die PDI-Objekte der IndraControl S20-Module; Übersicht der unterstützten PDI-Objekte ist in den Modul-Datenblättern zu finden. Dient als Basis für die anderen Funktionsbausteine
IndraControl XM mit IndraControl S20 Modulen	IH_S20IOPDIRead	Lesender Zugriff auf die PDI-Objekte der IndraControl S20-Module; Übersicht der unterstützten PDI-Objekte ist in den Modul-Datenblättern zu finden. Dient als Basis für die anderen Funktionsbausteine
IndraControl XM mit IndraControl S20 Modulen	IH_S20IOPDIWrite	Schreibender Zugriff auf die PDI-Objekte der IndraControl S20-Module; Übersicht der unterstützten PDI-Objekte ist in den Modul-Datenblättern zu finden. Dient als Basis für die anderen Funktionsbausteine
Analoge Eingangsmodule (AnalogIO)		
S20-AI-4-I	IH_S20AI4IU, Seite 942 IH_S20AI4IUType02	Werte normalisieren *Type02 inkl. Konfiguration
S20-AI-4-U	IH_S20AI4IU, Seite 942 IH_S20AI4IUType02	Werte normalisieren *Type02 inkl. Konfiguration
S20-AI-8	IH_S20AI8, Seite 946 IH_S20AI8Type02	Werte normalisieren *Type02 inkl. Konfiguration
S20-AI-4-RTD	IH_S20AI4RTD, Seite 948 IH_S20AI4RTDType02	Werte normalisieren *Type02 inkl. Konfiguration

Kategorie/Modul	Funktionsbaustein	Beschreibung
S20-AI-4-UTH	IH_S20AI4UTH, Seite 950 IH_S20AI4UTHType02	Werte normalisieren *Type02 inkl. Konfiguration
S20-AI-8-RTD	IH_S20AI8RTD, Seite 952 IH_S20AI8RTDType02	Werte normalisieren *Type02 inkl. Konfiguration
S20-AI-8-UTH	IH_S20AI8UTH, Seite 954 IH_S20AI8UTHType02	Werte normalisieren *Type02 inkl. Konfiguration
Analoge Ausgangsmodule (AnalogIO)		
S20-AO-4	IH_S20AO4, Seite 958 IH_S20AO4Type02	Werte normalisieren *Type02 inkl. Konfiguration
S20-AO-8	IH_S20AO8, Seite 956 IH_S20AO8Type02	Werte normalisieren *Type02 inkl. Konfiguration
Analoge Ein-/Ausgangsmodule (AnalogIO)		
S20-AIAO-2		In Vorbereitung
Technologiemodule (TechnologyIO)		
S20-AI6-AO2-SSI2	IH_S20Ai6Ao2Ssi2	Werte normalisieren
S20-CNT-INC-2/2	IH_S20CntInc, Seite 965	Werte normalisieren
S20-SSI-AO-1/1	IH_S20SsiAo1, Seite 961	Werte normalisieren
S20-RS-UNI	IH_S20RSCom, Seite 970 IH_S20RSComType02	Senden und Empfangen von seriellen Daten *Type02 inkl. Konfiguration

Tab. 23-1: RIH_IndraControl_S20-Bibliothek Funktionsbausteine

Die Bibliothek unterstützend folgende Funktionen für die Funktionsbausteine *Type02:

Funktionen für analoge Module	Beschreibung
IH_S20AIConfig	Die Funktion unterstützt beim Generieren der Konfigurationsdaten für Funktionsbausteine mit analogen Eingängen
IH_S20AOConfig	Die Funktion unterstützt beim Generieren der Konfigurationsdaten für Funktionsbausteine mit analogen Ausgängen
IH_S20TEMPConfig	Die Funktion unterstützt beim Generieren der Konfigurationsdaten für Funktionsbausteine mit RTD
IH_S20UTHConfig	Die Funktion unterstützt beim Generieren der Konfigurationsdaten für Funktionsbausteine mit UTH

Tab. 23-2: RIH_IndraControl_S20-Bibliothek Funktionen

Mit der Bibliothek ist es möglich, azyklisch, auf S20-Module via PDI-Objekte zuzugreifen. Die PDI-Objekte sind dem jeweiligen Datenblatt des S20-Modul zu entnehmen. Ein S20-Modul wird über den Eingang "Slot" adressiert.

Die Bibliothek unterstützt Sercos, Profinet, Profibus, Ethernet/IP, Ethernet (Modbus TCP) und S20-IO Lokal (aktuell getestet mit XM S20-IO Lokal, Sercos, Profinet und Profibus).



- Die Konfiguration des IndraControl S20-System erfolgt mit IndraWorks. Die Funktionsbausteine **ohne** "Suffix *Type02" lesen die Konfiguration der IndraControl S20-Module an einem IndraControl S20 Buskoppler.
- Die Funktionsbausteine **mit** "Suffix *Type02" (ab 14V22) funktionieren zusätzlich mit der IndraControl XM am Lokalbus und bieten die Möglichkeit, die Konfiguration aus der SPS in die IndraControl S20 Module zu schreiben. Es wird das PDI-Objekt 0x80 ParamTable beschrieben. Vor dem Schreiben wird die Konfiguration gelesen, zum Vermeiden von unnötigen Schreibvorgängen.

Bekanntes Problem:

Ein Feldbusanlauf überschreibt aktuell immer wieder die Konfiguration mit der Konfiguration aus IndraWorks.

Mit Feldbusanlauf ist z.B. gemeint:

- SPS Programm Download
- Profibus-/Profinet-Kabel am Buskoppler ziehen und wieder aufstecken
- Sercos Phase schalten

Workaround:

Parameterdaten aus den Gerätebeschreibungsdateien entfernen. Dann erfolgt die Konfiguration ausschließlich über die Funktionsbausteine. Das Prozessdatenabbild erfolgt weiterhin in IndraWorks.

Dokumentation

Die Dokumentation zu jedem einzelnen Funktionsbaustein finden Sie im Reiter "Dokumentation" innerhalb des IndraWorks-Bibliotheksmanager bei der Auswahl des gewünschten Funktionsbausteins.

Der Reiter "Dokumentation" beinhaltet folgende Informationen:

- Besonderheiten zu den Funktionsbausteinen
- Strukturen der Funktionsbausteine
- Enumerationen der Funktionsbausteine



Im folgenden sind die Codebeispiele und Fehlercodes aufgelistet. Codebeispiele dienen als Hilfestellung und weitere Beispiele finden Sie innerhalb der Bibliothek im Ordner "_CodeExample"

23.2 Fehlercodes

Fehlerbehandlung

Die Funktionsbausteine der Bibliothek verwenden folgende Fehlertabellen:

- **S20_IO_TABLE** (ERROR_TABLE=16#0210)
- **SERCOS_TABLE** (ERROR_TABLE=16#0010)
- **PROFINETIO_TABLE** (ERROR_TABLE=16#0200)
- **ETHERNET_IP_TABLE** (ERROR_TABLE=16#0151)
- **ETHERNET_TABLE** (ERROR_TABLE=16#0150)
- **F_RELATED_TABLE** (ERROR_TABLE=16#0170)

23.3 Diagnose (Diagnostics)

23.3.1 IH_S20IOModuleStateDetails

Beispielprogramm Im Beispiel wird die Diagnose der lokal an der XM-Steuerung angeschlossenen S20-Module gelesen.

Je S20-Modul kann maximal eine Diagnose gelesen werden, die letzte, höchstpriori Diagnose.

Die Struktur am Eingang "DiagInfoAdr" kann größenmäßig der Anzahl der angeschlossenen S20-Module entsprechen. Damit besteht die Möglichkeit von jedem angeschlossenen S20-Modul die Diagnose zu lesen. In der Realität melden selten alle Module gleichzeitig einen Fehler, daher kann die Größe der Struktur kleiner ausfallen als die Anzahl der angeschlossenen Module.

Empfehlung:

- Bei ≤ 10 angeschlossenen S20-Modulen entspricht die Größe der Anzahl der S20-Module
- Bei ≥ 10 angeschlossenen S20-Modulen ist die Größe der Struktur zwischen 10 und 20

TRUE am Eingang "Execute" startet das Auslesen der Diagnose.



Das funktioniert nur mit einer S20-Steuerung basierend auf Lokalbuss, z.B. IndraControl XM.

Deklaration:

Programm:

```
PROGRAM ProgIH_S20ModuleDiagnostics
VAR
  fbIH_S20IOModuleStateDetails : IH_S20IOModuleStateDetails; // Function block instance
  bExecute : BOOL; // TRUE: Function is enabled
  bDone : BOOL; // TRUE: Function Block has completed its task
  bActive : BOOL; // TRUE: Function Block is working on its actual task
  bError : BOOL; // TRUE: Indicates an error
  ErrorID : ERROR_CODE; // Class of error
  ErrorIdent : ERROR_STRUCT; // Detailed information about error
  arDiagInfoAdr : ARRAY[1..20] OF IH_S20_IO_DIAGNOSTICS; // Address of Diagnostics information array.
  // Use ADR () operator. Contains Diagnostic information of the modules.
  lwListOfDiagModules : LWORD; // Bitwise representation of module slots, where diagnostic messages exist
  udiActiveDiagCount : UDINT; // Number of diagnostic messages currently active/exist
  udiNoOfDiagInfo : UDINT; // Number of diagnostic messages read and processed
  udiMoreDiagInfoCount : UDINT; // Number of diagnostic messages still exist but not yet processed
END_VAR
```

Programm:

Programm:

```
fbIH_S20IOModuleStateDetails(
  Execute:= bExecute,
  Done=> bDone,
  Active=> bActive,
  Error=> bError,
  ErrorID=> ErrorID,
  ErrorIdent=> ErrorIdent,
  DiagInfoAdr:= ADR(arDiagInfoAdr),
  SizeOfDiagInfo:= SIZEOF(arDiagInfoAdr),
  ListOfDiagModules=> lwListOfDiagModules,
  ActiveDiagCount=> udiActiveDiagCount,
  NoOfDiagInfo=> udiNoOfDiagInfo,
  MoreDiagInfoCount=> udiMoreDiagInfoCount);
```

23.3.2 IH_S20IOState

Beispiel Program Im Beispiel wird ein Status WORD zur Diagnose vom S20-IO-Master einer XM-Steuerung gelesen.

TRUE am Eingang "Enable" startet das Auslesen der Diagnose.



Das funktioniert nur mit einer S20-Steuerung basierend auf Lokalbus, z.B. IndraControl XM.

Variable Deklaration:

Programm:

```
PROGRAM ProgIH_S20IOState
VAR
    fbIH_S20IOState: IH_S20IOState; // Function block instance
    bEnable       : BOOL; // TRUE: Function is enabled
    bDone         : BOOL; // TRUE: Function Block has completed its task
    bActive       : BOOL; // TRUE: Function Block is working on its actual task
    bError        : BOOL; // TRUE: Indicates an error
    ErrorID       : ERROR_CODE; // Class of error
    ErrorIdent    : ERROR_STRUCT; // Detailed information about error
    wState        : WORD; // S20 I/O bus diagnostic status
END_VAR
```

Programm:

Programm:

```
fbIH_S20IOState(
    Enable:= bEnable,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> ErrorID,
    ErrorIdent=> ErrorIdent,
    State=> wState);
```

23.3.3 IH_S20IOStateDetails

Beispielprogramm Im Beispiel wird die Detaildiagnose vom S20-IO-Master einer XM-Steuerung gelesen.

TRUE am Eingang "Execute" startet das Auslesen der Diagnose.



Das funktioniert nur mit einer S20-Steuerung basierend auf Lokalbus, z.B. IndraControl XM.

Variable Deklaration:

Programm:

```
PROGRAM ProgIH_S20IOStateDetails
VAR
    fbIH_S20IOStateDetails : IH_S20IOStateDetails; // Function block instance
    bExecute : BOOL; // TRUE: Function is enabled
    bDone : BOOL; // TRUE: Function Block has completed its task
    bActive : BOOL; // TRUE: Function Block is working on its actual task
    bError : BOOL; // TRUE: Indicates an error
    ErrorID : ERROR_CODE; // Class of error
    ErrorIdent : ERROR_STRUCT; // Detailed information about error
    stStateDetails : IH_S20_MASTER_STATE; // S20 I/O bus diagnostics data.
    // Refer structure: IH_S20_MASTER_STATE
    stDiagGlobalStatus : IH_S20_DIAG_GLOBAL_STATUS;
    // S20 I/O system different error count information.
```



```

//Refer structure: IH_S20_DIAG_GLOBAL_STATUS
END_VAR

```

Programm:*Programm:*

```

fbIH_S20IOStateDetails(
    Execute:= bExecute,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> ErrorID,
    ErrorIdent=> ErrorIdent,
    StateDetails=> stStateDetails,
    DiagGlobalStatus=> stDiagGlobalStatus);

```

23.4 I&M (Identification and Maintenance) (IM Data)

23.4.1 IH_S20ModuleIdentDetails

Beispielprogramm

Im Beispiel wird die Detaildiagnose vom S20-IO-Master einer XM-Steuerung gelesen.

Der verwendete Feldbus ist am Eingang "BusType" mithilfe der Variable "enBusType" definiert. Die Adresse/der Name des Buskopplers wird mit "stBusConfig" definiert. Der Modul-Steckplatz wird am Eingang "Slot" definiert.

TRUE am Eingang "Execute" startet das Auslesen der Objekte zur Identifikation (Gerätetypenschild) vom S20-AI-4-I.

Deklaration:*Programm:*

```

PROGRAM ProgIH_S20ModuleIdentDetails
VAR
    fbIH_S20ModuleIdentDetails : IH_S20ModuleIdentDetails; // Function block instance
    bExecute : BOOL; // Function block processing enabled( TRUE - once, edge-controlled )
    bDone : BOOL; // TRUE: Function Block has completed its task
    bActive : BOOL; // TRUE: Function Block is working on its actual task
    bError : BOOL; // TRUE: Indicates an error
    ErrorID : ERROR_CODE; // Class of error
    ErrorIdent : ERROR_STRUCT; // Detailed information about error
    enBusType : IH_S20_BUSTYPE; // Bus type selection
    stBusConfig : IH_S20_BUS_CONFIG; // Bus configuration parameters
    uiSlot : UINT := 1; // Module slot address. Bus coupler : 0,
    // 1st Module after coupler:1 ... 63rd Module after coupler: 63
    enIdentMode : IH_S20_IDENTMODE; // Enumeration for read/write of module Identification objects
    stManufacturerDetails : IH_S20_IDENT_MANUFACTURER;
    // Reference to read Manufacturer identification objects structure
    stGeneralDetails : IH_S20_IDENT_GENERAL; // Reference to read General identification objects structure
    stSpecialDetails : IH_S20_IDENT_SPECIAL; // Reference to read Special identification objects structure
    stUseOfTheDevice : IH_S20_IDENT_USEOFDEVICE;
    // Reference to read/write Use of the device identification objects
END_VAR

```

Variableninitialisierung für Sercos-Buskoppler:

Initialisierung der Variablen vom FB für das Analogmodul, angeschlossen am Sercos Buskoppler mit Sercos Adresse 69 und Modul an Steckplatz 1.

Schritt 1: Variableninitialisierung*Programm:*

```

enBusType := IH_S20_BUSTYPE.SERCOS; // Connected fieldbus coupler
stBusConfig.Sercos.BusMaster := IL_BUSMASTER.IL_BUSMASTER_0;
stBusConfig.Sercos.SercosAdr := 69; //Buscoupler address
uiSlot := 1; // Module position

```

Schritt 2: Programm*Programm:*

```
fbIH_S20ModuleIdentDetails(
    Execute:= bExecute,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> ErrorID,
    ErrorIdent=> ErrorIdent,
    BusType:= enBusType,
    BusConfig:= stBusConfig,
    Slot:= uiSlot,
    IdentMode:= enIdentMode,
    ManufacturerDetails:= stManufacturerDetails,
    GeneralDetails:= stGeneralDetails,
    SpecialDetails:= stSpecialDetails,
    UseOfTheDevice:= stUseOfTheDevice);
```

Strukturen zur Identifikation (Gerätetypenschild) eines S20-Moduls

- stManufacturerDetails - Hersteller Information (Read only)
 - stGeneralDetails - Modul - allgemein (Read only)
 - stSpecialDetails - Modul - speziell (Read only)
 - stUseOfTheDevice - Modul - speziell (Read/Write)
-



siehe Kapitel Standardobjekte in den Datenblatt vom S20-Modul

23.5 Azyklische Kommunikation (Acyclic Communication)**23.5.1 IH_S20ComConfiguration**

Beispielprogramm Das Beispielprogramm dient zum Erstellen eines Kommunikationshandle für S20 "Type02" Funktionsbausteine mit dem Eingang **S20BusHandle**.

Der verwendete Feldbus ist am Eingang "BusType" mithilfe der Variable "en-BusType" definiert. Die Adresse bzw. der Name des Buskopplers wird mit "stBusConfig" definiert.

TRUE am Eingang "bEnable" übernimmt die Konfiguration und der Funktionsbaustein generiert das Handle.



Das **S20BusHandle** gilt für alle S20 Module/Funktionsbausteine an einen Busmaster. Ein Busmaster kann ein S20-Buskoppler oder die XMxx-Steuerung mit lokal angereichten S20-Modulen sein.

Beispiel 1:**Schritt 1: Variablendeklaration am XMxx S20 IO Lokalbus, Modul am Slot 1****Deklaration:***Programm:*

```
PROGRAM ProgIH_S20ComInterfaceType02
VAR
    fbIH_S20ComConfiguration: IH_S20ComConfiguration; // Function block instance
    bEnableComConfig: BOOL; // Function block processing enabled(TRUE - once, edge-controlled)
    bInOperationComConfig: BOOL; //TRUE: Function Block is working on its actual task
    bErrorComConfig: BOOL; //TRUE: Indicates an error
    ErrorIDComConfig: ERROR_CODE; //Class of error
    ErrorIdentComConfig: ERROR_STRUCT; //Detailed information about error
```

```

enBusType:          IH_S20_BUSTYPE; //Bus type selection
stBusConfig:        IH_S20_BUS_CONFIG; //Bus configuration parameters
dwS20BusHandle:      DWORD; //Bus handle of S20IO configuration

fbIH_S20ComInterfaceType02: IH_S20ComInterfaceType02; // Function block instance
bExecute:           BOOL; // Function block processing enabled(TRUE - once, edge-controlled)
bDone:              BOOL; //TRUE: Function Block has completed its task
bActive:            BOOL; //TRUE: Function Block is working on its actual task
bError:             BOOL; //TRUE: Indicates an error
ErrorID:            ERROR_CODE; //Class of error
ErrorIdent:         ERROR_STRUCT; //Detailed information about error
enSetOpMode:        IH_S20_OP_MODE; //FB Operating mode selection
uiSlot:             UINT;
// Module slot address. Bus coupler : 0,
// 1st Module after coupler:1 ... 63rd Module after coupler: 63
IH_S20_PARAM; //PDI object mapping to bus parameter
tTimeout:           TIME := T#3S; // Timeout delay
uiNoOfRecBytes:     UINT; // No. of received bytes
arIndexData:        ARRAY[0..50] OF BYTE;
// Address of PDI Index data values.
// Common interface for read/write operation

```

END_VAR

Initialisierung der Variablen vom Funktionsbaustein angeschlossen an XM lokal und Modul an Steckplatz 1, im Programm.

Programm:

Programm:

```

//Implementation
//=====

// Assignment of BusType to S20IO local
enBusType := IH_S20_BUSTYPE.S20IO;

fbIH_S20ComConfiguration(
    Enable:= bEnableComConfig,
    InOperation=> bInOperationComConfig,
    Error=> bErrorComConfig,
    ErrorID=> ErrorIDComConfig,
    ErrorIdent=> ErrorIdentComConfig,
    BusType:= enBusType,
    BusConfig:= stBusConfig,
    S20BusHandle=> dwS20BusHandle);

uiSlot := 1; // Slot no
enSetOpMode := IH_S20_OP_MODE.DEVICETYPE_READ; //Setting Operating mode to read DeviceType

fbIH_S20ComInterfaceType02(
    Execute:= bExecute,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> ErrorID,
    ErrorIdent=> ErrorIdent,
    S20BusHandle:= dwS20BusHandle,
    SetOpMode:= enSetOpMode,
    Slot:= uiSlot,
    PDIIndex:= stPDIIndex,
    IndexDataAdr:= ADR(arIndexData),
    SizeOfIndexData:= SIZEOF(arIndexData),
    Timeout:= tTimeout,
    NoOfRecBytes=> uiNoOfRecBytes);

```

23.5.2 IH_S20ComInterfaceType02

Beispielprogramm

Das Beispielprogramm dient zum Lesen des Device-Typs von einem S20-Modul angereicht an eine XM-Steuerung.

Der verwendete Feldbus ist am Eingang "BusType" mithilfe der Variable "enBusType" definiert. Die Adresse bzw. der Name des Buskopplers wird mit "stBusConfig" definiert.

TRUE am Eingang "bEnable" übernimmt die Konfiguration und der Funktionsbaustein generiert das Handle.

Beispiel 1:

Schritt 1: Variablendeklaration am XMxx S20 IO Lokalbus, Modul am Slot 1

Deklaration:

Programm:

```
PROGRAM ProgIH_S20ComInterfaceType02
VAR
  fbIH_S20ComConfiguration: IH_S20ComConfiguration; // Function block instance
  bEnableComConfig:        BOOL; // Function block processing enabled( TRUE - once, edge-controlled )
  bInOperationComConfig:   BOOL; //TRUE: Function Block is working on its actual task
  bErrorComConfig :        BOOL; //TRUE: Indicates an error
  ErrorIDComConfig:        ERROR_CODE; //Class of error
  ErrorIdentComConfig:     ERROR_STRUCT; //Detailed information about error
  enBusType:               IH_S20_BUSTYPE; //Bus type selection
  stBusConfig:             IH_S20_BUS_CONFIG; //Bus configuration parameters
  dwS20BusHandle:          DWORD; //Bus handle of S20IO configuration

  fbIH_S20ComInterfaceType02: IH_S20ComInterfaceType02; // Function block instance
  bExecute:                BOOL; // Function block processing enabled( TRUE - once, edge-controlled )
  bDone:                   BOOL; //TRUE: Function Block has completed its task
  bActive:                 BOOL; //TRUE: Function Block is working on its actual task
  bError:                  BOOL; //TRUE: Indicates an error
  ErrorID:                 ERROR_CODE; //Class of error
  ErrorIdent:              ERROR_STRUCT; //Detailed information about error
  enSetOpMode:             IH_S20_OP_MODE; //FB Operating mode selection
  uiSlot:                  UINT ;
                          // Module slot address. Bus coupler : 0,
                          // 1st Module after coupler:1 ... 63rd Module after coupler: 63
  stPDIIndex:              IH_S20_PARAM; //PDI object mapping to bus parameter
  tTimeout:               TIME := T#3S; // Timeout delay
  uiNoOfRecBytes:          UINT; // No. of received bytes
  arIndexData:             ARRAY[0..50] OF BYTE;
                          // Address of PDI Index data values.
                          // Common interface for read/write operation

END_VAR
```

Initialisierung der Variablen vom Funktionsbaustein angeschlossen an XM lokal und Modul an Steckplatz 1, im Programm.

Programm:

Programm:

```
// Assignment of BusType to S20IO local
enBusType := IH_S20_BUSTYPE.S20IO;

fbIH_S20ComConfiguration(
  Enable:= bEnableComConfig,
  InOperation=> bInOperationComConfig,
  Error=> bErrorComConfig,
  ErrorID=> ErrorIDComConfig,
  ErrorIdent=> ErrorIdentComConfig,
  BusType:= enBusType,
  BusConfig:= stBusConfig,
  S20BusHandle=> dwS20BusHandle);

uiSlot := 1; // Slot no
enSetOpMode := IH_S20_OP_MODE.DEVICETYPE_READ; //Setting Operating mode to read DeviceType

fbIH_S20ComInterfaceType02(
  Execute:= bExecute,
  Done=> bDone,
  Active=> bActive,
  Error=> bError,
  ErrorID=> ErrorID,
  ErrorIdent=> ErrorIdent,
  S20BusHandle:= dwS20BusHandle,
  SetOpMode:= enSetOpMode,
  Slot:= uiSlot,
  PDIIndex:= stPDIIndex,
```

```

IndexDataAdr:= ADR(arIndexData),
SizeOfIndexData:= SIZEOF(arIndexData),
Timeout:= tTimeout,
NoOfRecBytes=> uiNoOfRecBytes);

```

23.5.3 IH_S20ComInterface

Funktionsbeschreibung Der Funktionsbaustein **IH_S20ComInterface** bildet die Grundlage für PDI-Zugriffe (azyklische Kommunikation) mit S20-Modulen über Busmaster.

Andere S20-Funktionsbausteine verwenden den Funktionsbaustein **IH_S20ComInterface** intern.

Strukturen und Enumerationen

Die Enumeration **IH_S20_BUSTYPE** bildet die möglichen Feldbusse ab.

IH_S20_BUSTYPE		
ENUM	Wert	Beschreibung
S20_LOCAL	16#0000	S20-Modul am S20 lokal an der IndraControl XM
SERCOS	16#0001	S20-Modul am Buskoppler Sercos
PROFINET	16#0002	S20-Modul am Buskoppler Profinet
ETHERNETIP	16#0003	S20-Modul am Buskoppler Ethernet/IP
PROFIBUS	16#0004	S20-Modul am Buskoppler Profibus
ETHERNET	16#0005	S20-Modul am Buskoppler Ethernet

Die Struktur **IH_S20_BUS_CONFIG** dient der feldbusspezifischen Adressierung der S20-Buskoppler

IH_S20_BUS_CONFIG		
STRUCT	Datentyp	Beschreibung
Sercos	IH_S20_SERCOS_CONFIG	Sercos-Buskopplerkonfiguration
Profinet	IH_S20_PROFINET_CONFIG	Profinet-Buskopplerkonfiguration
Ethernet/IP	IH_S20_ETHERNETIP_CONFIG	Ethernet/IP-Buskopplerkonfiguration
Profibus	IH_S20_PROFIBUS_CONFIG	ProfiBus-Buskopplerkonfiguration
Ethernet	IH_S20_ETHERNET_CONFIG	Ethernet-Buskopplerkonfiguration

Die Enumeration **IH_S20_OP_MODE** legt die Aktion fest, sendet ein definiertes Kommando oder PDI Objekt lesen/schreiben.

IH_S20_OP_MODE		
ENUM	Wert	Beschreibung
DEVICETYPE_READ	16#00	Lesen der Modulidentifikation DeviceType : 0037hex
DIAG_READ	16#01	Lesen des Diagnose PDI-Objekt, Diagnosezustand (0018hex: DiagState)
CONFIG_WRITE	16#02	Schreiben der Parametrierung auf das PDI-Objekt Parametertabelle (0080hex: ParaTable)
DIAGRESET_WRITE	16#03	Diagnose zurücksetzen 0019hex ResetDiag
OTHER_READ	16#04	Lesen eines beliebigen PDI-Objekts/Index der am Eingang "PDIIndex" angegeben wird

IH_S20_OP_MODE		
OTHER_WRITE	16#05	Schreiben eines beliebigen PDI-Objekts/Index der am Eingang "PDIIndex" angegeben wird
CONFIG_READ	16#08	Lesen der Parametrierung auf das PDI-Objekt Parametertabelle (0080hex: ParaTable)

Die Struktur **IH_S20_SERCOS_CONFIG** dient für die Angabe des Sercos-Busmasters und der Sercos-Adresse.

IH_S20_SERCOS_CONFIG						
STRUCT	Datentyp	Beschreibung	Min.-Wert	Max.-Wert	Defaultwert	Applikation
BusMaster	IL_BUSMASTER	Busmaster	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke am "Execute"
SercosAdr	UINT	Sercos-Adresse	0 (MLD: lokale Achse) 1 (andere Systeme)	511	0	

Die Struktur **IH_S20_PROFINET_CONFIG** dient für die Angabe des Profinet-Busmasters und des Stationsnamens.

IH_S20_PROFINET_CONFIG						
STRUCT	Datentyp	Beschreibung	Min.-Wert	Max.-Wert	Defaultwert	Applikation
BusMaster	IL_BUSMASTER	Busmaster	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke am "Execute"
IdentStr	STRING	Identifikationsstring des Feldbusteilnehmers	----	---	---	

Die Struktur **IH_S20_ETHERNETIP_CONFIG** dient für die Angabe des Ethernet/IP-Busmasters und der IP-Adresse.

IH_S20_ETHERNETIP_CONFIG						
STRUCT	Datentyp	Beschreibung	Min.-Wert	Max.-Wert	Defaultwert	Applikation
BusMaster	IL_BUSMASTER	Busmaster	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Steigende Flanke am "Execute"
CIPPath	STRING	Adresse des CIP-Teilnehmers "aaa.bbb.ccc.ddd"	--	--	--	

Die Struktur **IH_S20_PROFIBUS_CONFIG** dient für die Angabe des Profibus-Busmasters, des Segments und der Adresse.

IH_S20_PROFIBUS_CONFIG						
STRUCT	Datentyp	Beschreibung	Min.-Wert	Max.-Wert	Defaultwert	Applikation
Master	BYTE	DP-Master-Identifikation	---	---	---	Steigende Flanke am "Execute"
Segment	BYTE	Nummer des DP-Segments (0)	---	---	---	
Station	BYTE	Nummer des DP-Slaves (Busadresse)	---	---	---	

Die Struktur **IH_S20_ETHERNET_CONFIG** dient für die Angabe der IP-Adresse und des Port.

IH_S20_ETHERNET_CONFIG						
STRUCT	Datentyp	Beschreibung	Min.-Wert	Max.-Wert	Defaultwert	Applikation
Address	STRING	Verknüpfte IP-Adresse	--	--	"	Steigende Flanke am "Execute"
Port	UINT	TCP-Port des Modbus-TCP-Servers	0	65535	502	

Die Struktur **IH_S20_PARAM** dient zum Mappen des PDI-Index auf die feldbusspezifischen Parameter.

IH_S20_PARAM		
STRUCT	Datentyp	Beschreibung
Sercos	IH_S20_SIII_PARAM	Sercos-Parameter-Mapping
Profinet	IH_S20_PN_PARAM	Profinet-Index-Mapping
Ethernet/IP	IH_S20_ENIP_PARAM	Ethernet/IP-Attribut-Mapping
Profibus	IH_S20_PB_PARAM	Profibus-Index-Mapping
Ethernet	IH_S20_ETH_PARAM	ModbusTCP-Register-Mapping

Die Struktur **IH_S20_SIII_PARAM** dient zum Mappen des PDI-Index auf Sercos-Parameter.

IH_S20_SIII_PARAM						
STRUCT	Datentyp	Beschreibung	Min.-Wert	Max.-Wert	Defaultwert	Applikation
ParamType	IL_SIII_PARAM_TYPE	Standard (S) oder produktspezifische (P) IDN	IL_S_PARAM	IL_P_PARAM	IL_S_PARAM	Steigende Flanke am "Execute"
Set	USINT	Parametersatz	16#00	16#FF	16#00	
Nbr	UINT	Nummer des Datenbausteins (Index)	16#00	16#FFFF	16#00	
StructElem	USINT	Strukturelement (Subindex)	16#00	16#FF	16#00	

Die Struktur **IH_S20_PN_PARAM** dient zum Mappen des PDI-Index auf Profinet-Parameter.

IH_S20_PN_PARAM						
STRUCT	Datentyp	Beschreibung	Min.-Wert	Max.-Wert	Defaultwert	Applikation
Index	UINT	Index des Adressdatensatz	16#00	16#FFFF	16#00	Steigende Flanke am "Execute"
Subslot	UINT	Adressierung des Submoduls	16#01	16#8FFF	16#00	

Die Struktur **IH_S20_ENIP_PARAM** dient zum Mappen des PDI-Index auf Ethernet/IP-Attribute.

IH_S20_ENIP_PARAM						
STRUCT	Datentyp	Beschreibung	Min.-Wert	Max.-Wert	Defaultwert	Applikation
Class	UINT	ClassCode -16#69 für PDI-Zugriff	16#01	16#FF	16#69	Steigende Flanke am "Execute"
Instance	UINT	Instanznummer (Slot- Nr.)	16#00	16#FF	16#00	
Attribute	UINT	Attribut-ID (Index)	16#01	16#FF	16#01	

Die Struktur **IH_S20_PB_PARAM** dient zum Mappen des PDI-Index auf Profibus-Parameter.

IH_S20_PB_PARAM						
STRUCT	Datentyp	Beschreibung	Min.-Wert	Max.-Wert	Defaultwert	Applikation
Index	INT	Prozessdatenindex	16#00	16#7FFF	16#00	Steigende Flanke am "Execute"

Die Struktur **IH_S20_ETH_PARAM** dient zum Mappen des PDI-Index auf Modbus-TCP-Register.

IH_S20_ETH_PARAM						
STRUCT	Datentyp	Beschreibung	Min.-Wert	Max.-Wert	Defaultwert	Applikation
Index	UINT	Datenindexregister	16#00	16#FFFF	16#00	Steigende Flanke am "Execute"

Beispielprogramm Das Beispielprogramm beschreibt das Lesen/Schreiben der PDI-Objekte mit unterschiedlichen Feldbussen.

Der verwendete Feldbus ist am Eingang "BusType" mithilfe der Variable "enBusType" definiert. Die Adresse bzw. der Name des Buskopplers wird mit "stBusConfig" definiert. Der Modulsteckplatz wird am Eingang "Slot" definiert.

