SIEMENS

Foreword

Fundamental safety instructions 1

System functions – devices 2

System variables – devices 3

SIMOTION

System Functions/Variables Devices

List Manual

Valid as of version 5.5

08/2022

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER
indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING
indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION
indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

℗ 08/2022 Subject to change

Copyright © Siemens AG 2022.
All rights reserved

Foreword

Foreword

Scope

This document is valid for SIMOTION SCOUT V5.5:

- SIMOTION SCOUT V5.5 (engineering system of the SIMOTION product family)
- SIMOTION Kernel V3.0 bis V5.5

Chapters in this documentation

This documentation describes the generally valid system functions and system variables for the SIMOTION C, P, and D hardware platforms.

- System functions – devices (Page 17)
- System variables – devices (Page 667)

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product level V5.5:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

Table of contents

Foreword 3

1 Fundamental safety instructions 13

1.1 General safety instructions 13

1.2 Industrial security 14

1.3 Danger to life due to software manipulation when using removable storage media 15

2 System functions – devices 17

2.1 Alarms and messages 17
2.1.1 _acknowledgeAlarmSqId 17
2.1.2 _acknowledgeAllAlarmSqId 19
2.1.3 _getPendingAlarms 20
2.1.4 _readDiagnosticData 21
2.1.5 _resetAlarmId 25
2.1.6 _resetAllAlarmId 26
2.1.7 _sendProcessInterrupt 27
2.1.8 _writeAndSendMessage 30

2.2 Communication 32
2.2.1 Data transfer 32
2.2.1.1 _abortReadWriteRecordJobs 32
2.2.1.2 _provideRecord 33
2.2.1.3 _readRecord 40
2.2.1.4 _readVariableDiagnosticData 46
2.2.1.5 _readVariableRecord 49
2.2.1.6 _receiveRecord 53
2.2.1.7 _tcpCloseConnection 61
2.2.1.8 _tcpCloseServer 62
2.2.1.9 _tcpOpenClient 63
2.2.1.10 _tcpOpenServer 65
2.2.1.11 _tcpReceive 67
2.2.1.12 _tcpSend 69
2.2.1.13 _udpReceive 71
2.2.1.14 _udpSend 73
2.2.1.15 _writeRecord 75
2.2.1.16 _writeVariableRecord 80
2.2.1.17 _Xreceive 83
2.2.1.18 _Xsend 85
2.2.2 Interface handling - Configuration 90
2.2.2.1 _activateDpSlave 90
2.2.2.2 _activateDpSlaveAddress 94
2.2.2.3 _activateNameOfStation 95
2.2.2.4 _assignNameOfStationToDevice 97
2.2.2.5 _deactivateDpSlave 100
2.2.2.6 _enableDpInterfaceSynchronizationMode 104
2.2.2.7 _setDpSlaveAddress 106

2.2.2.8 _setIPConfig ... 107
2.2.2.9 _setNameOfStation ... 109
2.2.2.10 _setPnIpConfig ... 112
2.2.2.11 _setPnNameOfStation ... 116
2.2.2.12 _synchronizeDpInterfaces ... 119
2.2.2.13 _udpAddMulticastGroupMembership ... 122
2.2.2.14 _udpDropMulticastGroupMembership ... 124
2.2.3 Interface handling - Status ... 126
2.2.3.1 _getActiveDpSlaveAddress ... 126
2.2.3.2 _getActiveNameOfStation ... 127
2.2.3.3 _getDoIndexNumberFromLogAddress ... 131
2.2.3.4 _getDpStationAddressFromLogDiagnosticAddress ... 133
2.2.3.5 _getGeoAddressFromLogAddress ... 136
2.2.3.6 _getIPConfig ... 139
2.2.3.7 _getLogDiagnosticAddressFromDpStationAddress ... 141
2.2.3.8 _getNextLogAddress ... 144
2.2.3.9 _getPnInterfacePortNeighbour ... 147
2.2.3.10 _getPnIpConfig ... 152
2.2.3.11 _getPnNameOfStation ... 155
2.2.3.12 _getPnPortNeighbour ... 159
2.2.3.13 _getSegmentIdentification ... 163
2.2.3.14 _getStateOfAllDpSlaves ... 165
2.2.3.15 _getStateOfAllDpStations ... 169
2.2.3.16 _getStateOfDiagnosticDataCommand ... 175
2.2.3.17 _getStateOfDpSlave ... 176
2.2.3.18 _getStateOfIO ... 179
2.2.3.19 _getStateOfProcessInterruptCommand ... 186
2.2.3.20 _getStateOfRecordCommand ... 187
2.2.3.21 _getStateOfSingleDpSlave ... 189
2.2.3.22 _GetStateOfXCommand ... 194
2.2.3.23 _getStationType ... 195
2.3 drives ... 197
2.3.1 SIMODRIVE ... 197
2.3.1.1 _abortAllReadWriteDriveParameterJobs ... 197
2.3.1.2 _POSMOA_control ... 198
2.3.1.3 _POSMOA_nControl ... 200
2.3.1.4 _POSMOA_rwAllParameter ... 201
2.3.1.5 _POSMOA_rwParameter ... 204
2.3.1.6 _readDriveFaults ... 205
2.3.1.7 _readDriveMultiParameter ... 212
2.3.1.8 _readDriveMultiParameterDescription ... 220
2.3.1.9 _readDriveParameter ... 228
2.3.1.10 _readDriveParameterDescription ... 236
2.3.1.11 _RWPAR_cyclic ... 243
2.3.1.12 _writeDriveMultiParameter ... 244
2.3.1.13 _writeDriveParameter ... 253
2.3.2 SIMOVERT ... 261
2.3.2.1 _abortAllReadWriteDriveParameterJobs ... 261
2.3.2.2 _readDriveFaults ... 262
2.3.2.3 _readDriveMultiParameter ... 269
2.3.2.4 _readDriveMultiParameterDescription ... 277
2.3.2.5 _readDriveParameter ... 285

2.3.2.6 _readDriveParameterDescription ... 293
2.3.2.7 _RWPAR_cyclic ... 300
2.3.2.8 _writeDriveMultiParameter ... 301
2.3.2.9 _writeDriveParameter ... 310
2.3.3 SINAMICS ... 318
2.3.3.1 _abortAllReadWriteDriveParameterJobs ... 318
2.3.3.2 _LineModule_control ... 319
2.3.3.3 _readDriveFaults ... 320
2.3.3.4 _readDriveMultiParameter ... 327
2.3.3.5 _readDriveMultiParameterDescription ... 335
2.3.3.6 _readDriveParameter ... 343
2.3.3.7 _readDriveParameterDescription ... 351
2.3.3.8 _resetDriveObjectFault ... 358
2.3.3.9 _setDriveObjectSTW ... 360
2.3.3.10 _writeDriveMultiParameter ... 362
2.3.3.11 _writeDriveParameter ... 371
2.3.4 SIMATIC ... 379
2.3.4.1 _ET200S_FC_control ... 379

2.4 I/O modules ... 380
2.4.1 AS-Interface ... 380
2.4.1.1 _ASI_cmdInterface ... 380
2.4.1.2 _ASI_rdAsiMonDiagnostic ... 381
2.4.2 Function modules - FM350-1 ... 383
2.4.2.1 _FM3501_control2 ... 383
2.4.2.2 _FM3501_diagnostic ... 385
2.4.3 Function modules - FM350-2 ... 386
2.4.3.1 _FM3502_control ... 386
2.4.3.2 _FM3502_diagnostic ... 389
2.4.3.3 _FM3502_read ... 392
2.4.3.4 _FM3502_write ... 395
2.4.4 Function modules - FM352 ... 398
2.4.4.1 _FM352_control ... 398
2.4.4.2 _FM352_diagnostic ... 402
2.4.4.3 _FM352_initialize ... 403
2.4.5 Communication modules - CP340 ... 406
2.4.5.1 _CP340_byteToPrintData ... 406
2.4.5.2 _CP340_dintToPrintData ... 407
2.4.5.3 _CP340_dwordToPrintData ... 408
2.4.5.4 _CP340_getV24Signals ... 409
2.4.5.5 _CP340_intToPrintData ... 410
2.4.5.6 _CP340_printer ... 411
2.4.5.7 _CP340_printMsgText ... 412
2.4.5.8 _CP340_realToPrintData ... 413
2.4.5.9 _CP340_receive ... 414
2.4.5.10 _CP340_send ... 415
2.4.5.11 _CP340_setV24Signals ... 416
2.4.5.12 _CP340_wordToPrintData ... 417
2.4.6 Communication modules - CP341 ... 418
2.4.6.1 _CP341_byteToPrintData ... 418
2.4.6.2 _CP341_dintToPrintData ... 419
2.4.6.3 _CP341_dwordToPrintData ... 420
2.4.6.4 _CP341_getV24Signals ... 421

2.4.6.5 _CP341_intToPrintData....422
2.4.6.6 _CP341_printer....423
2.4.6.7 _CP341_printMsgText....424
2.4.6.8 _CP341_realToPrintData....425
2.4.6.9 _CP341_receive....426
2.4.6.10 _CP341_send....427
2.4.6.11 _CP341_setV24Signals....428
2.4.6.12 _CP341_wordToPrintData....429
2.4.7 Communication modules - ET200S-SI....430
2.4.7.1 _ET200S_SI04_flowRts....430
2.4.7.2 _ET200S_SI04_flowV24....431
2.4.7.3 _ET200S_SI04_flowXon....432
2.4.7.4 _ET200S_SI04_getV24Sig....433
2.4.7.5 _ET200S_SI04_receive....434
2.4.7.6 _ET200S_SI04_send....435
2.4.7.7 _ET200S_SI04_setV24Sig....436
2.4.7.8 _ET200S_SI08_flowRts....437
2.4.7.9 _ET200S_SI08_flowV24....438
2.4.7.10 _ET200S_SI08_flowXon....439
2.4.7.11 _ET200S_SI08_getV24Sig....440
2.4.7.12 _ET200S_SI08_receive....441
2.4.7.13 _ET200S_SI08_send....442
2.4.7.14 _ET200S_SI08_setV24Sig....443
2.4.7.15 _ET200S_SI32_flowRts....444
2.4.7.16 _ET200S_SI32_flowV24....445
2.4.7.17 _ET200S_SI32_flowXon....446
2.4.7.18 _ET200S_SI32_getV24Sig....447
2.4.7.19 _ET200S_SI32_receive....448
2.4.7.20 _ET200S_SI32_send....449
2.4.7.21 _ET200S_SI32_setV24Sig....450
2.4.8 Ident systems....451
2.4.8.1 _PIB_001KB....451
2.4.8.2 _PIB_016KB....453
2.4.8.3 _PIB_032KB....455
2.4.9 Weighing technology - SIWAREX FTA....457
2.4.9.1 _FTA_control....457

2.5 Additional system functions....469
2.5.1 Operating state....469
2.5.1.1 _changeOperationMode....469
2.5.1.2 _restart....470
2.5.2 File access....471
2.5.2.1 _directoryPathDelete....471
2.5.2.2 _fileClose....474
2.5.2.3 _fileCopy....478
2.5.2.4 _fileDelete....482
2.5.2.5 _filehandle....485
2.5.2.6 _fileOpen....490
2.5.2.7 _fileRead....496
2.5.2.8 _fileReadLn....500
2.5.2.9 _fileRename....504
2.5.2.10 _fileSetPosition....507
2.5.2.11 _fileWrite....512

2.5.2.12 _fileWriteLn....516
2.5.2.13 _getStateOfFile....520
2.5.3 Data storage....524
2.5.3.1 _resetUnitData....524
2.5.3.2 _saveConfigData....527
2.5.3.3 _saveDiagnosticFiles....528
2.5.3.4 _savePersistentMemoryData....530
2.5.3.5 _saveTraceAndDiagnosticFiles....532
2.5.4 Modular machines....534
2.5.4.1 _activateConfiguration....534
2.5.4.2 _activateDpSlave....539
2.5.4.3 _activateDpSlaveAddress....543
2.5.4.4 _activateNameOfStation....544
2.5.4.5 _activateTo....546
2.5.4.6 _assignNameOfStationToDevice....549
2.5.4.7 _deactivateDpSlave....552
2.5.4.8 _deactivateTo....556
2.5.4.9 _getActiveDpSlaveAddress....559
2.5.4.10 _getActiveNameOfStation....560
2.5.4.11 _getConfigurationData....564
2.5.4.12 _getIPConfig....568
2.5.4.13 _getObjectName....570
2.5.4.14 _getPnInterfacePortNeighbour....571
2.5.4.15 _getPnIpConfig....576
2.5.4.16 _getPnNameOfStation....579
2.5.4.17 _getPnPortNeighbour....583
2.5.4.18 _getStateOfAllDpSlaves....587
2.5.4.19 _getStateOfAllDpStations....591
2.5.4.20 _getStateOfDpSlave....597
2.5.4.21 _getStateOfIO....600
2.5.4.22 _getStateOfSingleDpSlave....607
2.5.4.23 _getStateOfTo....612
2.5.4.24 _setDpSlaveAddress....615
2.5.4.25 _setIPConfig....616
2.5.4.26 _setModeSelfAdaptingConfiguration....618
2.5.4.27 _setNameOfStation....619
2.5.4.28 _setPnIpConfig....622
2.5.4.29 _setPnNameOfStation....626
2.5.4.30 _synchronizeDpInterfaces....629
2.5.5 Controller....632
2.5.5.1 _CTRL_pid....632
2.5.5.2 _CTRL_piStep....634
2.5.5.3 _CTRL_pwm....635
2.5.6 System time....636
2.5.6.1 _getInternalTimeStamp....636
2.5.6.2 _getPNSyncCounter....637
2.5.6.3 _getTimeDifferenceOfInternalTimeStamps....638
2.5.7 Device functions....639
2.5.7.1 _changeOperationMode....639
2.5.7.2 _getDeviceId....640
2.5.7.3 _getMemoryCardId....641
2.5.7.4 _restart....642

2.5.7.5 _setDeviceErrorLED....643
2.6 Task system....644
2.6.1 _disableScheduler....644
2.6.2 _enableScheduler....645
2.6.3 _getCurrentTaskName....646
2.6.4 _getStateOfNTPClockSynchronisation....647
2.6.5 Diagnostics....648
2.6.5.1 _taskTraceStart....648
2.6.5.2 _taskTraceStop....649
2.6.5.3 _taskTraceUserEvent....650
2.6.5.4 _taskTraceWriteout....651
2.7 Technology....652
2.7.1 Shared functions....652
2.7.1.1 BEGIN_SYNC....652
2.7.1.2 END_SYNC....653
2.7.1.3 _activateTo....654
2.7.1.4 _deactivateTo....657
2.7.1.5 _getObjectName....660
2.7.1.6 _getStateOfTo....661
2.7.1.7 _resetTechnologicalErrors....664
2.7.1.8 _startSyncCommands....665
3 System variables – devices....667
3.1 systemClock....667
3.2 servoControlClock....667
3.3 servoControlClock_fast....667
3.4 ipoClock....668
3.5 ipoClock_fast....668
3.6 servoTaskCycle....668
3.7 servoTaskCycle_fast....669
3.8 traceControl....670
3.9 traceState....671
3.10 TCOFctGenOverride....673
3.11 userData....673
3.12 taskRuntimeMonitoring....674
3.13 taskRuntime....675
3.14 effectiveTaskRuntime....677
3.15 systemLoad....679
3.16 numberOfSummarizedTaskOverflow....680
3.17 ipoClock_2....680
3.18 modeOfDpInterfaceSynchronization....681
3.19 stateOfDpInterfaceSynchronization....682

3.20 stateOfDpSlaveSynchronization....683
3.21 persistentDataPowerMonitoring....684
3.22 _upsData....686
3.23 _cpuData....687
3.24 _cpuDataRW....688
3.25 fanbattery....689
3.26 _startupData....690
3.27 _configurationManagement....691
3.28 modeOfOperation....692
3.29 dccAuxClock....692
3.30 dccAux_2Clock....693
3.31 _imData....693
3.32 _driveStates....695
3.33 _automaticRestart....695
3.34 _simulation....697
3.35 PCShutdown....697

Fundamental safety instructions

1

1.1 General safety instructions

⚠ WARNING

Danger to life if the safety instructions and residual risks are not observed

The non-observance of the safety instructions and residual risks stated in the associated hardware documentation can result in accidents with severe injuries or death.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

⚠ WARNING

Danger to life caused by machine malfunctions caused by incorrect or changed parameterization

Incorrect or changed parameterization can cause malfunctions on machines that can result in injuries or death.