TRUE am Eingang "Execute" übernimmt die Konfiguration und der Funktionsbaustein führt die gewählte PDI-Aktion in "enSetOpMode" aus.

Deklaration:

Programm:

```

PROGRAM ProgIH_S20ComInterface
VAR
    fbIH_S20ComInterface: IH_S20ComInterface; // Function block instance
    bExecute: BOOL; // Function block processing enabled( TRUE - once, edge-controlled )
    bDone: BOOL; //TRUE: Function Block has completed its task
    bActive: BOOL; //TRUE: Function Block is working on its actual task
    bError: BOOL; //TRUE: Indicates an error
    ErrorID: ERROR_CODE; //Class of error
    ErrorIdent: ERROR_STRUCT; //Detailed information about error

```



```

enBusType: IH_S20_BUSTYPE //Bus type selection
stBusConfig: IH_S20_BUS_CONFIG; //Bus configuration parameters
enSetOpMode: IH_S20_OP_MODE; //FB Operating mode selection
uiSlot: UINT ; //Module slot address.Bus coupler:0,
//1st Module after coupler:1...63rd Module after coupler:63
stPDIIndex: IH_S20_PARAM; //PDI object mapping to bus parameter
tTimeout: TIME := T#3S; // Timeout delay
uiNoOfRecBytes: UINT; // No. of received bytes
arIndexData: ARRAY[0..50] OF BYTE; //Address of PDI Index data values.
//Common interface for read/write operation
END_VAR

```

Beispiel 1:

Initialisierung der Variablen vom Funktionsbaustein für das Analogmodul, angeschlossen am Sercos-Buskoppler mit Sercos-Adresse 69 und Modul an Steckplatz 1, zum Lesen des DeviceTyp.

Variableninitialisierung:

Programm:

```

enBusType:= IH_S20_BUSTYPE.SERCOS;
enSetOpMode:= IH_S20_OP_MODE.DEVICETYPE_READ;
uiSlot := 1;

```

Beispiel 2:

Initialisierung vom Funktionsbaustein zu Lesen der Diagnose am Profinet-Buskoppler.

Variableninitialisierung:

Programm:

```

enBusType:= IH_S20_BUSTYPE.PROFINET;
enSetOpMode:= IH_S20_OP_MODE.DIAG_READ;
uiSlot := 1;

```

Beispiel 3:

Initialisierung vom Funktionsbaustein zum Schreiben der Konfiguration für das S20-AI-4-I Modul am Profibus-Buskoppler.

Variablendeklaration:

```
arIndexData: ARRAY [0..14] OF BYTE := [0, 4(4), 9(0)];
```

Variableninitialisierung:

Programm:

```

enBusType:= IH_S20_BUSTYPE.PROFIBUS;
enSetOpMode:= IH_S20_OP_MODE.CONFIG_WRITE;
stBusConfig.Profibus.Master := 0;
stBusConfig.Profibus.Segment := 0;
stBusConfig.Profibus.Station := 2;
uiSlot := 1;

```

Beispiel 4:

Initialisierung vom Funktionsbaustein zum Zurücksetzen der Diagnose eines S20-Moduls am Ethernet/IP-Buskoppler.

Variableninitialisierung:

Programm:

```

enBusType:= IH_S20_BUSTYPE.ETHERNETIP;
enSetOpMode:= IH_S20_OP_MODE.DIAGRESET_WRITE;
uiSlot := 1;

```

Beispiel 5:

Initialisierung vom Funktionsbaustein zum Lesen des "Location Name" (Index: 0014hex) am Sercos-Buskoppler.

Variablendeklaration:

```
arIndexData: ARRAY [0..58] OF BYTE;
```

Variableninitialisierung:

Programm:

```
enBusType:= IH_S20_BUSTYPE.SERCOS;
enSetOpMode:= IH_S20_OP_MODE.OTHER_READ;
stBusConfig.Sercos.BusMaster := IL_BUSMASTER.IL_BUSMASTER_0;
stBusConfig.Sercos.SercosAdr := 69;

stPDIIndex.Sercos.ParamType := IL_SIII_PARAM_TYPE.IL_P_PARAM;
stPDIIndex.Sercos.Set := 1;
stPDIIndex.Sercos.Nbr := 16#14;
stPDIIndex.Sercos.StructElem := 0;
uiSlot := 1;
```

Beispiel 6:

Initialisierung vom Funktionsbaustein zum Schreiben des "Location Name" (Index: 0014hex) am Profinet-Buskoppler.

Variablendeklaration:

```
arIndexData: ARRAY [0..58] OF BYTE := [82, 57, 49, 55(0)];
```

Variableninitialisierung:

Programm:

```
enBusType:= IH_S20_BUSTYPE.PROFINET;
enSetOpMode:= IH_S20_OP_MODE.OTHER_WRITE;
stBusConfig.Profinet.BusMaster := IL_BUSMASTER.IL_BUSMASTER_0;
stBusConfig.Profinet.IdentStr := 's20-pn-bk'

stPDIIndex.Profinet.Index := 16#14;
stPDIIndex.Profinet.SubSlot := 1;
uiSlot := 1;
```

Programm:

Programm:

```
fbIH_S20ComInterface(
  Execute:= bExecute,
  Done=> bDone,
  Active=> bActive,
  Error=> bError,
  ErrorID=> ErrorID,
  ErrorIdent=> ErrorIdent,
  BusType:= enBusType,
  BusConfig:= stBusConfig,
  SetOpMode:= enSetOpMode,
  Slot:= uiSlot,
  PDIIndex:= stPDIIndex,
  IndexDataAdr:= ADR(arIndexData),
  SizeOfIndexData:= SIZEOF(arIndexData),
  Timeout:= tTimeout,
  NoOfRecBytes=> uiNoOfRecBytes);
```

23.6 Analoge Eingangsmodule (AnalogIO)

23.6.1 IH_S20AI4IU

Beispielprogramm Das Beispielprogramm dient zum Lesen der Analogwerte der Module **S20-AI-4-U** und **S20-AI-4-I**.

Der verwendete Feldbus ist am Eingang "BusType" mithilfe der Variable "enBusType" definiert. Die Adresse bzw. der Name des Buskopplers wird mit "stBusConfig" definiert. Der Modulsteckplatz wird am Eingang "Slot" definiert.

TRUE am Eingang "bEnable" übernimmt die Konfiguration und der Funktionsbaustein liest die aktuelle Konfiguration des Moduls.

Beispiel 1:

Schritt 1: Modulkonfiguration in IndraWorks Engineering für Kanal 1 zum Messen der Analogspannung von 10V, Filter 30Hz und 16 Samples

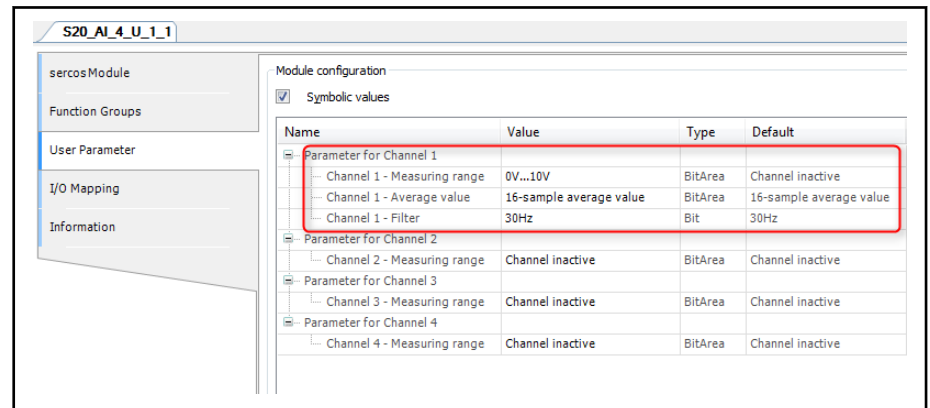


Abb. 23-2: Beispiel 1, Schritt 1

Schritt 2: Variablendeklaration für das Modul S20-AI-4-U am Sercos Buskoppler

Deklaration:

Programm:

```
PROGRAM ProgIH_S20AI4IU
VAR
  fbIH_S20AI4IU:    IH_S20AI4IU;
  bEnable:          BOOL; //TRUE: Function is enabled
  bInOperation:     BOOL; //TRUE: Function Block is working on its actual task
  bError:           BOOL; //TRUE: Indicates an error
  ErrorID:          ERROR_CODE; //Class of error
  ErrorIdent:       ERROR_STRUCT; //Detailed information about error
  enBusType:        IH_S20_BUSTYPE; //Bus type selection
  stBusConfig:      IH_S20_BUS_CONFIG; //Bus configuration parameters
  uiSlot:           UINT; //Module slot address.Bus coupler:0,
                  //1st Module after coupler:1...63rd Module after coupler:63
  arChannelConfig:  ARRAY[1..4] OF IH_S20_AI_CONFIG; //Structure for channel configuration
  arInProcData:     ARRAY[1..4] OF WORD; //Input Process variables
  arOutputData:     ARRAY[1..4] OF IH_S20_AI_DATA; //Structure for Output Data
  rCh1_Voltage:     REAL; // Ch-1 Voltage input value
END_VAR
```

Initialisierung der Variablen vom Funktionsbaustein für das Analogmodul, angeschlossen am Sercos Buskoppler mit Sercos-Adresse 69 und Modul an Steckplatz 1, im Programm.

Variableninitialisierung:

Programm:

```
enBusType := IH_S20_BUSTYPE.SERCOS; // Connected fieldbus coupler
stBusConfig.Sercos.BusMaster := IL_BUSMASTER.IL_BUSMASTER_0;
stBusConfig.Sercos.SercosAdr := 69; //Buscoupler address
uiSlot := 1; // Module position
//Ch-1 Configuration
arChannelConfig[1].HighLimit := 10.0; //10V
arChannelConfig[1].LowLimit := 0.0; //0v
```

Programm:*Programm:*

```
fbIH_S20AI4IU (
    Enable       := bEnable,
    InOperation=> bInOperation,
    Error=>       bError,
    ErrorID=>    ErrorID,
    ErrorIdent=> ErrorIdent,
    BusType      := enBusType,
    BusConfig    := stBusConfig,
    Slot         := uiSlot,
    ChannelConfig:= arChannelConfig,
    InProcData   := arInProcData,
    OutputData   := arOutputData);
```

Lesen der analogen Eingangswerte:

```
rCh1_Voltage:= arOutputData[1].NormalizedValue;
```

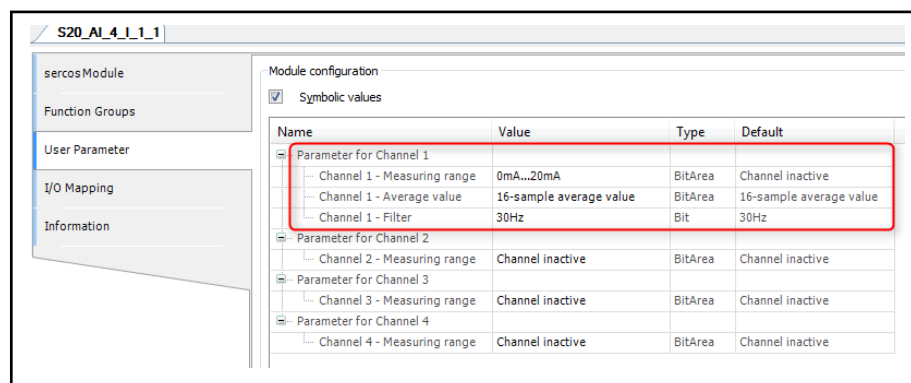
Beispiel 2:**Schritt 1: Modulkonfiguration in IndraWorks Engineering für Kanal 1 zum Messen vom Strom 0-20mA, Filter 30Hz und 16 Samples**

Abb. 23-3: Beispiel 2, Schritt1

Schritt 2: Variablendeklaration für das Modul S20-AI-4-I am Sercos-Buskoppler**Deklaration:***Programm:*

```
PROGRAM ProgIH_S20AI4IU
VAR
    fbIH_S20AI4IU:    IH_S20AI4IU;
    bEnable:          BOOL; //TRUE: Function is enabled
    bInOperation:      BOOL; //TRUE: Function Block is working on its actual task
    bError:            BOOL; //TRUE: Indicates an error
    ErrorID:           ERROR_CODE; //Class of error
    ErrorIdent:        ERROR_STRUCT; //Detailed information about error
    enBusType:         IH_S20_BUSTYPE; //Bus type selection
    stBusConfig:       IH_S20_BUS_CONFIG; //Bus configuration parameters
    uiSlot:            UINT; //Module slot address. Bus coupler : 0,
                        //1st Module after coupler:1 ... 63rd Module after coupler: 63
    arChannelConfig:   ARRAY[1..4] OF IH_S20_AI_CONFIG; //Structure for channel configuration
    arInProcData:      ARRAY[1..4] OF WORD; //Input Process variables
    arOutputData:      ARRAY[1..4] OF IH_S20_AI_DATA; //Structure for Output Data
    rCh1_Voltage:      REAL; // Ch-1 Voltage input value
END_VAR
```

Initialisierung der Variablen vom Funktionsbaustein für das Analogmodul, angeschlossen am Sercos Buskoppler, mit Sercos Adresse 69 und Modul an Steckplatz 1, im Programm.

Variableninitialisierung:

Programm:

```

enBusType      := IH_S20_BUSTYPE.SERCOS; // Connected fieldbus coupler
stBusConfig.Sercos.BusMaster := IL_BUSMASTER.IL_BUSMASTER_0;
stBusConfig.Sercos.SercosAdr := 69; //Buscoupler address
uiSlot         := 1; // Module position
//Ch-1 Configuration
arChannelConfig[1].HighLimit := 10.0; //10V
arChannelConfig[1].LowLimit  := 0.0; //0v

```

Programm:*Programm:*

```

fbIH_S20AI4IU (
    Enable      := bEnable,
    InOperation=> bInOperation,
    Error=>      bError,
    ErrorID=>    ErrorID,
    ErrorIdent=> ErrorIdent,
    BusType     := enBusType,
    BusConfig   := stBusConfig,
    Slot        := uiSlot,
    ChannelConfig := arChannelConfig,
    InProcData  := arInProcData,
    OutputData  := arOutputData);

```

Programm:

```

fbIH_S20AI4IU (
    Enable      := bEnable,
    InOperation=> bInOperation,
    Error=>      bError,
    ErrorID=>    ErrorID,
    ErrorIdent=> ErrorIdent,
    BusType     := enBusType,
    BusConfig   := stBusConfig,
    Slot        := uiSlot,
    ChannelConfig := arChannelConfig,
    InProcData  := arInProcData,
    OutputData  := arOutputData);

```

Lesen der analogen Eingangswerte:

```
rCh1_Current:= arOutputData[1].NormalisedValue;
```

Schritt 3: Schritt 3 ist gültig für Beispiel 1 und 2**Verknüpfen der Prozessdaten im E/A-Abbild am Modul in IndraWorks:**

Verknüpfen der Prozessdaten im E/A-Abbild mit dem im SPS Programm deklarierten Variablen für das Modul S20-AI-4-I, identisch für das Modul S20-AI-4-U.

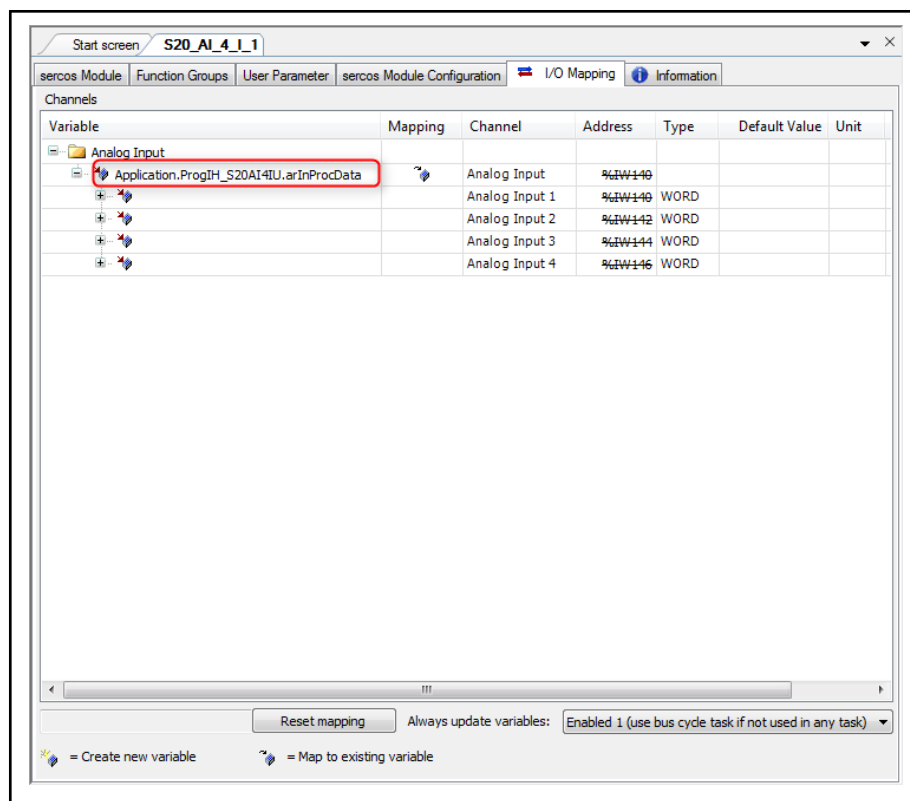


Abb. 23-4: Beispiel 2, Schritt3

23.6.2 IH_S20AI8

Beispielprogramm

Das Beispielprogramm dient zum Lesen von Analogwerten vom S20-AI-8 Modul.

Der verwendete Feldbus ist am Eingang "BusType" mithilfe der Variablen "enBusType" definiert. Die Adresse bzw. der Name des Buskopplers wird mit "stBusConfig" definiert. Der Modulsteckplatz wird am Eingang "Slot" definiert. TRUE am Eingang "bEnable" übernimmt die Konfiguration und der Funktionsbaustein liest die aktuelle Konfiguration des Moduls.

Schritt 1: Modulkonfiguration in IndraWorks Engineering für Kanal 1 zum Messen der Analogspannung von $\pm 5V$, Filter 30Hz und 16 Samples

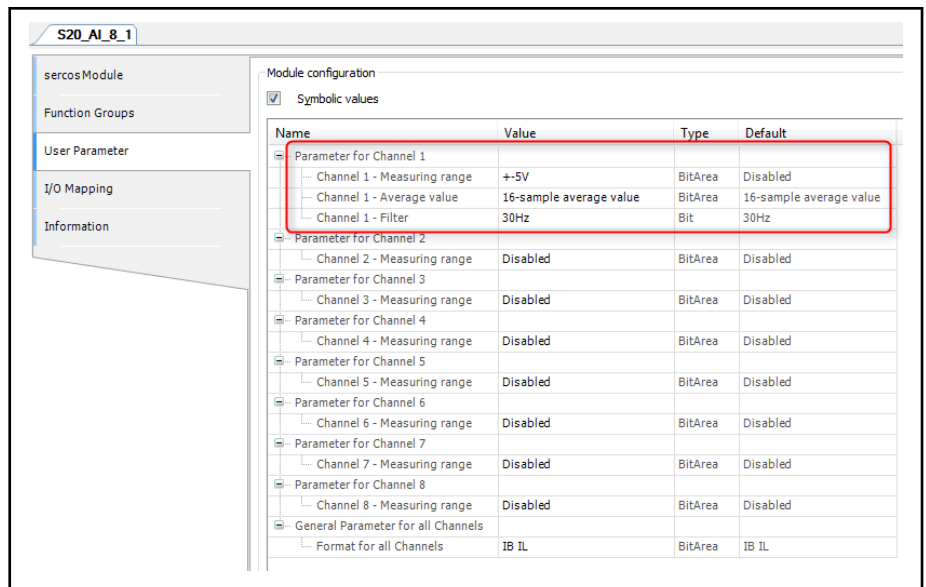


Abb. 23-5: Schritt 1

Schritt 2: Variablendeklaration für das Modul S20-AI-8 am Sercos-Buskoppler Deklaration:

Programm:

```
PROGRAM ProgIH_S20AI8
VAR
    fbIH_S20AI8:      IH_S20AI8; // FB Instance
    bEnable:          BOOL; //TRUE: Function is enabled
    bInOperation:     BOOL; //TRUE: Function Block is working on its actual task
    bError:           BOOL; //TRUE: Indicates an error
    ErrorID:          ERROR_CODE; //Class of error
    ErrorIdent:       ERROR_STRUCT; //Detailed information about error
    enBusType:        IH_S20_BUSTYPE; //Bus type selection
    stBusConfig:      IH_S20_BUS_CONFIG; //Bus configuration parameters
    uiSlot:           UINT; //Module slot address. Bus coupler : 0,
                    //1st Module after coupler:1 ... 63rd Module after coupler: 63
    arChannelConfig:  ARRAY[1..8] OF IH_S20_AI_CONFIG; //Structure for channel configuration
    arInProcData:     ARRAY[1..8] OF WORD; //Input Process variables
    arOutputData:     ARRAY[1..8] OF IH_S20_AI_DATA; //Structure for Output Data
    rCh1_Voltage:     REAL; // Ch-1 voltage value
END_VAR
```

Variablendeklaration für das Modul S20-AI-8 am Sercos Buskoppler:

Initialisierung der Variablen vom Funktionsbaustein für das Analogmodul, angeschlossen am Sercos-Buskoppler mit Sercos-Adresse 69 und Modul an Steckplatz 1, im Programm.

Variableninitialisierung:

Programm:

```
enBusType      := IH_S20_BUSTYPE.SERCOS; // Connected fieldbus coupler
stBusConfig.Sercos.BusMaster := IL_BUSMASTER.IL_BUSMASTER_0;
stBusConfig.Sercos.SercosAdr := 69; //Buscoupler address
uiSlot         := 1; // Module position

arChannelConfig[1].HighLimit := 5.0; //+5V
arChannelConfig[1].LowLimit  := -5.0; //-5V
```

Programm:

Programm:

```
fbIH_S20AI8 (
    Enable      := bEnable,
```

```

InOperation=> bInOperation,
Error=>       bError,
ErrorID=>     ErrorID,
ErrorIdent=>  ErrorIdent,
BusType      := enBusType,
BusConfig    := stBusConfig,
Slot         := uiSlot,
ChannelConfig:= arChannelConfig,
InProcData   := arInProcData,
OutputData   := arOutputData);

```

Lesen der analogen Eingangswerte:

```
rCh1_Voltage := arOutputData[1].NormalisedValue;
```

Schritt-3: Verknüpfen der Prozessdaten im E/A-Abbild am Modul in IndraWorks:

Verknüpfen der Prozessdaten im E/A-Abbild mit dem im SPS Programm deklarierten Variablen für das Modul S20-AI-8.

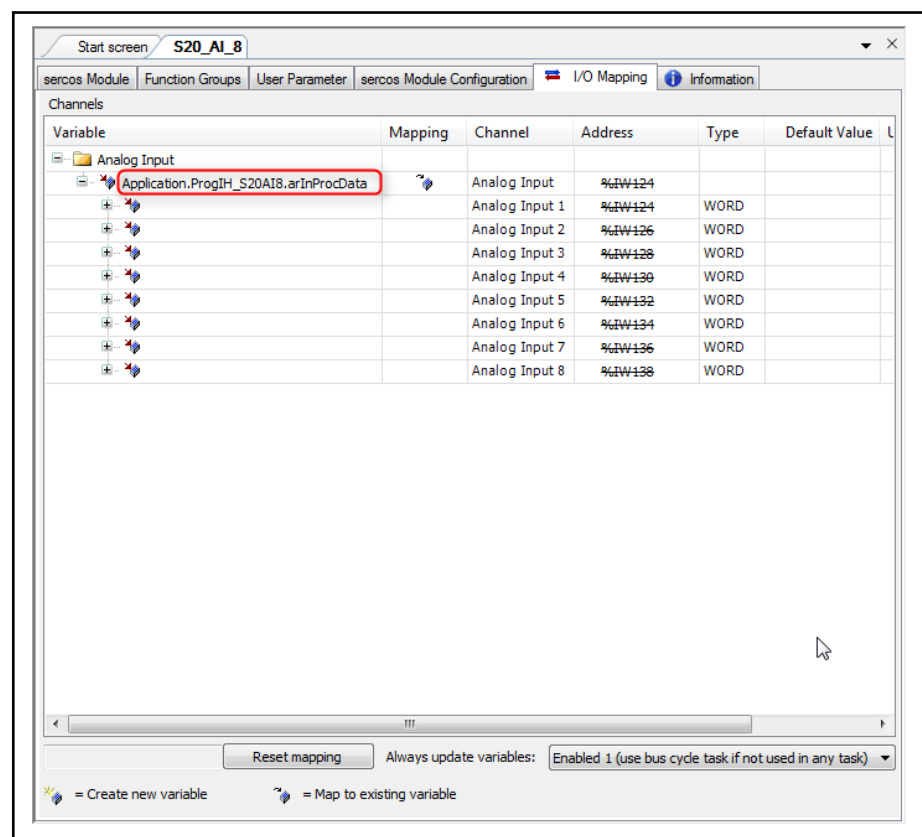


Abb. 23-6: Schritt 3

23.6.3 IH_S20AI4RTD

Beispielprogramm Das Beispielprogramm dient zum Lesen von Analogwerten vom S20-AI-4-RTD Modul.

Der verwendete Feldbus ist am Eingang "BusType" mithilfe der Variable "enBusType" definiert. Die Adresse bzw. der Name des Buskopplers wird mit "stBusConfig" definiert. Der Modulsteckplatz wird am Eingang "Slot" definiert.

TRUE am Eingang "bEnable" übernimmt die Konfiguration und der Funktionsbaustein liest die aktuelle Konfiguration des Moduls.

Schritt 1: Modulkonfiguration in IndraWorks Engineering für Kanal 1 zum Messen eines PT-100 Sensor mit der Auflösung 0.1°C in 3-Leiter-Technik und einem Widerstand von 100 Ohm, Filter 120 ms

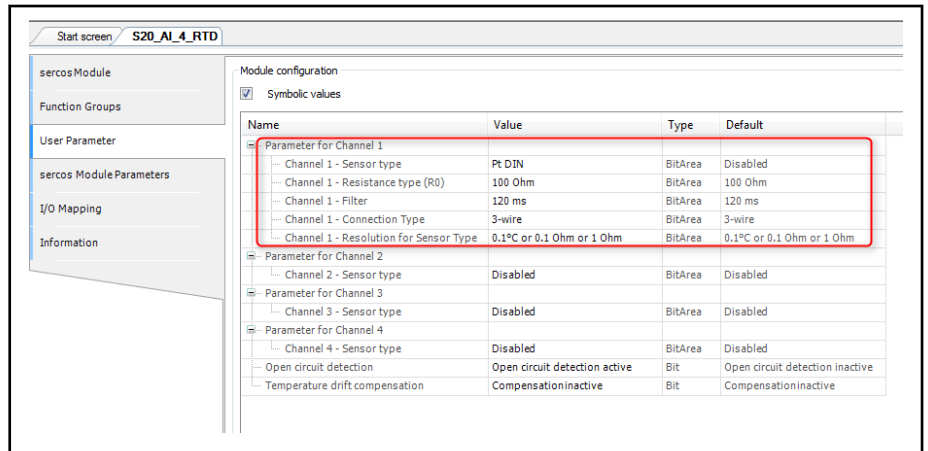


Abb. 23-7: Schritt 1

Schritt 2: Variablendeklaration für das Modul S20-AI-4-RTD am Sercos-Buskoppler

Deklaration:

Programm:

```
PROGRAM ProgIH_S20AI4RTD
VAR
  fbIH_S20AI4RTD:    IH_S20AI4RTD; // FB Instance
  bEnable:           BOOL; //TRUE: Function is enabled
  bInOperation:      BOOL; //TRUE: Function Block is working on its actual task
  bError:            BOOL; //TRUE: Indicates an error
  ErrorID:           ERROR_CODE; //Class of error
  ErrorIdent:        ERROR_STRUCT; //Detailed information about error
  enBusType:         IH_S20_BUSTYPE; //Bus type selection
  stBusConfig:       IH_S20_BUS_CONFIG; //Bus configuration parameters
  uiSlot:            UINT; //Module slot address. Bus coupler : 0,
                        //1st Module after coupler:1 ... 63rd Module after coupler: 63
  arInProcData:      ARRAY[1..4] OF WORD; //Input Process variables
  arOutputData:      ARRAY[1..4] OF IH_S20_TEMP_DATA; //Structure for Output Data
  rTemperature:      REAL; // read Temperature value
END_VAR
```

Variableninitialisierung für Sercos-Buskoppler:

Initialisierung der Variablen vom Funktionsbaustein für das Analogmodul, angeschlossen am Sercos-Buskoppler mit Sercos-Adresse 69 und Modul an Steckplatz 1, im Programm.

Variableninitialisierung:

Programm:

```
enBusType           := IH_S20_BUSTYPE.SERCOS; // Connected fieldbus coupler
stBusConfig.Sercos.BusMaster := IL_BUSMASTER.IL_BUSMASTER_0;
stBusConfig.Sercos.SercosAdr := 69; //BuscoupleF address
uiSlot              := 1; // Module position
```

Programm:

Programm:

```
fbIH_S20AI4RTD (
  Enable           := bEnable,
  InOperation=>    bInOperation,
  Error=>          bError,
```

```

ErrorID=>      ErrorID,
ErrorIdent=>   ErrorIdent,
BusType       := enBusType,
BusConfig     := stBusConfig,
Slot         := uiSlot,
InProcData    := arInProcData,
OutputData    := arOutputData);

```

Lesen der analogen Eingangswerte:

```
rTemperature := arOutputData[1].NormalizedValue;
```

Schritt 3: Verknüpfen der Prozessdaten im E/A-Abbild am Modul in IndraWorks

Das Verknüpfen der Prozessdaten im E/A-Abbild mit dem im SPS-Programm deklarierten Variablen für das Modul S20-AI-8-RTD.

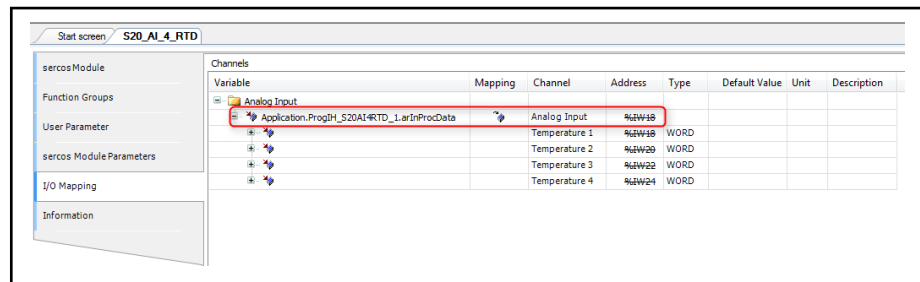


Abb. 23-8: Schritt 3

23.6.4 IH_S20AI4UTH

Beispielprogramm Das Beispielprogramm dient zum Lesen von Analogwerten vom S20-AI-4-UTH Modul.

Der verwendete Feldbus ist am Eingang "BusType" mithilfe der Variable "enBusType" definiert. Die Adresse bzw. der Name des Buskopplers wird mit "stBusConfig" definiert. Der Modulsteckplatz wird am Eingang "Slot" definiert. TRUE am Eingang "bEnable" übernimmt die Konfiguration und der Funktionsbaustein liest die aktuelle Konfiguration des Moduls.

Schritt 1: Modulkonfiguration in IndraWorks Engineering für Kanal 1 zum Messen eines K-Typ Sensor mit der Auflösung 0.1°C und dem Filter 120 ms

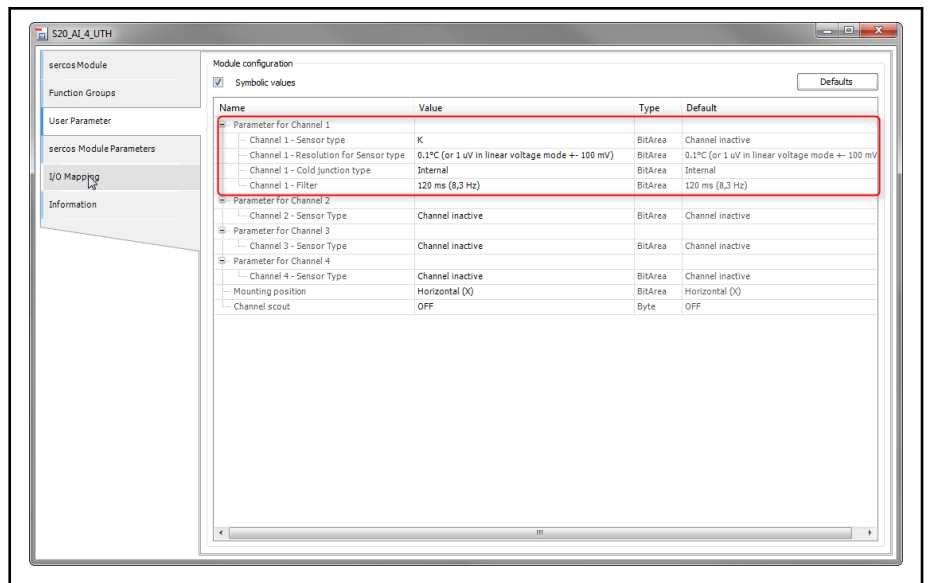


Abb. 23-9: Schritt 1

Schritt 2: Variablendeklaration für das Modul S20-AI-8-UTH am Sercos-Buskoppler

Deklaration:

Programm:

```

PROGRAM ProgIH_S20AI4UTH
VAR
  fbIH_S20AI4UTH:    IH_S20AI4UTH; // FB Instance
  bEnable:           BOOL; //TRUE: Function is enabled
  bInOperation:      BOOL; //TRUE: Function Block is working on its actual task
  bError:            BOOL; //TRUE: Indicates an error
  enErrorID:         ERROR_CODE; //Class of error
  stErrorIdent:      ERROR_STRUCT; //Detailed information about error
  stBusType:         IH_S20_BUSTYPE; ; //Bus type selection
  stBusConfig:       IH_S20_BUS_CONFIG; //Bus configuration parameters
  uiSlot:            UINT := 1; //Module slot address. Bus coupler : 0,
                        //1st Module after coupler:1 ... 63rd Module after coupler: 63
  arInProcData:      ARRAY [1..5] OF WORD; //Input process data
  arOutputData:      ARRAY [1..4] OF IH_S20_TEMP_DATA; //Contains normalized value
                        //and diagnostic code for channels 1...4.
  arOutProdata_0:    WORD; //Output process data(word 0)
  stAnalogData:      IH_S20_TEMP_DATA; //Contains normalized value
                        //and diagnostic code of analog voltage input +-5V
  rTemperature:      REAL; // Read temperature value
END_VAR

```

Variableninitialisierung für Sercos-Buskoppler:

Initialisierung der Variablen vom Funktionsbaustein für das Analogmodul, angeschlossen am Sercos-Buskoppler mit Sercos-Adresse 69 und Modul an Steckplatz 1, im Programm.

Variableninitialisierung:

Programm:

```

enBusType           := IH_S20_BUSTYPE.SERCOS; // Connected fieldbus coupler
stBusConfig.Sercos.BusMaster := IL_BUSMASTER.IL_BUSMASTER_0;
stBusConfig.Sercos.SercosAdr := 69; //Buscoupler address
uiSlot              := 1; // Module position

```

Programm:

Programm:

```

fbIH_S20AI4UTH (
    Enable           := bEnable,
    InOperation=>    bInOperation,
    Error=>           bError,
    ErrorID=>         enErrorID,
    ErrorIdent=>      stErrorIdent,
    BusType          := stBusType,
    BusConfig        := stBusConfig,
    Slot             := uiSlot,
    InProcData       := arInProcData,
    OutputData       := arOutputData,
    OutProcData_0=>  arOutProdata_0,
    AnalogData=>     stAnalogData);

```

Lesen der Eingangswerte:

```
rTemperature := arOutputData[1].NormalisedValue;
```

Schritt 3: Verknüpfen der Prozessdaten im E/A-Abbild am Modul in IndraWorks

Verknüpfen der Prozessdaten im E/A-Abbild mit dem im SPS Programm deklarierten Variablen für das Modul S20-AI-4-UTH.

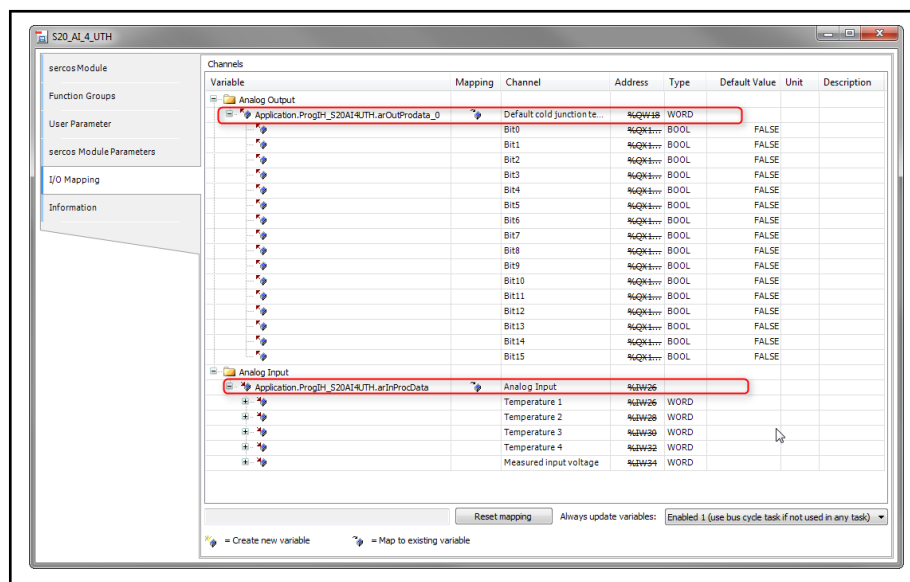


Abb. 23-10: Schritt 3

23.6.5 IH_S20AI8RTD**Beispielprogramm**

Das Beispielprogramm dient zum Lesen von Analogwerten vom S20-AI-8-RTD Modul.

Der verwendete Feldbus ist am Eingang "BusType" mithilfe der Variable "en-BusType" definiert. Die Adresse bzw. der Name des Buskopplers wird mit "stBusConfig" definiert. Der Modulsteckplatz wird am Eingang "Slot" definiert. TRUE am Eingang "bEnable" übernimmt die Konfiguration und der Funktionsbaustein liest die aktuelle Konfiguration des Moduls.

Schritt 1: Modulkonfiguration in IndraWorks Engineering für Kanal 1 zum Messen eines PT-100 Sensor mit der Auflösung 0.1°C in 3-Leiter Technik und einem Widerstand von 100 Ohm, Filter 120 ms

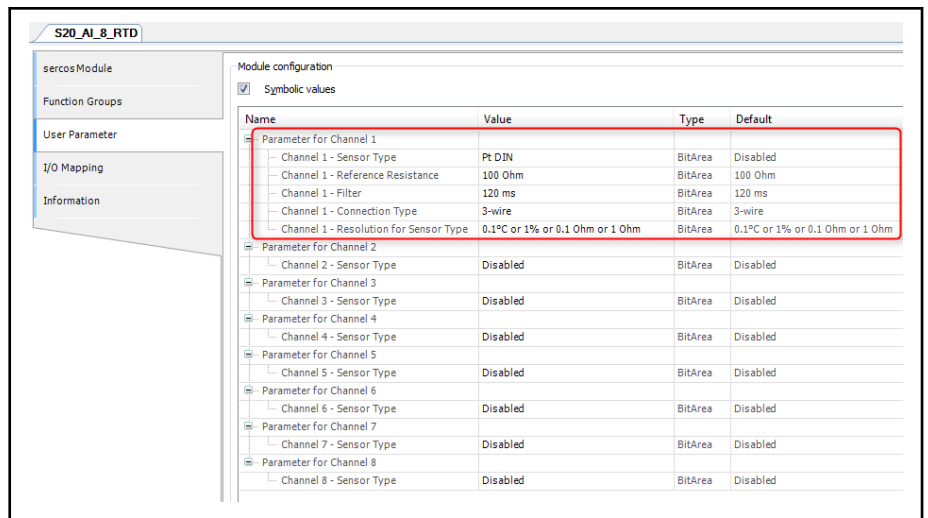


Abb. 23-11: Schritt 1

Schritt 2: Variablendeklaration für das Modul S20-AI-8-RTD am Sercos-Buskoppler

Deklaration:

Programm:

```

PROGRAM ProgIH_S20AI8RTD
VAR
  fbIH_S20AI8RTD: IH_S20AI8RTD; // FB Instance
  bEnable:       BOOL; //TRUE: Function is enabled
  bInOperation:  BOOL; //TRUE: Function Block is working on its actual task
  bError:        BOOL; //TRUE: Indicates an error
  ErrorID:       ERROR_CODE; //Class of error
  ErrorIdent:    ERROR_STRUCT; //Detailed information about error
  enBusType:     IH_S20_BUSTYPE; //Bus type selection
  stBusConfig:   IH_S20_BUS_CONFIG; //Bus configuration parameters
  uiSlot:        UINT; //Module slot address. Bus coupler : 0,
                  //1st Module after coupler:1 ... 63rd Module after coupler: 63
  arInProcData:  ARRAY[1..8] OF WORD; //Input Process variables
  arOutputData:  ARRAY[1..8] OF IH_S20_TEMP_DATA; //Structure for Output Data
  rTemperature:  REAL; // read Temperature value
END_VAR

```

Variableninitialisierung für Sercos-Buskoppler:

Initialisierung der Variablen vom Funktionsbaustein für das Analogmodul, angeschlossen am Sercos-Buskoppler, mit Sercos-Adresse 69 und Modul an Steckplatz 1, im Programm.

Variableninitialisierung:

Programm:

```

enBusType      := IH_S20_BUSTYPE.SERCOS; // Connected fieldbus coupler
stBusConfig.Sercos.BusMaster := IL_BUSMASTER.IL_BUSMASTER_0;
stBusConfig.Sercos.SercosAdr := 69; //Buscoupler address
uiSlot         := 1; // Module position

```

Programm:

Programm:

```

fbIH_S20AI8RTD (
  Enable      := bEnable,
  InOperation=> bInOperation,
  Error=>      bError,
  ErrorID=>    ErrorID,
  ErrorIdent=> ErrorIdent,

```

```

BusType      := enBusType,
BusConfig    := stBusConfig,
Slot         := uiSlot,
InProcData   := arInProcData,
OutputData   := arOutputData);

```

Lesen der analogen Eingangswerte:

```
rTemperature := arOutputData[1].NormalisedValue;
```

Schritt 3: Verknüpfen der Prozessdaten im E/A-Abbild am Modul in IndraWorks

Verknüpfen der Prozessdaten im E/A-Abbild mit dem im SPS Programm deklarierten Variablen für das Modul S20-AI-8-RTD.

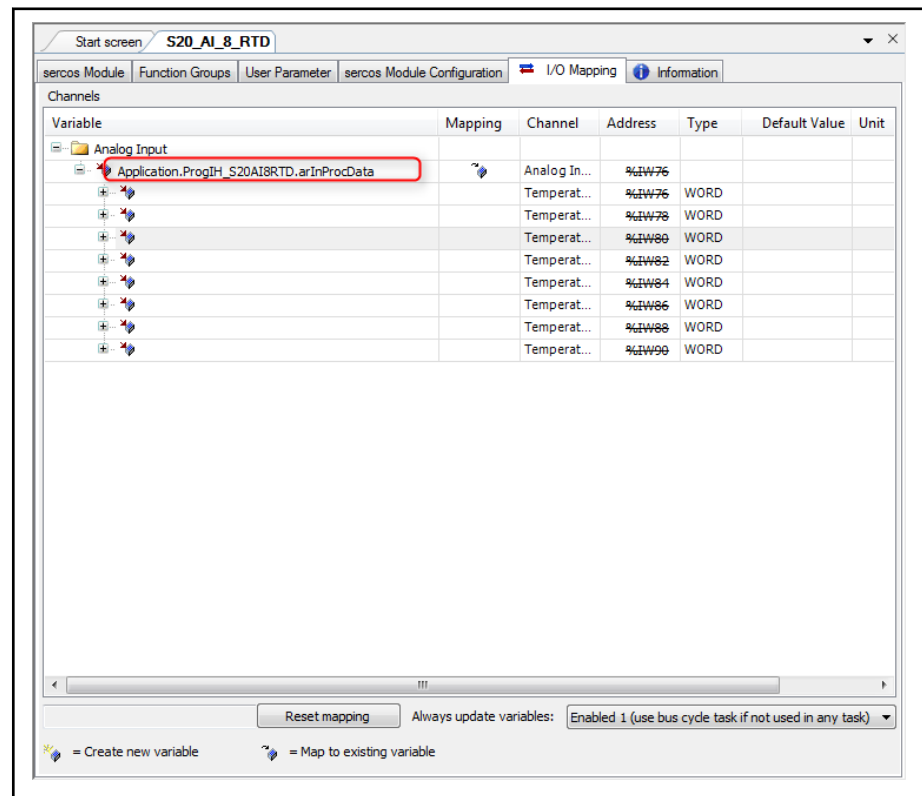


Abb. 23-12: Schritt 3

23.6.6 IH_S20AI8UTH

Beispielprogramm Das Beispielprogramm dient zum Lesen von Analogwerten vom S20-AI-8-UTH Modul.

Der verwendete Feldbus ist am Eingang "BusType" mithilfe der Variable "en-BusType" definiert. Die Adresse bzw. der Name des Buskopplers wird mit "stBusConfig" definiert. Der Modulsteckplatz wird am Eingang "Slot" definiert. TRUE am Eingang "bEnable" übernimmt die Konfiguration und der Funktionsbaustein liest die aktuelle Konfiguration des Moduls.

Schritt 1: Modulkonfiguration in IndraWorks Engineering für Kanal 1 zum Messen eines K-Typ Sensor mit der Auflösung 0.1°C und dem Filter 120 ms

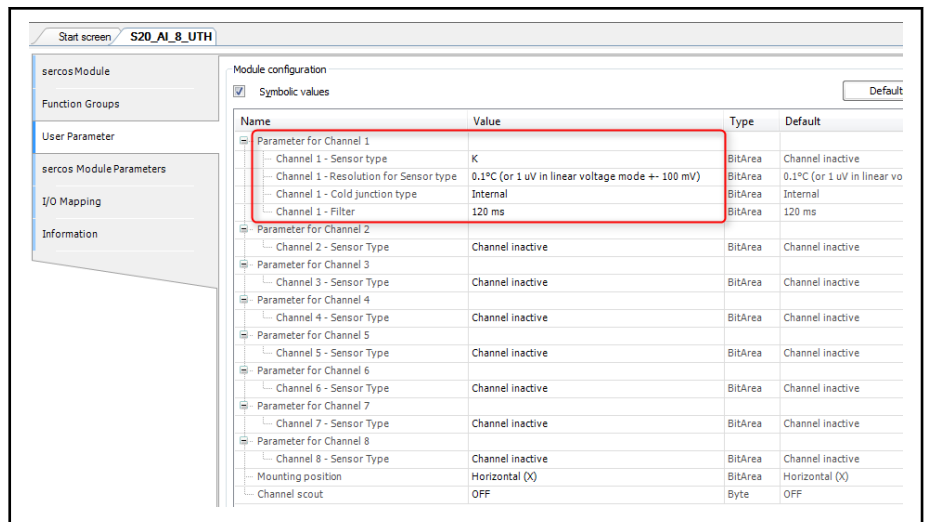


Abb. 23-13: Schritt 1

Schritt 2: Variablendeklaration für das Modul S20-AI-8-UTH am Sercos-Bus-koppler

Deklaration:

Programm:

```

PROGRAM ProgIH_S20AI8UTH
VAR
    fbIH_S20AI8UTH:    IH_S20AI8UTH; // FB Instance
    bEnable:           BOOL; //TRUE: Function is enabled
    bInOperation:      BOOL; //TRUE: Function Block is working on its actual task
    bError:            BOOL; //TRUE: Indicates an error
    enErrorID:         ERROR_CODE; //Class of error
    stErrorIdent:      ERROR_STRUCT; //Detailed information about error
    stBusType:         IH_S20_BUSTYPE; ; //Bus type selection
    stBusConfig:       IH_S20_BUS_CONFIG; //Bus configuration parameters
    uiSlot:            UINT := 1; //Module slot address. Bus coupler : 0,
                        //1st Module after coupler:1 ... 63rd Module after coupler: 63

    arInProcData:      ARRAY [1..9] OF WORD; //Input process data
    arOutputData:      ARRAY [1..8] OF IH_S20_TEMP_DATA; //Contains the normalized value
                        //and the diagnostic code for channels 1...8.
    arOutProdata_0:    WORD; //Output process data(word 0)
    stAnalogData:      IH_S20_TEMP_DATA; //Contains the normalized value
                        //and diagnostic code of analog voltage input ±5V
    rTemperature:      REAL; // Read temperature value
END_VAR

```

Variableninitialisierung für Sercos-Buskoppler:

Initialisierung der Variablen vom Funktionsbaustein für das Analogmodul, angeschlossen am Sercos-Buskoppler mit Sercos-Adresse 69 und Modul an Steckplatz 1, im Programm.

Variableninitialisierung:

Programm:

```

enBusType           := IH_S20_BUSTYPE.SERCOS; // Connected fieldbus coupler
stBusConfig.Sercos.BusMaster := IL_BUSMASTER.IL_BUSMASTER_0;
stBusConfig.Sercos.SercosAdr := 69; //Buscoupler address
uiSlot              := 1; // Module position

```

Programm:

Programm:

```

fbIH_S20AI8UTH (
    Enable           := bEnable,

```

```

InOperation=> bInOperation,
Error=> bError,
ErrorID=> enErrorID,
ErrorIdent=> stErrorIdent,
BusType      := stBusType,
BusConfig    := stBusConfig,
Slot         := uiSlot,
InProcData   := arInProcData,
OutputData   := arOutputData,
OutProcData_0=> arOutProdata_0,
AnalogData=> stAnalogData);

```

Lesen der Eingangswerte:

```
rTemperature := arOutputData[1].NormalisedValue;
```

Schritt 3: Verknüpfen der Prozessdaten im E/A-Abbild am Modul in IndraWorks

Verknüpfen der Prozessdaten im E/A-Abbild mit dem im SPS Programm deklarierten Variablen für das Modul S20-AI-8-UTH.

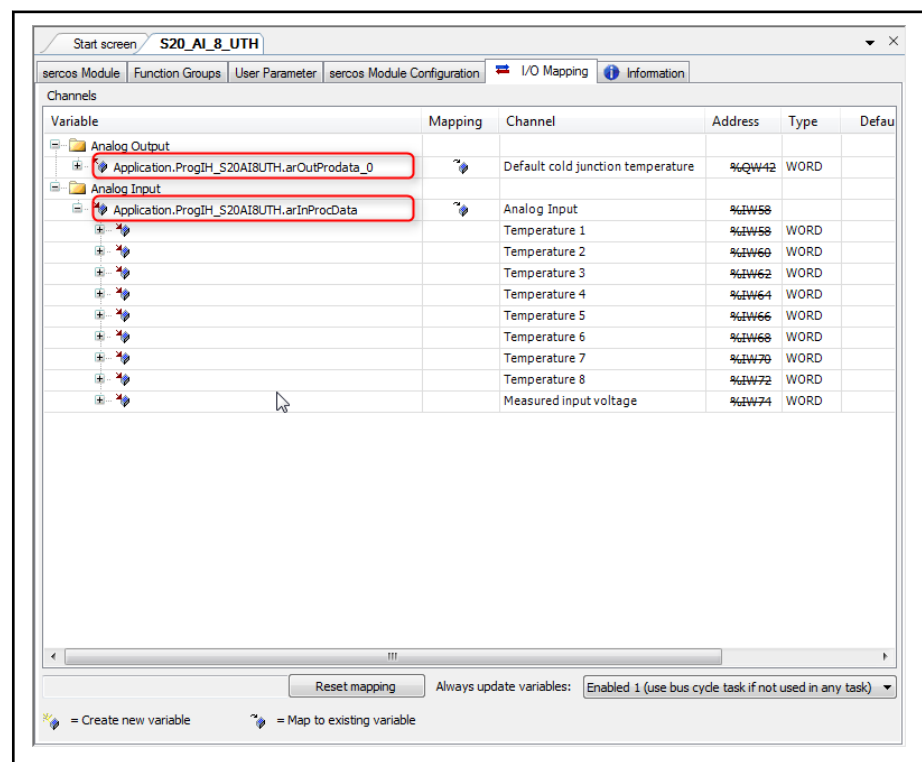


Abb. 23-14: Schritt 3

23.7 Analoge Ausgangsmodule (AnalogIO)

23.7.1 IH_S20AO8

Beispielprogramm Das Beispielprogramm dient zum Schreiben von Analogwerten vom S20-AO-8 Modul.

Der verwendete Feldbus ist am Eingang "BusType" mithilfe der Variable "en-BusType" definiert. Die Adresse bzw. der Name des Buskopplers wird mit "stBusConfig" definiert. Der Modulsteckplatz wird am Eingang "Slot" definiert.

TRUE am Eingang "bEnable" übernimmt die Konfiguration und der Funktionsbaustein liest die aktuelle Konfiguration des Moduls.

Schritt 1: Modulkonfiguration in IndraWorks Engineering für Kanal 1 zum Ausgeben von 0-10 V und Kanal-2 von +/-5 V

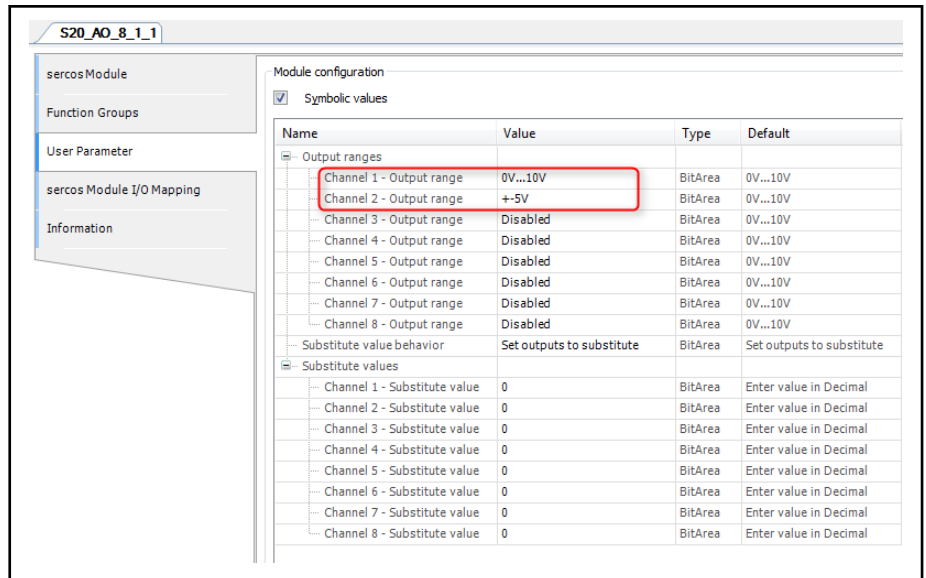


Abb. 23-15: Schritt 1

Schritt 2: Variablendeklaration für das Modul S20-AO-8 am Sercos-Buskoppler

Deklaration:

Programm:

```
PROGRAM ProgIH_S20AO8
VAR
  fbIH_S20AO8:      IH_S20AO8; // FB Instance
  bEnable:          BOOL; //TRUE: Function is enabled
  bInOperation:     BOOL; //TRUE: Function Block is working on its actual task
  bError:           BOOL; //TRUE: Indicates an error
  ErrorID:          ERROR_CODE; //Class of error
  ErrorIdent:       ERROR_STRUCT; //Detailed information about error
  enBusType:        IH_S20_BUSTYPE; //Bus type selection
  stBusConfig:      IH_S20_BUS_CONFIG; //Bus configuration parameters
  uiSlot:           UINT; //Module slot address. Bus coupler : 0,
                    //1st Module after coupler:1 ... 63rd Module after coupler: 63
  arChannelConfig:  ARRAY[1..8] OF IH_S20_AO_CONFIG; //Structure for channel configuration
  bBusy:           BOOL; //Busy Signal used for PDI handling in sequence
  arOutProcData:    ARRAY [1..8] OF WORD; // Outprocess data
  DiagCode:        ARRAY [1..8] OF STRING(14); //Channel diagnostic information
END_VAR
```

Variableninitialisierung für Sercos-Buskoppler:

Initialisierung der Variablen vom Funktionsbaustein für das Analogmodul, angeschlossen am Sercos-Buskoppler mit Sercos-Adresse 69 und Modul an Steckplatz 1 mit Festlegen der Limits je Kanal, im Programm.

Schritt 3: Variableninitialisierung

Programm:

```
enBusType          := IH_S20_BUSTYPE.SERCOS; // Connected fieldbus coupler
stBusConfig.Sercos.BusMaster := IL_BUSMASTER.IL_BUSMASTER_0;
stBusConfig.Sercos.SercosAdr := 69; //Buscoupler address
uiSlot             := 1; // Module position
//Ch-1 Configuration
arChannelConfig[1].HighLimit := 10.0;
arChannelConfig[1].LowLimit  := 0.0;
//Ch-2 Configuration
```

```
arChannelConfig[2].HighLimit := 5.0;
arChannelConfig[2].LowLimit  := -5.0;
```

Schritt 4: Programm:

Programm:

```
fbIH_S20AO8(
    Enable           := bEnable,
    InOperation=>    bInOperation,
    Error=>          bError,
    ErrorID=>        ErrorID,
    ErrorIdent=>     ErrorIdent,
    BusType          := enBusType,
    BusConfig        := stBusConfig,
    Slot             := uiSlot,
    ChannelConfig    := arChannelConfig,
    OutProcData      := arOutProcData,
    DiagCode         := DiagCode);
```

Ausgeben des analogen Ausgangswertes:

```
arChannelConfig[1].NormalizedValue := 10.0;
arChannelConfig[2].NormalizedValue := -5.0;
```

Schritt 5: Verknüpfen der Prozessdaten im E/A-Abbild am Modul in IndraWorks

Verknüpfen der Prozessdaten im E/A-Abbild mit dem im SPS Programm deklarierten Variablen für das Modul S20-AO-8.

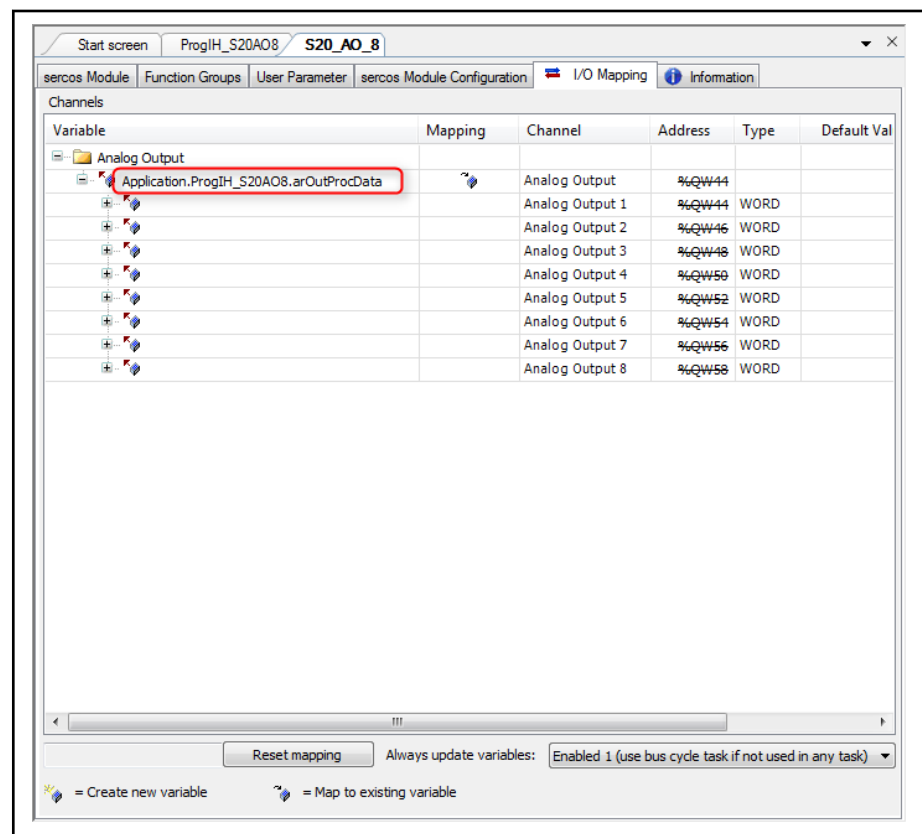


Abb. 23-16: Schritt 5

23.7.2 IH_S20AO4

Beispielprogramm Das Beispielprogramm dient zum Schreiben von Analogwerten vom S20-AO-4 Modul.

Der verwendete Feldbus ist am Eingang "BusType" mithilfe der Variable "enBusType" definiert. Die Adresse bzw. der Name des Buskopplers wird mit "stBusConfig" definiert. Der Modulsteckplatz wird am Eingang "Slot" definiert.

TRUE am Eingang "bEnable" übernimmt die Konfiguration und der Funktionsbaustein liest die aktuelle Konfiguration des Moduls.

Schritt 1: Modulkonfiguration in IndraWorks Engineering für Kanal 1 zum Ausgeben von 0-10 V und Kanal-2 von +/-5 V

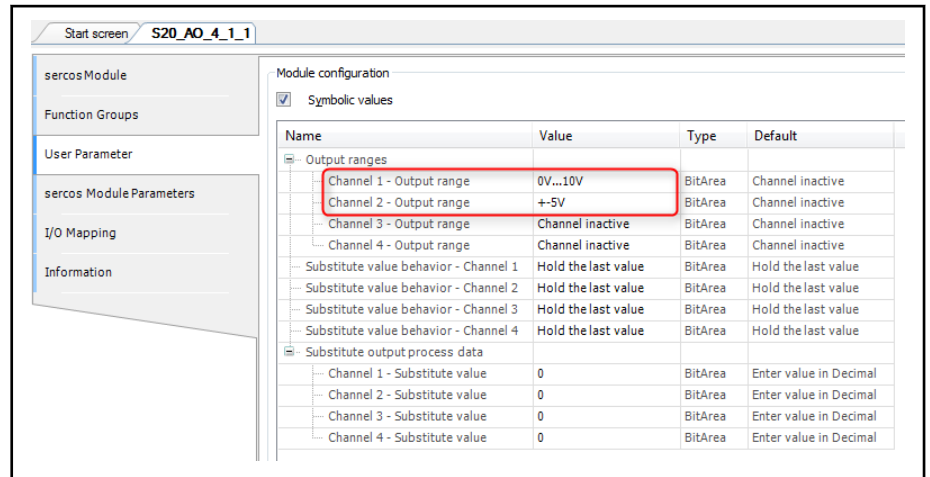


Abb. 23-17: Schritt 1

Schritt 2: Variablendeklaration für das Modul S20-AO-4 am Sercos-Buskoppler

Deklaration:

Programm:

```
PROGRAM ProgIH_S20AO4
VAR
    fbIH_S20AO4:      IH_S20AO4; // FB Instance
    bEnable:          BOOL; //TRUE: Function is enabled
    bInOperation:     BOOL; //TRUE: Function Block is working on its actual task
    bError:           BOOL; //TRUE: Indicates an error
    bErrorID:         ERROR_CODE; //Class of error
    ErrorIdent:       ERROR_STRUCT; //Detailed information about error
    enBusType:        IH_S20_BUSTYPE; //Bus type selection
    stBusConfig:      IH_S20_BUS_CONFIG; //Bus configuration parameters
    uiSlot:           UINT; //Module slot address. Bus coupler : 0,
                        //1st Module after coupler:1 ... 63rd Module after coupler: 63
    arChannelConfig:  ARRAY[1..4] OF IH_S20_AO_CONFIG; //Structure for channel configuration
    bBusy:           BOOL; //Busy Signal used for PDI handling in sequence
    arOutProcData:    ARRAY [1..4] OF WORD; //Outprocess data
    DiagCode:        ARRAY [1..4] OF STRING(14); //Diagnostics message
END_VAR
```

Variableninitialisierung für Sercos-Buskoppler:

Initialisierung der Variablen vom Funktionsbaustein für das Analogmodul, angeschlossen am Sercos-Buskoppler, mit Sercos-Adresse 69 und Modul an Steckplatz 1 mit Festlegen der Limits je Kanal, im Programm.

Variableninitialisierung:

Programm:

```
enBusType      := IH_S20_BUSTYPE.SERCOS; // Connected fieldbus coupler
stBusConfig.Sercos.BusMaster := IL_BUSMASTER.IL_BUSMASTER_0;
```

```

stBusConfig.Sercos.SercosAdr := 69; //Buscoupler address
uiSlot                        := 1; // Module position

//Ch-1 Configuration
arChannelConfig[1].HighLimit := 10.0;
arChannelConfig[1].LowLimit  := 0.0;
//Ch-2 Configuration
arChannelConfig[2].HighLimit := 5.0;
arChannelConfig[2].LowLimit  := -5.0;

```

Programm:*Programm:*

```

fbIH_S20AO4 (
    Enable           := bEnable,
    InOperation=>    bInOperation,
    Error=>           bError,
    ErrorID=>         ErrorID,
    ErrorIdent=>      ErrorIdent,
    BusType          := enBusType,
    BusConfig        := stBusConfig,
    Slot             := uiSlot,
    ChannelConfig     := arChannelConfig,
    OutProcData       := arOutProcData,
    DiagCode         := DiagCode);

```

Ausgeben der analogen Ausgangswerte:

```

arChannelConfig[1].NormalizedValue := 10.0;
arChannelConfig[2].NormalizedValue := -5.0;

```

Schritt 3: Verknüpfen der Prozessdaten im E/A-Abbild am Modul in IndraWorks

Verknüpfen der Prozessdaten im E/A-Abbild mit dem im SPS-Programm deklarierten Variablen für das Modul S20-AO-4.