- Protect the parameterization (parameter assignments) against unauthorized access.
- Respond to possible malfunctions by applying suitable measures (e.g. EMERGENCY STOP or EMERGENCY OFF).

1.2 Industrial security

Note

Industrial security

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions only form one element of such a concept.

Customer is responsible to prevent unauthorized access to its plants, systems, machines and networks. Systems, machines and components should only be connected to the enterprise network or the internet if and to the extent necessary and with appropriate security measures (e.g. use of firewalls and network segmentation) in place.

Additionally, Siemens' guidance on appropriate security measures should be taken into account. For more information about industrial security, please visit http://www.siemens.com/industrialsecurity.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends to apply product updates as soon as available and to always use the latest product versions. Use of product versions that are no longer supported, and failure to apply latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under http://www.siemens.com/industrialsecurity..

⚠ WARNING

Danger to life as a result of unsafe operating states resulting from software manipulation

Manipulation of the software, e.g. viruses, trojans, malware or worms, can cause unsafe operating states in your system that may lead to death, serious injury, and property damage.

- Keep the software up to date.
- Incorporate the automation and drive components into a state-of-the-art, integrated industrial security concept for the installation or machine.
- Make sure that you include all installed products in the integrated industrial security concept.
- Protect files on removable storage media against malware through appropriate protective measures, e.g. virus scanners.

1.3 Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

1.4 Note regarding the general data protection regulation

Siemens observes the principles of data protection, in particular the principle of data minimization (privacy by design). For this product this means:

The product does not process / store any personal data, only technical functional data (e.g. time stamp). If the user links this data with other data (e.g. shift plans) or stores personal data on the same medium (e.g. hard disk) and thus establishes a personal reference, the user must ensure compliance with data protection regulations.

System functions – devices

2

2.1 Alarms and messages

2.1.1 _acknowledgeAlarmSqId

This function acknowledges a single pending alarm of type 'AlarmSQ'.

Additional references

Additional information is available at:

- Basic Functions function manual
- or in the SIMOTION online help

Return value: DWORD

Description of the return values:

16#00000000	Alarm could be acknowledged without errors.
16#00008001	Message name not permitted. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_ILLEGAL_EVENT_ID
16#00008002	Loss of messages due to overflow (no more space in the message list). All 40 entries in the message list have been assigned. Entry is not made in the message list. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_LAST_ENTRY_USED
16#00008003	Loss of messages due to overflow (signal not yet sent, signal overflow). Send buffer for messaging clients is still being used by the last event. Entry is not made in the message list. Error may also occur if function calls come quickly one after the other with rising and falling edge. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_LAST_SIGNAL_USED
16#00008005	No display unit registered (message is entered into the list anyway). ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_EVENT_ID_NOT_USED
16#00008007	No job has been started yet with this message name (first call with Sig = FALSE). Falling edge (outgoing message) arrived without prior rising edge (incoming message) Entry is not made in the message list. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_IV_FIRST_CALL
16#00008009	Internal error. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_INTERNAL_ERROR
16#00008010	Entry was rejected; message acknowledgement memory full. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_NO_ENTRY

Parameters:

id

Direction:	Input parameter
Data type:	StructAlarmId
Parameter index:	1

ID of the alarm that is to be acknowledged.

2.1.2 _acknowledgeAllAlarmSqId

This function acknowledges all pending alarms of type 'AlarmSQ'.

Additional references

Additional information is available at:

- Basic Functions function manual
- or in the SIMOTION online help

Return value: DWORD

Description of the return values:

16#00000000	All alarms have been acknowledged without error
16#00008001	Message name not permitted. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_ILLEGAL_EVENT_ID
16#00008002	Loss of messages due to overflow (no more space in the message list). All 40 entries in the message list have been assigned. Entry is not made in the message list. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_LAST_ENTRY_USED
16#00008003	Loss of messages due to overflow (signal not yet sent, signal overflow). Send buffer for messaging clients is still being used by the last event. Entry is not made in the message list. Error may also occur if function calls come quickly one after the other with rising and falling edge. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_LAST_SIGNAL_USED
16#00008005	No display unit registered (message is entered into the list anyway). ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_EVENT_ID_NOT_USED
16#00008007	No job has been started yet with this message name (first call with Sig = FALSE). Falling edge (outgoing message) arrived without prior rising edge (incoming message) Entry is not made in the message list. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_IV_FIRST_CALL
16#00008009	Internal error. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_INTERNAL_ERROR
16#00008010	Entry was rejected; message acknowledgement memory full. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_NO_ENTRY

2.1.3 _getPendingAlarms

The function _getPendingAlarms provides come messages for 'AlarmS' type alarms and all non-acknowledged messages for type 'AlarmSQ'

Additional references

Additional information is available at:

- Basic Functions function manual
- or in the SIMOTION online help

Return value: StructRetGetPendingAlarmState

EnumAlarmIdType

ALARM_S (0)	Alarm type AlarmS
ALARM_SQ (1)	Alarm type AlarmSQ

EnumAlarmIdState

OUTGOING_ALARM (0)	Outgoing alarm
INCOMING_ALARM (1)	Incoming alarm

StructRetGetPendingAlarmState
Return value for _getPendingAlarms

Structure	Name	Data type	Unit
numberOfPendingAlarms	Id of the alarm	UINT	-
alarm	Array with states of the individual alarms	ARRAY [0..39] OF StructPendingAlarmState	-

StructPendingAlarmState
State of an alarm of type AlarmS/SQ

Structure	Name	Data type	Unit
id	Id of the alarm	StructAlarmId	-
_type	Type of the alarm	EnumAlarmIdType	-
state	State of the alarm	EnumAlarmIdState	-

The return value of the function consists of a structure that contains the number of 'AlarmS' and 'AlarmSQ' alarms that have not been acknowledged, and an array with the signal status of the associated alarm.

2.1.4 _readDiagnosticData

This function is used to output diagnostic data of a station or a module via the user program.

For PROFIBUS, the diagnostics for the slave are fetched, i.e. the slave supplies the complete diagnostic information. The structure of the diagnostic data is described in IEC 61158-6-3.

For PROFINET, the subslot-specific diagnostics are read (i.e. the data set 0x800A). The diagnostics for a subslot are supplied. The structure of the diagnostic data is described in IEC 61158-6-10.

Maximum 240 bytes of diagnostic data are read.

A maximum of 8 jobs can be issued simultaneously.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual
- Function Manual Supplement to the Command Interface for AS-Interface Master Modules
- or in the SIMOTION online help

Return value: StructRetDiagnosticData

StructRetDiagnosticData
Return parameter for _readDiagnosticData

Structure	Name	Data type	Unit
functionResult	Return value of function	DINT	-
dataLength	Length of diagnostic data read	UDINT	-
data	Diagnostic data	ARRAY [0..239] OF USINT	-

The return value consists of 'functionResult,' 'dataLength,' and 'data'.

Description of the return value:

functionResult

'functionResult' reports the job status.

16#00000000	Data set transfer okay, job completed with positive result. Job okay.
16#00007001	Must be repeated in the next program cycle. Initial call, initiation of data transfer okay (only when command is issued asynchronously).
16#00007002	Must be repeated in the next program cycle. Intermediate call, data transfer still active. (only when command is issued asynchronously).
16#00007003	Data set transfer aborted. Job aborted.
16#FFFF8090	Error while reading diagnostic data, job aborted. Specified logical diagnostic address invalid: No assignment is available in the system data blocks (SDBs). The logical diagnostic address does not exist in the project.

16#FFFF8091	Error while reading diagnostic data, job aborted. The _readDiagnosticData function cannot reach the specified logical diagnostic address. The logical diagnostics address is available in the project, but cannot be reached at the moment (e.g. the station is currently switched off, or the module has been unplugged).
16#FFFF809D	Error, job aborted. An I slave/I device cannot read diagnostic data of the higher-level master/controller.
16#FFFF809E	Error, job aborted. Attempt to abort a non-active function.
16#FFFF809F	Error, job aborted. Function not executable.
16#FFFF80A0	Error while reading diagnostic data, job aborted. Negative acknowledgment when reading from the module: - Module was removed during the read operation - Module defective
16#FFFF80A2	Error while reading diagnostic data, job aborted. Protocol error in Layer2: - Module not available
16#FFFF80A3	Error while reading diagnostic data, job aborted. Protocol error involving user interface/user: - Module not available
16#FFFF80A4-A7	Error while reading diagnostic data, job aborted. Error numbers according to IEC 61158 reserved - Query the cause at the module's manufacturer
16#FFFF80A8	Error while reading diagnostic data, job aborted. Error because of version conflict.
16#FFFF80A9	Error while reading diagnostic data, job aborted. Function not supported by the module.
16#FFFF80AA-AF	Error while reading diagnostic data, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF80B0	Error while reading diagnostic data, job aborted. - Protocol error in Layer2. - Protocol error involving user interface/user. - System function not supported for this module type.
16#FFFF80B2	Error during data set transfer, job aborted. Module reports access to an invalid slot/subslot.
16#FFFF80C2	Error while reading diagnostic data, command can be repeated immediately. The module is currently executing the maximum possible jobs for one CPU.
16#FFFF80C3	Error while reading diagnostic data, command can be repeated immediately. Required resources are presently occupied: - In the _readDiagnosticData function - In the module
16#FFFF80C4	Error while reading diagnostic data, command can be repeated immediately. Communication errors: - Parity error - SW Ready not set - Error in block length administration - Checksum error on CPU side - Checksum error on module side
16#FFFF80C5	Error while reading diagnostic data, can be repeated in the next program cycle. Distributed I/O not available at present.

16#FFFF80C6	Error while reading diagnostic data, can be repeated in the next program cycle. Reading of diagnostic data was aborted due to priority class abort (restart or background).
16#FFFF80CF	Error while reading diagnostic data, can be repeated in the next program cycle. Another _readDiagnosticData function call is currently active under this 'commandId'.

dataLength

'dataLength' reports the length of the diagnostic data read.

Data

'data' contains the read data (byte array, maximum of 240 bytes).

The structure and meaning of the data can be found in the documentation for the relevant DP station or the relevant module.

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of the logical address. This specification is optional. With INPUT, the logical address is in the input range. With OUTPUT, the logical address is in the output range. Does not have to be specified for diagnostic addresses.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical address or the diagnostic address of the DP station/module.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	3

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. With IMMEDIATELY, the transition takes place immediately (asynchronous call of). With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call of). With ABORT_CURRENT_COMMAND and the same 'commandId' as in the previous function call with IMMEDIATELY, the active function is aborted.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	4

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND.

2.1.5 _resetAlarmId

This function resets a single pending alarm of type 'AlarmS' or 'AlarmSQ'.

Additional references

Additional information is available at:

- Basic Functions function manual
- or in the SIMOTION online help

Return value: UDINT

Description of the return values:

16#00000000	Alarm could be reset without errors.
16#00008001	Message name not permitted. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_ILLEGAL_EVENT_ID
16#00008002	Loss of messages due to overflow (no more space in the message list). All 40 entries in the message list have been assigned. Entry is not made in the message list. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_LAST_ENTRY_USED
16#00008003	Loss of messages due to overflow (signal not yet sent, signal overflow). Send buffer for messaging clients is still being used by the last event. Entry is not made in the message list. Error may also occur if function calls come quickly one after the other with rising and falling edge. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_LAST_SIGNAL_USED
16#00008005	No display unit registered (message is entered into the list anyway). ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_EVENT_ID_NOT_USED
16#00008007	No job has been started yet with this message name (first call with Sig = FALSE). Falling edge (outgoing message) arrived without prior rising edge (incoming message) Entry is not made in the message list. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_IV_FIRST_CALL
16#00008009	Internal error. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_INTERNAL_ERROR
16#00008010	Entry was rejected; message acknowledgement memory full. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_NO_ENTRY

Parameters:

id

Direction:	Input parameter
Data type:	StructAlarmId
Parameter index:	1

ID of the alarm that is to be reset.

2.1.6 _resetAllAlarmId

This function resets all pending alarms of type 'AlarmS' or 'AlarmSQ'.

Additional references

Additional information is available at:

- Basic Functions function manual
- or in the SIMOTION online help

Return value: UDINT

Description of the return values:

16#00000000	Alarm could be reset without errors.
16#00008001	Message name not permitted. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_ILLEGAL_EVENT_ID
16#00008002	Loss of messages due to overflow (no more space in the message list). All 40 entries in the message list have been assigned. Entry is not made in the message list. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_LAST_ENTRY_USED
16#00008003	Loss of messages due to overflow (signal not yet sent, signal overflow). Send buffer for messaging clients is still being used by the last event. Entry is not made in the message list. Error may also occur if function calls come quickly one after the other with rising and falling edge. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_LAST_SIGNAL_USED
16#00008005	No display unit registered (message is entered into the list anyway). ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_EVENT_ID_NOT_USED
16#00008007	No job has been started yet with this message name (first call with Sig = FALSE). Falling edge (outgoing message) arrived without prior rising edge (incoming message) Entry is not made in the message list. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_IV_FIRST_CALL
16#00008009	Internal error. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_INTERNAL_ERROR
16#00008010	Entry was rejected; message acknowledgement memory full. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_NO_ENTRY

2.1.7 _sendProcessInterrupt

This function causes the DP slave to trigger a process interrupt on the DP master.

Additional references

Additional information is available at:

- Basic Functions function manual
- or in the SIMOTION online help

Return value: DINT

The return value reports the job status.