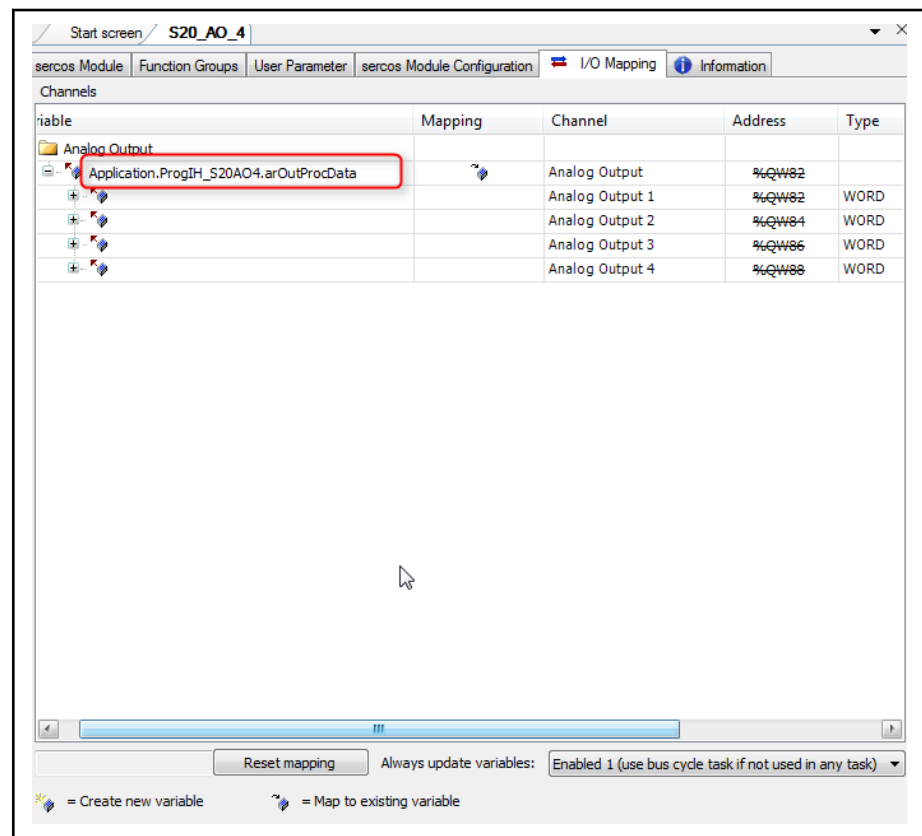


Abb. 23-18: Schritt 3

23.8 Technologiemodule (TechnologyIO)

23.8.1 IH_S20SsiAo1

Beispielprogramm Beispiel 1: Modul S20-SSI-AO-1/1 am Sercos-Buskoppler S20-S3-BK+

Das Beispielprogramm dient zum Lesen des SSI-Kanals und Schreiben von Analogwerten vom S20-SSI-AO-1/1 Modul.

Der verwendete Feldbus ist am Eingang "BusType" mithilfe der Variable "enBusType" definiert. Die Adresse bzw. der Name des Buskopplers wird mit "stBusConfig" definiert. Der Modulsteckplatz wird am Eingang "Slot" definiert.

TRUE am Eingang "bEnable" übernimmt die Konfiguration und der Funktionsbaustein liest die aktuelle Konfiguration des Moduls.

Schritt 1: Modulkonfiguration in IndraWorks Engineering für den Ausgang $\pm 10V$ und den SSI-Kanal

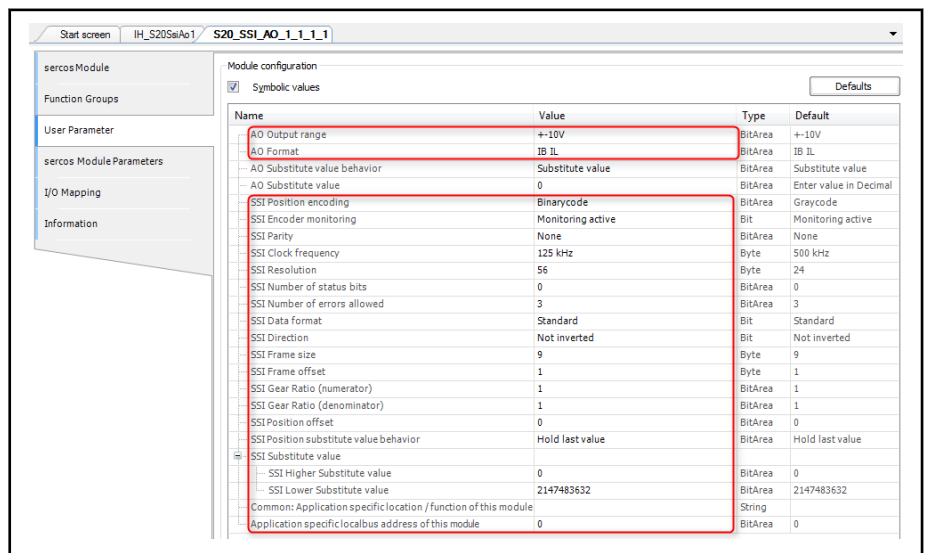


Abb. 23-19: Schritt 1

Schritt 2: Variablendeklaration für das Modul S20-SSI-AO-1/1 am Sercos-Buskoppler

Deklaration:

Programm:

```
PROGRAM ProgIH_S20SsiAo1
VAR
  fbIH_S20SsiAo1: IH_S20SsiAo1; // Function block instance
  bEnable: BOOL; //TRUE: Function is enabled
  bInOperation: BOOL; //TRUE: Function Block is working on its actual task
  bError: BOOL; //TRUE: Indicates an error
  ErrorID: ERROR_CODE; //Class of error
  ErrorIdent: ERROR_STRUCT; //Detailed information about error
  enBusType: IH_S20_BUSTYPE; //Bus type selection
  stBusConfig: IH_S20_BUS_CONFIG; //Bus configuration parameters
  uiSlot: UINT; //Module slot address. Bus coupler : 0,
  //1st Module after coupler:1 ... 63rd Module after coupler: 63
  bSetZero: BOOL; // Resetting position to Zero
  stAOChannelConfig: IH_S20_AO_CONFIG; // Structure for AO channel configuration
  arInProcData: ARRAY [1..6] OF WORD; // Input process variable
  arOutProcData: ARRAY [1..6] OF WORD; // Output process variable
  stSSIOutData: IH_S20_SSI_DATA; // Structure for SSI output status and position data
  stAODiagCode: IH_S20_DIAG_CODE; // Enumeration for handling AO diagnostics
  wDiagIn: WORD; // SSI status word on sercos
```

RIH_IndraControl_S20.library

```

arSSIPosition:    ARRAY [1..4] OF WORD; // SSI position on sercos
wAnalogOut:       WORD; // Analog control on sercos
byDiagOut:        BYTE; // SSI control word on sercos
rEnc_Position:    REAL; // Encoder actual position
END_VAR

```

Variableninitialisierung für Sercos-Buskoppler:

Initialisierung der Variablen vom Funktionsbaustein für das Technologiemodul, angeschlossen am Sercos-Buskoppler mit Sercos-Adresse 69 und Modul an Steckplatz 1, im Programm.

Variableninitialisierung:*Programm:*

```

enBusType          := IH_S20_BUSTYPE.SERCOS; // Connected fieldbus coupler
stBusConfig.Sercos.BusMaster := IL_BUSMASTER.IL_BUSMASTER_0;
stBusConfig.Sercos.SercosAdr := 69; //Buscoupler address
uiSlot             := 1; // Module position

// AO Channel configuration
stAoConfig.HighLimit := 10.0;
stAoConfig.LowLimit  := 0.0;

```

Programm:*Programm:*

```

fbIH_S20SsiAo1 (
  Enable          := bEnable,
  InOperation=>   bInOperation,
  Error=>          bError,
  ErrorID=>        ErrorID,
  ErrorIdent=>     ErrorIdent,
  BusType         := enBusType,
  BusConfig       := stBusConfig,
  Slot            := uiSlot,
  SetZero         := bSetZero,
  AOChannelConfig:= stAOChannelConfig,
  InProcData      := arInProcData,
  OutProcData     := arOutProcData,
  SSIOutData=>    stSSIOutData,
  AODiagCode=>    stAODiagCode);

IF enBusType = IH_S20_BUSTYPE.SERCOS THEN // Process data mapping function on sercos
IH_S20SsiAo1DataMap(
  DiagIn          := wDiagIn,
  SSIPosition     := arSSIPosition,
  OutProcData     := arOutProcData,
  InProcData=>    arInProcData,
  AnalogOut=>     wAnalogOut,
  DiagOut=>       byDiagOut);
END_IF

```



Besonderheit am Sercos zum Swappen der Prozessdaten. Zur Unterstützung wird die Funktion **IH_S20SsiAo1DataMap** verwendet.

Der Rückgabewert der Funktion **IH_S20SsiAo1DataMap** bei erfolgreicher Abarbeitung ist "TRUE".

Lesen der Eingangswerte und Schreiben der Ausgangswerte

```

stAoConfig.NormalizedValue := 5.0; // Normalised analog
output Value

rEnc_Position := SSIOutData.ActPos; // Encoder actual po-
sition

```

Schritt 3: Verknüpfen der Prozessdaten im E/A-Abbild am Modul in IndraWorks

Verknüpfen der Prozessdaten im E/A-Abbild, mit dem im SPS-Programm deklarierten Variablen für das Modul S20-SSI-AO-1/1.

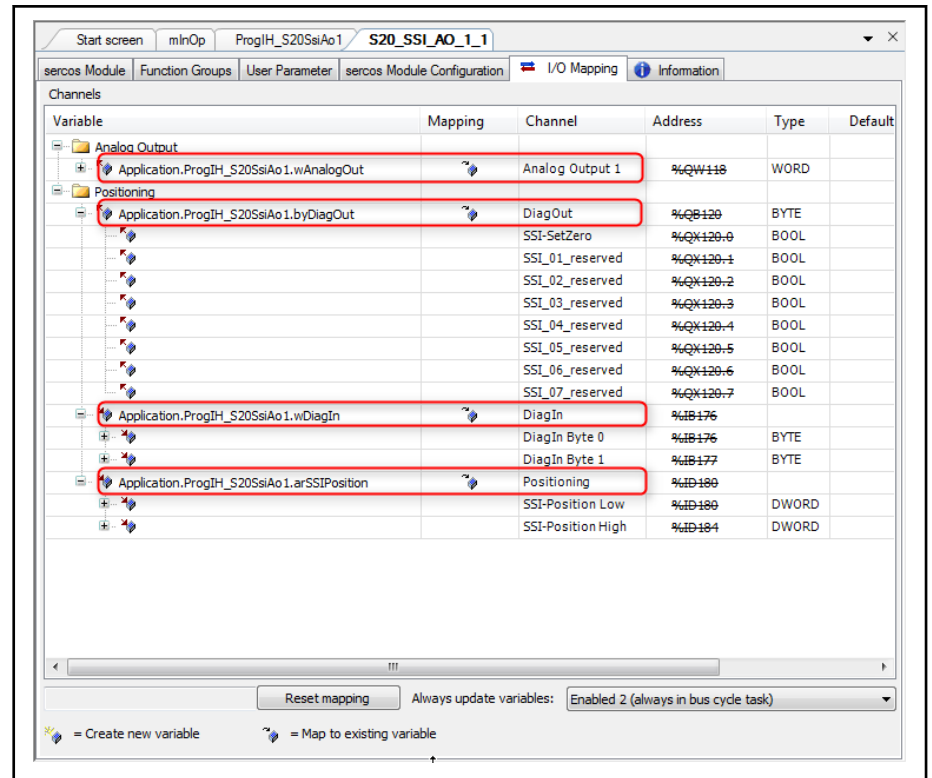


Abb. 23-20: Beispiel 1, Schritt 3

Beispiel 2: Modul S20-SSI-AO-1/1 am Profinet-Buskoppler S20-PN-BK+

Das Beispielprogramm dient zum Lesen des SSI-Kanals und Schreiben von Analogwerten vom S20-SSI-AO-1/1 Modul.

Der verwendete Feldbus ist am Eingang "BusType" mithilfe der Variable "enBusType" definiert. Die Adresse bzw. der Name des Buskopplers wird mit "stBusConfig" definiert. Der Modulsteckplatz wird am Eingang "Slot" definiert.

TRUE am Eingang "bEnable" übernimmt die Konfiguration und der Funktionsbaustein liest die aktuelle Konfiguration des Moduls.

Schritt 1: Modulkonfiguration in IndraWorks Engineering für den Ausgang $\pm 10V$ und den SSI-Kanal

Konfiguration ist analog zum Beispiel 1 am Sercos-Buskoppler.

Schritt 2: Variablendeklaration für das Modul S20-SSI-AO-1/1 am Profinet-Buskoppler

Deklaration:

Programm:

```
PROGRAM ProgIH_S20SsiAo1
VAR
  fbIH_S20SsiAo1:  IH_S20SsiAo1; // Function block instance
  bEnable:         BOOL; //TRUE: Function is enabled
  bInOperation:    BOOL; //TRUE: Function Block is working on its actual task
  bError:          BOOL; //TRUE: Indicates an error
  ErrorID:         ERROR_CODE; //Class of error
  ErrorIdent:      ERROR_STRUCT; //Detailed information about error
  enBusType:       IH_S20_BUSTYPE; //Bus type selection
  stBusConfig:     IH_S20_BUS_CONFIG; //Bus configuration parameters
```

```

uiSlot:          UINT; //Module slot address. Bus coupler : 0,
                  //1st Module after coupler:1 ... 63rd Module after coupler: 63
bSetZero:        BOOL; // Resetting position to Zero
stAOChannelConfig: IH_S20_AO_CONFIG; // Structure for AO channel configuration
arInProcData:     ARRAY [1..6] OF WORD; // Input process variable
arOutProcData:    ARRAY [1..6] OF WORD; // Output process variable
stSSIOutData:     IH_S20_SSI_DATA; // Structure for SSI output status and position data
stAODiagCode:     IH_S20_DIAG_CODE; // Enumeration for handling AO diagnostics
END_VAR

```

Variableninitialisierung für Profinet-Buskoppler:

Initialisierung der Variablen vom Funktionsbaustein für das Technologiemodul, angeschlossen am Profinet-Buskoppler mit Stationsname "s20-pn-bk" und Modul an Steckplatz 1, im Programm.

Variableninitialisierung:

Programm:

```

enBusType        := IH_S20_BUSTYPE.PROFINET // Connected fieldbus coupler
stBusConfig.Profinet.BusMaster := IL_BUSMASTER.IL_BUSMASTER_0;
stBusConfig.Profinet.IdentStr  := 's20-pn-bk'; //Buscoupler address
uiSlot           := 1; // Module position

// AO Channel configuration
stAoConfig.HighLimit := 10.0;
stAoConfig.LowLimit  := 0.0;

```

Programm:

Programm:

```

fbIH_S20SsiAo1 (
  Enable           := bEnable,
  InOperation=>    bInOperation,
  Error=>           bError,
  ErrorID=>         ErrorID,
  ErrorIdent=>      ErrorIdent,
  BusType          := enBusType,
  BusConfig        := stBusConfig,
  Slot             := uiSlot,
  SetZero          := bSetZero,
  AOChannelConfig  := stAOChannelConfig,
  InProcData       := arInProcData,
  OutProcData      := arOutProcData,
  SSIOutData=>     stSSIOutData,
  AODiagCode=>     stAODiagCode);

```

Lesen der Eingangswerte und Schreiben der Ausgangswerte

```

stAoConfig.NormalizedValue := 5.0; // Normalised analog
output Value

rEnc_Position := SSIOutData.ActPos; // Encoder actual po-
sition

```

Verknüpfen der Prozessdaten im E/A-Abbild am Modul in IndraWorks:

Das Verknüpfen der Prozessdaten im E/A-Abbild mit dem im SPS-Programm deklarierten Variablen für das Modul S20-SSI-AO-1/1.

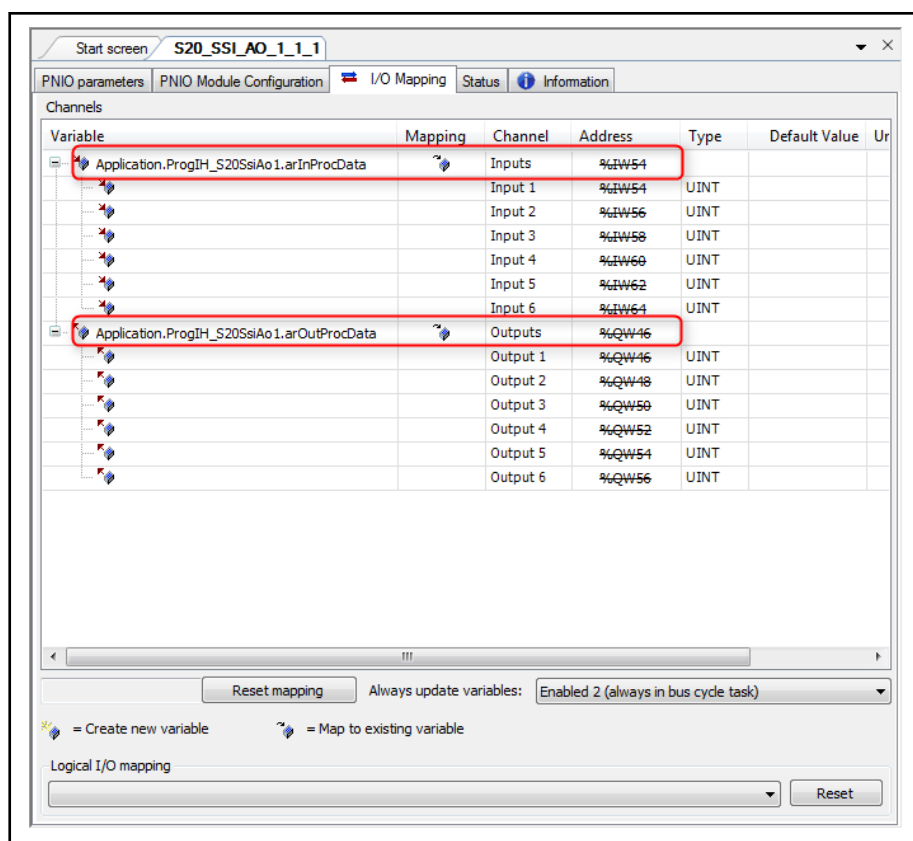


Abb. 23-21: Beispiel 2, Schritt 2

23.8.2 IH_S20CntInc

Beispielprogramm **Beispiel 1: Modul S20-CNT-INC-2/2 am Sercos-Buskoppler S20-S3-BK+**

Das Beispielprogramm dient zum Lesen des Counterwertes Kanal 1 des S20-CNT-INC-2/2 Modul mit einem zyklischen Wert 0-5000.

Der verwendete Feldbus ist am Eingang "BusType" mithilfe der Variable "en-BusType" definiert. Die Adresse bzw. der Name des Buskopplers wird mit "stBusConfig" definiert. Der Modulsteckplatz wird am Eingang "Slot" definiert. TRUE am Eingang "bEnable" übernimmt die Konfiguration und der Funktionsbaustein liest die aktuelle Konfiguration des Moduls.

Schritt 1: Modulkonfiguration in IndraWorks Engineering für den Counter Kanal 1

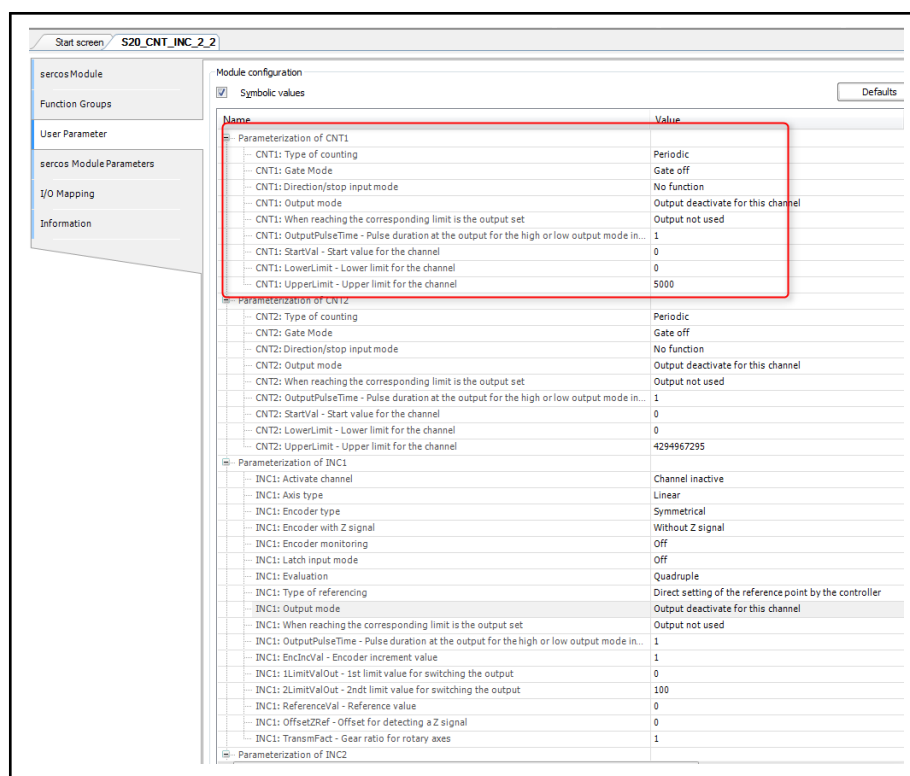


Abb. 23-22: Beispiel 1, Schritt 1

Schritt 2: Variablendeklaration für das Modul S20-CNT-INC-2/2 am Sercos-Buskoppler

Deklaration:

Programm:

```

PROGRAM ProgIH_S20CntInc
VAR
    fbIH_S20CntInc:    IH_S20CntInc; // FB Instance
    bEnable:          BOOL; //TRUE: Function is enabled
    bInOperation:     BOOL; //TRUE: Function Block is working on its actual task
    bError:           BOOL; //TRUE: Indicates an error
    ErrorID:          ERROR_CODE; //Class of error
    ErrorIdent:       ERROR_STRUCT; //Detailed information about error
    enBusType:        IH_S20_BUSTYPE := IH_S20_BUSTYPE.SERCOS; //Bus type selection
    stBusConfig:      IH_S20_BUS_CONFIG; //Bus configuration parameters
    uiSlot:           UINT; //Module slot address. Bus coupler : 0,
                        //1st Module after coupler:1 ... 63rd Module after coupler: 63
    byChannelSelect:  BYTE; // Channel selection
    arCntConfig:      ARRAY[1..2] OF IH_S20_CNT_CONFIG; //Structure for Counter configuration
    arIncConfig:      ARRAY[1..2] OF IH_S20_INC_CONFIG; //Structure for incremental Encoder configuration
    arInProcData:     ARRAY[1..14] OF WORD; //Input Process variable
    arOutProcData:    ARRAY[1..14] OF WORD; //Output Process variable
    bResetDiag:       BOOL; //It deletes the diagnostics memory
                        //and acknowledge the diagnostic messages in the module
    bBusy:            BOOL; //Busy Signal used for PDI handling in sequence
    arCntOutputData:  ARRAY[1..2] OF IH_S20_CNT_DATA; //Structure for counter Output Data
    arIncOutputData:  ARRAY[1..2] OF IH_S20_INC_DATA; //Structure for incremental encoder Output Data
    bCntStart:        BOOL; //TRUE- Start counting
    bCntStop:         BOOL; //TRUE-Stop counting
    bCntDir:          BOOL; //TRUE- Count direction negative
    bCntReset:        BOOL; //TRUE- Count reset
    wCntDiagIn:       WORD; // Cnt Status
    arCntCounterIn:   ARRAY[1..2] OF DWORD; //Counter values
    wIncDiagIn:       WORD; // Inc Status
    arIncPosition:    ARRAY[1..4] OF DWORD; // INC position
    wCntDiagOut:      WORD; //CNT control
    arCntCounterOut:  ARRAY[1..4] OF DWORD; // Counter limits
    wIncDiagOut:      WORD; //INC control
END_VAR

```

Variableninitialisierung für Sercos-Buskoppler:

Initialisierung der Variablen vom Funktionsbaustein für das Technologiemodul, angeschlossen am Sercos-Buskoppler, mit Sercos-Adresse 69 und Modul an Steckplatz 1, im Programm.

Variableninitialisierung:

Programm:

```

enBusType           := IH_S20_BUSTYPE.SERCOS; // Connected fieldbus coupler
stBusConfig.Sercos.BusMaster := IL_BUSMASTER.IL_BUSMASTER_0;
stBusConfig.Sercos.SercosAdr := 69; //Buscoupler address
uiSlot              := 1; // Module position
byChannelSelect      := 2#0000_0001; //Channel-1 is selected

```

Programm:

Programm:

```

fbIH_S20CntInc (
    Enable:= bEnable,
    InOperation=> bInOperation,
    Error=> bError,
    ErrorID=> ErrorID,
    ErrorIdent=> ErrorIdent,
    BusType:= enBusType,
    BusConfig:= stBusConfig,
    Slot:= uiSlot,
    ChannelSelect:= byChannelSelect,
    CntConfig:= arCntConfig,
    IncConfig:= arIncConfig,
    InProcData:= arInProcData,
    OutProcData:= arOutProcData,
    ResetDiag:= bResetDiag,
    CntOutputData=> arCntOutputData,
    IncOutputData=> arIncOutputData);

IF enBusType = IH_S20_BUSTYPE.SERCOS THEN //Process data mapping function on Sercos
    IH_S20CntIncDataMap(
        CntDiagIn:= wCntDiagIn,
        CntCounterIn:= arCntCounterIn,
        IncDiagIn:= wIncDiagIn,
        IncPosition:= arIncPosition,
        OutProcData:= arOutProcData,
        InProcData=> arInProcData,
        IncDiagOut=> wIncDiagOut,
        CntDiagOut:= wCntDiagOut,
        CntCounterOut:= arCntCounterOut);
END_IF

```

Bedienen, Kommandieren des Counters:

Programm:

```

arCntConfig[1].CntStart      := bCntStart; //Start counting
arCntConfig[1].CntStop       := bCntStop; //Stop counting
arCntConfig[1].CntDir        := bCntDir; // Count direction control
arCntConfig[1].CntReset      := bCntReset; // Count reset

```



Besonderheit am Sercos zum Swappen der Prozessdaten. Zur Unterstützung wird die Funktion **IH_S20CntIncDataMap** verwendet.

Der Rückgabewert der Funktion **IH_S20CntIncDataMap** bei erfolgreicher Abarbeitung ist "TRUE".

Schritt 3: Verknüpfen der Prozessdaten im E/A-Abbild am Modul in IndraWorks

Verknüpfen der Prozessdaten im E/A-Abbild mit dem im SPS-Programm deklarierten Variablen für das Modul S20-CNT-INC-2/2.

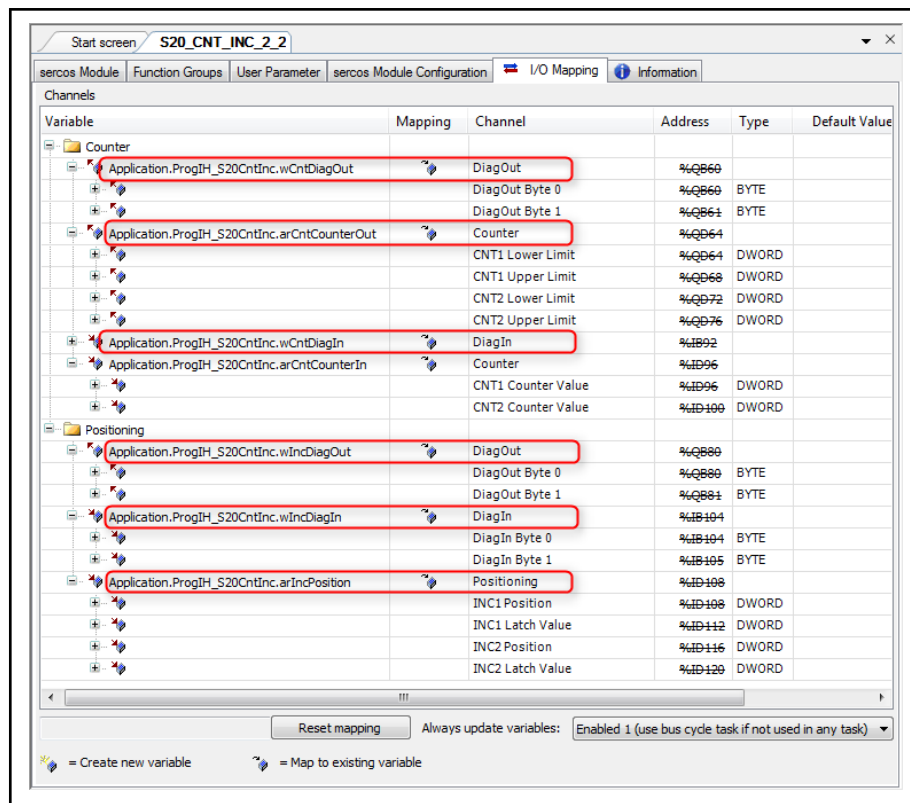


Abb. 23-23: Beispiel 1, Schritt 3

Beispiel 2: Modul S20-CNT-INC-2/2 am Profinet-Buskoppler S20-PN-BK+

Das Beispielprogramm dient zum Lesen des Counterwertes Kanal 1 des S20-CNT-INC-2/2 Modul mit einem zyklischen Wert 0-5000.

Der verwendete Feldbus ist am Eingang "BusType" mithilfe der Variable "enBusType" definiert. Die Adresse bzw. der Name des Buskopplers wird mit "stBusConfig" definiert. Der Modulsteckplatz wird am Eingang "Slot" definiert.

TRUE am Eingang "bEnable" übernimmt die Konfiguration und der Funktionsbaustein liest die aktuelle Konfiguration des Moduls.

Schritt 0: Modulkonfiguration in IndraWorks Engineering für den Counter Kanal 1, analog zum Sercos-Beispiel 1

Schritt 1: Variablendeklaration für das Modul S20-CNT-INC-2/2 am Profinet-Buskoppler

Deklaration:

Programm:

```
PROGRAM ProgIH_S20CntInc
VAR
  fbIH_S20CntInc:  IH_S20CntInc; // FB Instance
  bEnable:         BOOL; //TRUE: Function is enabled
  bInOperation:    BOOL; //TRUE: Function Block is working on its actual task
  bError:          BOOL; //TRUE: Indicates an error
  ErrorID:         ERROR_CODE; //Class of error
  ErrorIdent:      ERROR_STRUCT; //Detailed information about error
  enBusType:       IH_S20_BUSTYPE := IH_S20_BUSTYPE.SERCOS; //Bus type selection
  stBusConfig:     IH_S20_BUS_CONFIG; //Bus configuration parameters
  uiSlot:          UINT; //Module slot address. Bus coupler : 0,
```

```

//1st Module after coupler:1 ... 63rd Module after coupler: 63
byChannelSelect: BYTE; // Channel selection
arCntConfig: ARRAY[1..2] OF IH_S20_CNT_CONFIG; //Structure for Counter configuration
arIncConfig: ARRAY[1..2] OF IH_S20_INC_CONFIG; //Structure for incremental Encoder configuration
arInProcData: ARRAY[1..14] OF WORD; //Input Process variable
arOutProcData: ARRAY[1..14] OF WORD; //Output Process variable
bResetDiag: BOOL; //It deletes the diagnostics memory
//and acknowledge the diagnostic messages in the module
bBusy: BOOL; //Busy Signal used for PDI handling in sequence
arCntOutputData: ARRAY[1..2] OF IH_S20_CNT_DATA; //Structure for counter Output Data
arIncOutputData: ARRAY[1..2] OF IH_S20_INC_DATA; //Structure for incremental encoder Output Data
bCntStart: BOOL; //TRUE- Start counting
bCntStop: BOOL; //TRUE-Stop counting
bCntDir: BOOL; //TRUE- Count direction negative
bCntReset: BOOL; //TRUE- Count reset
END_VAR

```

Initialisierung der Variablen vom Funktionsbaustein für das Technologiemodul, angeschlossen am Profinet-Buskoppler mit Stationsname "s20-pn-bk" und Modul an Steckplatz 1, im Programm.

Variableninitialisierung:

Programm:

```

enBusType := IH_S20_BUSTYPE.PROFINET; // Connected fieldbus coupler
stBusConfig.Profinet.BusMaster := IL_BUSMASTER.IL_BUSMASTER_0;
stBusConfig.Profinet.IdentStr := 's20-pn-bk'; //Buscoupler address
uiSlot := 1; // Module position
byChannelSelect := 2#0000_0001; //Channel-1 is selected

```

Programm:

Programm:

```

fbIH_S20CntInc (
  Enable:= bEnable,
  InOperation=> bInOperation,
  Error=> bError,
  ErrorID=> ErrorID,
  ErrorIdent=> ErrorIdent,
  BusType:= enBusType,
  BusConfig:= stBusConfig,
  Slot:= uiSlot,
  ChannelSelect:= byChannelSelect,
  CntConfig:= arCntConfig,
  IncConfig:= arIncConfig,
  InProcData:= arInProcData,
  OutProcData:= arOutProcData,
  ResetDiag:= bResetDiag,
  CntOutputData=> arCntOutputData,
  IncOutputData=> arIncOutputData);

```

Bedienen, Kommandieren des Counters:

Programm:

```

arCntConfig[1].CntStart := bCntStart; //Start counting
arCntConfig[1].CntStop := bCntStop; //Stop counting
arCntConfig[1].CntDir := bCntDir; // Count direction control
arCntConfig[1].CntReset := bCntReset; // Count reset

```

Schritt 4: Verknüpfen der Prozessdaten im E/A-Abbild am Modul in IndraWorks

Verknüpfen der Prozessdaten im E/A-Abbild mit dem im SPS-Programm deklarierten Variablen für das Modul S20-CNT-INC-2/2.