Description of the return value:

16#00000000	Data set transfer okay, job completed with positive result. Job okay.
16#00007001	Must be repeated in the next program cycle. Initial call, initiation of process interrupt okay.
16#00007002	Must be repeated in the next program cycle. Intermediate call, the triggered process alarm has not yet been acknowledged by the DP master.
16#00007003	Data set transfer aborted. Job aborted.
16#FFFF8090	Error: Job aborted. Incorrect initial address of address range in transfer memory.
16#FFFF8091	Error: Job aborted. Process interrupt disabled by configuration.
16#FFFF809E	Error: Job aborted. Attempt to abort a non-active function.
16#FFFF809F	Error: Job aborted. Function not executable.
16#FFFF80B5	Error: Job aborted. Call in DP master not permitted.
16#FFFF80C3	Error: Can be repeated in the next program cycle. Required resources are presently occupied: - In the _sendProcessInterrupt function - In the module
16#FFFF80C5	Error, can be repeated in the next program cycle. Distributed I/O not available at present.
16#FFFF80CF	Error, can be repeated in the next program cycle. Another _sendProcessInterrupt function call is currently active under this 'commandId'.

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of logical base address of slot in the DP slave that is triggering the alarm. With INPUT, the logical address is in the input range. With OUTPUT, the logical address is in the output range.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical base address of slot in the DP slave that is triggering the alarm.

details

Direction:	Input parameter
Data type:	DWORD
Parameter index:	3

Process interrupt details.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	4

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. With IMMEDIATELY, the transition takes place immediately (asynchronous call of). With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call of). With ABORT_CURRENT_COMMAND and the same 'commandId' as in the previous function call with IMMEDIATELY, the active function is aborted.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	5

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND.

2.1.8 _writeAndSendMessage

This function enters an event specified by the 'EventNumber', 'Info1', 'Info2', and 'eventClass' parameters in the diagnostics buffer and optionally sends it to all logged-on operator panels (OPs) simultaneously.

Additional references

Further information about the device diagnostics: Diagnostic buffer can found in the:

- Overview of Service and Diagnostics Options, Product Information
- or in the SIMOTION online help

Return value: DINT

Description of the return values:

0	Job OK
701	No station logged on
-701	Invalid event number
-702	Send buffer full

Parameters:

eventNumber

Direction:	Input parameter
Data type:	USINT
Parameter index:	1

Specifies a number that identifies the event throughout the project. The number is specified by the user during message configuration.

event

Direction:	Input parameter
Data type:	EnumIncomingOutgoing
Parameter index:	2

EnumIncomingOutgoing

INCOMING (0)	Incoming event
OUTGOING (1)	Outgoing event

Specifies event details. INCOMING designates the event as incoming. OUTGOING designates the event as outgoing.

eventClass

Direction:	Input parameter
Data type:	EnumEventClass
Parameter index:	3

EnumEventClass

PREDEFINED_MODULE_INFORMATION (8)	Internal
PREDEFINED_APPLICATION_EVENTS (9)	Internal
USER_EVENTS_1 (10)	Free user events 1 / Class A
USER_EVENTS_2 (11)	Free user events 2 / Class B

Specifies the event class. For USER_EVENTS_1 (=Class 1) and USER_EVENTS_2 (=Class 2), the event is specified by the user during message configuration.

send (optional)

Direction:	Input parameter
Data type:	EnumYesNo
Parameter index:	4
System default:	YES

EnumYesNo

NO (91)	No
YES (173)	Yes

Specifies the mode of operation for the event. NO enters the event in the diagnostics buffer only.

YES also sends the event to all logged-on OPs.

Note

Information about Yes

The functionality was previously not supported by the OP.

info1 (optional)

Direction:	Input parameter
Data type:	UINT
Parameter index:	5
System default:	16#0000

Specifies the auxiliary value. 'info1' corresponds to auxiliary value 1, which can be specified in the message text during message configuration.

info2 (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	6
System default:	16#00000000

Specifies the auxiliary value. 'info2' corresponds to auxiliary value 2, which can be specified in the message text during message configuration.

2.2 Communication

2.2.1 Data transfer

2.2.1.1 _abortReadWriteRecordJobs

This function aborts all initiated jobs of _readRecord and/or _writeRecord.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual
- or in the online help

Return value: DINT

The return value reports the job status.

Description of the return values:

16#00000000	Job completed without errors.
16#FFFF809F	Error, job aborted. Function not executable.
16#FFFF80C5	Error, can be repeated in the next program cycle. Function could not be executed. This may have the following reasons: - The function _abortReadWriteRecordJobs is already being executed. - The function _readRecord or _writeRecord is currently being called from another task.

Parameters:

mode (optional)

Direction:	Input parameter
Data type:	EnumDeviceAbortReadWriteJobs
Parameter index:	1
System default:	READ_AND_WRITE_JOBS

EnumDeviceAbortReadWriteJobs

READ_AND_WRITE_JOBS (0)	Abort of all read and write jobs
READ_JOBS (1)	Abort of all read jobs
WRITE_JOBS (2)	Abort of all write jobs

Specifies which jobs are to be deleted. Both _readRecord jobs and _writeRecord jobs are aborted with READ_AND_WRITE_JOBS. Only the _readRecord jobs are aborted with READ_JOBS. Only the _writeRecord jobs are aborted with WRITE _JOBS.

2.2.1.2 _provideRecord

The function block checks cyclically whether the I-Device has been sent a request to make a data set available. It sends the requested data set and a response to the higher-level controller.

The 'mode' input parameter is used to specify the operating mode of the _provideRecord function block. Jobs are started with a rising edge on the 'execute' input parameter. The job execution and tracking is shown in the 'busy', 'done', 'error' and 'errorId' output parameters. The 'done' output parameter shows that the job has been processed without an error. Any errors are shown in the 'error' and 'errorId' output parameters.

Parameters:

execute

Direction:	Input parameter
Data type:	BOOL
Parameter index:	1

A job is started with a rising edge with the operating mode specified in the 'mode' parameter.

mode

Direction:	Input parameter
Data type:	INT
Parameter index:	2

Operating mode to be executed.

0	With 'mode'=0, a cyclic check is performed as to whether the I-device has received a request to provide a data set. If a request is present for a data set provision for the I-device from a higher-level controller, the 'actualLogAddress' output parameter shows the logical address from which a data set is to be sent to the higher-level controller. The 'actualIoId' output parameter shows whether it is an input or output area of the I-device. The 'recordNumber' and 'reqDataLength' output parameters show the number and length of the data set to be sent.
1	Reserved.
2	With 'mode=2', the request for the provision of a data set from a slot/subslot of the I-device is received. If a request is present for the I-device from a higher-level controller, the _provideRecord function block describes the output parameter.
3	Send positive response to the higher-level controller. With 'mode=3', the _provideRecord function block checks the request of the higher-level controller to provide a data set. The requested data set is provided in the 'data' parameter and a positive acknowledgment is sent to the higher-level controller.

4	Send negative response to the higher-level controller. With 'mode=4', the _provideRecord function block checks the request of the higher-level controller to provide a data set. The request is rejected by the function block and a negative acknowledgment is sent to the higher-level controller. The reason for the rejection is shown in the 'responseCode' parameter.

ioId

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	3

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output range on the I device.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	4

Logical base address on the I device.

dataLength

Direction:	Input parameter
Data type:	UDINT
Parameter index:	5

Length of the data to be sent in the 'data' parameter in bytes.

It is specified whether the data set requested by the higher-level controller should be transferred in full or if only a certain length of the data set is to be transferred.

Note

Data sets can be sent or received with a maximum size of 32 Kbyte.

responseCode

Direction:	Input parameter
Data type:	BYTE
Parameter index:	6

Response code to the higher-level controller indicating whether the request for the provision of a data set is accepted or rejected. 'Zero' with 'mode=3' or response code according to the table for 'mode=4'.

In an error situation, the documentation for the associated module should be consulted for the specific error analysis.

00	Error-free
A0	Negative acknowledgment when sending a data set.
A2	Protocol error in Layer2.
A8	Error because of version conflict.
A9	Function not supported by the module.
AA - AF	Error in the data set transfer on the application side. Job aborted. Error numbers, manufacturer-specific in accordance with IEC 61158.
B0	Data set unknown to module.
B2	Access to an invalid slot/subslot.
B3	Module is reporting a type conflict.
B4	Module reports access to an invalid area.
B5	Module is not ready.
B6	Module rejects access.
B7	Module reports an illegal range for a parameter or value.
B8	Module is reporting an invalid parameter.
B9	Module is reporting an invalid type.
BA - BF	Error in data set transfer with regard to access Job aborted. Error numbers, manufacturer-specific in accordance with IEC 61158.
C0	Conflict when sending data sets.
C2	Module is currently processing the possible maximum of jobs.
C3	Module is reporting that all resources are in use.
C8 - CF	Error in data set transfer with regard to resources. Job aborted. Error numbers, manufacturer-specific in accordance with IEC 61158.

actualMode

Direction:	Output parameter
Data type:	INT
Parameter index:	7

'actualMode' shows the mode that has been started via 'execute'. All responses in 'done', 'busy', 'error' refer to 'actualMode'.

done

Direction:	Output parameter
Data type:	BOOL
Parameter index:	8
System default:	FALSE

Job completed without errors according to the 'mode' parameter.

busy

Direction:	Output parameter
Data type:	BOOL
Parameter index:	9
System default:	FALSE

Job active according to the 'mode' parameter.

error

Direction:	Output parameter
Data type:	BOOL
Parameter index:	10
System default:	FALSE

Job completed with errors according to the 'mode' parameter.

errorId

Direction:	Output parameter
Data type:	DWORD
Parameter index:	11
System default:	0

Error specification

In an error situation, the documentation for the associated module should be consulted for the specific error analysis.

16#00000000	Error-free
16#FFFF8090	Specified logical base address invalid.
16#FFFF8091	Specified logical base address cannot be reached.
16#FFFF809A	The PROFINET interface does not support this service.

16#FFFF809C	Internal temporary collision of DPV1 jobs.
16#FFFF809D	The interface is not configured as an I-device. a controller interface cannot send this data set.
16#FFFF809F	The function cannot be executed.
16#FFFF80A1	Negative acknowledgment when sending; function block cannot be used for the module.
16#FFFF80A2	Protocol error in Layer2.
16#FFFF80A3	PROFINET IO: General CM error.
16#FFFF80A4-A7	Error during data set transfer. Job aborted. Error numbers, manufacturer-specific in accordance with IEC 61158.
16#FFFF80A8	Error during data set transfer. Job aborted. Error because of version conflict.
16#FFFF80A9	Function not supported by the module.
16#FFFF80AA-AF	Error during data set transfer on the application side; job aborted, error numbers manufacturer-specific in accordance with IEC 61158.
16#FFFF80B0	Data set unknown to module.
16#FFFF80B2	Module reports access to an invalid slot/subslot.
16#FFFF80B3	Module reports type conflict.
16#FFFF80B4	Module reports access to an invalid area.
16#FFFF80B5	Module is not ready.
16#FFFF80B6	Module rejects access.
16#FFFF80B7	Module reports an illegal range for a parameter or value.
16#FFFF80B8	Module reports an invalid parameter.
16#FFFF80B9	Module reports an invalid type.
16#FFFF80BA-BF	Error in data set transfer with regard to access Job aborted. Error numbers, manufacturer-specific in accordance with IEC 61158.
16#FFFF80C0	Conflict when sending a data set.
16#FFFF80C2	The module is currently processing the possible maximum of jobs.
16#FFFF80C3	Module is reporting that all resources are in use.
16#FFFF80C4	Communication error
16#FFFF80C5	Distributed I/O not available.
16#FFFF80C6	Data set transfer has been aborted because of priority class abort.
16#FFFF80C7	Interruption due to restart or cold start of the DP master.
16#FFFF80C8-CE	Error in data set transfer with regard to resources. Job aborted. Error numbers, manufacturer-specific in accordance with IEC 61158.
16#FFFF80CF	The function block is currently processing another job.

actualLogAddress

Direction:	Output parameter
Data type:	DINT
Parameter index:	12
System default:	0

Logical address on the I device for which a data set is to be sent to the higher-level controller.

actualIoId

Direction:	Output parameter
Data type:	EnumIoIdType
Parameter index:	13
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Logical address on the I device for which a data set is to be sent to the higher-level controller.

recordNumber

Direction:	Output parameter
Data type:	UDINT
Parameter index:	14
System default:	0

Number of the data set to be sent.

reqDataLength

Direction:	Output parameter
Data type:	UDINT
Parameter index:	15
System default:	0

Length of the data to be sent requested from the higher-level controller.

clientId

Direction:	Output parameter
Data type:	UDINT
Parameter index:	16
System default:	0

Unique assignment to higher-level communication partner (e.g. PN controller, supervisor) for this data set job.

data

Direction:	Input/output parameter
Data type:	ARRAY [..] OF BYTE
Parameter index:	17

Data to be sent.

2.2.1.3 _readRecord

This function causes a data set to be transferred from an I/O module to the user program.

It is used for CP and FM and dynamic parameter assignment of analog and digital modules and to read/write parameters from a drive.

A maximum of 8 jobs can be issued simultaneously.

Note

Only one job should be issued per peripheral module in order to ensure processing.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual
- or in the online help

Return value: StructRetReadRecord

StructRetReadRecord
Return parameter for _readRecord

Structure	Name	Data type	Unit
functionResult	Return value	DINT	-
dataLength	Length of data read	UDINT	-
data	User data	ARRAY [0..239] OF USINT	-

Description of the return values:

The return value consists of 'functionResult,' 'dataLength,' and 'data'.

In an error situation, the documentation for the associated module should be consulted for the specific error analysis.

'functionResult'

reports the job status.

16#00000000	Data set transfer okay, job completed with positive result. Job okay.
16#00007001	Must be repeated in the next program cycle. Initial call, initiation of data transfer okay (only when command is issued asynchronously).
16#00007002	Must be repeated in the next program cycle. Intermediate call, data transfer still active. (only when command is issued asynchronously).
16#00007003	Data set transfer aborted. Job aborted.
16#FFFF8090	Error during data set transfer, job aborted. Specified logical base address invalid: No assignment is available in SDBs, or there is no base address.