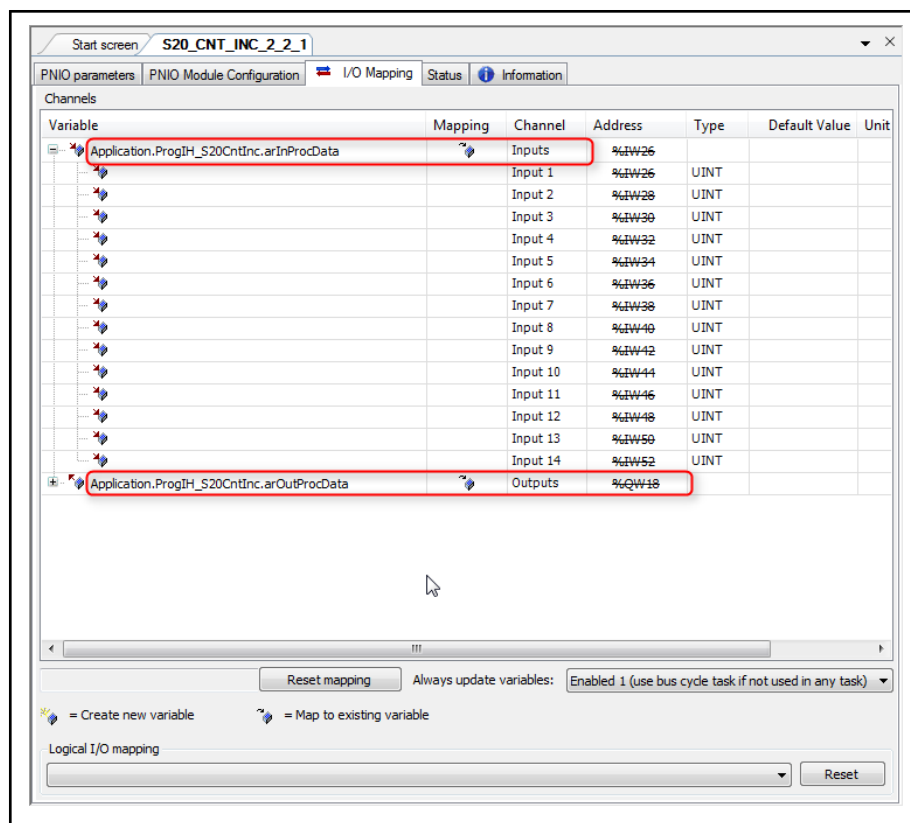


Abb. 23-24: Beispiel 2, Schritt 4

23.8.3 IH_S20RSCom

Beispielprogramm IH_S20RSCom

Beispiel 1: Modul IH_S20RSCom am Sercos-Buskoppler S20-S3-BK+

Das Beispielprogramm dient zur seriellen Kommunikation des S20-RS-UNI-Modul.

Der verwendete Feldbus ist am Eingang "BusType" mithilfe der Variable "en-BusType" definiert. Die Adresse bzw. der Name des Buskopplers wird mit "stBusConfig" definiert. Der Modulsteckplatz wird am Eingang "Slot" definiert.

TRUE am Eingang "bEnable" übernimmt die Konfiguration und der Funktionsbaustein liest die aktuelle Konfiguration des Moduls.

Schritt 1: Modulkonfiguration in IndraWorks Engineering für das RS Uni

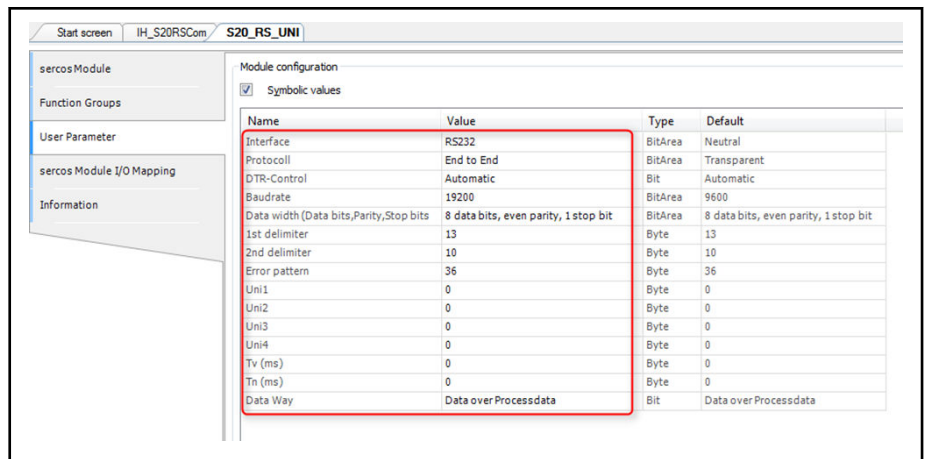


Abb. 23-25: Beispiel 1, Schritt 1

Schritt 2: Variablendeklaration für das Modul IH_S20RSCoM am Sercos-Bus-koppler

Deklaration:

Programm:

```

PROGRAM ProgIH_S20_RSCoM
VAR
    fbIH_S20RSCoM      : IH_S20RSCoM; // Function block instance
    bEnable            : BOOL; // TRUE: Function is enabled
    bDone              : BOOL; // TRUE: Function Block has completed its task
    bActive            : BOOL; // TRUE: Function Block is working on its actual task
    bError             : BOOL; // TRUE: Indicates an error
    ErrorID            : ERROR_CODE; // Class of error
    ErrorIdent         : ERROR_STRUCT; // Detailed information about error
    enBusType          : IH_S20_BUSTYPE; // Bus type selection
    stBusConfig        : IH_S20_BUS_CONFIG; // Bus configuration parameters
        uiSlot          : UINT; // Module slot address. Bus coupler : 0,
        // 1st Mod. after coupler:1 ... 63rd Mod. after coupler: 63
    bSend              : BOOL;
    arSendData         : ARRAY[0..339] OF BYTE;
    uiSizeOfSendData   : UINT;
    bRecv              : BOOL;
    arRecvData         : ARRAY[0..4095] OF BYTE;
    uiSizeOfRecvData   : UINT;
    arInProcData       : ARRAY[0..19] OF BYTE;
    arOutProcData      : ARRAY[0..19] OF BYTE;
    tTimeout           : TIME := T#3S;
    enAddDiag          : IH_S20_RS_DIAG;
    bRecvBufferNotEmpty : BOOL;
    uiRecvBufferSize   : UINT;
    uiRecvCounter      : UINT;
END_VAR

```

Variableninitialisierung für Sercos-Buskoppler:

Initialisierung der Variablen vom Funktionsbaustein für das Technologiemo-
dul, angeschlossen am Sercos-Buskoppler mit Sercos-Adresse 69 und Modul
an Steckplatz 1, im Programm.

Variableninitialisierung:

Programm:

```

enBusType          := IH_S20_BUSTYPE.SERCOS; // Connected fieldbus coupler
stBusConfig.Sercos.BusMaster:= IL_BUSMASTER.IL_BUSMASTER_0;
stBusConfig.Sercos.SercosAdr:= 69; //Buscoupler address
uiSlot             := 1; // Module position

```

Programm:

Programm:

```

fbIH_S20RSCom(
    Enable:= bEnable,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> ErrorID,
    ErrorIdent=> ErrorIdent,
    BusType:= enBusType,
    BusConfig:= stBusConfig,
    Slot:= uiSlot,
    Send:= bSend,
    SendDataAdr:= ADR(arSendData),
    SizeOfSendData:= uiSizeOfSendData,
    Recv:= bRecv,
    RecvDataAdr:= ADR(arRecvData),
    SizeOfRecvData:= uiSizeOfRecvData,
    InProcDataAdr:= ADR(arInProcData),
    SizeOfInProcData:= SIZEOF(arInProcData),
    OutProcDataAdr:= ADR(arOutProcData),
    SizeOfOutProcData:= SIZEOF(arOutProcData),
    Timeout:= tTimeout,
    AddDiag=> enAddDiag,
    RecvBufferNotEmpty=> bRecvBufferNotEmpty,
    RecvBufferSize=> uiRecvBufferSize,
    RecvCounter=> uiRecvCounter);

```

Senden und Empfangen der Daten:

bSend := TRUE; // To Send the data

bRecv := TRUE; // To Receive the data

Schritt 3: Verknüpfen der Prozessdaten im E/A-Abbild am Modul in IndraWorks

Verknüpfen der Prozessdaten im E/A-Abbild mit dem im SPS-Programm deklarierten Variablen für das Modul S20-RS-UNI.

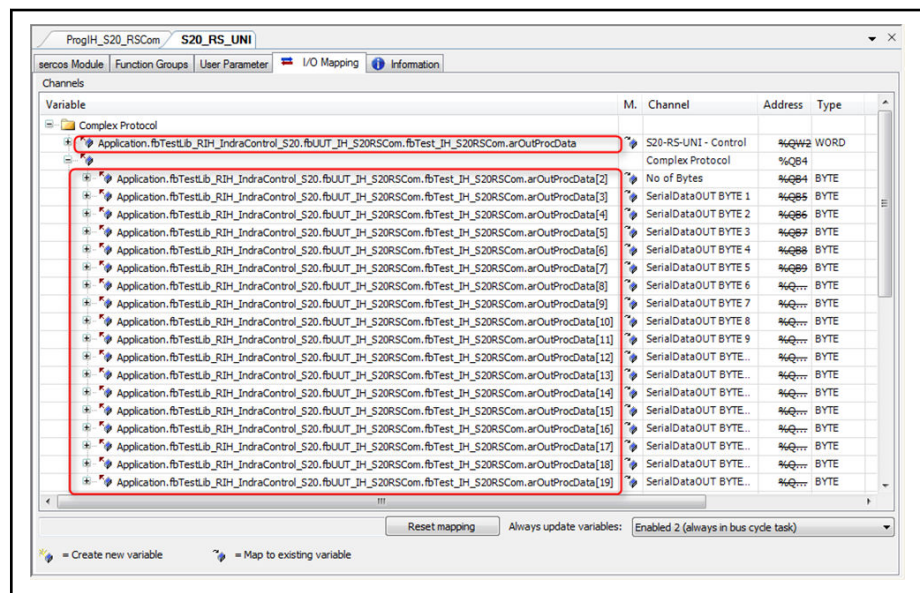


Abb. 23-26: Beispiel 1, Schritt 3, Ausgangsdaten

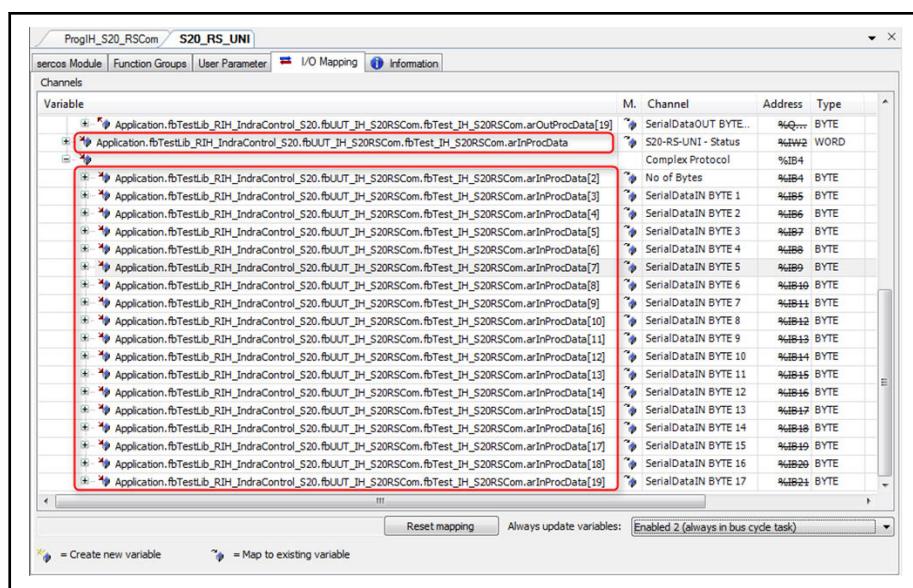


Abb. 23-27: Beispiel 1, Schritt 3, Eingangsdaten

Beispiel 2: Modul IH_S20RSCom am Profibus-Buskoppler S20-PB-BK

Das Beispielprogramm dient zur seriellen Kommunikation des S20-RS-UNI-Modul.

Der verwendete Feldbus ist am Eingang "BusType" mithilfe der Variable "enBusType" definiert. Die Adresse bzw. der Name des Buskopplers wird mit "stBusConfig" definiert. Der Modulsteckplatz wird am Eingang "Slot" definiert.

TRUE am Eingang "bEnable" übernimmt die Konfiguration und der Funktionsbaustein liest die aktuelle Konfiguration des Moduls.

Schritt 1: Modulkonfiguration in IndraWorks Engineering für das RS Uni

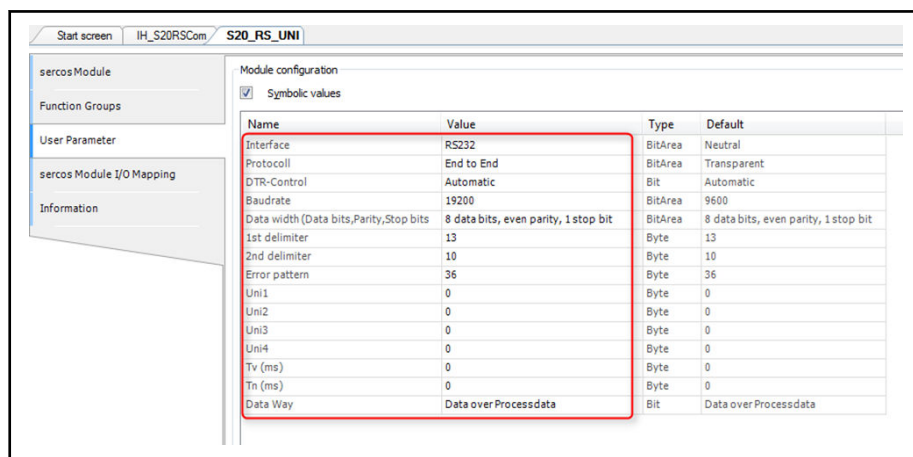


Abb. 23-28: Beispiel 2, Schritt 1

Schritt 2: Variablendeklaration für das Modul IH_S20RSCom am Profibus-Buskoppler

Deklaration:

Programm:

```
PROGRAM ProgIH_S20_RSCom
VAR
fbIH_S20RSCom      : IH_S20RSCom; // Function block instance
```

RIH_IndraControl_S20.library

```

bEnable      : BOOL; // TRUE: Function is enabled
bDone        : BOOL; // TRUE: Function Block has completed its task
bActive      : BOOL; // TRUE: Function Block is working on its actual task
bError       : BOOL; // TRUE: Indicates an error
ErrorID      : ERROR_CODE; // Class of error
ErrorIdent   : ERROR_STRUCT; // Detailed information about error
enBusType    : IH_S20_BUSTYPE; // Bus type selection
stBusConfig  : IH_S20_BUS_CONFIG; // Bus configuration parameters
uiSlot       : UINT; // Module slot address. Bus coupler : 0,
              // 1st Module after coupler:1 ... 63rd Module after coupler: 63
bSend        : BOOL;
arSendData   : ARRAY[0..339] OF BYTE;
uiSizeOfSendData : UINT;
bRecv        : BOOL;
arRecvData   : ARRAY[0..4095] OF BYTE;
uiSizeOfRecvData : UINT;
arInProcData : ARRAY[0..19] OF BYTE;
arOutProcData : ARRAY[0..19] OF BYTE;
tTimeout     : TIME := T#3S;
enAddDiag    : IH_S20_RS_DIAG;
bRecvBufferNotEmpty : BOOL;
uiRecvBufferSize : UINT;
uiRecvCounter : UINT;
END_VAR

```

Variableninitialisierung für Profibus-Buskoppler:

Initialisierung der Variablen vom Funktionsbaustein für das Technologiemodul, angeschlossen am Profibus-Buskoppler mit Profibus-Adresse 3 und Modul an Steckplatz 1, im Programm.

Variableninitialisierung:

Programm:

```

enBusType      := IH_S20_BUSTYPE.PROFIBUS; // Connected fieldbus coupler
stBusConfig.Profibus.Master := IL_BUSMASTER.IL_BUSMASTER_0;
stBusConfig.Profibus.Segment := 0;
stBusConfig.Profibus.Station := 3; // Bus coupler node address
uiSlot         := 1; // Module position

```

Programm:

Programm:

```

fbIH_S20RSCom(
    Enable:= bEnable,
    Done=> bDone,
    Active=> bActive,
    Error=> bError,
    ErrorID=> ErrorID,
    ErrorIdent=> ErrorIdent,
    BusType:= enBusType,
    BusConfig:= stBusConfig,
    Slot:= uiSlot,
    Send:= bSend,
    SendDataAdr:= ADR(arSendData),
    SizeOfSendData:= uiSizeOfSendData,
    Recv:= bRecv,
    RecvDataAdr:= ADR(arRecvData),
    SizeOfRecvData:= uiSizeOfRecvData,
    InProcDataAdr:= ADR(arInProcData),
    SizeOfInProcData:= SIZEOF(arInProcData),
    OutProcDataAdr:= ADR(arOutProcData),
    SizeOfOutProcData:= SIZEOF(arOutProcData),
    Timeout:= tTimeout,
    AddDiag=> enAddDiag,
    RecvBufferNotEmpty=> bRecvBufferNotEmpty,
    RecvBufferSize=> uiRecvBufferSize,
    RecvCounter=> uiRecvCounter);

```

Senden und Empfangen der Daten:

bSend := TRUE; // To Send the data

bRecv := TRUE; // To Receive the data

Schritt 3: Verknüpfen der Prozessdaten im E/A-Abbild am Modul in IndraWorks

Verknüpfen der Prozessdaten im E/A-Abbild mit dem im SPS-Programm deklarierten Variablen für das Modul S20-RS-UNI.

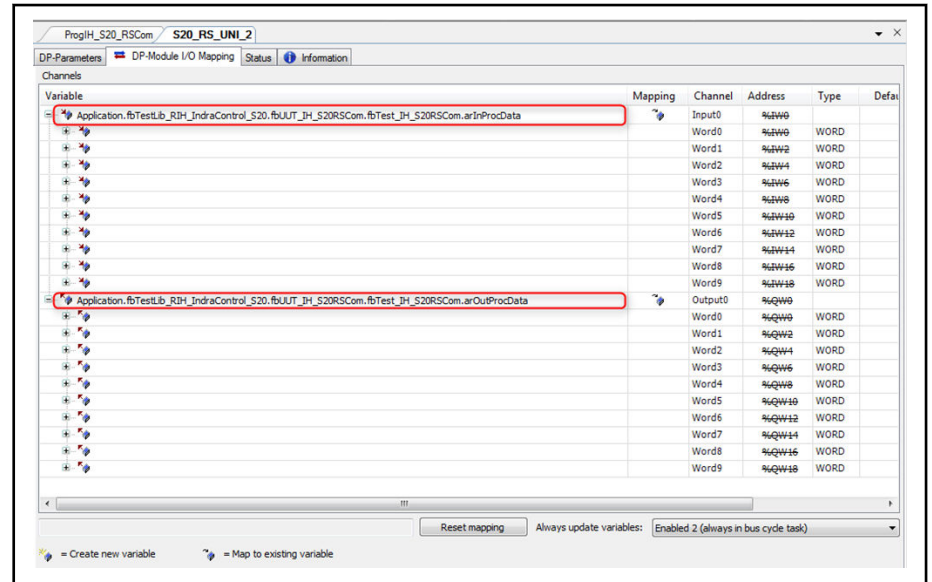


Abb. 23-29: Beispiel 2, Schritt 3

Beispielprogramm IH_S20RSCComType02

Beispiel 3: Modul IH_S20RSCComType02 am Sercos-Buskoppler S20-S3-BK+
Das Beispielprogramm dient zur seriellen Kommunikation des S20-RS-UNI-Modul.

Der verwendete Feldbus, die Adresse bzw. der Name des Buskopplers wird mit dem Funktionsbaustein IH_S20ComConfiguration definiert und mithilfe dem S20BusHandle übergeben. Der Modulsteckplatz wird am Eingang "Slot" definiert.

TRUE am Eingang "bEnable" übernimmt die Konfiguration und der Funktionsbaustein liest die aktuelle Konfiguration des Moduls.

Schritt 1: Modulkonfiguration in IndraWorks Engineering für das RS Uni

Siehe [Beispiel 1](#)

Schritt 2: Variablendeklaration für das Modul IH_S20RSCCom am Sercos-Buskoppler

Deklaration:

Programm:

```
PROGRAM ProgIH_S20RSCComType02
VAR
    fbIH_S20ComConfiguration: IH_S20ComConfiguration; // Function block instance
    bEnableComConfig:        BOOL; // Function block processing enabled( TRUE - once, edge-controlled )
    bInOperationComConfig:    BOOL; //TRUE: Function Block is working on its actual task
    bErrorComConfig:          BOOL; //TRUE: Indicates an error
    ErrorIDComConfig:         ERROR_CODE; //Class of error
    ErrorIdentComConfig:      ERROR_STRUCT; //Detailed information about error
    enBusType:                IH_S20_BUSTYPE; //Bus type selection
    stBusConfig:              IH_S20_BUS_CONFIG; //Bus configuration parameters
    dwS20BusHandle:           DWORD; //Bus handle of S20IO configuration

    fbIH_S20RSCComType02:     IH_S20RSCComType02; // Function block instance
    bEnable:                  BOOL; // TRUE: Function is enabled
    bDone:                    BOOL; // TRUE: Function Block has completed its task
    bActive:                  BOOL; // TRUE: Function Block is working on its actual task
    bError:                   BOOL; // TRUE: Indicates an error
    ErrorID:                  ERROR_CODE; // Class of error
```

RIH_IndraControl_S20.library

```

ErrorIdent:          ERROR_STRUCT; // Detailed information about error
stRSConfig:          IH_S20_RS_CONFIG;
bSend:               BOOL;
arSendData:          ARRAY[0..339] OF BYTE;
uiSizeOfSendData:    UINT;
bRecv:               BOOL;
arRecvData:          ARRAY[0..4095] OF BYTE;
uiSizeOfRecvData:    UINT;
arInProcData:        ARRAY[0..19] OF BYTE;
arOutProcData:       ARRAY[0..19] OF BYTE;
tTimeout:            TIME := T#3S;
  enAddDiag:          IH_S20_RS_DIAG;
  bRecvBufferNotEmpty: BOOL;
  uiRecvBufferSize:  UINT;
  uiRecvCounter:      UINT;

```

END_VAR

Variableninitialisierung und Programm für Sercos-Buskoppler:

Programm:

//Step-1: Variable Initiliasation:

```

enBusType := IH_S20_BUSTYPE.SERCOS; // Connected fieldbus coupler
stBusConfig.Sercos.BusMaster := IL_BUSMASTER.IL_BUSMASTER_0;
stBusConfig.Sercos.SercosAdr := 69; //Buscoupler address
uiSlot := 1; // Module position

```

```

// Channel configuration
stRSConfig.InterfaceType := IH_S20_RS_INTERFACE.RS232;
stRSConfig.Protocol := IH_S20_RS_PROTOCOL.END_TO_END;
stRSConfig.Baudrate := IH_S20_RS_BAUDRATE.BAUDRATE_19200;
stRSConfig.DataWidth := IH_S20_RS_DATAWIDTH.DATABIT8_EVEN_STOPBIT1;
stRSConfig.FirstDelimiter := 16#0D;
stRSConfig.SecondDelimiter := 16#0A;
stRSConfig.ErrorPattern := 16#24;
stRSConfig.Uni1 := 16#00;
stRSConfig.Uni2 := 16#00;
stRSConfig.Uni3 := 16#00;
stRSConfig.Tv := 16#00;
stRSConfig.Tn := 16#00;

```

//Step-2: Program:

```

fbIH_S20ComConfiguration(
  Enable:= bEnableComConfig,
  InOperation=> bInOperationComConfig,
  Error=> bErrorComConfig,
  ErrorID=> ErrorIDComConfig,
  ErrorIdent=> ErrorIdentComConfig,
  BusType:= enBusType,
  BusConfig:= stBusConfig,
  S20BusHandle=> dwS20BusHandle);

```

```

fbIH_S20RSComType02(
  Enable:= bEnable,
  Done=> bDone,
  Active=> bActive,
  Error=> bError,
  ErrorID=> ErrorID,
  ErrorIdent=> ErrorIdent,
  S20BusHandle:= dwS20BusHandle;
  Slot:= uiSlot,
  RSConfig:= stRSConfig,
  Send:= bSend,
  SendDataAdr:= ADR(arSendData),
  SizeOfSendData:= uiSizeOfSendData,
  Recv:= bRecv,
  RecvDataAdr:= ADR(arRecvData),
  SizeOfRecvData:= uiSizeOfRecvData,
  InProcDataAdr:= ADR(arInProcData),
  SizeOfInProcData:= SIZEOF(arInProcData),
  OutProcDataAdr:= ADR(arOutProcData),
  SizeOfOutProcData:= SIZEOF(arOutProcData),
  Timeout:= tTimeout,
  AddDiag=> enAddDiag,
  RecvBufferNotEmpty=> bRecvBufferNotEmpty,
  RecvBufferSize=> uiRecvBufferSize,
  RecvCounter=> uiRecvCounter);

```

Senden und Empfangen der Daten:

bSend := TRUE; // To Send the data

bRecv := TRUE; // To Receive the data

Schritt 3: Verknüpfen der Prozessdaten im E/A-Abbild am Modul in IndraWorks

Siehe [Beispiel 1](#), [Schritt 3](#)

24 RIH_IndraControl_S20_KNX

24.1 Allgemeines

Die Bibliothek "RIH_IndraControl_S20_KNX" enthält Funktionsbausteine für den Zugriff des SPS-Programms auf das IndraControl S20 KNX-Modul.

Die Bibliothek wird von IndraControl XM-Steuerungen unterstützt (IndraControl S20-Sercos-Buskoppler).

Funktionen IndraControl XM-Steuerungen ③ beinhalten eine Softwareschnittstelle, die aus dem "IH_S20KNXClient" und den Funktionsbausteinen für die einzelnen Datenpunkttypen besteht, siehe Tabelle [Tab. 24-1 "KNX-Services und Datenpunkttypen"](#) auf Seite 980.

Der Funktionsbaustein "IH_S20KNXClient" ③ verwaltet die Kommunikation zum Objektserver ⑤ mit Hilfe eines Kommunikationsprotokolls ④ auf Basis des seriellen FT1.2-Protokolls.

Der KNX-Objektserver ⑤ wird vom S20 KNX-1 Modul ⑥ bereitgestellt und beinhaltet die "ServerItems" und "DataPoints".

Die Konfiguration und die Parametrierung des KNX-Objektserver ⑤ erfolgt mit der ETS-Software und mit einem Programmiertool¹⁾ ⑨ auf dem S20-KNX-1 Modul ⑥.

Das Modul S20 KNX-1 und angeschlossene KNX-Geräte ⑦ werden durch ein externes KNX-Netzteil ⑧ versorgt.

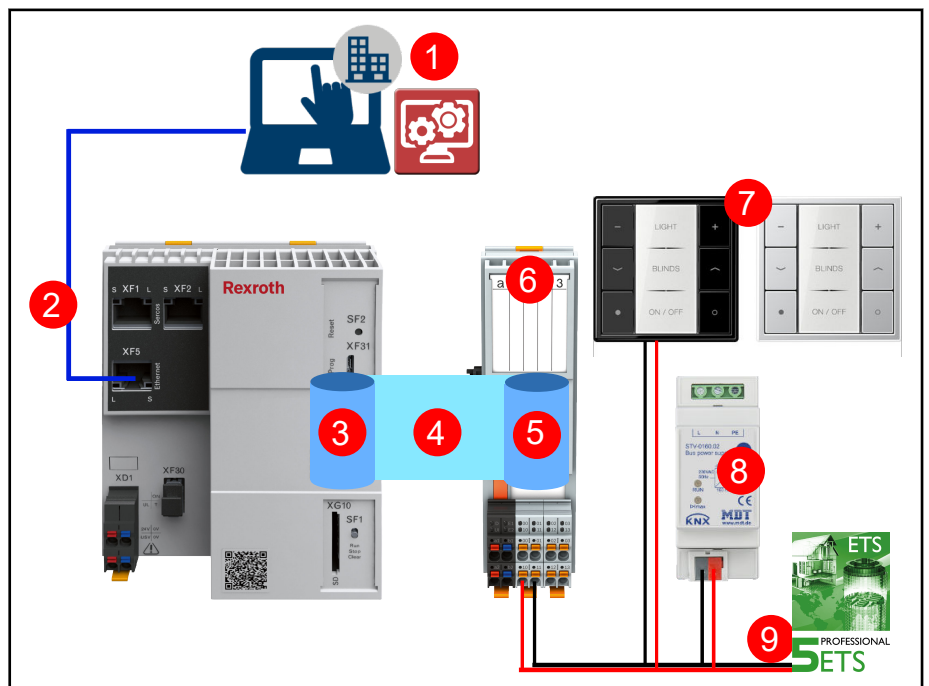


Abb. 24-1: Funktionsschema – Modul S20 KNX-1 und Netztopologie

Folgende Services und Datenpunkttypen werden mit der Bibliothek und dem Modul S20-KNX-1 unterstützt:

1) USB KNX Interface, oder ETH KNX Gateway mit Interface

KNX-Services und Datenpunkttypen

Service	Funktionsbaustein	Beschreibung	Datenbreite
IndraControl XM und S20-S3-BK+	IH_S20KNXClient	Kommunikation zur S20 KNX-1 Modul (benötigt vom Funktionsbaustein "IH_S20ComConfiguration" das S20BusHandle)	
DPT1	IH_S20KNXDPT1	Binär	1 Bit
DPT2	IH_S20KNXDPT2	Binär mit Priorität	2 Bit
DPT3	IH_S20KNXDPT3	Dimmen auf/ab	4 Bit
DPT4	IH_S20KNXDPT4	Zeichen	1 Byte
DPT5	IH_S20KNXDPT5	Prozentwert	1 Byte
DPT6	IH_S20KNXDPT6	Wert mit Vorzeichen	1 Byte
DPT7	IH_S20KNXDPT7	Wert ohne Vorzeichen	2 Bytes
DPT8	IH_S20KNXDPT8	Wert mit Vorzeichen	2 Bytes
DPT9	IH_S20KNXDPT9	Gleitkommawert	2 Bytes
DPT10	IH_S20KNXDPT10	Zeit	3 Bytes
DPT11	IH_S20KNXDPT11	Datum	3 Bytes
DPT12	IH_S20KNXDPT12	Wert ohne Vorzeichen	4 Bytes
DPT13	IH_S20KNXDPT13	Wert mit Vorzeichen	4 Bytes
DPT14	IH_S20KNXDPT14	Gleitkommawert	4 Bytes
DPT16	IH_S20KNXDPT16	Text / String	14 Bytes
DPT18	IH_S20KNXDPT18	Szene mit Lernfunktion	1 Byte

Tab. 24-1:



DPT15 und DPT17 werden nicht vom Modul S20 KNX-1 unterstützt!

Die Dokumentation zu den Funktionsbausteinen der Bibliothek RIH_IndraControl_S20_KNX finden Sie im IndraWorks Bibliotheksmanager.

Öffnen Sie hierzu den Bibliotheksmanager in der Applikation und selektieren Sie den entsprechenden Funktionsbaustein in der Bibliothek RIH_IndraControl_S20_KNX.

Der Ordner "Dokumentation" beinhaltet folgende Informationen zu den Funktionsbausteinen:

- Besonderheiten
- Strukturen
- Enumerationen
- Programmcodebeispiele für die unterschiedlichen DPT-Funktionsbausteine im Ordner **_CodeExamples** (weitere Beispiele und Inbetriebnahmevideos finden Sie im Internet beim Produkt)



Die empfohlene SPS-Zykluszeit für den Funktionsbaustein IH_S20KNXClient beträgt 10ms.

Diagnose und Fehlercodes

Die Fehlercodes der Funktionsbausteine sind in folgender Tabelle aufgelistet.



Angaben zu Fehlercodes des Moduls S20 KNX-1 sind im Datenblatt des Moduls zu finden.