16#FFFF8091	Error during data set transfer, job aborted. The '_readRecord' function cannot reach the specified logical base address.
16#FFFF809C	Error, job aborted. Internal temporary collision of DPV1 jobs. The parameter job previously issued with _writeRecord is no longer valid. - After a wait time of several 100 ms, use _writeRecord to reissue the parameter job. - Then use _readRecord to read out the requested data.
16#FFFF809D	Error, job aborted. An I slave/I device interface is unable to read data sets from the higher-level master/controller.
16#FFFF809E	Error, job aborted. Attempt to abort a non-active function.
16#FFFF809F	Error, job aborted. Function not executable.
16#FFFF80A0	Error during data set transfer, job aborted. Negative acknowledgment when reading from the module: - Module was removed during the read operation - Module defective
16#FFFF80A2	Error during data set transfer, job aborted. Protocol error in Layer2: - Module not available
16#FFFF80A3	Error during data set transfer, job aborted. Protocol error involving user interface/user: - Module not available.
16#FFFF80A4-A7	Error during data set transfer, job aborted. Error numbers, reserved in accordance with IEC 61158. - Query the cause at the module's manufacturer.
16#FFFF80A8	Error during data set transfer, job aborted. Error because of version conflict.
16#FFFF80A9	Error during data set transfer, job aborted. Function not supported by the module.
16#FFFF80AA-AF	Error during data set transfer, job aborted. Error numbers, manufacturer-specific in accordance with IEC 61158. - Query the cause at the module's manufacturer.
16#FFFF80B0	Error during data set transfer, job aborted. - System function not supported for this module type. - Data set unknown to module.
16#FFFF80B1	Error during data set transfer, job aborted. Incorrect length specified in 'dataLength' parameter.
16#FFFF80B2	Error during data set transfer, job aborted. Module reports access to an invalid slot/subslot.
16#FFFF80B3	Error during data set transfer, job aborted. Module reports type conflict.
16#FFFF80B4	Error during data set transfer, job aborted. Module reports access to an invalid area.

16#FFFF80B5	Error during data set transfer, job aborted. Module is not ready. Within the PROFIdrive profile (parameter model), this value has the following meaning: - Access is temporarily denied due to an internal processing status. - A write request was not issued before a read request. - A write and read request is already pending. - A read request is already pending.
16#FFFF80B6	Error during data set transfer, job aborted. Module rejects access.
16#FFFF80B7	Error during data set transfer, job aborted. Module reports an illegal range for a parameter or value.
16#FFFF80B8	Error during data set transfer, job aborted. Module reports an invalid parameter.
16#FFFF80B9	Error during data set transfer, job aborted. Module reports an invalid type.
16#FFFF80BA-BF	Error during data set transfer, job aborted. Error numbers, manufacturer-specific in accordance with IEC 61158. - Query the cause at the module's manufacturer.
16#FFFF80C0	Error during data set transfer, can be repeated in the next program cycle. The module lists the data set, but the module does not yet have any read data.
16#FFFF80C2	Error during data set transfer, command can be repeated immediately. The module is currently executing the maximum possible jobs for one CPU.
16#FFFF80C3	Error during data set transfer, command can be repeated immediately. Required resources are presently occupied: - In the _readRecord function - In the module
16#FFFF80C4	Error during data set transfer, command can be repeated immediately. Communication errors: - Parity error - SW Ready not set - Error in block length administration - Checksum error on CPU side - Checksum error on module side
16#FFFF80C5	Error during data set transfer, Can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF80C6	Error during data set transfer, can be repeated in the next program cycle. Data set transfer has been aborted due to priority class abort (restart or background).
16#FFFF80CF	Error during data set transfer, can be repeated in the next program cycle. Another _readRecord function call is currently active under this 'commandId'.

'dataLength'

reports the length of the data read.

'data'

contains the read data (byte array, maximum of 240 bytes).

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of logical base address of the I/O module. With INPUT, the logical address of the I/O module is in the input range. With OUTPUT, the logical address of the I/O module is in the output range.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical base address of the I/O module.

Note

You can used both an input and an output address when hybrid modules are used.

recordNumber

Direction:	Input parameter
Data type:	UDINT
Parameter index:	3

Specification of the number of the data set of the peripheral module to be read.

offset

Direction:	Input parameter
Data type:	UDINT
Parameter index:	4

Specifies the offset of the first byte of the data to be read during transfer of the data set of the I/O module. The parameter is not currently supported.

dataLength (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	5
System default:	240

Length of data to be read during the transfer of the data set of the I/O module.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	6

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. With IMMEDIATELY, the transition takes place immediately (asynchronous call of). With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call of). With ABORT_CURRENT_COMMAND and the same 'commandId' as in the previous function call with IMMEDIATELY, the active function is aborted.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	7

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND.

2.2.1.4 _readVariableDiagnosticData

This function block is used to output diagnostic data of a station or module via the user program. The diagnostics format for PROFINET IO V2.2 and PROFIBUS DP is described in the IEC 61158-6 standard. Maximum 65535 bytes of diagnostics data are read.

Parameters:

EXECUTE

Direction:	Input parameter
Data type:	BOOL
Parameter index:	1

Starts a new job with a rising edge.

IOID

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	2

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Specifies the I/O direction.

LOGADDR

Direction:	Input parameter
Data type:	DINT
Parameter index:	3

Specifies the logical address of the module.

DIAGDATA

Direction:	Input/output parameter
Data type:	ARRAY [..] OF BYTE
Parameter index:	4

Specifies the diagnostic data to be read.

DONE

Direction:	Output parameter
Data type:	BOOL
Parameter index:	5
System default:	FALSE

Message that the block was ended.

BUSY

Direction:	Output parameter
Data type:	BOOL
Parameter index:	6
System default:	FALSE

Message that the block is being processed.

ERROR

Direction:	Output parameter
Data type:	BOOL
Parameter index:	7
System default:	FALSE

Message that the block was executed with errors.

ERRORID

Direction:	Output parameter
Data type:	DWORD
Parameter index:	8
System default:	0

Description of the error messages:

In an error situation, the documentation for the associated module should be consulted for the specific error analysis.

16#00000000	Error-free
16#FFFF8090	Specified logical diagnostic address invalid: No assignment is available in the system data blocks (SDBs). The logical diagnostic address does not exist in the project.
16#FFFF8091	The _readDiagnosticData function cannot reach the specified logical diagnostic address. The logical diagnostics address is available in the project, but cannot be reached at the moment (e.g. the station is currently switched off, or the module has been unplugged).
16#FFFF809D	An I slave/I device cannot read diagnostic data of the higher-level master/ controller.
16#FFFF809E	Attempt to abort a non-active function.
16#FFFF809F	Function not executable.
16#FFFF80A0	Negative acknowledgment when reading from the module: - Module was removed during the read operation - Module defective
16#FFFF80A2	Protocol error in Layer2: - Module not available.
16#FFFF80A3	Protocol error involving user interface/user: - Module not available
16#FFFF80A4-A7	Error numbers, reserved in accordance with IEC 61158. - Query the cause at the module's manufacturer.

16#FFFF80A8	Error because of version conflict.
16#FFFF80A9	Function not supported by the module.
16#FFFF80B0	Data set unknown to module.
16#FFFF80B2	Module reports access to an invalid slot/subslot.
16#FFFF80B4	Module reports access to an invalid area.
16#FFFF80B5	Module is not ready.
16#FFFF80B6	Module rejects access.
16#FFFF80C0	The module has the data set, but no read data is available.
16#FFFF80C2	The module is currently processing the possible maximum of jobs.
16#FFFF80C3	Required resources are presently occupied.
16#FFFF80C4	Communication error
16#FFFF80C5	Distributed I/O not available.
16#FFFF80C6	Data set transfer has been aborted because of priority class abort.
16#FFFF80CF	The function block is currently processing another job.

LEN

Direction:	Output parameter
Data type:	UDINT
Parameter index:	9
System default:	0

Length of the read data set in bytes.

2.2.1.5 _readVariableRecord

This function block causes a data transfer from an I/O module to the user program. Maximum blocks of 32 K (32768 bytes) are read.

This function block or the device system function _readRecord can be used for reading.

Additional references

Further information is available at:

- SIMOTION Communication System Manual
- or in the online help

Parameters:

EXECUTE

Direction:	Input parameter
Data type:	BOOL
Parameter index:	1

Starts a new job with a rising edge.

Note

If the input 'EXECUTE' is set back to 'FALSE' after starting a job, the output signals ERROR, ERRORID, DONE and LEN are pending for exactly one cycle.

IOID

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	2

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Specifies the I/O direction.

LOGADDR

Direction:	Input parameter
Data type:	DINT
Parameter index:	3

Specifies the logical address of the module.

DSNR

Direction:	Input parameter
Data type:	UDINT
Parameter index:	4

Specifies the data set number.

RECORD

Direction:	Input/output parameter
Data type:	ARRAY [..] OF BYTE
Parameter index:	5

Specifies the data set to be read.

DONE

Direction:	Output parameter
Data type:	BOOL
Parameter index:	6
System default:	FALSE

Message that the block was ended.

BUSY

Direction:	Output parameter
Data type:	BOOL
Parameter index:	7
System default:	FALSE

Message that the block is being processed.

ERROR

Direction:	Output parameter
Data type:	BOOL
Parameter index:	8
System default:	FALSE

Message that the block was executed with errors.

ERRORID

Direction:	Output parameter
Data type:	DWORD
Parameter index:	9
System default:	0

Description of the error messages:

In an error situation, the documentation for the associated module should be consulted for the specific error analysis.

16#00000000	Error-free
16#FFFF8090	Specified logical base address invalid.
16#FFFF8091	Specified logical base address cannot be reached.
16#FFFF809B	The target memory provided is not large enough.
16#FFFF809C	Internal temporary collision of DPV1 jobs.
16#FFFF809D	An I slave / I device interface cannot read any data sets.
16#FFFF809E	Error, job aborted. Attempt to abort a non-active function.
16#FFFF809F	Error, job aborted. Function not executable.
16#FFFF80A0	Negative acknowledgment when reading a module.
16#FFFF80A2	Error during data set transfer, job aborted. Protocol error in Layer2: - Module not available
16#FFFF80A3	Error during data set transfer, job aborted. Protocol error involving user interface/user: - Module not available.
16#FFFF80A8	Error because of version conflict.
16#FFFF80A9	Function not supported by the module.

16#FFFF80AA-AF	Error during data set transfer, job aborted. Error numbers, manufacturer-specific in accordance with IEC 61158. - Query the cause at the module's manufacturer.
16#FFFF80B0	Data set unknown to module.
16#FFFF80B1	Incorrect length specified in 'dataLength' parameter.
16#FFFF80B2	Module reports access to an invalid slot/subslot.
16#FFFF80B3	Module reports type conflict.
16#FFFF80B4	Module reports access to an invalid area.
16#FFFF80B5	Module is not ready.
16#FFFF80B6	Module rejects access.
16#FFFF80B7	Module reports an illegal range for a parameter or value.
16#FFFF80B8	Module reports an invalid parameter.
16#FFFF80B9	Module reports an invalid type.
16#FFFF80BA-BF	Error during data set transfer, job aborted. Error numbers, manufacturer-specific in accordance with IEC 61158. - Query the cause at the module's manufacturer.
16#FFFF80BB	Error during data set transfer, job aborted. Error numbers, manufacturer-specific in accordance with IEC 61158. - Query the cause at the module's manufacturer.
16#FFFF80C0	The module has the data set, but no read data is available.
16#FFFF80C2	The module is currently processing the possible maximum of jobs.
16#FFFF80C3	Required resources are presently occupied.
16#FFFF80C4	Communication error
16#FFFF80C5	Distributed I/O not available.
16#FFFF80C6	Data set transfer has been aborted because of priority class abort.
16#FFFF80CF	The function block is currently processing another job.

LEN

Direction:	Output parameter
Data type:	UDINT
Parameter index:	10
System default:	0

Length of the read data set in bytes.

2.2.1.6 _receiveRecord

The function block performs a cyclical check as to whether the I device has a request for data set receipt It makes the data set available at the output parameters and sends a response to the higher-level controller.

The 'mode' input parameter is used to specify the operating mode of the _receiveRecord function block. A jobs is started with a rising edge on the 'execute' input parameter. The job execution and tracking is shown in the 'busy', 'done', 'error' and 'errorId' output parameters. The 'done' output parameter shows that the job has been processed without an error. Any errors are shown in the 'error' and 'errorId' output parameters.

Parameters:

execute

Direction:	Input parameter
Data type:	BOOL
Parameter index:	1

A job is started with a rising edge with the operating mode specified in the 'mode' parameter.

mode

Direction:	Input parameter
Data type:	INT
Parameter index:	2

Operating mode to be executed.

0	With 'mode=0', a cyclic check is performed as to whether the I-device has received a request to receive a data set. If a data set is present for the I-device from a higher-level controller, the 'actualLogAddress' output parameter describes the address to which the present data set is directed. The 'actualIoId' output parameter shows whether it is an input or output area of the I-device. The 'recordNumber' and 'dataLength' output parameters show the number and length of the data set to be received.
1	Reserved.
2	'mode=2': Receives a data set for a slot/subslot of the I-device. If a data set from the higher-level controller is present for the I-device, the _receiveRecord function block transfers the data set to the 'data' parameter.
3	'mode=3': Send positive response to the higher-level controller. The _receiveRecord function block checks the request from the higher-level controller to receive a data set. The _receiveRecord function block accepts the present data set if the 'responseCode=0' input parameter is set and sends a positive acknowledgment to the higher-level controller.

4	'mode=4': Send negative response to the higher-level controller. The _receiveRecord function block checks the request from the higher-level controller to receive a data set. The present data set is rejected by the function block and a negative acknowledgment is sent to the higher-level controller. The reason for the rejection is shown in the 'responseCode' parameter.

ioId

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	3

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output range on the I device.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	4

Logical base address on the I device.

responseCode

Direction:	Input parameter
Data type:	BYTE
Parameter index:	5

Response code to the higher-level controller indicating whether the pending data set is to be accepted or rejected. 'Zero' with 'mode=3' or response code according to the table for 'mode=4'.

In an error situation, the documentation for the associated module should be consulted for the specific error analysis.

00	Error-free
A1	Negative acknowledgment when receiving a data set.
A2	Protocol error in Layer2.
A8	Error during data set transfer; job aborted, error due to version conflict.
A9	Function not supported by the module.
AA - AF	Error in the data set transfer on the application side. Job aborted. Error numbers, manufacturer-specific in accordance with IEC 61158.
B0	Data set unknown to module.
B1	Incorrect length specified in 'dataLength'.
B2	Access to an invalid slot/subslot.
B3	Module reports type conflict.
B4	Module reports access to an invalid area.
B5	Module is not ready.
B6	Module rejects access.
B7	Module reports an illegal range for a parameter or value.
B8	Module is reporting an invalid parameter

B9	Module is reporting an invalid type.
BA - BF	Error in data set transfer with regard to access Job aborted. Error numbers, manufacturer-specific in accordance with IEC 61158.
C1	Conflict when receiving data sets on the module.
C2	Module is currently processing the possible maximum of jobs.
C3	Module is reporting that all resources are in use.
C8 - CF	Error in data set transfer with regard to resources. Job aborted. Error numbers, manufacturer-specific in accordance with IEC 61158.

actualMode

Direction:	Output parameter
Data type:	INT
Parameter index:	6

'actualMode' shows the mode that has been started via 'execute'. All responses in 'done', 'busy', 'error' refer to 'actualMode'.

done

Direction:	Output parameter
Data type:	BOOL
Parameter index:	7
System default:	FALSE

Job completed without errors according to the 'mode' parameter.

busy

Direction:	Output parameter
Data type:	BOOL
Parameter index:	8
System default:	FALSE

Job active according to the 'mode' parameter.

error

Direction:	Output parameter
Data type:	BOOL
Parameter index:	9
System default:	FALSE

Job completed with errors according to the 'mode' parameter.

errorId

Direction:	Output parameter
Data type:	DWORD
Parameter index:	10
System default:	0

Error specification

In an error situation, the documentation for the associated module should be consulted for the specific error analysis.

16#00000000	Error-free
16#FFFF8090	Specified logical base address invalid.
16#FFFF8091	Specified logical base address cannot be reached.
16#FFFF809A	The PROFINET interface does not support this service.

16#FFFF809B	The target memory provided is not large enough.
16#FFFF809C	Internal temporary collision of DPV1 jobs.
16#FFFF809D	The interface is not configured as an I-device. a controller interface cannot receive this data set.
16#FFFF809F	Function not executable.
16#FFFF80A1	Negative acknowledgment when receiving; function block cannot be used for the module.
16#FFFF80A2	Protocol error in Layer2.
16#FFFF80A3	PROFINET IO: General CM error.
16#FFFF80A4-A7	Error during data set transfer. Job aborted. Error numbers, manufacturer-specific in accordance with IEC 61158.
16#FFFF80A8	Error during data set transfer. Job aborted. Error because of version conflict.
16#FFFF80A9	Function not supported by the module.
16#FFFF80AA-AF	Error in the data set transfer on the application side. Job aborted. Error numbers, manufacturer-specific in accordance with IEC 61158.
16#FFFF80B0	Data set unknown to module.
16#FFFF80B1	Incorrect length specified in 'dataLength' parameter.
16#FFFF80B2	Module reports access to an invalid slot/subslot.
16#FFFF80B3	Module reports type conflict.
16#FFFF80B4	Module reports access to an invalid area.
16#FFFF80B5	Module is not ready.
16#FFFF80B6	Module rejects access.
16#FFFF80B7	Module reports an illegal range for a parameter or value.
16#FFFF80B8	Module reports an invalid parameter.
16#FFFF80B9	Module reports an invalid type.
16#FFFF80BA-BF	Error in data set transfer with regard to access Job aborted. Error numbers, manufacturer-specific in accordance with IEC 61158.
16#FFFF80C1	Conflict in the receipt of a data set.
16#FFFF80C2	The module is currently processing the possible maximum of jobs.
16#FFFF80C3	Module is reporting that all resources are in use.
16#FFFF80C4	Communication error.
16#FFFF80C5	Distributed I/O not available.
16#FFFF80C6	Data set transfer has been aborted because of priority class abort.
16#FFFF80C7	Interruption due to restart or cold start of the DP master.
16#FFFF80C8-CE	Error in data set transfer with regard to resources. Job aborted. Error numbers, manufacturer-specific in accordance with IEC 61158.
16#FFFF80CF	The function block is currently processing another job.

actualLogAddress

Direction:	Output parameter
Data type:	DINT
Parameter index:	11
System default:	0

Logical address on the I-Device for which a data set is being made available by the higher-level controller

actualIoId

Direction:	Output parameter
Data type:	EnumIoIdType
Parameter index:	12
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output range on the I-Device for which a data set is being made available by the higher-level controller

recordNumber

Direction:	Output parameter
Data type:	UDINT
Parameter index:	13
System default:	0

Number of the data set received.

dataLength

Direction:	Output parameter
Data type:	UDINT
Parameter index:	14
System default:	0

Length of the data received in the 'data' parameter in bytes.

clientId

Direction:	Output parameter
Data type:	UDINT
Parameter index:	15
System default:	0

Unique assignment to higher-level communication partner (e.g. PN controller, supervisor) for this data set job.

data

Direction:	Input/output parameter
Data type:	ARRAY [..] OF BYTE
Parameter index:	16

Received data.

2.2.1.7 _tcpCloseConnection

The "_tcpCloseConnection" function closes a connection established with "_tcpOpenServer" and "_tcpOpenClient", and releases the communications resources used by this connection.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual
- SIMOTION Basic Functions Function Manual
- or in the SIMOTION online help

Return value: DINT

Description of the return values:

16#00000000	Connection has been closed.
16#00008090	Error in transfer parameter
16#FFFF80C3	Invalid "connectionId".
16#FFFF8FFE	Connection does not exist or has already been closed.
16#FFFFFFFD	Internal error.

Parameters:

connectionId

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

Connection ID

2.2.1.8 _tcpCloseServer

The function is called if no more new connections are to be accepted. (However, it does not abort existing connections.)

Additional references

Further information is available at:

- SIMOTION Communication System Manual
- SIMOTION Basic Functions Function Manual
- or in the SIMOTION online help

Return value: DINT

Description of the return values:

16#00000000	Waiting state ended.
16#FFFF8090	Error in transfer parameter
16#FFFF80C3	Invalid port number.

Parameters:

port

Direction:	Input parameter
Data type:	UINT
Parameter index:	1

Port that is waiting for a connection request.

2.2.1.9 _tcpOpenClient

The _tcpOpenClient system function submits a connection request to a server addressed via the parameters 'serverAddress' and 'serverPort'. Once a connection has been successfully established, the function provides a connection ID which is required for subsequent write and read calls (_tcpsend, _tcpReceive).

The function can only be called of from motion tasks or the background task.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual
- SIMOTION Basic Functions Function Manual
- or in the SIMOTION online help

Return value: StructRetTcpOpenClient

StructRetTcpOpenClient
Return parameter for _tcpOpenClient

Structure	Name	Data type	Unit
functionResult	Result	DINT	-
connectionId	New connection Id	DINT	-

Description of the return values:

functionResult

16#00000000	Connection successfully established
16#00007001	Function started (only with nextCommand = IMMEDIATELY)
16#00007002	Function active (only with nextCommand = IMMEDIATELY)
16#00008090	Error in transfer parameter
16#FFFF8FFF	Address combination of own port, destination IP address, and destination port multiply assigned.
16#FFFF8FFB	Connection partner refuses connection.
16#FFFF8FFA	No connection partner available.
16#FFFFFFFC	Function is already active in another task.
16#FFFFFFFD	Internal error.
16#FFFFFFFF	Resource bottleneck

Parameters:

port

Direction:	Input parameter
Data type:	UINT
Parameter index:	1

Own port number.

serverAddress

Direction:	Input parameter
Data type:	ARRAY [0..3] OF USINT
Parameter index:	2

IP address of server.

serverPort

Direction:	Input parameter
Data type:	UINT
Parameter index:	3

Port number of the server.

nextCommand

Direction:	Input parameter
Data type:	EnumTcpNextCommandMode
Parameter index:	4

EnumTcpNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous

Specifies condition for transition to next command. With IMMEDIATELY, the transition takes place immediately. With WHEN_COMMAND_DONE, the transition takes place when the command is finished.

2.2.1.10 _tcpOpenServer

The '_tcpOpenServer' function waits at the port specified by the 'port' parameter for connection requests from one or more clients. Once a connection has been successfully established, the function provides a connection ID which is required for subsequent write and read calls (_tcpsend, _tcpReceive). After this, the function becomes available again so that further connections can be established.

If no further connection is to be established, the resources used by '_tcpOpenServer' can be released again by calling '_tcpCloseServer'.

This does not automatically close any previously established connections; they have to be disconnected by calling _tcpCloseConnection.

The function can only be called of from motion tasks or the background task.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual
- SIMOTION Basic Functions Function Manual
- or in the SIMOTION online help

Return value: StructRetTcpOpenServer

StructRetTcpOpenServer
Return parameter for _tcpOpenServer

Structure	Name	Data type	Unit
functionResult	Result	DINT	-
connectionId	New connection Id	DINT	-
clientAddress	IP address of the connecting client	ARRAY [0..3] OF USINT	-
clientPort	Port number of the connecting client	UINT	-

Description of the return values:

16#00000000	Connection successfully established
16#00007001	Function started.
16#00007002	Function active.
16#FFFF8090	Error in transfer parameter
16#FFFF80C1	There are currently no resources available to execute this system function. The user program can call this function until this return value no longer appears.
16#FFFF8FFF	Server port has been assigned several times.
16#FFFFFFFC	Function is already active in another task.
16#FFFFFFFD	Internal error.
16#FFFFFFFF	Resource bottleneck (for example, connection requests exceed amount permitted in the 'backlog' parameter).

Parameters:

port

Direction:	Input parameter
Data type:	UINT
Parameter index:	1

Port at which the function is waiting for a connection request.

The connection request can be made via any interface.

backlog

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

The 'backlog' parameter describes the maximum number of connection requests that the server can backlog while processing a currently running connection request.

nextCommand

Direction:	Input parameter
Data type:	EnumTcpNextCommandMode
Parameter index:	3

EnumTcpNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous

Specifies condition for transition to next command. With IMMEDIATELY, the transition takes place immediately. With WHEN_COMMAND_DONE, the transition takes place when the command is finished.

2.2.1.11 _tcpReceive

The function '_tcpReceive' receives data via a connection previously established with '_tcpOpenServer' or '_tcpOpenClient'. Valid receive data is available if the 'functionResult' return value contains the value 16#00000000. The 'dataLength' return value contains the number of received data bytes in this case.

Data can be received in arbitrary packet sizes. Generally the packet sizes do not correspond to that of the transmitting side. The user program is responsible for putting together the number of data bytes required for a complete data structure.

Example:

```
Example
_tcpSend(...,dataLength := 123, ...);
_tcpSend(...,dataLength := 234, ...);
```

can occur on the receive side, e.g. in the following divisions: 60, 101, 153, 43 or 145, 212 or ...

The sequence of the received data agrees in any case with the sender sequence.

Negative values in 'functionResult' indicate an error in the data transfer. In this case, the connection must be disconnected by calling '_tcpCloseConnection'.

The function can only be called of from motion tasks or the background task.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual
- SIMOTION Basic Functions Function Manual
- or in the SIMOTION online help

Return value: StructRetTcpReceive

StructRetTcpReceive
Return parameter for _tcpReceive

Structure	Name	Data type	Unit
functionResult	Result	DINT	-
dataLength	Number of received user data bytes	UDINT	-

Description of the return values:

The return value consists of 'functionResult' and 'dataLength'.

'functionResult'

contains error detection.

'dataLength'

contains the number of received data bytes.

16#00000000	New receive data is available.

16#00007001	Function started (only with nextCommand = IMMEDIATELY)
16#00007002	Function active (only with nextCommand = IMMEDIATELY)
16#FFFF80C3	Invalid "connectionId".
16#FFFF8FFC	Connection termination.
16#FFFF8FFE	Invalid connection ID.
16#FFFF8FFD	Connection closed by communication partner.
16#FFFFFFFC	Function is already active in another task.
16#FFFFFFFD	Internal error.

Parameters:

connectionId

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

Connection ID

nextCommand

Direction:	Input parameter
Data type:	EnumTcpNextCommandMode
Parameter index:	2

EnumTcpNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous

Specifies condition for transition to next command. With IMMEDIATELY, the transition takes place immediately. With WHEN_COMMAND_DONE, the transition takes place when the command is finished.

receiveVariable

Direction:	Input parameter
Data type:	ARRAY [0..4095] OF BYTE
Parameter index:	3

User data range.

2.2.1.12 _tcpSend

The function _tcpSend sends data via a connection previously established with _tcpOpenServer or _tcpOpenClient. Negative values in 'functionResult' indicate an error in the data transfer. In this case, the connection must be disconnected by calling _tcpCloseConnection. With an asynchronous call (nextCommand = IMMEDIATELY), the function must be called until it returns the value 0 (or a negative value in case of an error). The function can only be called of from motion tasks or the background task.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual
- SIMOTION Basic Functions Function Manual
- See the SIMOTION MCC Programming and Operating Manual
- or in the SIMOTION online help

Return value: DINT

Description of the return values:

16#00000000	Transferred data sent successfully.
16#00007001	Function started (only with nextCommand = IMMEDIATELY)
16#00007002	Function active (only with nextCommand = IMMEDIATELY)
16#FFFF80C3	Invalid "connectionId".
16#FFFF8FFC	Connection termination.
16#FFFF8FFE	Invalid connection ID.
16#FFFF8FFD	Connection closed by communication partner.
16#FFFFFFFC	Function is already active in another task.
16#FFFFFFFD	Internal error.

Parameters:

connectionId

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

Connection ID

nextCommand

Direction:	Input parameter
Data type:	EnumTcpNextCommandMode
Parameter index:	2

EnumTcpNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous

With WHEN_COMMAND_DONE, the system function remains active until all data has been sent. The execution of the system function is aborted prematurely if an error occurs during transmission. With IMMEDIATELY, the system function comes back straight away. The return value indicates whether all data has been sent (return value = 0), if an error has occurred (return value < 0)or if the command is still active.

dataLength

Direction:	Input parameter
Data type:	UDINT
Parameter index:	3

Number of user data bytes to be sent.

data

Direction:	Input parameter
Data type:	ARRAY [0..4095] OF BYTE
Parameter index:	4

User data range.

2.2.1.13 _udpReceive

This function receives a UDP telegram at a port specified via a transfer parameter.

"_udpReceive" can only be called from motion tasks or the BackgroundTask.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual
- SIMOTION Basic Functions Function Manual
- or in the SIMOTION online help

Return value: StructRetUdpReceive

StructRetUdpReceive
Return parameter for _udpReceive

Structure	Name	Data type	Unit
functionResult	Result	DINT	-
sourceAddress	IP address of the transmitter	ARRAY [0..3] OF USINT	-
sourcePort	Port number of the transmitter	UINT	-
dataLength	Number of received user data bytes	UDINT	-

Description of the return values:

16#00000000	Communication on this port completed without errors
16#00007001	Communication on this port started
16#00007002	Communication on this port active
16#00007003	Communication on this port aborted
16#FFFFFFFF	Resource bottleneck
16#FFFFFFFD	Internal error
16#FFFFFFFC	Receive function already active on this port
16#FFFFFFFB	Reception errors
16#FFFFFFF8	Incorrect parameter

Parameters:

port

Direction:	Input parameter
Data type:	UINT
Parameter index:	1

Port number of receiver

communicationMode (optional)

Direction:	Input parameter
Data type:	EnumUdpCommunicationMode
Parameter index:	2
System default:	DO_NOT_CLOSE_ON_EXIT

EnumUdpCommunicationMode

CLOSE_ON_EXIT (0)	Socket will be closed
DO_NOT_CLOSE_ON_EXIT (1)	Socket remains

Information on longevity of assigned communication resources. With CLOSE_ON_EXIT, communication resources are released after data receipt. With DO_NOT_CLOSE_ON_EXIT, the communication resources persist after data receipt.

nextCommand (optional)

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	3
System default:	IMMEDIATELY

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. With IMMEDIATELY, the transition takes place immediately. With WHEN_COMMAND_DONE, the transition takes place when the command is finished. With ABORT_CURRENT_COMMAND and the same port number as in the previous function call with IMMEDIATELY, the active function is aborted.

receiveVariable

Direction:	Input parameter
Data type:	ARRAY [0..1399] OF BYTE
Parameter index:	4

Received user data bytes.