Potenzielle Fehlerquellen:

- Modul S20 KNX-1 hat keine gültige KNX (ETS) Konfiguration
- KNX-Spannungsversorgung fehlt oder ist überlastet
- Fehler in der KNX-Verdrahtung/Installation

Diagnose und Fehlercodes

Additional 1	Additional 2	ErrorID	Beschreibung
16#0001	16#00000002	INPUT_RANGE_ERROR	Falscher Slot
16#0001	16#00000005	INPUT_RANGE_ERROR	TimeOut außerhalb des Bereichs (2s..60min)
16#0002	16#00000001	INPUT_INVALID_ERROR	Falsches Modul oder falscher Slot
16#0006	16#00000001	STATE_MACHINE_ERROR	Internal State Machine Error Funktionsbaustein neu starten
16#0007	16#00000000	RESOURCE_ERROR	Null Pointer, ungültiger S20BusHandle
16#0008	16#00000000	DEVICE_ERROR	TimeOut
16#0201	16#00000001	INPUT_RANGE_ERROR	DPT ID außerhalb vom Bereich 1..1000
	16#00000002	INPUT_RANGE_ERROR	DPT Name außerhalb vom Bereich
	16#00000003	INPUT_RANGE_ERROR	DPT Value außerhalb vom Bereich, der Eingangsbereich ist DPT-spezifisch
	16#00000004	INPUT_RANGE_ERROR	ReadWrite außerhalb vom Bereich
16#0206	16#00000001	DEVICE_ERROR	DPT ist deaktiviert in der KNX-Konfiguration (ETS)
	16#00000002	DEVICE_ERROR	IH_S20KNXClient nicht bereit (busy) • KNX Client initialisiert • KNX Client nicht Enabled
	16#00000003	DEVICE_ERROR	Antworttelegramm nicht identifizierbar
	16#00000004	DEVICE_ERROR	KNX-Bus nicht verbunden
	16#00000005	DEVICE_ERROR	DPT-Konfiguration mit ETS-Software (ETS download) geändert
	16#xxxx_00yy	DEVICE_ERROR	Negative Antwort bei einer Lese-/Schreibe-Aktion • xxxx - Falsche DPT ID 16#0001 - 16#03E8 • 00yy - Fehlernummern 0-11 Siehe Tab. 24-3 "Übersicht KNX-Protokollfehler" auf Seite 982

Additional 1	Additional 2	ErrorID	Beschreibung
16#0207	16#00000001	RESOURCE_ERROR	DPT Type stimmt nicht mit der Objektserver-Konfiguration überein
	16#00000002	RESOURCE_ERROR	Datenaustausch-Token zum KNX-Client verbraucht, es können bis zu 260 Aufträge gleichzeitig bearbeitet werden, warten auf freie Token und nochmal probieren
	16#00000003	RESOURCE_ERROR	DPT ist nicht konfiguriert im S20 KNX-1 Modul
	16#00000004	RESOURCE_ERROR	DPT Konfiguration (Flags) erfüllen nicht die Voraussetzung für tiefgründiges Lesen (Deep Read) Write, Transmit und Update Flag muss aktiviert sein in der ETS-Konfiguration

Tab. 24-2:

Übersicht KNX-Protokollfehler

Fehlernummer	Beschreibung
0	Kein Fehler
1	Interner Fehler
2	Element nicht gefunden
3	Puffer zu klein
4	Item nicht schreibbar
5	Service nicht unterstützt
6	Falscher Service Parameter (Bad)
7	Falsche ID (Bad)
8	Falsches Kommando (Bad) / Wert (value)
9	Falsche Länge
10	Message inkonsistent
11	Objektserver busy (beschäftigt)

Tab. 24-3:

Vorgehensweise im Fehlerfall

Im Fehlerfall sollten folgende Prüfungen durchgeführt werden:

1. KNX-Spannungsversorgung prüfen
2. ETS-Konfiguration prüfen
3. Status und Fehlercode der Funktionsbausteine prüfen

DPT Status/Communication Status/Error

- **IH_S20_KNX_STATUS** zeigt an, was der Funktionsbaustein **IH_S20KNXClient** aktuell bearbeitet oder in Fehler steht
- Jeder KNX-DPT-Funktionsbaustein hat einen Ausgang **Valid**, der im Zustand TRUE signalisiert, dass ein KNX-Datenpunkt gültig ist und dass der Status dem Status im KNX-Bus der realen KNX-Anwendung entspricht.

Ist **Valid** im Zustand FALSE, kann das verschiedene Ursachen haben:

- Initialer Zustand, KNX ist noch nicht initialisiert
- Kein DPT aus dem Anwenderprogramm gelesen oder geschrieben, keine Werteänderung in der realen KNX-Anwendung
- **IH_S20KNXClient** auf anstehende Diagnose prüfen
- S20-KNX-1-Modul in Fehler (Diagnose-Code und LED am S20-KNX-1-Modul prüfen)
- KNX-Spannung fehlt (wird am S20-KNX-1-Modul mit Fehler 0x3402 signalisiert und ist erkennbar an E2-LED Rot)

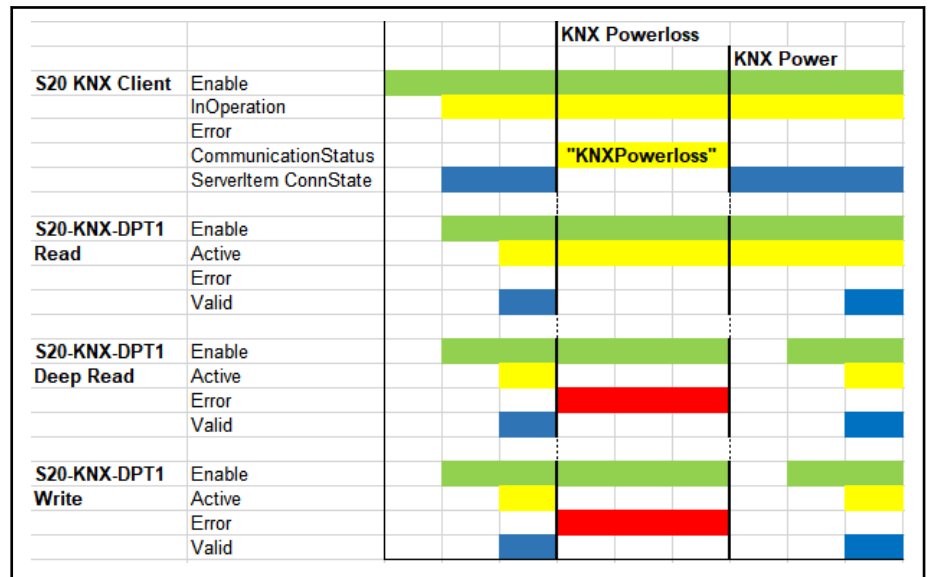


Abb. 24-2: Signal-Zeit-Diagramm KNX-Spannungsfehler-Verhalten

Im Zustand KNX-Spannungsfehler können KNX-Datenpunkte weiterhin aus dem Puffer des **IH_S20KNXClient** mit den KNX-DPT-Funktionsbausteinen gelesen werden. Schreiben und "Tiefes" Lesen führen zum Fehler, weil hierzu auf den KNX-Bus zugegriffen wird. Steht die KNX-Spannung wieder an, wird **Valid** nach der ersten Aktualisierung vom jeweiligen Datenpunkt auf dem KNX-Bus wieder **TRUE**.

IH_S20KNXClient

Der Funktionsbaustein **IH_S20KNXClient** bildet die Softwareschnittstelle zwischen dem Modul S20 KNX-1 und den KNX-Datenpunktyp-Funktionsbausteinen (mittels **KNXHandle**).

Für jedes konfigurierte Modul S20 KNX-1 ist eine eigenständige Instanz des Funktionsbausteins **IH_S20KNXClient** erforderlich.

Um den Funktionsbaustein **IH_S20KNXClient** aufrufen zu können, ist zusätzlich der Funktionsbaustein **IH_S20ComConfiguration** erforderlich. Der Funktionsbaustein stellt den notwendigen **S20BusHandle** für den Funktionsbaustein **IH_S20KNXClient** bereit.

Analog arbeiten die Funktionsbausteine der einzelnen Datenpunkttypen, die Schnittstelle ist das **KNXHandle** am Funktionsbaustein **IH_S20KNXClient**.

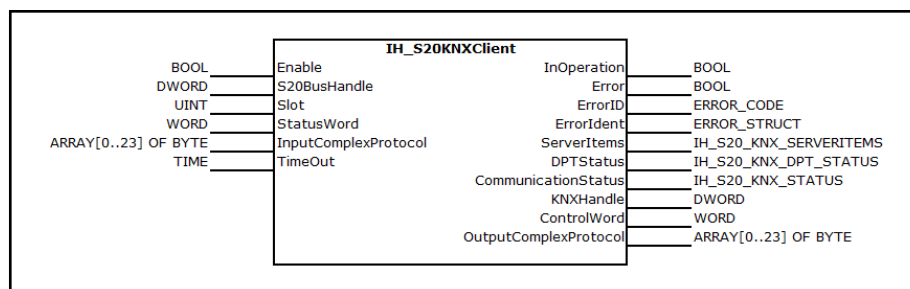


Abb. 24-3: Funktionsbaustein IH_S20KNXClient

Damit die Kommunikation zum Modul S20 KNX-1 aufgebaut wird, muss der Eingang "Enable" am **IH_S20KNXClient** auf **TRUE** gesetzt werden.

Beim Wechsel von **FALSE** nach **TRUE** am Eingang "Enable" erfolgt die Initialisierung der Schnittstelle.

Dazu werden alle konfigurierten Datenpunkte des Moduls S20 KNX-1 ausgelesen und in einem internen Datenbereich vorgehalten (maximal 1000 Datenpunkte).

Die Initialisierung ist abgeschlossen, sobald alle Datenpunkte eingelesen sind, der Ausgang "InOperation" wechselt dann auf **TRUE**.

Das Modul S20 KNX-1 ist jetzt betriebsbereit.

Der Ausgang "CommunicationStatus" zeigt die Betriebsbereitschaft mit dem Status "CLIENT_READY" an.

Der Initialisierungsprozess kann einige Sekunden dauern und wird am Ausgang "CommunicationStatus" mit "INIT_*****" angezeigt.



Mit jeder neuen Flanke von **FALSE** nach **TRUE** am Eingang "Enable" erfolgt eine Initialisierung.

Nach Abschluss der Initialisierung werden alle ankommenden Telegramme der einzelnen KNX-Datenpunkte im Betrieb analysiert und in den RAW-Daten des Funktionsbausteins **IH_S20KNXClient** abgespeichert (Puffer im Funktionsbaustein für alle DPT).

Die Kommunikation zwischen den Funktionsbausteinen der verschiedenen KNX-Datenpunkttypen und dem Funktionsbaustein **IH_S20KNXClient** erfolgt über das **KNXHandle** innerhalb der Applikation.

Der Ausgangsparameter "DPTStatus" stellt dem Anwender folgende RAW-Daten aus der KNX-Konfiguration zur Verfügung:

- DPTID
- DPT-Typ
- Konfigurationsflags der ETS-Konfiguration
- State
 - TransmissionStatus
 - ReadrequestFlag
 - UpdateFlag: Flag wird gesetzt, wenn der ActValue der betreffenden DPT-ID aktualisiert wurde
 - ValidFlag
- KNX-Nutzdaten (14 Byte)

Die Auswertung dieser Information und Aufbereitung erfolgt durch die Funktionsbausteine der einzelnen Datenpunkttypen (Ausgangsparameter nicht zwingend erforderlich/optional).

Der Ausgangsparameter "ServerItems" stellt folgende Informationen über das geladene KNX-Projekt der SPS-Applikation zur Verfügung:

- Hardware-Typ
- Hardware-Version
- Firmware-Version
- KNX Manufacturer Code DEV
- KNX Manufacturer Code APP
- ETS Applikation ID
- ETS-Applikationsversion
- Seriennummer
- Busstatus (KNX)
- Protokollversion
- Indication Sending
- Maximale Anzahl an Datenpunkten
- Anzahl der konfigurierten Datenpunkte (abhängig vom angelegten ETS-Projekt, ob mit 250 Datenpunkten oder mehr)

Die verfügbaren Ein- und Ausgangsparameter der Funktionsbausteine werden von den enthaltenen bzw. konfigurierten Datenpunkttypen bestimmt.

IH_S20KNXDPTxx

Die einzelnen Funktionsbausteine IH_S20KNXDPTxx besitzen einen Eingang "ReadWrite" mit der Enumeration "IH_S20_KNX_RW".

Folgende Lese/Schreib-Modi werden unterstützt:

IH_S20_KNX_RW

ReadWrite Mode	Beschreibung
IDLE	"ActValue" (aktueller Wert) wird gelesen (Default Mode)
WRITEONCE	"SetValue" wird einmalig geschrieben. Wird zusätzlich der Eingang "Threshold" verwendet, wird nur einmal geschrieben, wenn "SetValue" größer oder gleich dem Threshold-Wert ist. Ein Schreiben kann forciert werden, wenn "ActValue" gleich dem "ACTValue" ist und "Threshold" zu Null gesetzt ist. Zyklisches Ausführen ist begrenzt auf 100 ms.
WRITEAUTO	"SetValue" wird immer geschrieben, wenn dieser Wert sich zur letzten Vorgabe ändert. Wird zusätzlich der Eingang "Threshold" verwendet, wird immer geschrieben, wenn "SetValue" größer oder gleich dem "Threshold"-Wert ist und verschieden zum vorhergehenden "SetValue" ist. Es wird einmalig nach einer steigenden Flanke an Enable geschrieben, wenn der Wert verschieden zu "ActValue" und größer "Threshold" ist. Zyklisches Schreiben ist begrenzt auf 100 ms.
READVIABUS	Lesen direkt vom KNX-Bus wird erzwungen. Es wird aktiv gelesen auf dem KNX-Bus

Waren WRITEONCE, WRITEAUTO und READVIABUS erfolgreich, erscheint ein Done für einen SPS-Zyklus, zyklisches Ausführen ist begrenzt auf 100 ms.

Wird ein Threshold-Wert verwendet und es soll der Wert "0" über SetValue geschrieben werden oder ein Schreiben erzwungen werden, ist der Threshold mit Null vorzugeben.

24.2 Applikationsbeispiel

Im Nachfolgenden wird eine Beispielapplikation beschrieben, angefangen bei der KNX-Konfiguration auf dem Modul S20-KNX-1 bis zum Applikationsprogramm auf der Steuerung zum Lesen und Schreiben eines KNX-Datenpunkts.

- Voraussetzung**
1. Modul S20-KNX-1 ist betriebsbereit, an einer XM-Steuerung oder S20-S3-BK+ angeschlossen und in Betrieb
 2. KNX-Netzteil und Versorgung vorhanden
 3. KNX-Geräte mit KNX-Netzteil und S20-KNX-1 verbunden
 4. KNX-Programmiergerät vorhanden, z.B. KNX USB-Interface oder KNX IP-Gateway, und mit dem S20-KNX-1 und Engineering-PC verbunden
 5. ETS5-Software von der KNX-Webseite heruntergeladen und installiert
 6. Notwendige KNX-Beschreibungsdateien können direkt in der ETS5-Software nachgeladen werden, sofern die Dateien im ETS-Katalog verfügbar sind oder sind vom KNX-Gerätehersteller zu besorgen und in die ETS5-Software zu importieren

- ETS-Projekt erstellen**
1. ETS5-Software öffnen und ein neues Projekt erstellen:
 - Einen Namen vergeben
 - Backbone "IP" auswählen
 - Topologie "Create Line 1.1" anhaken und "TP" auswählen
 - Gruppenadressstil "Three Level" auswählen

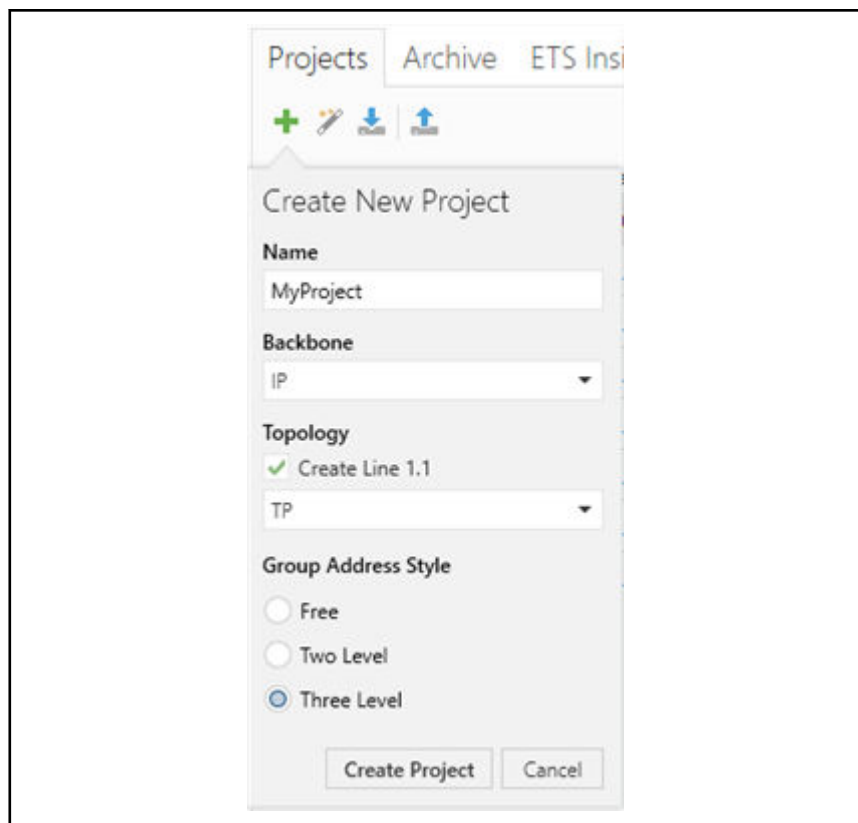


Abb. 24-4: Neues Projekt erstellen

Die Projektansicht zeigt sich wie folgt:

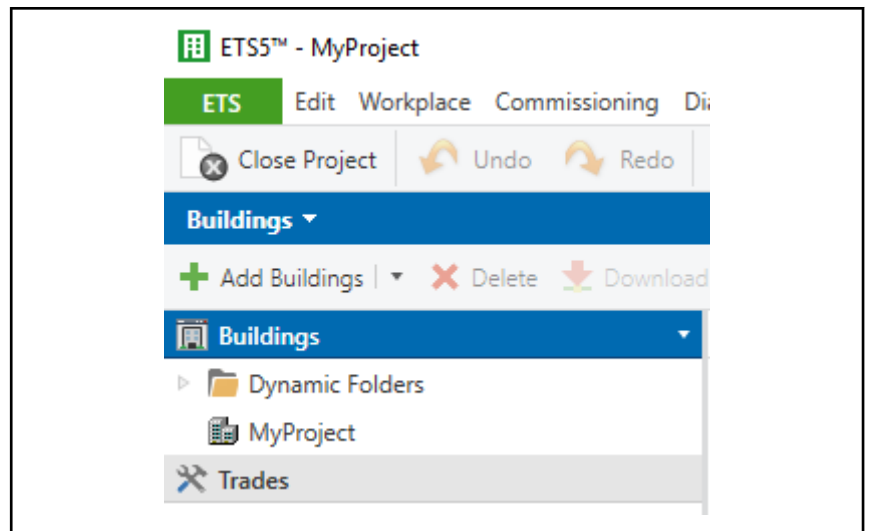


Abb. 24-5: ETS-Projektansicht

2. Dem Projekt einen Raum hinzufügen:
 1. Rechtsklick auf "MyProject" und Raum hinzufügen
 2. Gewünschten Raumnamen vergeben und erstellen

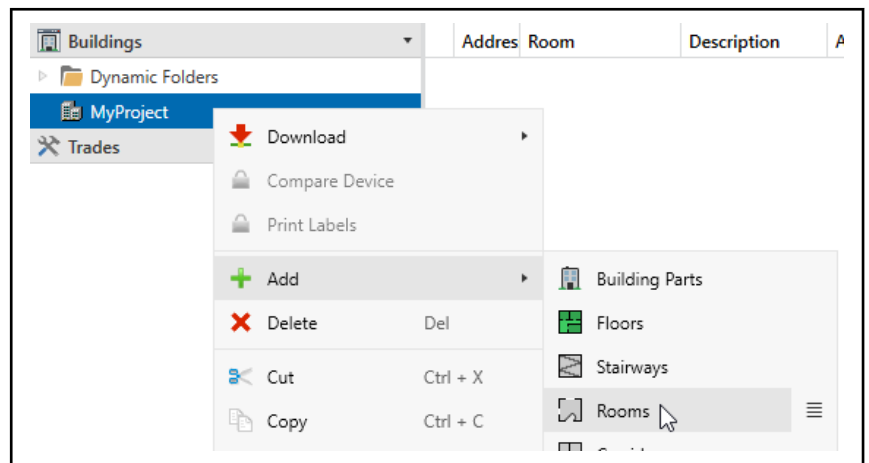


Abb. 24-6: Raum hinzufügen

3. Geräte hinzufügen, analog zum Raum:

Rechtsklick auf den Raum und Geräte hinzufügen. Dabei öffnet sich automatisch der Gerätekatalog. Im Katalog die gewünschten Geräte auswählen und dem Raum hinzufügen. Dies kann per Drag&Drop erfolgen. Hier im Beispiel das Modul "S20-KNX-1" und den Schalter/Taster "4072.02LED" (Fa. Albrecht Jung) hinzufügen, optional das KNX-Netzteil MDT STV-320.02



KNX-Netzteile sind Bestandteil der Konfiguration und gewährleisten bei der Validierung, dass die KNX-Bus-Versorgung ausreichend ist.

4. Modul S20-KNX-1 für den Raum "MyRoom" konfigurieren, das Gerät auswählen und Registerkarte "Parameter" auswählen. Im Menü **General** muss bei "Send Indications" "Enabled" angewählt sein.

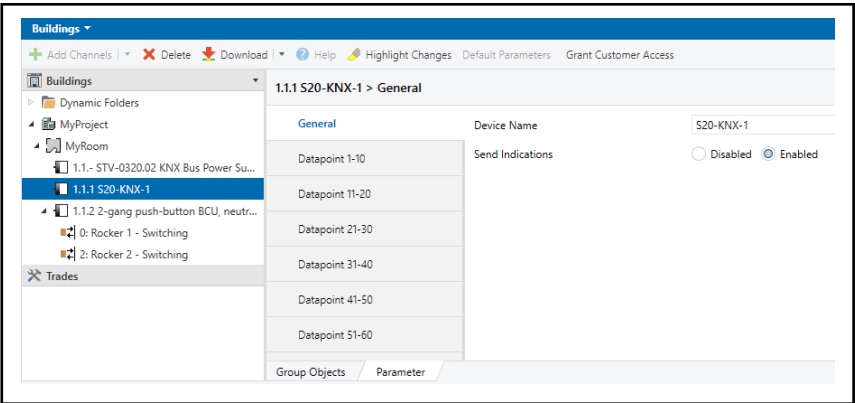


Abb. 24-7: Konfiguration "MyRoom"

5. Vom Menü **General** auf **Datapoint 1-10** wechseln und jeweils für "Datapoint type 1" und Datapoint type 2 den Typ "DPT 1 - Binary-1 Bit" auswählen.

Zum besseren Überblick empfiehlt es sich, bei jedem Punkt "Datapoint description" eine Kurzbeschreibung anzugeben.

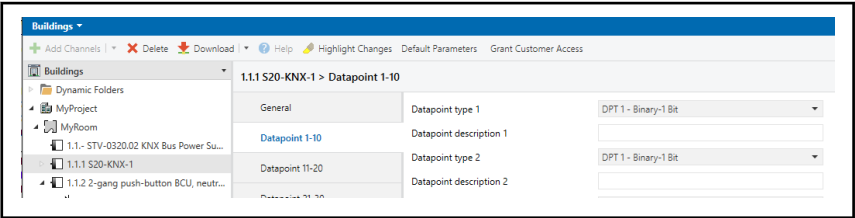


Abb. 24-8: Konfiguration Datapoint 1-10

6. Beim Wechsel auf die Registerkarte "Group Objects" erscheinen die beiden angelegten Datenpunkte. Bei jedem Datenpunkt werden über "Properties" die entsprechenden Flags gesetzt.

Number	Name	Object Function	Description	Group Address	Length	C	R	W	T	U	Data Type
1	Group object 1	DPT 1			1 bit	C	R	W	T	U	1-bit, switch
2	Group object 2	DPT 1			1 bit	C	R	W	T	U	1-bit, switch

Abb. 24-9: Registerkarte "Group Objects"

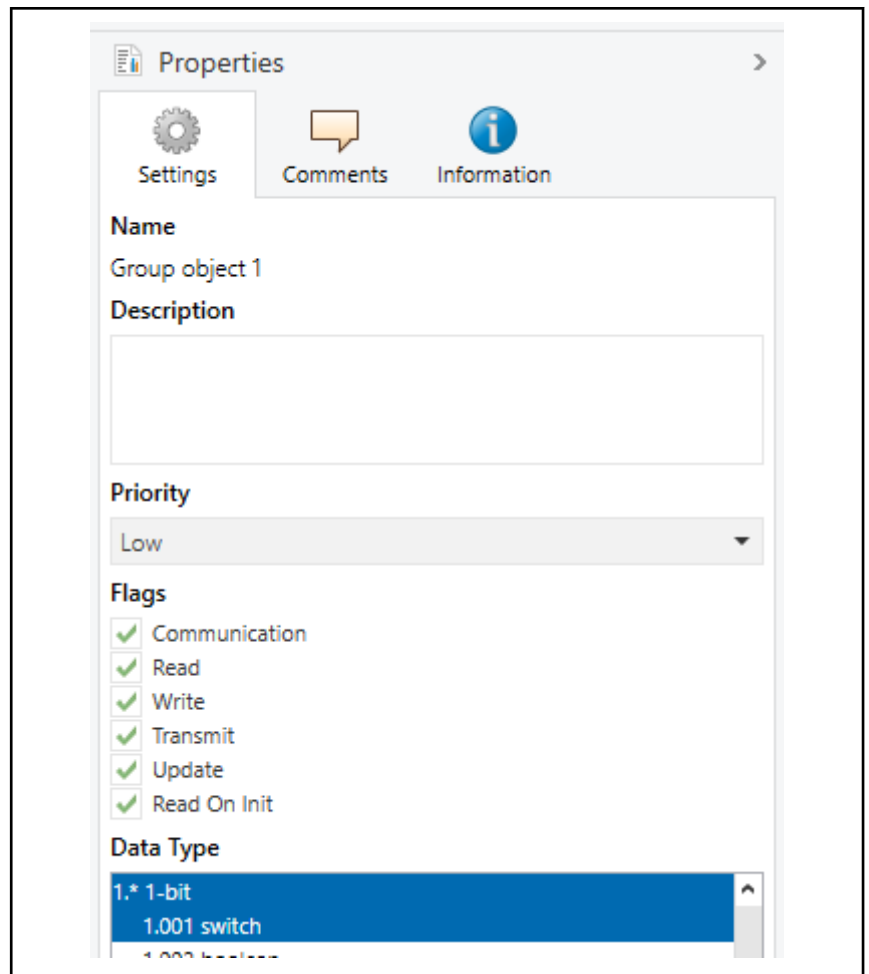


Abb. 24-10: Registerkarte "Properties"

7. Am Taster können die Standardeinstellungen, bis auf das LED-Verhalten, beibehalten werden. In "Function" muss das Verhalten von "Button-press display" auf "Status indicator (via LED object)" umgestellt werden. Es wird automatisch ein neuer Datenpunkt am Taster eingefügt. Für diesen Datenpunkt sind die Flags unter "Properties" ebenfalls zu aktivieren.

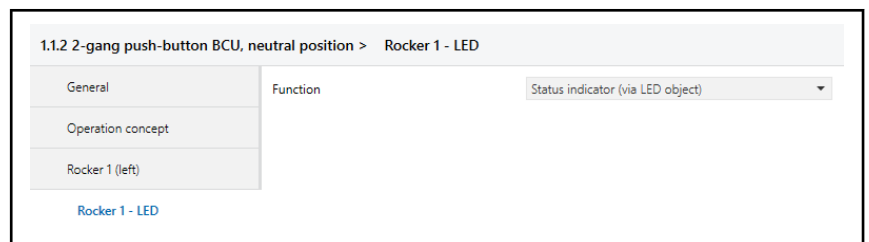


Abb. 24-11: Einstellungen Taster

Die einzelnen KNX-Geräte sind vorbereitet. Jetzt müssen diese Geräte mit Hilfe von Gruppen miteinander verknüpft werden.

8. Über Menu **Workplace ► Open New Panel ► Group Addresses** wird die Gruppenansicht hinzugefügt

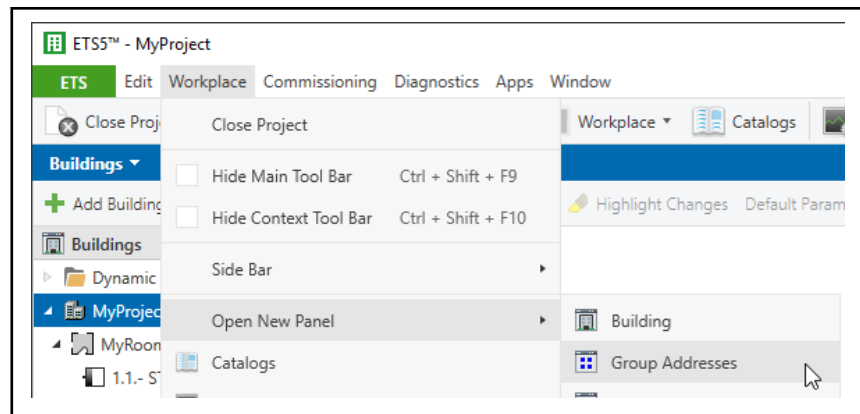


Abb. 24-12: Menü "Workplace"

9. Mit der Funktion **Add Main Groups** die Gruppen hinzufügen:
 1. Erste Ebene "MainGroup"
 2. Nächste Ebene "MiddleGroup"
 3. Gruppenadresse "RockerSwitch1"
 4. Gruppenadresse "RockerLED"

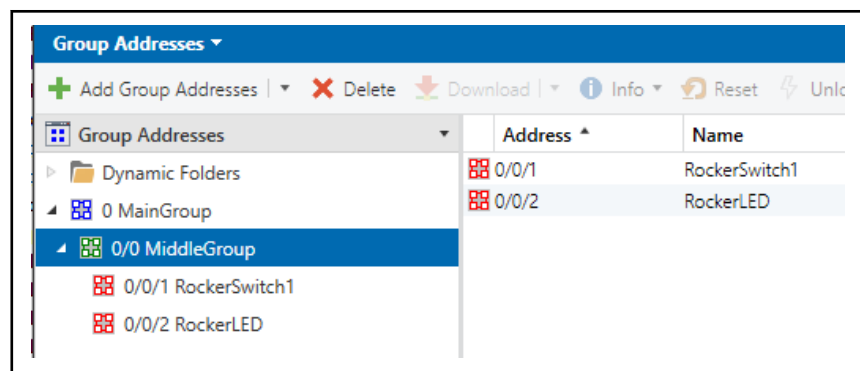


Abb. 24-13: Gruppen hinzufügen

10. Die Datenpunkte können per Drag&Drop den Gruppenadressen hinzugefügt werden. Das Hinzufügen zu einer Gruppe ist notwendig zum eindeutigen Adressieren auf dem KNX. Es erfolgt das Verknüpfen der Datenpunkte für den Datenaustausch zum Lesen und Schreiben. Einer Gruppenadresse werden alle Datenpunkte des gleichen Datenpunktyps zugewiesen, die miteinander interagieren sollen.

Für das Beispiel müssen "Group object 1" mit "Rocker1 - Switching" zur Gruppenadresse "0/0/1 RockerSwitch1" und "Group object 2" mit "LED left - Status indication" zu "0/0/2 RockerLED" verknüpft werden.

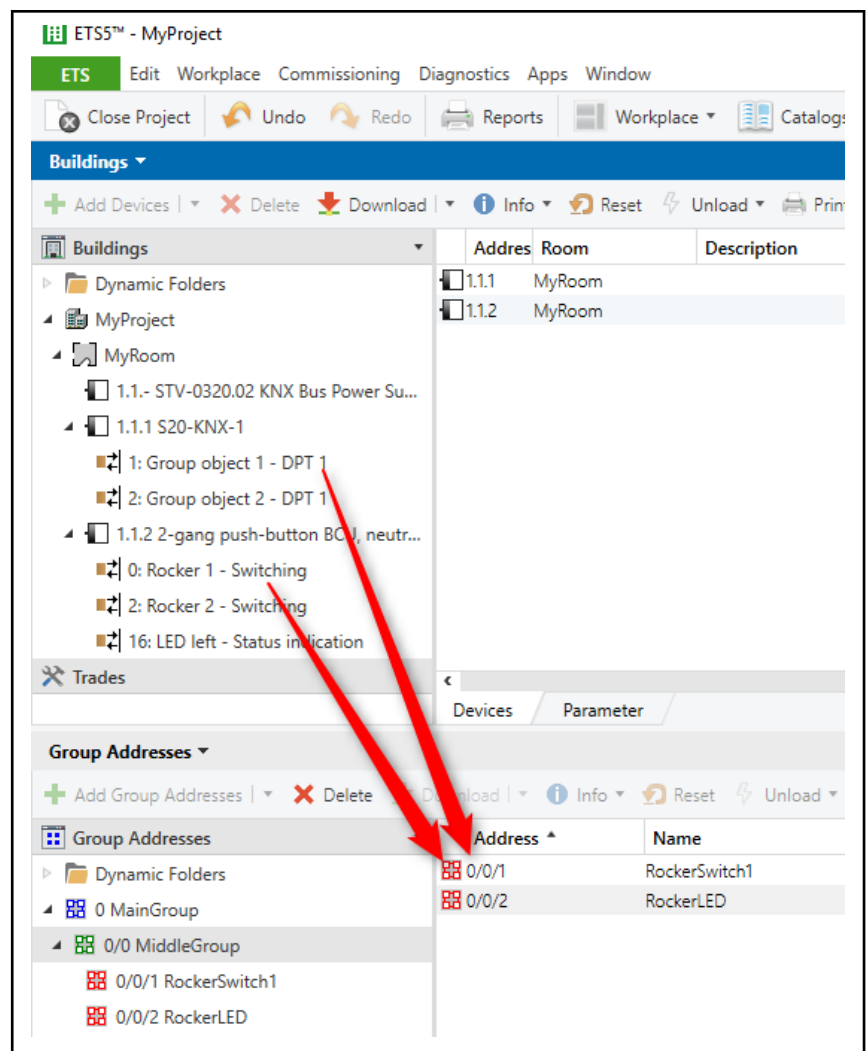


Abb. 24-14: Zuweisung von Datenpunkten



Ohne die Zuweisung von Datenpunkten zu einer Gruppenadresse kann dieser Datenpunkt mit der Applikation S20-KNX-1 nicht verwendet werden und führt bei Zugriff zu einem Fehler "DPT ist deaktiviert in der KNX Konfiguration (ETS)".