2.2.1.14 _udpSend

This function sends a UDP telegram to the receiver identified by the IP address and port number.

UDP is a wireless network protocol without flow control, which means that a positive return value from '_udpsend()' does not guarantee that a data packet has reached the receiver.

There are two possibilities if flow control is required:

- Flow control can be implemented as part of the application.
- Flow control use via TCP.

The function can only be called of from motion tasks or the background task.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual
- or in the SIMOTION online help

Return value: DINT

Description of the return values:

16#00000000	Data successfully sent
16#FFFFFFFF	Resource bottleneck Local send/receive buffers overloaded
16#FFFFFFFE	Port not available
16#FFFFFFFD	Internal error
16#FFFFFFFB	Reception errors
16#FFFFFFF9	Error while sending (e.g. no buffer available, faulty IP parameterization, address resolution failed).
16#FFFFFFF8	Incorrect parameter (e.g. user data greater than 1470 bytes).
16#FFFFFFF5	A broadcast address if not permitted for the destinationAddress parameter.

Parameters:

sourcePort

Direction:	Input parameter
Data type:	UINT
Parameter index:	1

Port number of transmitter

destinationAddress

Direction:	Input parameter
Data type:	ARRAY [0..3] OF USINT
Parameter index:	2

IP address of receiver

destinationPort

Direction:	Input parameter
Data type:	UINT
Parameter index:	3

Port number of receiver

communicationMode (optional)

Direction:	Input parameter
Data type:	EnumUdpCommunicationMode
Parameter index:	4
System default:	DO_NOT_CLOSE_ON_EXIT

EnumUdpCommunicationMode

CLOSE_ON_EXIT (0)	Socket will be closed
DO_NOT_CLOSE_ON_EXIT (1)	Socket remains

Information on longevity of assigned communication resources. With CLOSE_ON_EXIT, communication resources are released after the data transfer. With DO_NOT_CLOSE_ON_EXIT, the communication resources persist after the data transfer.

dataLength

Direction:	Input parameter
Data type:	UDINT
Parameter index:	5

Number of user data bytes to be sent.

data

Direction:	Input parameter
Data type:	ARRAY [0..1399] OF BYTE
Parameter index:	6

User data range.

2.2.1.15 _writeRecord

This function causes a data set to be transferred from the ST program to an I/O module. It is used for CP and FM and dynamic parameter assignment of analog and digital modules and to read/write parameters from a drive.

A maximum of 8 jobs can be issued simultaneously.

Note

Only one job should be issued per peripheral module in order to ensure processing.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual
- or in the SIMOTION online help

Return value: DINT

The return value reports the job status.

In an error situation, the documentation for the associated module should be consulted for the specific error analysis.

Description of the return values:

16#00000000	Data set transfer okay, job completed with positive result. Job okay.
16#00007001	Must be repeated in the next program cycle. Initial call, initiation of data transfer okay.
16#00007002	Must be repeated in the next program cycle. Intermediate call, data transfer still active.
16#00007003	Data set transfer aborted. Job aborted.
16#FFFF8090	Error during data set transfer, job aborted. Specified logical base address invalid: No assignment is available in SDBs, or there is no base address.
16#FFFF8091	Error during data set transfer, job aborted. The _writeRecord function cannot reach the specified logical base address.
16#FFFF809D	Error, job aborted. An I slave/I device interface is unable to write data sets to the higher-level master/controller.
16#FFFF809E	Error, job aborted. Attempt to abort a non-active function.
16#FFFF809F	Error, job aborted. Function not executable.
16#FFFF80A1	Error during data set transfer, job aborted. Negative acknowledgment when writing to module: - Module was removed during the write operation. - Module defective

16#FFFF80A2	Error during data set transfer, job aborted. Protocol error in Layer2: - Module not available.
16#FFFF80A3	Error during data set transfer, job aborted. Protocol error involving user interface/user: - Module not available.
16#FFFF80A4-A7	Error during data set transfer, job aborted. Error numbers reserved according to IEC 61158 - Query the cause at the module's manufacturer.
16#FFFF80A8	Error during data set transfer, job aborted. Error because of version conflict.
16#FFFF80A9	Error during data set transfer, job aborted. Function not supported by the module.
16#FFFF80AA-AF	Error during data set transfer, job aborted. Error numbers, manufacturer-specific in accordance with IEC 61158. - Query the cause at the module's manufacturer.
16#FFFF80B0	Error during data set transfer, job aborted. - System function not supported for this module type. - Data set unknown to module. - Writing of data sets with static data (e.g. DS0, DS1) not permitted.
16#FFFF80B1	Error during data set transfer, job aborted. Incorrect length specified in 'dataLength' parameter.
16#FFFF80B2	Error during data set transfer, job aborted. Module reports access to an invalid slot/subslot.
16#FFFF80B3	Error during data set transfer, job aborted. Module reports type conflict.
16#FFFF80B4	Error during data set transfer, job aborted. Module reports access to an invalid area.
16#FFFF80B5	Error during data set transfer, job aborted. Module is not ready. Within the PROFIdrive profile (parameter model), this value has the following meaning: - Access is temporarily denied due to an internal processing status.
16#FFFF80B6	Error during data set transfer, job aborted. Module rejects access.
16#FFFF80B7	Error during data set transfer, job aborted. Module reports an illegal range for a parameter or value. Within the PROFIdrive profile (parameter model), this value has the following meaning: - Error in write-request header.
16#FFFF80B8	Error during data set transfer, job aborted. Module reports an invalid parameter.
16#FFFF80B9	Error during data set transfer, job aborted. Module reports an invalid type.
16#FFFF80BA-BF	Error during data set transfer, job aborted. Error numbers, manufacturer-specific in accordance with IEC 61158. - Query the cause at the module's manufacturer.
16#FFFF80C1	Error during data set transfer, command can be repeated immediately. Data of the preceding write job on the module for the same data set has not yet been processed by the module.

16#FFFF80C2	Error during data set transfer, command can be repeated immediately. The module is currently executing the maximum possible jobs for one CPU.
16#FFFF80C3	Error during data set transfer, command can be repeated immediately. Required resources are presently occupied: - In the _writeRecord function. - In the module
16#FFFF80C4	Error during data set transfer, command can be repeated immediately. Communication errors: - Parity error - SW Ready not set - Error in block length administration - Checksum error on CPU side - Checksum error on module side
16#FFFF80C5	Error during data set transfer, can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF80C6	Error during data set transfer, can be repeated in the next program cycle. Data set transfer has been aborted due to priority class abort (restart or background).
16#FFFF80CF	Error during data set transfer, can be repeated in the next program cycle. Another _writeRecord function call is currently active under this 'commandId'.

Parameters:

ioId

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of logical base address of the I/O module. With INPUT, the logical address of the I/O module is in the input range. With OUTPUT, the logical address of the I/O module is in the output range.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical base address of the I/O module.

Note

You can used both an input and an output address when hybrid modules are used.

recordNumber

Direction:	Input parameter
Data type:	UDINT
Parameter index:	3

Specification of the number of the data set of the peripheral module to be written.

offset

Direction:	Input parameter
Data type:	UDINT
Parameter index:	4

Specifies the offset of the first byte of the data to be read during transfer of the data set of the I/O module. The parameter is not currently supported.

dataLength

Direction:	Input parameter
Data type:	UDINT
Parameter index:	5

Specifies the length of data to be written [bytes].

data

Direction:	Input parameter
Data type:	ARRAY [0..239] OF USINT
Parameter index:	6

Specifies the start address of the data to be written (byte array).

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	7

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. With IMMEDIATELY, the transition takes place immediately (asynchronous call of). With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call of). With ABORT_CURRENT_COMMAND and the same 'commandId' as in the previous function call with IMMEDIATELY, the active function is aborted.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	8

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition 'ABORT_CURRENT_COMMAND'.

2.2.1.16 _writeVariableRecord

The function block causes a data set to be transferred from the user program to an I/O module. Maximum 32768 bytes are written.

This function block or the device system function _writeRecord can be used for writing.

Parameters:

EXECUTE

Direction:	Input parameter
Data type:	BOOL
Parameter index:	1

Starts a new job with a rising edge.

IOID

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	2

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

I/O direction

LOGADDR

Direction:	Input parameter
Data type:	DINT
Parameter index:	3

Logical address of the module.

DSNR

Direction:	Input parameter
Data type:	UDINT
Parameter index:	4

Data set number

LEN (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	5
System default:	0

Length of the read data set in bytes

RECORD

Direction:	Input/output parameter
Data type:	ARRAY [..] OF BYTE
Parameter index:	6

Data set to be written

DONE

Direction:	Output parameter
Data type:	BOOL
Parameter index:	7
System default:	FALSE

Block was ended

BUSY

Direction:	Output parameter
Data type:	BOOL
Parameter index:	8
System default:	FALSE

The function block is being processed.

ERROR

Direction:	Output parameter
Data type:	BOOL
Parameter index:	9
System default:	FALSE

The function block was executed with error.

ERRORID

Direction:	Output parameter
Data type:	DWORD
Parameter index:	10
System default:	0

Description of the error messages:

In an error situation, the documentation for the associated module should be consulted for the specific error analysis.

16#00000000	Error-free
16#FFFF8090	Specified logical base address invalid.
16#FFFF8091	Specified logical base address cannot be reached.
16#FFFF809C	Internal temporary collision of DPV1 jobs.
16#FFFF809D	An I slave / I device interface cannot read any data sets.

16#FFFF809E	Error, job aborted. Attempt to abort a non-active function.
16#FFFF809F	Error, job aborted. Function not executable.
16#FFFF80A1	Negative acknowledgment when writing to the module.
16#FFFF80A2	Error during data set transfer, job aborted. Protocol error in Layer2: - Module not available
16#FFFF80A3	Error during data set transfer, job aborted. Protocol error involving user interface/user: - Module not available
16#FFFF80A8	Error because of version conflict
16#FFFF80A9	Function not supported by the module.
16#FFFF80AA-AF	Error during data set transfer, job aborted. Error numbers, manufacturer-specific in accordance with IEC 61158. - Query the cause at the module's manufacturer.
16#FFFF80B0	Data set unknown to module.
16#FFFF80B1	Incorrect length specified in 'LEN' parameter.
16#FFFF80B2	Module reports access to an invalid slot/subslot.
16#FFFF80B3	Module reports type conflict.
16#FFFF80B4	Module reports access to an invalid area.
16#FFFF80B5	Module is not ready.
16#FFFF80B6	Module rejects access.
16#FFFF80B7	Module reports an illegal range for a parameter or value.
16#FFFF80B8	Module reports an invalid parameter.
16#FFFF80B9	Module reports an invalid type.
16#FFFF80BA-BF	Error during data set transfer, job aborted. Error numbers, manufacturer-specific in accordance with IEC 61158. - Query the cause at the module's manufacturer.
16#FFFF80BB	Error during data set transfer, job aborted. Error numbers, manufacturer-specific in accordance with IEC 61158. - Query the cause at the module's manufacturer.
16#FFFF80C1	Data of the preceding write job on the module for the same data set has not yet been processed by the module.
16#FFFF80C2	The module is currently processing the possible maximum of jobs.
16#FFFF80C3	Required resources are presently occupied.
16#FFFF80C4	Communication error
16#FFFF80C5	Distributed I/O not available.
16#FFFF80C6	Data set transfer has been aborted because of priority class abort.
16#FFFF80CF	The function block is currently processing another job.
16#FFFF80Dx to FFFF80Fx	These error messages are manufacturer-specific. You will find more detailed information in the documentation of the module in question.

2.2.1.17 _Xreceive

This function supplies the job data on the receiver side.

Additional references

Additional information is available at:

- SIMOTION Basic Functions Function Manual
- SIMOTION Communication System Manual
- or in the SIMOTION online help

Return value: StructRetXreceive

StructRetXreceive
Output parameters for _Xreceive

Structure	Name	Data type	Unit
functionResult	Return value of function	DINT	-
dataLength	Return value, number of bytes received	UDINT	-
data	Received data [200]	ARRAY [0..199] OF BYTE	-

The return value of the function consists of 'functionResult', 'dataLength', and 'data'.

Description of the return values:

Asynchronous start:	
16#000000xx	functionResult: Execution was completed without errors (xx length of data).
16#00000000	Data length: Number of received bytes
16#00007000	functionResult: Job is still active
Synchronous call:	
16#000000xx	functionResult: Execution was completed without errors (xx length of data).
16#00000000	Data length: Number of received bytes
16#000080xx	functionResult: Job interrupted with errors. xx reports a unique error number.
Locally detected error	
16#00008092	Permitted data length exceeded (error in Xsend).
16#FFFFFF51	This function is supported in the simulation.
Communication error	
16#000080A1	Connection termination
The return value is a structure	

000000xx	xx=Length of data. For a synchronous call, you must ensure that the last byte in the 'functionResult' contains the length of the received data.

Parameters:

messageId

Direction:	Input parameter
Data type:	UDINT
Parameter index:	1

Specifies the job identifier. This must match the messageId for Xsend.

nextCommand

Direction:	Input parameter
Data type:	EnumXNextCommandEnable
Parameter index:	2

EnumXNextCommandEnable

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous

Specifies the command transition. With IMMEDIATELY, the transition takes place immediately after the command is issued. With WHEN_COMMAND_DONE, the transition takes place when the command has been executed or aborted.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	3

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status.

2.2.1.18 _Xsend

This function initiates an Xsend job.

Additional references

Additional information is available at:

- SIMOTION Basic Functions Function Manual
- SIMOTION Communication System Manual
- or in the SIMOTION online help

Return value: DINT

Return value of function:

The return value depends on call parameter 'nextCommand' (asynchronous or synchronous) of the function.

For an asynchronous call (nextCommand = IMMEDIATELY), the function returns immediately after initiation.

For a synchronous call (nextCommand = WHEN_COMMAND_DONE), the function does not return until the job has been completely executed.

Description of the return values:

16#00007001	Job initiated, still active (only possible for asynchronous job).
16#00000000	Execution completed without errors. Locally detected error. (User can review and modify program)
16#00008090	Remote address invalid. I/O address is not specified here.
16#00008092	Length of data to be sent is not permitted. Communication error (user cannot change anything).
16#000080A0	Protocol error
16#000080A1	Connection termination (check wiring). Error at the communication partner.
16#000080B1	'messageId' not recognized in target system (check Xreceive).
16#000080B5	Remote partner in the STOP state. Temporary error (resource bottleneck: quantity framework should be checked)
16#000080C0	Connection is already occupied by other jobs.
16#000080C1	Temporary local resource bottleneck in client (sender)
16#000080C2	Temporary resource bottleneck in the target system in server (receiver)
16#000080C3	Error when connection is established
16#FFFFFF51	This function is supported in the simulation.