Damit ist das ETS-Projekt erstellt und kann auf die KNX-Geräte verteilt werden.

11. Bei Erstinbetriebnahme benötigen die KNX-Geräte einmalig die Vergabe der physikalischen Adresse.
 1. **Download ► Download all** auswählen, physikalische Adressen werden vergeben
 2. Anweisungen von ETS folgen, beim Download müssen die Programmierknöpfe der KNX-Geräte bei Aufforderung betätigt werden

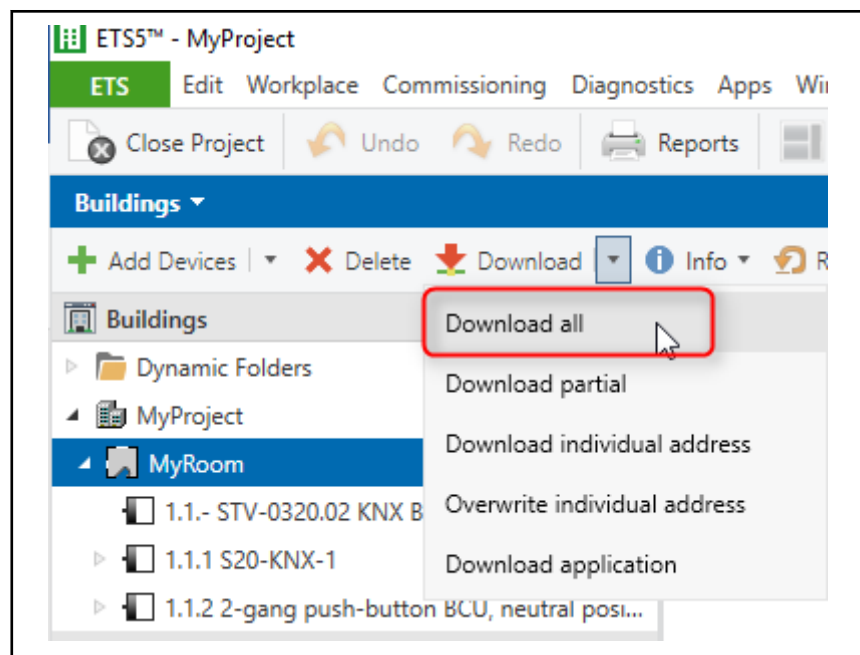


Abb. 24-15: Start ETS-Download

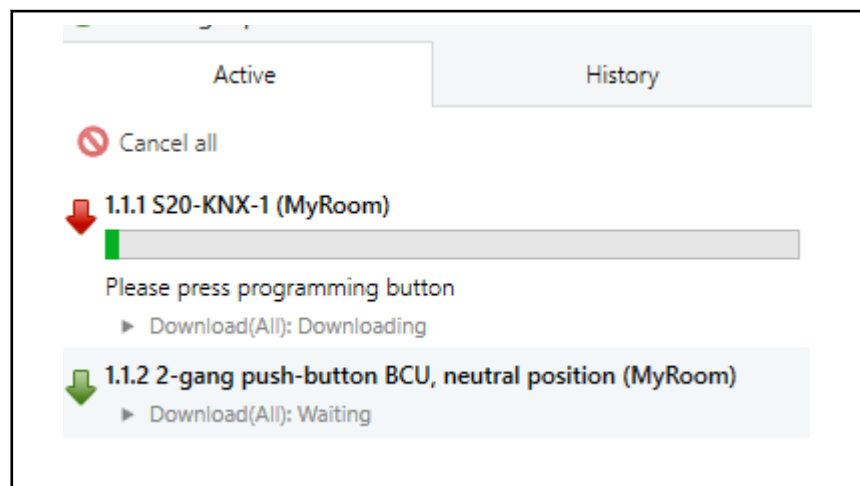


Abb. 24-16: Aufforderung den Programmieraster auf den KNX-Gerät zu betätigen (Programmiermodus aktivieren)

Bei weiteren Änderungen der Datenpunkte ist ein **Download** ► **Download partial** oder **Download** ► **Download application** ausreichend. Dazu muss der Programmieraster nicht mehr betätigt werden.



Nach einem kompletten Download oder bei der Erstinbetriebnahme ist das Modul S20-KNX-1 spannungslos zu machen. Dazu muss die Steuerung oder der Buskoppler mit dem Modul ausgeschaltet und wieder eingeschaltet werden.

Wurde nur ein Datenpunkt hinzugefügt oder geändert, muss kein Neustart durchgeführt werden.

Beispiel SPS-Applikation

1. Die Bibliothek **RIH_IndraControl_S20** hinzufügen.
Es ist der Funktionsbaustein **IH_S20ComConfiguration** anzulegen.

Der Funktionsbaustein stellt das **S20BusHandle** zur Verfügung, mit dem festgelegt wird, an welchem Busmaster (XM lokal oder an einem Buskoppler) das S20-KNX-1 Modul betrieben wird.

- Die Bibliothek **RIH_IndraControl_S20_KNX** hinzufügen.

Es ist der Funktionsbaustein **IH_S20KNXClient** anzulegen.

Der Funktionsbaustein **IH_S20KNXClient** stellt zusammen mit dem Funktionsbaustein **KNXHandle** die Schnittstelle zum S20-KNX-1 Modul und den KNX-Datenpunkttypen zur Verfügung. Für jedes S20-KNX-1 Modul ist eine neue Funktionsbausteininstanz vom **IH_S20KNXClient** notwendig.

- Zum Erfassen vom Zustand des Tasters kann der Datenpunkt **DPT1** der Gruppe "0/0/1 RockerSwitch1" verwendet werden.

Es ist der Funktionsbaustein **IH_S20KNXDPT1** für jede Datenpunkt-ID "DPTID" mit Datenpunkttyp DPT1 als eigene Instanz anzulegen.

- Beispiel Programm Code

Lesen des DPT ID 1 von der Gruppe 0/0/1 RockerSwitch1, zum Lesen des Zustandes des DPT ID 0, welches der Tasterstellung des realen Taster entspricht

Schreiben und Lesen von DPT ID 2 von der Gruppe 0/0/2 RockerLED, zum Schreiben und Lesen des DPT ID 16, der die LED vom Taster ansteuert

Beispiel ST

Programm:

```

PROGRAM KNX_Example
VAR
    bInit                : BOOL := TRUE;
    fbIH_S20ComConfiguration : IH_S20ComConfiguration; //FB Instance
    bEnableComConfig      : BOOL; //Enable
    enBusType              : IH_S20_BUSTYPE := IH_S20_BUSTYPE.S20IO; //Bus type selection
    enBusConfig            : IH_S20_BUS_CONFIG; //Bus_config for field bus device
    dwS20BusHandle         : DWORD;

    fbIH_S20KNXClient      : IH_S20KNXClient; //FB Instance
    bEnableClient          : BOOL; //Enable
    uiSlot: UINT           := 3; //Slot of the S20-KNX-1 module
    tTimeout: TIME         := T#2S; //Timeout
    stServerItems          : IH_S20_KNX_SERVERITEMS; //KNX Server Status
    enDPTStatus            : IH_S20_KNX_DPT_STATUS; //RAW DPT data
    enCommStatus           : IH_S20_KNX_STATUS; //State of the KNX Client FB
    dwKNXHandle            : DWORD; //KNX communication Handle for DPT FBs
    wStatusWord            : WORD; //Status of S20-KNX-1 module
    arInputComplexProtocol : ARRAY [0..23] OF BYTE; //Input data of S20-KNX-1 module
    wControlWord           : WORD; //Control of S20-KNX-1 module
    arOutputComplexProtocol : ARRAY [0..23] OF BYTE; //Output data of S20-KNX-1 module

    fbIH_S20KNXDPT1_R      : IH_S20KNXDPT1; //FB Instance
    bEnableDPT1_R          : BOOL; //Enable
    bDone_DPT1_Read        : BOOL; //Done, indicates when the value is changed
    bValidDPT1_R           : BOOL; //Valid indicates the KNX valid flag of the DPTID
    bDPT1Value_R           : BOOL; //actual value of the KNX DPT

    fbIH_S20KNXDPT2_W      : IH_S20KNXDPT1; //FB Instance
    bEnableDPT1_W          : BOOL; //Enable
    bDone_DPT2_Write       : BOOL; //Done, indicates when the value is changed
    uiDPTID_DPT2: UINT     := 2; //DPT ID
    enReadWriteDPT         : IH_S20_KNX_RW := IH_S20_KNX_RW.WRITEAUTO;
    //Write on change in SetValue and read after write success

    bSetValueDPT2         : BOOL; //Value which is written
    bValidDPT2_W           : BOOL; //Valid indicates the KNX valid flag of the DPTID
    bDPT2Value_R           : BOOL; //actual value of the KNX DPT
END_VAR

```

RIH_IndraControl_S20_KNX

Programm:

```

IF bInit THEN
    bEnableComConfig := TRUE;
    bEnableClient     := TRUE;
    bInit             := FALSE;
END_IF

fbIH_S20ComConfiguration(
    Enable      := bEnableComConfig,
    BusType     := enBusType,
    BusConfig   := enBusConfig,
    S20BusHandle => dwS20BusHandle);
fbIH_S20KNXClient(
    Enable      := bEnableClient AND fbIH_S20ComConfiguration.InOperation,
    S20BusHandle := dwS20BusHandle,
    Slot        := uiSlot,
    StatusWord  := wStatusWord,
    InputComplexProtocol := arInputComplexProtocol,
    Timeout     := tTimeout,
    ServerItems := stServerItems,
    DPTStatus   => enDPTStatus,
    CommunicationStatus := enCommStatus,
    KNXHandle   => dwKNXHandle,
    ControlWord := wControlWord,
    OutputComplexProtocol => arOutputComplexProtocol);
//Read DPT ID 1 of Group 0/0/1 RockerSwitch1 to read DPT ID 0
fbIH_S20KNXDPT1_R(
    Enable      := bEnabledDPT1_R AND fbIH_S20KNXClient.InOperation,
    Done        => bDone_DPT1_Read,
    DPTID       := 1,
    Timeout     := tTimeout,
    KNXHandle   := dwKNXHandle,
    Valid       => bValidDPT1_R,
    ActValue    => bDPT1Value_R);
//Write and Read DPT ID 2 of Group 0/0/2 RockerLED to write DPT ID 16
fbIH_S20KNXDPT2_W(
    Enable      := bEnabledDPT1_W AND fbIH_S20KNXClient.InOperation,
    Done        => bDone_DPT2_Write,
    DPTID       := uiDPTID_DPT2,
    ReadWrite   := enReadWriteDPT,
    SetValue    := bSetValueDPT2,
    Timeout     := tTimeout,
    KNXHandle   := dwKNXHandle,
    Valid       => bValidDPT2_W,
    ActValue    => bDPT2Value_R);

```

Beispiel CFC*Programm:*

```

PROGRAM KNX_Example_CFC
VAR
    fbIH_S20ComConfiguration : IH_S20ComConfiguration; //FB Instance
    bEnableComConfig         : BOOL := TRUE;
    enBusType                : IH_S20_BUSTYPE := IH_S20_BUSTYPE.S20IO; //Bus type selection
    enBusConfig              : IH_S20_BUS_CONFIG; //Bus-config for field bus device

    fbIH_S20KNXClient        : IH_S20KNXClient; //FB Instance
    bEnableClient            : BOOL;
    enCommStatus             : IH_S20_KNX_STATUS; //State of the KNX Client FB
    wKNXStatus               : WORD; //Status of S20-KNX-1 module
    wKNXControl              : WORD; //Control of S20-KNX-1 module
    arKNXComplexProtocolInput : ARRAY [0..23] OF BYTE; //Input data of S20-KNX-1 module
    arKNXComplexProtocolOutput : ARRAY [0..23] OF BYTE; //Output data of S20-KNX-1 module

    fbIH_S20KNXDPT1_R        : IH_S20KNXDPT1; //FB Instance
    bEnabledDPT1_R           : BOOL;

    fbIH_S20KNXDPT2_W        : IH_S20KNXDPT1; //FB Instance
    bEnabledDPT1_W           : BOOL;
    enReadWriteDPT           : IH_S20_KNX_RW := IH_S20_KNX_RW.WRITEAUTO;
    bSetValueDPT2            : BOOL; //Value which is written
    uiDPTID_DPT2             : UINT := 2; //DPT ID
END_VAR

```

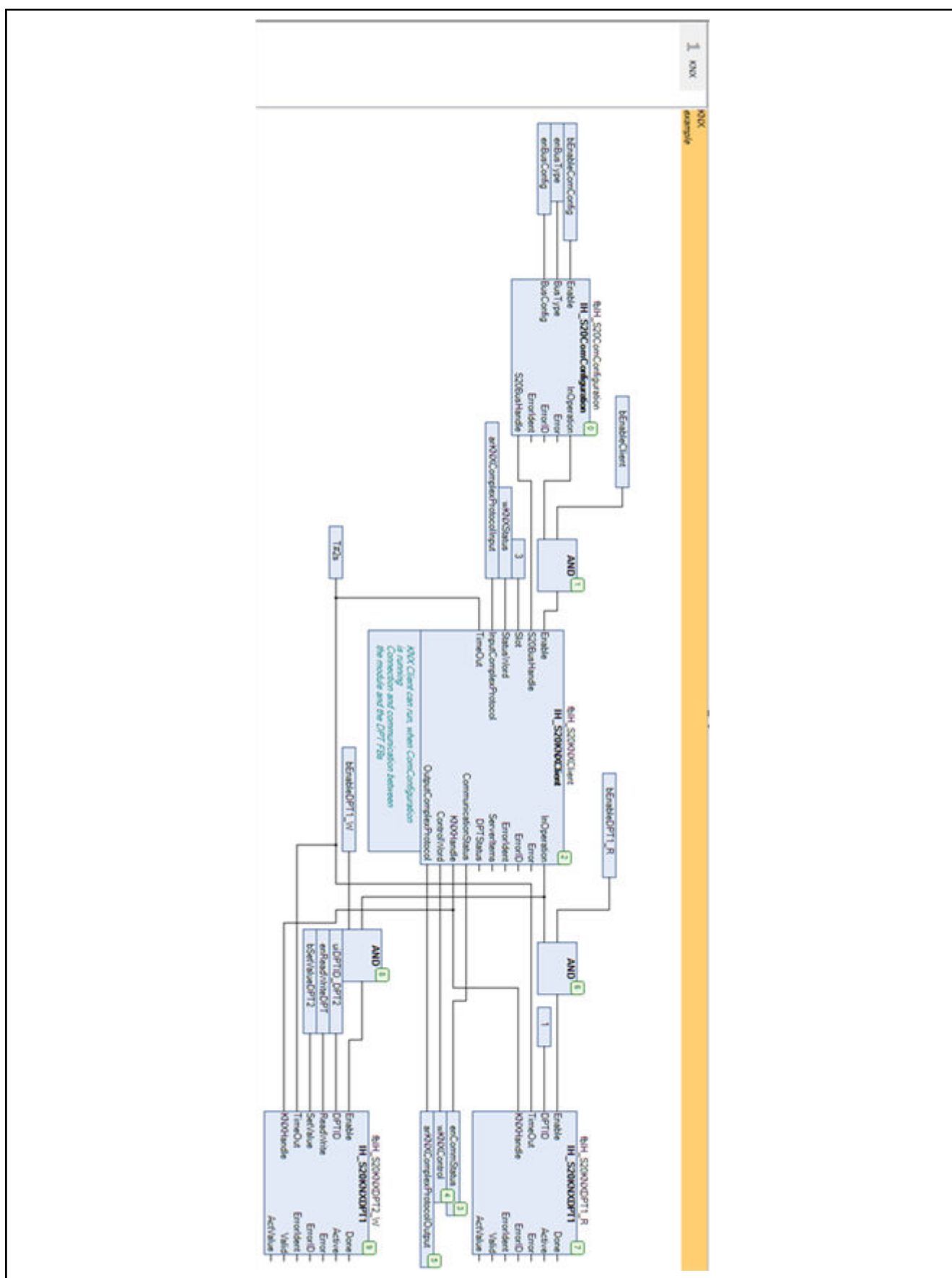


Abb. 24-17: Beispiel CFC

25 BACnet-Bibliotheken

25.1 Übersicht

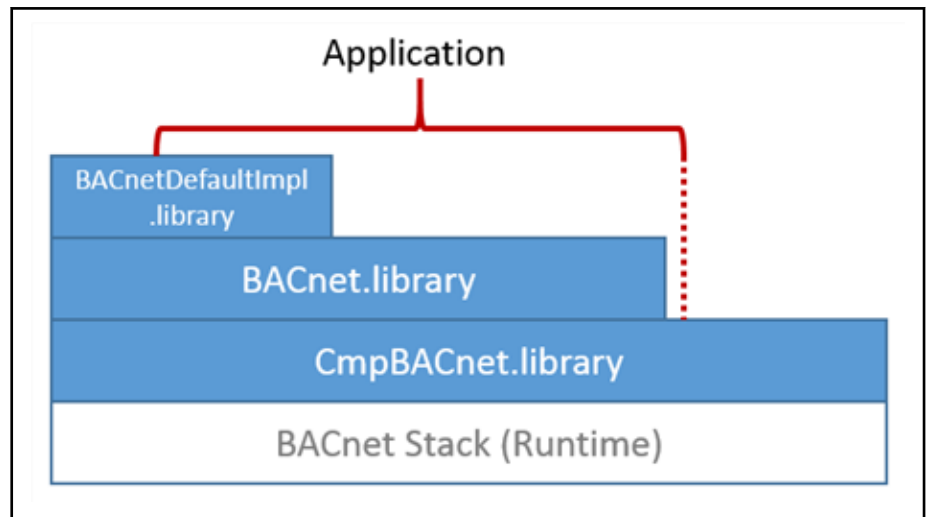


Abb. 25-1: Übersicht BACnet-Bibliotheken

BACnetDefaultImpl.library	Default-Implementierungen von Anwendungsfällen, die bei Bedarf in der Applikation genutzt werden können, z.B. Handler für Device-Communication-Control oder ReInit-Device
BACnet.library	Anwenderschnittstelle für die unterstützten Server-Objekte und Client-Requests
CmpBACnet.library	Integration des BACnet Stacks (IEC-Interface). Für die Applikation sind hier bei Bedarf nur ausgewählte Methoden oder Spezialfälle sinnvoll, ansonsten sollten die Funktionsbausteine der BACnet.library verwendet werden

Tab. 25-1: Kurzbeschreibung BACnet-Bibliotheken

Beschreibung der BACnet-Bibliotheken

Eine ausführliche Schnittstellen- und Funktionsbeschreibung steht als integrierte Hilfe in der jeweiligen Bibliothek zur Verfügung und kann im Bibliotheksverwalter eingesehen werden.

BACnetClientReadProperty (FB)

FUNCTION_BLOCK PUBLIC FINAL BACnetClientReadProperty EXTENDS BACnetClientBase

Common Behaviour Model conform (ETrigA) FB to execute a Read-Property-Request in the BAC if execution completed successfully and xDone is TRUE, result is filled with valid data, in any otl data in result are not valid.

Scope	Name	Type
Input	xExecute	BOOL
Output	xDone	BOOL
	xBusy	BOOL

Abb. 25-2: BACnet-Bibliotheken Dokumentation im Bibliotheksverwalter

25.2 Programmierbeispiele

25.2.1 Server: NotificationClass → SetRecipientList

Programm:

```

VAR
  rNotifyDest : BACnet.CmpBACnet.IEC_BACNET_DESTINATION;
END_VAR

rNotifyDest.recipient.tag                := BACnet.CmpBACnet.IEC_BACNET_RECIPIENT_TYPE.RECIPIENT_DEVICE;
rNotifyDest.recipient.recipient.device.type := BACnet.CmpBACnet.IEC_BACNET_OBJECT_TYPE.OBJ_DEVICE;
rNotifyDest.recipient.recipient.device.instNumber := 1; // Instance-Number (Device-ID) from the Client
rNotifyDest.processID := NC_1.ProcessId; // example
rNotifyDest.issueConfNotification := 1; // 0 = Unconfirmed / 1 = Confirmed
// Transitions (To-Offnormal, To-Fault, To-Normal)
BACnet.InitializeBACnetBitString( str:='111', bitStr:=rNotifyDest.transitions );
// Valid Week-Days
BACnet.InitializeBACnetBitString( str:='1111111', bitStr:=rNotifyDest.validDays );
rNotifyDest.fromTime.hour := 0;
rNotifyDest.fromTime.minute := 0;
rNotifyDest.fromTime.second := 0;
rNotifyDest.fromTime.hundredths := 0;
rNotifyDest.toTime.hour := 23;
rNotifyDest.toTime.minute := 59;
rNotifyDest.toTime.second := 59;
rNotifyDest.toTime.hundredths := 99;

NC_1.SetRecipientList( 1, ADR(rNotifyDest) );

```

25.2.2 Server / Client: IBACnetEventConsumer → BACnetEventCallback

Mit dem Interface IBACnetEventConsumer (Methode BACnetEventCallback) können verschiedene Ereignisse eventgesteuert verarbeitet werden.

Folgender Beispielcode registriert COV-Benachrichtigungen, gilt aber exemplarisch für alle Events (siehe CmpBACnet.library → IEC_BACNET_CB_TYPE im Ordner BACnetEnums):

- Es muss ein Funktionsbaustein erstellt werden, der das Interface IBACnetEventConsumer implementiert
- Die Ausgänge und Variablen im nachfolgenden Beispielcode sind exemplarisch für das hier gezeigte Event

Programm:

```

FUNCTION BLOCK FB_CB_CONF_COV_NOTIFICATION IMPLEMENTS BACnet.IBACnetEventConsumer
VAR_OUTPUT
  EvCnt      : UINT := 0; // call count
  LastEvtValid : BOOL := FALSE;
  LastCovNotifyInfo : BACnet.CmpBACnet.IEC_BACNET_COV_NOTIF_INFO;
  LastPropertyValue : BACnet.CmpBACnet.IEC_BACNET_PROPERTY_VALUE;
  LastRealValue   : REAL;
END_VAR
VAR
  prEvtConfCovNotify : POINTER TO BACnet.CmpBACnet.EVT_BACNET_CONF_COV_NOTIFICATION;
END_VAR

```

- Dieses Interface gibt die Methode BACnetEventCallback vor, die generische Übergabeparameter besitzt
- Um die jeweiligen Parameter des spezifischen Events verarbeiten zu können, muss der generische Pointer pParameter auf die jeweilige Event-Struktur gecastet werden
- Die Event-Informationen werden hier in diesem Beispiel dann auf die Ausgänge des FBs kopiert

Programm:

```

METHOD BACnetEventCallback : BOOL
VAR_INPUT
    EventId : DWORD; // ID of the occurred event (see CmpBACnet.library -> GVL EVENTIDS)
    usParamId : WORD; // ID of the parameter structure (see CmpBACnet.library -> GVL EVENTIDS)
    usVersion : WORD; // Version of the parameter structure (see CmpBACnet.library -> GVL EVENTIDS)
    pParameter: POINTER TO BYTE; // Pointer to the event specific parameter, that is specified by ID
                                // (e.g. EVT_BACNET_ACKALARM)
END_VAR

// NOTE: no breakpoint possible, because this event callback is called from the BACstack task,
// which is not an IEC task!

// Check EventId
IF (    EventId = BACnet.CmpBACnet.EVT_ConfCOVNotification
    AND usParamId = BACnet.CmpBACnet.EVT_PARAMID_CmpBACnet_ConfCOVNotification
) THEN
    LastEvtValid := TRUE;
    EvCnt := EvCnt + 1;

    // Event parameter
    prEvtConfCovNotify := pParameter;
    LastCovNotifyInfo := prEvtConfCovNotify^.pServiceInfo^;
    LastPropertyValue := LastCovNotifyInfo.listOfValues^;
    // Note: the buffer in 'PropertyContents' is stack internal, that means you have to copy
    // the value here in the callback!
    LastRealValue := BACnet.GetRealFromContents(LastPropertyValue.value);

ELSE // unexpected EventId
    LastEvtValid := FALSE;
END_IF

// Return
BACnetEventCallback := TRUE; // TRUE = no calls of other registered callbacks of this event

(* Note
prEvtConfCovNotify^.pCBStatus^ := BACnet.CmpBACnet.IEC_BACNET_CB_STATUS.CB_STATUS_DEFAULT
=> default: the stack handles the default actions (and not the hook / callback)

prEvtConfCovNotify^.pCBStatus^ := BACnet.CmpBACnet.IEC_BACNET_CB_STATUS.CB_STATUS_OK
=> the hook or application generates the reply, see the ((Response)-'CbCompletion' functions,
which are mentioned in the corresponding event strukturen,
e.g. see documentation of EVT_BACNET_CONF_COV_NOTIFICATION.
*)

```

- In einem Programmbaustein ist dann eine Instanz dieses Funktionsbausteins anzulegen:

```
fbCbConfCovNotification : FB_CB_CONF_COV_NOTIFICATION;
```

- Diese FB-Instanz wird dann am BACnet-Gerät für einen bestimmten Event-Typ (→ IEC_BACNET_CB_TYPE) registriert und kann auch wieder deregistriert werden (UnregisterHook):

```
eBACnetStatus := BACnet_Server.RegisterHook( BACnet.CmpBACnet.IEC_BACNET_CB_TYPE.CB_CONF_COV_NOTIFICATION, fbCbConfCovNotification );
```



Weitere Beispiele finden Sie in der allgemeinen BACnet-Beschreibung der Dokumentation "IndraWorks 15VRS Feldbusse" (DOK-IWORKS-FB*****V15-APRS-DE-P).

26 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

Abkürzungen.....	15
Active.....	18
Allgemeine Fehlercodes	
Profinet IO-Controller.....	196
PROFINET IO-Device.....	220
Änderungsverlauf.....	1
Anregungen.....	16
ANSI Z535.6-2006.....	3
Anwendungsphasen.....	1

B

BACnet Bibliotheken	
Programmierbeispiele.....	998
BACnet-Bibliotheken	
BACnet.library.....	997
BACnetDefaultImpl.library.....	997
CmpBACnet.library.....	997
Übersicht.....	997
Bausteintypen.....	17
Bausteintypen und ihre Schnittstellen.....	18
Beschwerde.....	16
Bibliothek	
RIH_Dali.....	745
RIH_IndraControl_S20.....	927
RIH_Modbus.....	811
RIH_Safety.....	711
RIH_Safety_ProfibusDP.....	713
RIH_Safety_ProfinetIO.....	725
RIH_Safety_SercosIII.....	737
RIL_Ethercat.library.....	265
RIL_EtherNetIP.....	223
RIL_EtherNetIPAdapter.....	253
RIL_InlineIOModules.....	573
RIL_IoLink.....	901
RIL_MappingList.....	331
RIL_ProfibusDP_02.....	31
RIL_ProfibusDP Slave.....	57
RIL_ProfinetIO.....	71
RIL_ProfinetIODevice.....	211
RIL_SercosIII.....	341
RIL_SercosIII DeviceReplacement.....	563
RMB_SercosIII_Util.....	431

C

CommandAborted.....	18
CommunicationState.....	217
ConfigState.....	217

D

Darstellung von Informationen	
Bezeichnungen und Abkürzungen.....	15
Dokumentation	
Änderungsverlauf.....	1
Done.....	18

DP_DEVICELIST.....	36
DP_MASTERINFO.....	37
DP_MASTERLIST.....	36
DP_SLAVELIST.....	35

E

Ein- und Ausgänge flankengesteuerter Funktionsbausteine, abschließende Bearbeitung.....	22
Ein- und Ausgänge flankengesteuerter Funktionsbausteine, permanente Bearbeitung...	28
Ein- und Ausgänge zustandsgesteuerter Funktionsbausteine, abschließende Bearbeitung.....	19
Ein- und Ausgänge zustandsgesteuerter Funktionsbausteine, permanente Bearbeitung...	25
Enable.....	18
Error_Code_DP.....	55
Error_Code_FDL.....	54
Error_Code_OB.....	54
Error_Tables	
ETHERCAT_TABLE.....	328
ETHERNET_IP_TABLE.....	250, 260
PB_DP_TABLE Profibus-DP Fehler.....	52
PROFINETIO_TABLE.....	195, 219
ErrorID	
RIL_ProfibusDP_02.....	52
ErrorIdent.....	52
ETHERCAT_TABLE.....	328
EtherNet/IP adapter	
CIP, Allgemeine Fehlercodes.....	262
Erweiterten Fehlercodes für Connection Manager.....	263
Fehlercodes, allgemeine.....	260
Fehlercodes, spezielle.....	261
Execute.....	18
ExecuteLock.....	18

F

Feedback.....	16
Fehlercodes	
IH_PBDPAck_PSDx_Inline.....	718
IH_PBDPAck_PSDx_S20.....	723
IH_PNIOAck_PSDx_Inline.....	736
IH_PNIOAck_PSDx_S20.....	731
IH_SIIIAck_SSDx_S20.....	743
IL_AI2Sf.....	580
IL_AI4Ef.....	585
IL_AI4IU.....	589
IL_AI8SfIs.....	594
IL_AO1Sf.....	626
IL_AO2SfBp.....	630
IL_AO2UI.....	634
IL_CntCount.....	641
IL_CntFrequency.....	647

Index

IL_CntPulse.....	650	IL_SIIIMasterState.....	393
IL_CntTime.....	656	IL_SIIIMasterStateDetails.....	394
IL_DPMasterState.....	39	IL_SIIIPParameterBackup.....	379
IL_ENIPAdapterBackup.....	236	IL_SIIIPParameterRestore.....	385
IL_ENIPAdapterRestore.....	239	IL_SIIIReadParameter.....	360
IL_ENIPAdapterState.....	255	IL_SIIIReadRealParameter.....	364
IL_ENIPAdapterStateDetails.....	256	IL_SIIIReadStringParameter.....	366
IL_Incln.....	694	IL_SIIIRemoteSlaveState.....	396
IL_PBDPSlaveDPV1GetWriteData.....	66	IL_SIIIRemoteSlaveStateDetails.....	398
IL_PBDPSlaveDPV1Polling.....	61	IL_SIIISetSercosPhase.....	425
IL_PBDPSlaveDPV1Response.....	64	IL_SIIISvcRead.....	345
IL_PNIOAssignName.....	93	IL_SIIISvcWrite.....	350
IL_PNIOCheckTopology.....	98	IL_SIIISvcWriteParameter.....	369
IL_PNIOControllerState.....	118	IL_SIIISvcWriteRealParameter.....	372
IL_PNIOControllerStateDetails.....	120	IL_SlaveMapListAddEntry.....	337
IL_PNIODeviceBackup.....	88	IL_SlaveMapListInit.....	334
IL_PNIODeviceReplacement.....	100	IL_Ssi.....	684
IL_PNIODeviceResetToDefault.....	102	IL_SsiIn.....	689
IL_PNIODeviceRestore.....	90, 91	IL_Status.....	258
IL_PNIODeviceSetIP.....	104	IL_Temp2Rtd.....	611
IL_PNIODeviceState.....	213	IL_Temp2Uth.....	605
IL_PNIODeviceStateDetails.....	214	IL_Temp4_8Rtd.....	617
IL_PNIODeviceStateDetailsXMAC.....	216	IL_Temp4_8RtdEf.....	623
IL_PNIODiagnosis.....	125	MB_SIIIDRn5020.....	557
IL_PNIOGetConfigDeviceNameList.....	126	MB_SIIIFcP5020.....	561
IL_PNIOGetControllerList.....	128	MB_SIIIPlcDriveDirectValveControl.....	521
IL_PNIOGetDiagDeviceNameList.....	129	MB_SIIIPlcDriveEfcFrequencyType01.....	496
IL_PNIOGetLiveList.....	106	MB_SIIIPlcDriveFlowControl.....	503
IL_PNIOGetLogMsg.....	107	MB_SIIIPlcDrivePosInterpolationType01.....	458
IL_PNIOGetPortData.....	110	MB_SIIIPlcDrivePositioningType01.....	454
IL_PNIOIoLinkBackupDevice.....	185	MB_SIIIPlcDrivePressureDirectValve- Control.....	537
IL_PNIOIoLinkCall.....	147	MB_SIIIPlcDrivePressureFlowControl.....	508
IL_PNIOIoLinkGetDeviceAccessLock.....	152	MB_SIIIPlcDrivePressureSwivelAngleSla- ve.....	517
IL_PNIOIoLinkIdent.....	157	MB_SIIIPlcDriveTorqueDirectValveCont- rol.....	527
IL_PNIOIoLinkIdentDetails.....	163	MB_SIIIPlcDriveTorqueFlowControl.....	532
IL_PNIOIoLinkRemoteDeviceState.....	168	MB_SIIIPlcDriveTorqueType01.....	448
IL_PNIOIoLinkRestoreDevice.....	188	MB_SIIIPlcDriveType01.....	440
IL_PNIOIoLinkSetDeviceAccessLock.....	173	MB_SIIIPlcDriveVelocityAltCtrl.....	542
IL_PNIOIoLinkSystemCommand.....	178	MB_SIIIPlcDriveVelocityType01.....	444
IL_PNIOReadRecord.....	76	MB_SIIIPlcPressureSwivelAngleControl.....	513
IL_PNIOReadWriteIMData.....	139	MB_SIIIPowerSupplyDcDclslandGrid.....	484
IL_PNIORemoteDeviceState.....	131	MB_SIIIPowerSupplyDriveIntegrated.....	470
IL_PNIORemoteDeviceStateDetails.....	133	MB_SIIIPowerSupplyStandalone.....	478
IL_PNIORestoreTopologie.....	116	RIH_Dali.....	810
IL_PNIOServiceRecord.....	80	RIH_Modbus.....	900
IL_PNIOStoreTopologie.....	112	RIL_EtherNetIP.....	251
IL_PNIOWriteRecord.....	84	Fehlertabellen	
IL_Pwm2Frequency.....	659	ETHERCAT_TABLE.....	328
IL_Pwm2PulseDirection.....	666	ETHERNET_IP_TABLE.....	250, 260
IL_Pwm2Pwm.....	662	PB_DP_TABLE Profibus-DP Fehler.....	52
IL_Pwm2SinglePulse.....	669	PROFINETIO_TABLE.....	195, 219
IL_RsCom.....	702	FM_Error_Code.....	53
IL_Sgi2f.....	600		
IL_SIIICommand.....	357		
IL_SIIIGetConfigSlaveAdrList.....	388		
IL_SIIIGetDiagSlaveAdrList.....	390		
IL_SIIIGetLiveSlaveAdrList.....	391		
IL_SIIIGetSercosPhase.....	424		

G

Gefahrenhinweise..... 3

H

Helpdesk..... 1001

Hotline..... 1001

I

IH_DaliAddress..... 749

IH_DaliAddToGroup..... 770

IH_DaliAddToScene..... 772

IH_DaliCallScene..... 751

IH_DaliDim..... 747

IH_DaliDirectPower..... 755

IH_DaliOff..... 774

IH_DaliOn..... 776

IH_DaliProgramShortAddress..... 783

IH_DaliQueryActualLevel..... 766

IH_DaliQueryDTR..... 794

IH_DaliQueryGroup..... 800

IH_DaliQueryParameter..... 796

IH_DaliQueryRandomAddress..... 803

IH_DaliQuerySceneLevel..... 781

IH_DaliQueryStatus..... 760

IH_DaliReadWrite..... 763

IH_DaliRemoveFromGroup..... 753

IH_DaliRemoveFromScene..... 758

IH_DaliReset..... 791

IH_DaliSearchAddress..... 779

IH_DaliStoreParameter..... 788

IH_INLN_DALI_ADDRTYPE..... 806

IH_INLN_DALI_DIMMODE..... 806

IH_INLN_DALI_LIGHTLEVEL..... 807

IH_INLN_DALI_OPMODE..... 807

IH_INLN_DALI_PARAMETER..... 808

IH_INLN_DALI_TOKEN..... 806

IH_INLN_MB_CONFIG..... 895

IH_INLN_MB_DATA..... 894

IH_INLN_MB_DEV_IDCODE..... 895

IH_INLN_MB_EVENTLOG..... 897

IH_INLN_MB_FIFODATA..... 895

IH_INLN_MB_RECORD..... 897

IH_INLN_MB_SUBREQUESTS..... 897

IH_INLN_MB_TOKEN..... 896

IH_MBGetCommEventCounter (FC 11)..... 813

IH_MBGetCommEventLog (FC 12)..... 816

IH_MBMaskWriteRegister (FC 22)..... 820

IH_MBReadCoils (FC 1)..... 823

IH_MBReadDeviceIdentification (FC 43)..... 827

IH_MBReadDiagnostics (FC 8)..... 834

IH_MBReadDiscreteInputs (FC 2)..... 837

IH_MBReadExceptionStatus (FC 7)..... 841

IH_MBReadFIFOQueue (FC 24)..... 844

IH_MBReadFileRecord (FC 20)..... 847

IH_MBReadHoldingRegisters (FC 3)..... 851

IH_MBReadInputRegisters (FC 4)..... 855

IH_MBReadWriteMultipleRegisters (FC 23)..... 859

IH_MBRTUMaster..... 863

IH_MBRTUSlave..... 868

IH_MBWriteFileRecord (FC 21)..... 875

IH_MBWriteMultipleCoils (FC 15)..... 879

IH_MBWriteMultipleRegisters (FC 16)..... 883

IH_MBWriteSingleCoil (FC 5)..... 888

IH_MBWriteSingleRegister (FC 6)..... 891

IH_PBDPAck_PSDx_Inline..... 713

IH_PBDPAck_PSDx_S20..... 718

IH_PNIOAck_PSDx_Inline..... 731

IH_PNIOAck_PSDx_S20..... 725

IH_S20AI4IU..... 942

IH_S20AI4RTD..... 948

IH_S20AI4UTH..... 950

IH_S20AI8..... 946

IH_S20AI8RTD..... 952

IH_S20AI8UTH..... 954

IH_S20AO4..... 958

IH_S20AO8..... 956

IH_S20CntInc..... 965

IH_S20ComConfiguration..... 934

IH_S20ComInterface..... 937

IH_S20ComInterfaceType02..... 935

IH_S20IOModuleStateDetails..... 931

IH_S20IOState..... 932

IH_S20IOStateDetails..... 932

IH_S20ModuleIdentDetails..... 933

IH_S20RSCom..... 970

IH_S20SsiAo1..... 961

IH_SIIIAck_SSDx_S20..... 737

IL_ADDRESSTYPE..... 337

IL_AI2Sf..... 576

IL_AI4Ef..... 580

IL_AI4IU..... 585

IL_AI8Sfls..... 590

IL_AO1Sf..... 623

IL_AO2SfBp..... 626

IL_AO2UI..... 630

IL_BACKUP_ERRORS..... 386

IL_BACKUP_OPTIONS..... 386

IL_BUSMASTER..... 31, 189, 246, 312, 429

IL_BUSSLAVE..... 67, 218, 258, 339

IL_CntCount..... 634

IL_CntFrequency..... 641

IL_CntPulse..... 647

IL_CntTime..... 650

IL_DPBaudrateGet..... 48

IL_DPDeviceListGet..... 47

IL_DPDevInfoGet..... 49

IL_DPIdent..... 39

IL_DPMasterState..... 37

IL_DPPrjSlaveListGet..... 49

IL_DPReadDiag..... 43

IL_DPSlaveDiagListGet..... 50

IL_DPSycFr..... 45

IL_DPV1Read..... 40

IL_DPV1Write..... 42

IL_ECANT_ADDRESS_TYPE..... 313

Index

IL_ECAT_BASE_DATA_TYPE.....	313	IL_ENIP_IDENTITY_ATTRIBUTES.....	247
IL_ECAT_COE_DIAG_HISTORY_FLAGS.....	314	IL_ENIP_REMOTE_ADAPTER_STATE_DE-	
IL_ECAT_COE_DIAG_HISTORY_STRUCT.....	315	TAILS.....	248
IL_ECAT_COE_DIAG_MSG_EXT.....	316	IL_ENIP_SCANNER_STATE.....	249
IL_ECAT_COE_DIAG_MSG_RAW.....	317	IL_ENIPAdapterBackup.....	233
IL_ECAT_COE_DIAG_TYPE.....	317	IL_ENIPAdapterRestore.....	236
IL_ECAT_DL_STATUS.....	318	IL_ENIPAdapterState.....	253
IL_ECAT_ESC_FEATURES.....	318	IL_ENIPAdapterStateDetails.....	255
IL_ECAT_MASTER_BUS_STATISTIC.....	319	IL_ENIPGetAttributeSingle.....	224
IL_ECAT_MASTER_STATE.....	319	IL_ENIPGetIdentityObjAllAttributes.....	226
IL_ECAT_MBX_CONFIG_PROTOCOLS.....	320	IL_ENIPRemoteAdapterState.....	239
IL_ECAT_MBX_ONLINE_PROTOCOLS.....	320	IL_ENIPRemoteAdapterStateDetails.....	241
IL_ECAT_MBX_SIZE_INFO.....	321	IL_ENIPScannerState.....	243
IL_ECAT_MEMORY_INFO.....	321	IL_ENIPScannerStateDetails.....	245
IL_ECAT_PORT_ERROR_COUNTER.....	321	IL_ENIPSendUnconnectedMessage.....	228
IL_ECAT_PORT_STATE.....	322	IL_ENIPSetAttributeSingle.....	230
IL_ECAT_SLAVE_CONFIG_INFO.....	322	IL_FBUS_COMMUNICATION_STATE....	190, 250,
IL_ECAT_SLAVE_ERROR_COUNTER.....	323		418
IL_ECAT_SLAVE_IDENTITY_INFO.....	324	IL_FBUS_SLAVE_RESULT.....	68
IL_ECAT_SLAVE_ONLINE_INFO.....	324	IL_FBUS_SLAVE_SERVICE.....	67
IL_ECAT_SLAVE_STATE.....	325	IL_FBUS_SLAVE_STATE.....	190, 250
IL_ECAT_SOE_ATTRIBUTE.....	326	IL_FIELDBUSOBJECT.....	337
IL_ECAT_SOE_DATA_LENGTH.....	326	IL_FIELDBUSTYPE.....	338
IL_ECAT_SOE_DATA_STATUS.....	327	IL_Incln.....	689
IL_ECAT_SOE_DATA_TYPE.....	327	IL_INLN_AI_CONFIG	
IL_ECAT_SOE_ELEMENT.....	327	Deklaration.....	702
IL_ECAT_SOE_PARAM_TYPE.....	328	IL_AI2Sf.....	577
IL_ECAT_STATE.....	328	IL_AI4Ef.....	582
IL_ECATCoeReadDiagHistory.....	300	IL_AI4IU.....	587
IL_ECATCoeReadSdo.....	266	IL_AI8Sfls.....	591
IL_ECATCoeWriteSdo.....	269	IL_INLN_AI_DATA.....	703
IL_ECATGetConfigEthercatAddrList.....	282	IL_AI2Sf.....	578
IL_ECATGetNumConfigSlaves.....	280	IL_AI4Ef.....	582
IL_ECATGetNumOnlineSlaves.....	281	IL_AI4IU.....	587
IL_ECATGetOnlineEthercatAddrList.....	284	IL_AI8Sfls.....	592
IL_ECATGetSlaveConfigInfo.....	285	IL_INLN_AI_RANGE	
IL_ECATGetSlaveOnlineInfo.....	287	Deklaration.....	702
IL_ECATMasterBusStatistic.....	290	IL_AI2Sf.....	579
IL_ECATMasterGetBusState.....	304	IL_AI4Ef.....	583
IL_ECATMasterSetBusState.....	305	IL_AI4IU.....	588
IL_ECATMasterState.....	289	IL_AI8Sfls.....	592
IL_ECATMasterStateDecoding.....	311	IL_INLN_AO_CONFIG	
IL_ECATRemoteSlaveClearErrorCounter.....	299	IL_AO2UI.....	632
IL_ECATRemoteSlaveGetDevState.....	307	IL_INLN_AO_DATA	
IL_ECATRemoteSlaveGetErrorCounter.....	297	IL_AO2SfBp.....	627
IL_ECATRemoteSlaveGetErrorCounterPeri-		IL_INLN_AO_RANGE	
od.....	295	IL_AO2UI.....	632
IL_ECATRemoteSlaveSetDevState.....	308	IL_INLN_CNT_COUNT	
IL_ECATRemoteSlaveSetErrorCounterPeri-		IL_CntCount.....	636
od.....	294	IL_INLN_CNT_FREQ	
IL_ECATRemoteSlaveState.....	292	IL_CntFrequency.....	643
IL_ECATSlaveStateDecoding.....	312	IL_INLN_CNT_FREQ_CONTROL	
IL_ECATSoeAttributeDecoding.....	311	IL_CntFrequency.....	643
IL_ECATSoeCommand.....	277	IL_INLN_CNT_GATE	
IL_ECATSoeIdnCoding.....	310	IL_CntCount.....	638
IL_ECATSoeRead.....	271	IL_INLN_CNT_INCONFIG	
IL_ECATSoeWrite.....	275	Deklaration.....	704
IL_ENIP_ADAPTER_STATE.....	259	IL_CntCount.....	637

IL_CntFrequency.....	644	IL_Ssi.....	672
IL_CntTime.....	653	IL_INLN_SSI_CTRL	
IL_INLN_CNT_OUTBEHAVE		IL_Ssi.....	675
IL_CntCount.....	638	IL_INLN_SSI_DATA	
IL_INLN_CNT_PULSE_FREQ		IL_Ssi.....	677
IL_CntPulse.....	649	IL_INLN_SSI_FREQ	
IL_INLN_CNT_PULSE_WIDTH		IL_SsiIn.....	687
IL_CntCount.....	639	IL_INLN_SSI_OPMODE	
IL_INLN_CNT_SG_OPERATION		IL_Ssi.....	677
IL_CntCount.....	638	IL_INLN_SSI_PARITY	
IL_CntFrequency.....	644	IL_SsiIn.....	687
IL_CntTime.....	653	IL_INLN_SSI_PATH	
IL_INLN_CNT_TIME		IL_Ssi.....	675
IL_CntTime.....	652	IL_INLN_TEMP_CONFIG	
IL_INLN_CNT_TIME_OUT_RELATION		Deklaration.....	704
IL_CntTime.....	654	IL_Temp2Rtd.....	607
IL_INLN_CNT_TIME_RES		IL_Temp4_8Rtd.....	613
IL_CntTime.....	654	IL_INLN_TEMP_CONNECTIONTYPE	
IL_INLN_DIAG_CODE		Deklaration.....	704
Deklaration.....	703	IL_Temp2Rtd.....	609
IL_AI2Sf.....	578	IL_Temp4_8Rtd.....	615
IL_AI4Ef.....	583	IL_INLN_TEMP_DATA	
IL_AI4IU.....	588	IL_Temp2Rtd.....	608
IL_AI8Sfls.....	592	IL_Temp2Uth.....	602
IL_AO1Sf.....	624	IL_Temp4_8Rtd.....	614
IL_AO2SfBp.....	628	IL_Temp4_8RtdEf.....	620
IL_AO2UI.....	632	IL_INLN_TEMP_EF_CONFIG	
IL_Sgi2f.....	597	Deklaration.....	706
IL_Temp2Rtd.....	608	IL_Temp4_8RtdEf.....	619
IL_Temp2Uth.....	603	IL_INLN_TEMP_EF_ODS	
IL_Temp4_8Rtd.....	614	Deklaration.....	706
IL_Temp4_8RtdEf.....	620	IL_Temp4_8RtdEf.....	621
IL_INLN_INC_CONFIG		IL_INLN_TEMP_EF_R0	
IL_Incln.....	691	Deklaration.....	706
IL_INLN_INC_DATA		IL_Temp4_8RtdEf.....	621
IL_Incln.....	691	IL_INLN_TEMP_EF_SENSOR	
IL_INLN_INC_MODE		Deklaration.....	707
IL_Incln.....	692	IL_Temp4_8RtdEf.....	621
IL_INLN_PWM_PULSE		IL_INLN_TEMP_R0	
IL_Pwm2Pwm.....	661	Deklaration.....	705
IL_INLN_PWM_TIME_BASE		IL_Temp2Rtd.....	608
IL_Pwm2SinglePulse.....	668	IL_Temp4_8Rtd.....	614
IL_INLN_RS_BAUDRATE.....	708	IL_INLN_TEMP_SENSOR	
IL_INLN_RS_CONFIG.....	707	Deklaration.....	705
IL_INLN_RS_DATAWIDTH.....	709	IL_Temp2Rtd.....	609
IL_INLN_RS_DIAG.....	710	IL_Temp4_8Rtd.....	615
IL_INLN_RS_INTERFACE.....	709	IL_INLN_UTH_CJCOMP	
IL_INLN_RS_MODULETYPE.....	709	IL_Temp2Uth.....	603
IL_INLN_RS_PROTOCOL.....	708	IL_INLN_UTH_CONFIG	
IL_INLN_SF_BP		IL_Temp2Uth.....	602
IL_AO2SfBp.....	628	IL_INLN_UTH_SENSOR	
IL_INLN_SGI_CONFIG		IL_Temp2Uth.....	603
IL_Sgi2f.....	596	IL_IOL_BACKUP_MODE.....	142
IL_INLN_SGI_DATA		IL_IOL_DEVICE_STATUS.....	140
IL_Sgi2f.....	597	IL_IOL_DEVICE_STATUS_DETAILS.....	140
IL_INLN_SGI_RANGE		IL_IOL_EVENT_QUALIFIER.....	141
IL_Sgi2f.....	597	IL_IOL_EVENT_QUALIFIER_INSTANCE.....	141
IL_INLN_SSI_CONFIG		IL_IOL_EVENT_QUALIFIER_MODE.....	141

Index

IL_IOL_EVENT_QUALIFIER_SOURCE.....	142	IL_PNIORemoteDeviceState.....	129
IL_IOL_EVENT_QUALIFIER_TYPE.....	142	IL_PNIORemoteDeviceStateDetails.....	131
IL_IOL_IN_PROC_SPOOL_TYPE01L.....	917	IL_PNIORestoreTopology.....	112
IL_IOL_OUT_PROC_SPOOL_TYPE01.....	917	IL_PNIOServiceRecord.....	76
IL_IOL_RESTORE_MODE.....	142	IL_PNIOSStoreTopology.....	110
IL_IOL_SERVICE.....	142	IL_PNIOWriteRecord.....	81
IL_IoLinkCall.....	906	IL_PSDx_ACK (alt).....	713
IL_IoLinkDirectValveControlType01.....	913	IL_Pwm2Frequency.....	656
IL_IoLinkErrorText.....	910	IL_Pwm2PulseDirection.....	663
IL_IoLinkIdent.....	911	IL_Pwm2Pwm.....	659
IL_PBDPSlaveDPV1GetWriteData.....	64	IL_Pwm2SinglePulse.....	666
IL_PBDPSlaveDPV1Polling.....	59	IL_RsCom.....	694
IL_PBDPSlaveDPV1Response.....	61	IL_SERCOS_DATA_STATUS.....	372
IL_PNIO_CONTROLLER_STATE.....	190	IL_Sgi2f.....	594
IL_PNIO_DEVICE_STATE.....	216	IL_Sgi2fOffset.....	600
IL_PNIO_DEVICE_XMAC.....	217	IL_SIII_DATA_LENGTH.....	427
IL_PNIO_DEVICE_XMAC_PORT.....	218	IL_SIII_DATA_TYPE.....	427
IL_PNIO_IM_RWMODE.....	135	IL_SIII_DEVICE_STATUS.....	402
IL_PNIO_IM0.....	133	IL_SIII_DEVICE_STATUS_UNION.....	402
IL_PNIO_IM1.....	134	IL_SIII_DEVICE_STATUS_UNION_BITS.....	403
IL_PNIO_IM2.....	134	IL_SIII_DEVICE_TYPE.....	406
IL_PNIO_IM3.....	134	IL_SIII_DIAGNOSTIC_NUMBER.....	404
IL_PNIO_IM4.....	134	IL_SIII_DIAGNOSTIC_NUMBER_CLASS.....	406
IL_PNIO_PORT_DATA.....	191	IL_SIII_DIAGNOSTIC_NUMBER_R.....	405
IL_PNIO_REMOTE_DEVICE_INFO.....	191	IL_SIII_DIAGNOSTIC_NUMBER_SOUR- CE_GROUP.....	405
IL_PNIO_REMOTE_DEVICE_STATE.....	192	IL_SIII_DIAGNOSTIC_NUMBER_SOURCE_TY- PE.....	405
IL_PNIO_TOPO_STATE.....	192	IL_SIII_DRIVE_DIAG.....	406
IL_PNIO_TOPO_STATUS.....	193	IL_SIII_DRV_ERROR.....	407
IL_PNIOAssignName.....	91	IL_SIII_DRV_ERROR_BITS.....	407
IL_PNIOCheckTopology.....	93	IL_SIII_DRV_MESSAGE.....	409
IL_PNIOControllerState.....	116	IL_SIII_DRV_MESSAGE_BITS.....	409
IL_PNIOControllerStateDetails.....	118	IL_SIII_DRV_WARNING.....	408
IL_PNIODeviceBackup.....	84	IL_SIII_DRV_WARNING_BITS.....	408
IL_PNIODeviceReplacement.....	98	IL_SIII_ELEMENT.....	427
IL_PNIODeviceResetToDefault.....	100	IL_SIII_IO_DIAG.....	410
IL_PNIODeviceRestore.....	88	IL_SIII_IO_DIAG_STATUS_CODE.....	411
IL_PNIODeviceSetIP.....	102	IL_SIII_IO_DIAG_STATUS_CODE_TYPE.....	411
IL_PNIODeviceState.....	211	IL_SIII_IoLinkProxy.....	903
IL_PNIODeviceStateDetails.....	213	IL_SIII_MASTER_STATE_DETAILS.....	398
IL_PNIODeviceStateDetailsXMAC.....	214	IL_SIII_PARAM_TYPE.....	428
IL_PNIODiagnosis.....	120	IL_SIII_REMOTE_SLAVE_STATE_DE- TAILS.....	399
IL_PNIOGetConfigDeviceNameList.....	125	IL_SIII_REMOTE_SLAVE_STATE_EX.....	401
IL_PNIOGetControllerList.....	127	IL_SIII_REMOTE_SLAVE_STATE_EX_BITS.....	401
IL_PNIOGetDiagDeviceNameList.....	128	IL_SIII_SERCOS_PHASE.....	428
IL_PNIOGetLiveList.....	104	IL_SIII_TOPOLOGY_PORT_STATUS.....	403
IL_PNIOGetLogMsg.....	106	IL_SIIIAtribute.....	418
IL_PNIOGetPortData.....	107	IL_SIIICommand.....	351
IL_PNIOIoLinkBackupDevice.....	180	IL_SIIIElementsToldn.....	421
IL_PNIOIoLinkCall.....	143	IL_SIIIGetConfigSlaveAdrList.....	387
IL_PNIOIoLinkGetDeviceAccessLock.....	149	IL_SIIIGetDiagSlaveAdrList.....	388
IL_PNIOIoLinkIdent.....	154	IL_SIIIGetLiveSlaveAdrList.....	390
IL_PNIOIoLinkIdentDetails.....	158	IL_SIIIGetSercosPhase.....	422
IL_PNIOIoLinkRemoteDeviceState.....	164	IL_SIIIMasterState.....	391
IL_PNIOIoLinkRestoreDevice.....	185	IL_SIIIMasterStateDetails.....	393
IL_PNIOIoLinkSetDeviceAccessLock.....	169	IL_SIIIPParameterBackup.....	373
IL_PNIOIoLinkSystemCommand.....	175		
IL_PNIOReadRecord.....	73		
IL_PNIOReadWriteIMData.....	135		

IL_SIIParameterRestore.....	379
IL_SIIReadParameter.....	358
IL_SIIReadRealParameter.....	363
IL_SIIReadStringParameter.....	365
IL_SIIRemoteSlaveState.....	394
IL_SIIRemoteSlaveStateDetails.....	396
IL_SIISetSercosPhase.....	424
IL_SIIStringToldn.....	426
IL_SIISvcRead.....	343
IL_SIISvcWrite.....	348
IL_SIIWriteParameter.....	367
IL_SIIWriteRealParameter.....	370
IL_SlaveMapListAddEntry.....	334
IL_SlaveMapListInit.....	333
IL_Ssi.....	669
IL_Ssiln.....	684
IL_Status.....	256
IL_Temp2Rtd.....	605
IL_Temp2Uth.....	600
IL_Temp4_8Rtd.....	611
IL_Temp4_8RtdEf.....	617
InGear.....	18
InOperation.....	18
InSync.....	18
InTorque.....	18
InVelocity.....	18
IoDrvModbusComPort.....	919
IoDrvModbusSerialSlave.....	921
IoDrvModbusTCP.....	920
IoDrvModbusTCPSlave.....	922

K

KNX.....	979
Applikationsbeispiel.....	986
Funktionen.....	979
Konfiguration	
R IB-IL-AO-2/SF.....	626
R IB-IL-AO-2/U/BP.....	626
R-IB IL AI 2/SF.....	576
R-IB IL AI 4/EF.....	580
R-IB IL AI 4/I.....	585
R-IB IL AI 4/U.....	585
R-IB IL AI8/IS.....	590
R-IB IL AI8/SF.....	590
R-IB IL AO 2/UI-PAC.....	630
R-IB IL CNT.....	634, 641, 647, 650
R-IB IL INC-IN-PAC.....	689
R-IB IL PWM/2-PAC.....	656, 659, 663, 666
R-IB IL RS232 PRO.....	694
R-IB IL RS485/422 PRO.....	694
R-IB IL SGI 2/F-PAC.....	594
R-IB IL SSI.....	669
R-IB IL SSI IN-PAC.....	684
R-IB IL TEMP 2 UTH-PAC.....	600
R-IB IL TEMP 4/8 RTD-EF-PAC.....	617
R-IB IL TEMP 4/8 RTD-PAC.....	611
R-IB IL TEMP2 RTD-PAC.....	605

R-IL_AO1Sf.....	623
Kritik.....	16
Kundenfeedback.....	16

L

Library	
RIH_IndraControl_S20.....	927
LinkState.....	216

M

MB_EFC_FUNCTION_CODE.....	496
MB_POWERSUPPLY_ACTUAL_MODE.....	487
MB_POWERSUPPLY_ERROR_MODE.....	488
MB_POWERSUPPLY_FUNCTION.....	490
MB_POWERSUPPLY_MODE.....	487
MB_POWERSUPPLY_TARGET_MODE.....	489
MB_PS_TARGET_MODE_DC_DC_IS-	
LAND_GRID.....	490
MB_SIII_CONTROLWORD.....	458
MB_SIII_DRIVE_SHUTDOWN_MODE.....	460
MB_SIII_DRN_ENABLE.....	561
MB_SIII_DRN_VAR_VEL_ENABLE.....	562
MB_SIII_EFC_CONTROLWORD.....	497
MB_SIII_EFC_STATUSWORD.....	497
MB_SIII_FCP_DUALPUMPSTATUS.....	562
MB_SIII_FCP_MASTERSLAVE.....	562
MB_SIII_FCP_PARAGROUP.....	562
MB_SIII_HOMING_MODE.....	460
MB_SIII_HYDR_CONTROL_MODE.....	542
MB_SIII_IN_PROC_EFC_FREQ.....	496
MB_SIII_IN_PROC_FLOW.....	543
MB_SIII_IN_PROC_POS_INTERPOLATION...	463
MB_SIII_IN_PROC_POSITIONING.....	462
MB_SIII_IN_PROC_PRESSURE_FLOW.....	543
MB_SIII_IN_PROC_PRESSURE_SPOOL.....	549
MB_SIII_IN_PROC_SPOOL.....	547
MB_SIII_IN_PROC_SWIVEL.....	545
MB_SIII_IN_PROC_SWIVEL_SLAVE.....	546
MB_SIII_IN_PROC_TORQUE.....	462
MB_SIII_IN_PROC_TORQUE_FLOW.....	548
MB_SIII_IN_PROC_TORQUE_SPOOL.....	548
MB_SIII_IN_PROC_VEL_ALT_CTRL.....	550
MB_SIII_IN_PROC_VELOCITY.....	461
MB_SIII_OUT_PROC_EFC_FREQ.....	496
MB_SIII_OUT_PROC_FLOW.....	543
MB_SIII_OUT_PROC_POS_INTERPOLATI-	
ON.....	463
MB_SIII_OUT_PROC_POSITIONING.....	462
MB_SIII_OUT_PROC_PRESSURE_FLOW.....	544
MB_SIII_OUT_PROC_PRESSURE_SPOOL....	550
MB_SIII_OUT_PROC_SPOOL.....	547
MB_SIII_OUT_PROC_SWIVEL.....	545
MB_SIII_OUT_PROC_SWIVEL_SLAVE.....	546
MB_SIII_OUT_PROC_TORQUE.....	462
MB_SIII_OUT_PROC_TORQUE_FLOW.....	549
MB_SIII_OUT_PROC_TORQUE_SPOOL.....	548
MB_SIII_OUT_PROC_VEL_ALT_CTRL.....	551

Index

MB_SIII_OUT_PROC_VELOCITY.....	461	Vorgehensweise im Fehlerfall.....	982
MB_SIII_POSCTRLWORD.....	460	RIH_IndraControl_S20.library.....	927
MB_SIII_STATUSWORD.....	459	RIH_Modbus.library.....	811
MB_SIIIDRn5020.....	551	RIH_Safety_ProfibusDP.library.....	713
MB_SIIIEfcCodeTolDn.....	491	RIH_Safety_ProfinetIO.library.....	725
MB_SIIIEfcFrequencyPlcDrive (alt).....	492	RIH_Safety_SercosIII.library.....	737
MB_SIIIFcP5020.....	557	RIH_Safety.library.....	711
MB_SIIIPlcDriveDirectValveControl.....	517	RIL_Ethercat.library.....	265
MB_SIIIPlcDriveEfcFrequencyType01.....	492	Azyklische Kommunikation CoE.....	266
MB_SIIIPlcDriveFlowControl.....	498	Azyklische Kommunikation SoE.....	271
MB_SIIIPlcDrivePosInterpolationType01.....	454	Datentypen.....	312
MB_SIIIPlcDrivePositioningType01.....	448	Diagnose.....	280
MB_SIIIPlcDrivePressureDirectValveControl....	532	Einführung und Übersicht.....	265
MB_SIIIPlcDrivePressureFlowControl.....	503	Fehlercodes.....	328
MB_SIIIPlcDrivePressureSwivelAngleControl.....	508	Hilfsfunktionen.....	310
MB_SIIIPlcDrivePressureSwivelAngleSlave.....	513	Zustandsmaschine.....	304
MB_SIIIPlcDriveTorqueDirectValveControl.....	521	RIL_EtherNetIP.library.....	223
MB_SIIIPlcDriveTorqueFlowControl.....	527	RIL_EtherNetIPAdapter.library.....	253
MB_SIIIPlcDriveTorqueType01.....	444	RIL_InlineIOModules.library.....	573
MB_SIIIPlcDriveType01.....	436	RIL_IoLink.library.....	901
MB_SIIIPlcDriveVelocityAltCtrl.....	537	RIL_MappingList.library.....	331
MB_SIIIPlcDriveVelocityType01.....	441	RIL_ProfibusDP_02.....	
MB_SIIIPositioningPlcDrive (alt).....	454	mehrere DP-Master.....	31
MB_SIIIPowerSupplyDcDcIslandGrid.....	480	RIL_ProfibusDP_02.library.....	31
MB_SIIIPowerSupplyDriveIntegrated.....	465	RIL_ProfibusDPSlave.library.....	57
MB_SIIIPowerSupplyStandalone.....	473	RIL_ProfinetIO.library.....	71
MB_SIIIPowerSupplyType03 (obsolet).....	465	RIL_ProfinetIODevice.library.....	211
MB_SIIIPowerSupplyType04 (obsolet).....	473	RIL_SercosIII.library.....	341
MB_SIIITorquePlcDrive (alt).....	444	RIL_SercosIIIDeviceReplacement.....	563
MB_SIIIVelocityPlcDrive (alt).....	441	RMB_SercosIII_Util.library.....	431
Modbus RTU.....	919	S	
Bibliotheken.....	919	Sercos-IO-Statuscode.....	411
ModbusRequest.....	924	Service-Hotline.....	1001
ModbusServer.....	922	Shutdown.....	18
ModbusSlaveComPort.....	919	Sicherheitshinweise.....	3
ModbusTCPSlave.....	920	Signalgrafik.....	3
P		Signalwörter.....	3
PNIO_CHANNELERROR.....	193	Spezielle Fehlercodes	
PNIO_DIAGINFO.....	194	Profinet IO IO-Link Device.....	209
PNIO_DIAGTYPE.....	194	Profinet IO IO-Link Master.....	209
PNIO_LIVELIST.....	194	Profinet IO-Controller.....	198
PnsState.....	216	PROFINET IO-Device.....	222
Profinet IO-Controller		Profinet IO-Status	199
Profinet IO-Status	199	Standardschnittstellen an Funktionsbausteinen.....	17
Spezielle Fehlercodes.....	198	Support.....	1001
PROFINET IO-Device		Symbole.....	5
Spezielle Fehlercodes.....	222	U	
R		Übersicht Bibliotheken und Zielsysteme.....	2
RIH_Dali.library.....	745	W	
RIH_IndraControl_S20_KNX.....	979	Warnhinweise.....	3
Allgemeines.....	979	Z	
Diagnose und Fehlercodes.....	981	Zu dieser Dokumentation.....	1
IH_S20KNXClient.....	983		
IH_S20KNXDPTxx.....	985		

Gültigkeit der Dokumentation.....	1

Notizen

Notizen

Bosch Rexroth AG

Postfach 13 57

97803 Lohr a.Main, Deutschland

Bgm.-Dr.-Nebel-Str. 2

97816 Lohr a.Main, Deutschland

Tel. +49 932 18 0

Fax +49 9352 18 8400

www.boschrexroth.com/electrics



R911393274