Parameters:

communicationMode

Direction:	Input parameter
Data type:	EnumXCommunicationMode
Parameter index:	1

EnumXCommunicationMode

ABORT_CONNECTION (0)	Disconnecting
HOLD_CONNECTION (1)	Connection remains active

Specifies whether the communication connection persists. ABORT_CONNECTION closes the connection after data transfer. HOLD_CONNECTION keeps the connection open after data transfer.

address

Direction:	Input parameter
Data type:	StructXsendDestAddr
Parameter index:	2

StructXsendDestAddr
Address buffer for _XSend

Structure	Name	Data type	Unit
deviceId	Device ID	USINT	-
remoteSubnetIdLength	Length of remote subnet ID	USINT	-
remoteStaddrLength	Length of remote station address	USINT	-
nextStaddrLength	Length of router address	USINT	-
remoteSubnetId	Remote subnet ID	ARRAY [0..5] OF USINT	-
remoteStaddr	Remote station address	ARRAY [0..5] OF USINT	-
nextStaddr	Router address	ARRAY [0..5] OF USINT	-

Specifies the destination address of the communications partner.

For C230-2, C240, C240 PN: The value 1=x8 or 2=x9 must be entered for deviceId.

For P320-4 The value 1=x101 or 2=x102 must be entered for deviceId.

For D4x5-2: The value 1=x126 or 2=x136 must be entered for deviceId.

For D410-2 DP/PN: The value 1=x24, 1=x21, or 2=x21 must be entered for deviceId.

The value 0 is entered for remoteSubnetIdLength and nextStaddrLength.

The value 1 is entered for remoteStaddrLength if the target system is MPI or DP.

The value 6 is entered if the target system is Ethernet.

Example (for MPI):

Address of deviceId	= 1/2
remoteSubnetIdLength	= 0
remoteStaddrLength	= 1

messageId

Direction:	Input parameter
Data type:	UDINT
Parameter index:	9

Specifies the job identifier. This job identifier must match the messageId for Xreceive.

nextCommand

Direction:	Input parameter
Data type:	EnumXNextCommandEnable
Parameter index:	10

EnumXNextCommandEnable

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous

Specifies the command transition. With IMMEDIATELY, the transition takes place immediately after the command is issued. With WHEN_COMMAND_DONE, the transition takes place when the command has been executed or aborted.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	11

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status.

data

Direction:	Input parameter
Data type:	ARRAY [0..199] OF BYTE
Parameter index:	13

ARRAY of BYTE[0 to 199]

Specifies the communication data.

dataLength

Direction:	Input parameter
Data type:	UDINT
Parameter index:	14

Specifies the data length of the Xsend job in bytes.

2.2.2 Interface handling - Configuration

2.2.2.1 _activateDpSlave

This system function establishes cyclical communication for the entire DP station (slave) or the PROFINET station (device), even if the station has multiple logical addresses (diagnostic addresses).

This system function cannot be called when the _deactivateDpSlave system function is active with the same 'logicalAddressOfDpStation'.

This system function applies to both PROFIBUS and PROFINET.

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- SIMOTION Communication System Manual
- or in the online help

Return value: DINT

The return value reports the job status.

Description of the return values:

16#00000000	Abort job. No jobs are now being executed for the logical address.
16#00000001	PROFIBUS slave/IO device activated - job completed with positive result. Job okay.
16#00000005	PROFIBUS slave/IO device is activated. The system function has not waited for a 'station recovery' alarm. Job completed with positive result. Job okay. A PROFIBUS slave/IO device can also be activated if it is switched off or is not located on the bus. If the PROFIBUS slave/IO device is switched back on or joins the bus, it continues to be treated as activated.
16#00007000	Functionality not active. Initial call required with REQUEST_TRUE
16#00007001	Functionality set up, only for asynchronous execution.
16#00007002	Functionality active, only for asynchronous execution.
16#00007003	Abort job. The job has already been accepted, but its execution could not continue as feedback is expected from the system.
16#FFFF8090	Specified logical base address invalid.
16#FFFF8091	No configuration present for the specified DP station address or device number.
16#FFFF8092	"Activate" requested while "Activate" or "Deactivate" is being executed.
16#FFFF8093	Not a valid DP station
16#FFFF8098	Functionality terminated after timeout.
16#FFFF8099	Internal error, function terminated.
16#FFFF80C1	"Activate" requested while "Deactivate" is being executed.
16#FFFF80C3	Temporary bottleneck: No additional function resource currently available.

Parameters:

logicalAddressOfDpStation

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

The logical diagnostic address of the DP station should be specified for PROFIBUS. The logical diagnostic address of the IO device/I device should be specified for PROFINET.

reqActDeactGetStateMode

Direction:	Input parameter
Data type:	EnumReqActDeactGetStateMode
Parameter index:	2

EnumReqActDeactGetStateMode

REQUEST_TRUE (0)	Start functionality
REQUEST_FALSE (1)	Query status with REQUEST_TRUE
REQUEST_ABORT (2)	Abort of the function execution

REQUEST_TRUE starts the functionality. REQUEST_FALSE scans the status of the functionality started with REQUEST_TRUE. REQUEST_ABORT aborts the function execution.

dpAlarmMode

Direction:	Input parameter
Data type:	EnumDeviceDpAlarmMode
Parameter index:	3

EnumDeviceDpAlarmMode

SET_DP_ALARM (0)	Set up user interface
DO_NOT_SET_DP_ALARM (1)	Suppress user interface call

Specifies the alarm mode. SET_DP_ALARM generates the standard DP alarms (e.g. station failure) that are triggered by the function call. DO_NOT_SET_DP_ALARM suppresses the standard DP alarms that are triggered by the function call.

Note

The alarm is only suppressed if it occurs within the time set in parameter 'timeToWaitForStationAlarm'.

Alarms that occur later are not suppressed.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	4

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and the function cannot be completed.

With WHEN_COMMAND_DONE, the transition takes place when the command is finished. With ABORT_CURRENT_COMMAND and the same 'commandId' as in the previous function call with IMMEDIATELY, the active function is aborted.

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	5
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of the logical diagnostic address of the distributed station or IO device or I device. For INPUT, the logical diagnostic address of the distributed station or IO device or I device is in the input range. For OUTPUT, the logical diagnostic address of the distributed station or IO device or I device is in the output range. Diagnostic addresses are always of type INPUT.

timeToWaitForStationAlarm (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	6
System default:	60

The system function waits for the specified time (in seconds) for the station recovery alarm to arrive. After this time, the internal processing is completed. The system function signals "16#00000005"; i.e. the station is "activated".

Behavior when the 'timeToWaitForStationAlarm' input parameter (wait time) is set to the following values:

- T#0s: The system function does not wait for a station recovery alarm.
- T#60s (default): Behavior as in version < V4.4; the system waits for the station recovery alarm for up to 60 seconds.
- > T#0s: The system function waits for the station recovery alarm for up to the specified wait time.
- The version V4.3 value is applied as the system default.

2.2.2.2 _activateDpSlaveAddress

This system function activates all DP slave addresses set beforehand.

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Return value: DINT

Description of the return values:

16#FFFF8090	Invalid address Outside of valid range / invalid DP segment adapter
16#FFFF8099	Internal error, function terminated.
16#FFFF80C1	Resources for executing the functionality are not available.
16#FFFFFF51	This function is supported in the simulation.

Parameters:

logicalAddressCommunicationAdapter

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

Specifies the logical diagnostic address of the interface.

2.2.2.3 _activateNameOfStation

The function activates all names previously set for the interfaces with the system function _setNameOfStation. Internally, SIMOTION changes to STOP and then triggers a restart.

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Return value: DINT

The return value returns the job status and name.

Description of the return values:

16#FFFF8090	The specified logical diagnostic address is invalid.
16#FFFF8091	No valid logical diagnostic address for a PROFINET interface.
16#FFFF8092	PROFINET interface is configured, but not available. Can only occur if the PROFINET interface is inserted in SIMOTION via a module.
16#FFFF8095	The SIMOTION restart necessary for activation of NameOfStation did not take place.
16#FFFF8096	A device name was not set with _setNameOfStation before this system function was called.
16#FFFF8097	The job was not changed in the STOP operating mode.
16#FFFF8099	Internal error. Function terminated.
16#FFFF80C1	There are currently no resources available to execute this system function. The user program can continue to call this system function until this return value no longer appears.
16#FFFF80CF	A system function call for the relevant PROFINET interface is already in progress. The user program can call this function until this return value no longer appears.

Parameters:

logicalAddressPnInterface

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

Specifies logical base address of PNIO interface (diagnostic address).

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	2
System default:	INPUT

EnumIoldType

INPUT (198)	Input address
OUTPUT (199)	Output address

The iold parameter indicates whether the logical base address of the Profinet interface is an input address or an output address.

2.2.2.4 _assignNameOfStationToDevice

The function block assigns the PROFINET device name 'nameOfStation' to an IO device with the MAC address 'MAC'.

Before setting the name, the function implicitly resets the communication parameters of the IO device in order to set the IO device into a defined initial state for the communication if it was previously in operation in another plant, for example. Whereby, a permanent IP address is deleted. NameOfStation is always set permanently. A temporary setting is not supported in contrast to the system function _setPnNameOfStation.

Additional references

Further information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Parameters:

EXECUTE

Direction:	Input parameter
Data type:	BOOL
Parameter index:	1

Starts a new job with a rising edge.

Note

If the input 'EXECUTE' is set back to 'FALSE' after starting a job, the output signals ERROR, ERRORID, DONE and LEN are pending for exactly one cycle.

ioId

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	2

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Specifies the I/O direction.

logicalAddressPnInterface

Direction:	Input parameter
Data type:	DINT
Parameter index:	3

PROFINET interface of the SIMOTION module. The interface must be configured for the IO controller operating mode.

MAC

Direction:	Input parameter
Data type:	STRING
Parameter index:	4

MAC address (as string) of the device whose name is to be set.

NameOfStation

Direction:	Input parameter
Data type:	STRING
Parameter index:	5

Name of the device to be set.

DONE

Direction:	Output parameter
Data type:	BOOL
Parameter index:	6
System default:	FALSE

Message that the block was completed correctly.

BUSY

Direction:	Output parameter
Data type:	BOOL
Parameter index:	7
System default:	FALSE

Message that the block is being processed.

ERROR

Direction:	Output parameter
Data type:	BOOL
Parameter index:	8
System default:	FALSE

Message that the block was executed with errors.

ERRORID

Direction:	Output parameter
Data type:	DWORD
Parameter index:	9
System default:	0

Description of the error messages:

In an error situation, the documentation for the associated module should be consulted for the specific error analysis.

16#00000000	Error-free - job performed successfully
16#FFFF8009	Internal error
16#FFFF800A	The specified logical diagnostic address is invalid.
16#FFFF800B	PROFINET interface is configured, but not available.
16#FFFF800C	PROFINET interface is not configured as IO controller.
16#FFFF800D	The STRING specified in the 'MAC' parameter does not correspond to the format specification for the MAC address.
16#FFFF800E	The STRING specified in the 'nameOfStation'' parameter is too long.
16#FFFF800F	The contents of the STRING specified in the 'NameOfStation' parameter do not comply with the conventions that apply for a 'NameOfStation'.
16#FFFF8010	A system function call for the relevant PROFINET interface is already in progress. The user program can call this function until this return value no longer appears.
16#FFFF8011	Job was rejected by the IO device.
16#FFFF8012	There are currently no resources available to execute this system function. The user program can call this function until this return value no longer appears.

2.2.2.5 _deactivateDpSlave

This system function aborts cyclical communication for the entire DP station (slave) or the PROFINET station (device), even if the station has multiple logical addresses (diagnostic addresses).

This system function cannot be called when the _activateDpSlave system function is active with the same 'logicalAddressOfDpStation'. This system function applies to both PROFIBUS and PROFINET.

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- SIMOTION Communication System Manual
- or in the online help

Return value: DINT

The return value reports the job status.

Description of the return values:

16#00000000	Abort job. No jobs are now being executed for the logical address.
16#00000002	PROFIBUS slave/IO device deactivated. Job completed with positive result. Job okay.
16#00000006	PROFIBUS slave/IO device is deactivated. The system function has not waited for a 'station failure' alarm. Job completed with positive result. Job okay. A PROFIBUS slave/IO device can also be deactivated if it is switched off or is not located on the bus. If the PROFIBUS slave/IO device is switched back on or joins the bus, it continues to be treated as deactivated.
16#00007000	Functionality not active.
16#00007001	Functionality initiated, only for asynchronous execution.
16#00007002	Functionality active, only for asynchronous execution.
16#00007003	Abort job. The job has already been accepted, but its execution could not continue as feedback is expected from the system.
16#FFFF8090	Specified logical base address invalid.
16#FFFF8091	No configuration present for the specified DP station address or device number.
16#FFFF8092	"Deactivate" requested while "Activate" or "Deactivate" is being executed.
16#FFFF8093	Not a valid DP station
16#FFFF8098	Functionality of system function has been terminated with timeout.
16#FFFF8099	Internal error, function terminated.
16#FFFF80C1	"Activate" requested while "Deactivate" is being executed.
16#FFFF80C3	Temporary bottleneck: No additional function resource currently available.

Parameters:

logicalAddressOfDpStation

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

The logical diagnostic address of the DP station should be specified for PROFIBUS. The logical diagnostic address of the IO device/I device should be specified for PROFINET.

reqActDeactGetStateMode

Direction:	Input parameter
Data type:	EnumReqActDeactGetStateMode
Parameter index:	2

EnumReqActDeactGetStateMode

REQUEST_TRUE (0)	Start functionality
REQUEST_FALSE (1)	Query status with REQUEST_TRUE
REQUEST_ABORT (2)	Abort of the function execution

REQUEST_TRUE starts the functionality. REQUEST_FALSE scans the status of the functionality started with REQUEST_TRUE. REQUEST_ABORT aborts the function execution.

dpAlarmMode

Direction:	Input parameter
Data type:	EnumDeviceDpAlarmMode
Parameter index:	3

EnumDeviceDpAlarmMode

SET_DP_ALARM (0)	Set up user interface
DO_NOT_SET_DP_ALARM (1)	Suppress user interface call

Specifies the alarm mode. SET_DP_ALARM generates the standard DP alarms (e.g. station failure) that are triggered by the function call. DO_NOT_SET_DP_ALARM suppresses the standard DP alarms that are triggered by the function call.

Note

The alarm is only suppressed if it occurs within the time set in parameter 'timeToWaitForStationAlarm'.

Alarms that occur later are not suppressed.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	4

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and the function cannot be completed.

With WHEN_COMMAND_DONE, the transition takes place when the command is finished. With ABORT_CURRENT_COMMAND and the same 'commandId' as in the previous function call with IMMEDIATELY, the active function is aborted.

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	5
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of the logical diagnostic address of the distributed station, IO device, or I device. For INPUT, the logical diagnostic address of the distributed station, IO device, or I device is in the input range. For OUTPUT, the logical diagnostic address of the distributed station, IO device, or I device is in the output range. Diagnostic addresses can be of type INPUT or OUTPUT.

timeToWaitForStationAlarm (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	6
System default:	0

The system function waits for this time (in seconds) for the station failure alarm to arrive. After this time, the internal processing is completed. The system function reads the activation state and signals "16#00000006"; i.e. the station is "deactivated".
Behavior when the timeToWaitForStationAlarm input parameter (wait time) is set to the following values:
0: The system function does not wait for a station restoration alarm.
60: (default): Behavior as in version < V4.4; the system waits for the station failure alarm for up to 60 seconds.
> 0: The system function waits for the station failure alarm for up to the specified wait time in seconds.
The Version V4.3 value applies by default.

2.2.2.6 _enableDpInterfaceSynchronizationMode

This function activates automatic synchronization of the DP master interface with the DP slave interface and/or the alarm messages required for synchronization of the DP interfaces. This function can only be executed synchronously.

Additional references

For additional information on the synchronization of the interfaces, refer to the function manuals:

- SIMOTION Motion Control, TO followingObject, TO cam
- SIMOTION Basic Functions for Modular Machines
- or in the online help

Return value: DINT

Description of the return values:

16#00007000	DP alarm messages activated: The return value signals that the function call has been successfully executed.
16#FFFF8000	DP alarm messages cannot be activated: The return value signals that the function call is not permitted in this system state. The activation of alarm messages can be changed only in unsynchronized operation.
16#FFFF8090	Activation of DP alarm messages is not supported: Return value signals that the function is not supported by the platform.

Parameters:

dpInterfaceSyncMode

Direction:	Input parameter
Data type:	EnumDpInterfaceSyncMode
Parameter index:	1

EnumDpInterfaceSyncMode

NO_ALARMMESSAGES (0)	Alarm messages not enabled
MASTER_SLAVE_ALARMMESSAGES_1 (1)	Alarm messages for DP-master-slave synchronization activated
SLAVE_ALARMMESSAGES_1 (2)	Alarm messages for DP-slave synchronization activated
AUTOMATIC_INTERFACE_SYNCHRONIZATION (3)	Automatic DP-master-slave synchronization activated

With NO_ALARMMESSAGES, no synchronization-specific alarm messages are enabled (system default).

MASTER_SLAVE_ALARMMESSAGES_1 enables the following events to initiate PeripheralFaultTask: _SC_DP_CLOCK_DETECTED, _SC_DP_SYNCHRONIZATION_LOST

SLAVE_ALARMMESSAGES_1 enables the following events to initiate PeripherialFaultTask: _SC_DP_SLAVE_SYNCHRONIZED, _SC_DP_SLAVE_NOT_SYNCHRONIZED

AUTOMATIC_INTERFACE_SYNCHRONIZATION enables the following event to initiate the PeripherialFaultTask: _SC_DP_SYNCHRONIZATION_LOST

If one DP interface is configured as an equidistant DP slave and the other DP interface is configured as an equidistant DP master, a DP cycle clock is detected at the DP slave interface and is automatically synchronized.

Note

Nothing is provided for switching between the modes. If the automatic DP master slave synchronization has been set via AUTOMATIC_INTERFACE_SYNCHRONIZATION in the StartupTask, this mode will remain activated until power off.

2.2.2.7 _setDpSlaveAddress

This system function sets a new DP address (node number) of a DP interface on a SIMOTION device.

Additional references

Further information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Return value: DINT

Description of the return values:

16#00000000	The system function is successfully completed
16#FFFF8090	Invalid address Outside of valid range / invalid DP segment adapter
16#FFFF8091	Invalid DP slave address.
16#FFFF8092	DP adapter is not configured as a slave.
16#FFFF8093	DP adapter is not configured passively.
16#FFFF8094	Requested slave address not stored.
16#FFFF8099	Internal error, function terminated.
16#FFFF80C1	Resources for implementing the functionality are occupied.
16#FFFFFF51	This function is supported in the simulation.

Parameters:

logicalAddressCommunicationAdapter

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

Specifies the logical diagnostic address of the interface.

dpSlaveAddress

Direction:	Input parameter
Data type:	SINT
Parameter index:	2

Communication address.

2.2.2.8 _setIPConfig

Note

Applies from SIMOTION version V4.4

Use the '_setPnIpConfig' system function instead of '_setIpConfig' when you create a new SIMOTION project.

This function sets the IP configuration of a selected Ethernet interface. The parameters are not permanently saved. The configured IP address data are not backed up with RamtoRom.

Additional references

Further information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Return value: UDINT

Description of the return values:

16#00000000	IP configuration was set successfully
16#FFFF80C1	Resource bottleneck
16#FFFF8099	Internal error.
16#FFFF8090	Incorrect return value output in the transfer parameter 'ethernetInterface', subnet mask or gateway.

Parameters:

ethernetInterface

Direction:	Input parameter
Data type:	EnumInterfaceID
Parameter index:	1

EnumInterfaceID

IE_01 (0)	Ethernet interface 1: All platforms
IE_02 (1)	Ethernet interface 2: All platforms

Selected Ethernet interface

ipAddress

Direction:	Input parameter
Data type:	ARRAY [0..5] OF USINT
Parameter index:	2

Ethernet interface IP address to be transferred

subnetMask

Direction:	Input parameter
Data type:	ARRAY [0..5] OF USINT
Parameter index:	3

Ethernet interface subnet screen form to be transferred

gatewayAddress

Direction:	Input parameter
Data type:	ARRAY [0..5] OF USINT
Parameter index:	4

Ethernet interface gateway address to be transferred

2.2.2.9 _setNameOfStation

Note

Applies from SIMOTION version V4.4

Use the '_setPnNameOfStation' system function instead of '_setNameOfStation' when you create a new SIMOTION project.

This function saves the transferred NameOfStation. The stored 'NameOfStation' only then takes effect using _activateNameOfStation.

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Return value: DINT

The return value returns the job status and name.

Description of the return values:

16#0000000	System function completed successfully. 'NameOfStation' is successfully set during the next activation using _activateNameOfStation.
16#FFFF8090	The specified logical diagnostic address is invalid.
16#FFFF8091	No valid logical diagnostic address for a PROFINET interface or a port of a PROFINET interface.
16#FFFF8092	PROFINET interface is configured, but not available. Can only occur if the PROFINET interface is inserted in SIMOTION via a module.
16#FFFF8093	The STRING specified in the 'nameOfStation" parameter is too long. Only 239 characters are permitted. According to the standard, 240 characters are permitted, but this is not currently supported by SIMOTION.
16#FFFF8094	An abort is no longer possible in the current system function status.
16#FFFF8098	The contents of the STRING specified under the parameter 'nameOfStation' do not respect the usual 'NameOfStation' conventions.
16#FFFF8099	Internal error. Function terminated.
16#FFFF809A	Setting the 'NameOfStation' is not supported by the PROFINET interface.
16#FFFF80A0	In HW Config, the options "Obtain device names in another way" or "Obtain IP address in another way" are deselected for the respective communication interface. Select both of these options for the corresponding communication interface in HW Config. Function terminated.
16#FFFF80C1	There are currently no resources available to execute this system function. The user program can call this function until this return value no longer appears.
16#FFFF80CF	A system function call for the relevant PROFINET interface is already in progress. The user program can call this function until this return value no longer appears.

Parameters:

logicalAddressPnInterface

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

Specifies the logical diagnostic address of the PROFINET interface, or the logical diagnostic addresses of the associated ports on the PROFINET interface, from which the active device name is to be read.

The ports' logical diagnostic addresses clearly identify the PROFINET interface. The requested active 'NameOfStation' is supplied.

NameOfStation

Direction:	Input parameter
Data type:	STRING
Parameter index:	2

Specification of the station name. This name is active in the specified Profinet interface (x120) x127, (x130, x150), x1400. If the name to be set exceeds a length of 63 characters, it must be divided into 63-character sections separated by the "." character.

requestMode

Direction:	Input parameter
Data type:	EnumReqSysFunctMode
Parameter index:	3

EnumReqSysFunctMode

REQUEST_TRUE (0)	Start functionality
REQUEST_FALSE (1)	Status query of functionality started with REQUEST_TRUE
REQUEST_ABORT (2)	Abort of the function execution

REQUEST_TRUE starts the functionality. REQUEST_FALSE scans the status of the functionality started with REQUEST_TRUE. REQUEST_ABORT aborts the function execution.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	4

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

This parameter specifies a 'CommandId' of type 'commandIdType' that is unique from all others in the system. The user can make the system generate a 'syncCommandId' by calling _getSyncCommandId. The same syncCommandId is used for subsequent calls.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	6

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and the function cannot be completed.

With WHEN_COMMAND_DONE, the transition takes place when the command is finished. With ABORT_CURRENT_COMMAND and the same 'commandId' as in the previous function call with IMMEDIATELY, the active function is aborted.

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	7
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

The ioId parameter indicates whether logicalAddressPnInterface of the Profinet interface is an input address or an output address.

2.2.2.10 _setPnIpConfig

Note

Applies from SIMOTION version V4.4

Use the '_setPnIpConfig' system function instead of '_setIpConfig' when you create a new SIMOTION project.

This function sets the IP configuration of a selected PROFINET interface.

Additional references

Further information is available at:

- SIMOTION Communication System Manual
- or in the online help

Return value: UDINT

Return value UDINT:

16#00000000	IP configuration has been set successfully and is active.
16#00007001	First call, the trigger to set the IP configuration OK (only when command is issued asynchronously). Must be repeated in the next program cycle.
16#00007002	Intermediate call, the setting of the IP configuration is still active. Repeat in the next program cycle where necessary until the IP configuration has been set successfully or canceled (only when the command is issued asynchronously).
16#00007003	Setting IP configuration aborted (only when command is issued asynchronously).
16#FFFF8090	No valid logical diagnostic address for a PROFINET interface or a port of a PROFINET interface.
16#FFFF8091	No valid logical diagnostic address for a PROFINET interface or a port of a PROFINET interface.
16#FFFF8092	PROFINET interface is configured, but not available. Can only occur if the PROFINET interface is inserted into the SIMOTION device via a module.
16#FFFF8094	An abort is not possible in the current system function status.
16#FFFF8099	An error occurred while the function was being processed.
16#FFFF80A5	An attempt has been made to set an IP subnet which is already functioning at another PROFINET interface on the device. The subnet in which a PROFINET interface is located is defined in its IP configuration. The rule is that each PROFINET interface on a device must be located in its own IP subnet.
16#FFFF80A7	The IP configuration could not be set because there are illegal values in the 'ipAddress', 'subnetMask', and/or 'gatewayAddress' parameters.
16#FFFF80C1	There are currently no resources available to execute this system function. The user program can call this function until this return value no longer appears.

16#FFFF80C2	An attempt is made to change the configuration of the PROFINET interface while a job for changing the PROFINET interface is running.
16#FFFF80FF	There is no configuration information on a PROFINET interface at the logical address. To remedy this, load a project with a configuration for the PROFINET interface into the device.

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

States the diagnostic address. Selects whether the diagnostic address in the 'logicalAddressPnInterface' parameter is an input or output address.

logicalAddressPnInterface

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

States the diagnostic address of interfaces / ports. You must specify a valid diagnostics address of interfaces/ports of PROFINET interfaces, from which the IP configuration is to be set. This can be used to address both Ethernet interfaces with basic PROFINET data (PN-IE) and PN-IO interfaces of a SIMOTION device.

ipAddress

Direction:	Input parameter
Data type:	ARRAY [0..3] OF USINT
Parameter index:	3

States the IP address of the Ethernet interface to be transferred.

subnetMask

Direction:	Input parameter
Data type:	ARRAY [0..3] OF USINT
Parameter index:	4

States the subnet screen of the Ethernet interface to be transferred.

gatewayAddress

Direction:	Input parameter
Data type:	ARRAY [0..3] OF USINT
Parameter index:	5

States the gateway address of the Ethernet interface to be transferred.

storeIpConfigPermanent (optional)

Direction:	Input parameter
Data type:	EnumYesNo
Parameter index:	6
System default:	NO

EnumYesNo

NO (91)	No
YES (173)	Yes

States the validity of the IP configuration set. If the input parameter is set to NO, the set IP configuration remains active until the next start-up (e.g. download or network on/off). If the input parameter is set to YES, the set IP configuration remains active even after the next start-up (e.g. download or network on/off).

Call options _setPnIpConfig

The following are possible call options of the system function _setPnIpConfig and their results after a successful execution:

_setPnIpConfig and 'obtain IP address in another way' not set.

After an error-free execution, the IP configuration stated on _setPnIpConfig is temporarily active. The IP configuration loaded via a project is then temporarily suspended. After a restart, the IP configuration from the project will be reactivated. The input parameter 'storeIpConfigPermanent' of Systemfunktion _setPnIpConfig has no effect.

_setPnIpconfig with 'storeIpConfigPermanent=YES' and 'Obtain IP address in another way' set.

After an error-free execution, the IP configuration stated on _setPnIpConfig is active. The IP configuration is written into the permanent files. After the next restart, the IP configuration set via _setPnIpConfig remains active.

_setPnIpConfig with 'storeIpConfigPermanent=NO' and 'Obtain IP address in another way' set.

After an error-free execution, the IP configuration stated on _setPnIpConfig is active. After the next restart, an invalid IP configuration (IP address = subnet screen = gateway address = 0.0.0.0) is active.

nextCommand (optional)

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	7
System default:	IMMEDIATELY

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Selection of synchronous or asynchronous call of the system function. Familiar functionality.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	8

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND. The same 'commandId' is used for subsequent calls.

2.2.2.11 _setPnNameOfStation

Note

Applies from SIMOTION version V4.4

Use the '_setPnNameOfStation' system function instead of '_setNameOfStation' when you create a new SIMOTION project.

This function saves the transferred NameOfStation. The system function is completed using 'OK'. The return value 'OK' means 'NameOfStation effective'. Subsequent activation using '_activateNameOfStation' is no longer required and is even acknowledged with an error in the system function return value. This name is active in the specified Profinet interface (x120) x127, (x130, x150), x1400. If the name to be set exceeds a length of 63 characters, it must be divided into 63-character sections separated by the "." character.

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Return value: DINT

The return value returns the job status and name.

Description of the return values:

16#0000000	The device name has been set and activated successfully.
16#00007001	Initial call, the trigger to set the device name is still active. Repetition necessary in the next program cycle (only when command is issued asynchronously).
16#00007002	Intermediate call, the setting of the device name is still active. Repetition necessary in the next program cycle until the setting of the device name has been completed successfully or aborted (only when command is issued asynchronously).
16#00007003	Setting the device name aborted. If an abort is not possible, the system function returns 16#FFFF8094.
16#FFFF8090	The specified logical diagnostic address is invalid.
16#FFFF8091	No valid logical diagnostic address for an interface or a port of a PROFINET interface.
16#FFFF8092	PROFINET interface is configured, but not available. This error only occurs if the PROFINET interface is inserted in SIMOTION via a module.
16#FFFF8093	The STRING specified in the 'nameOfStation'' parameter is too long.
16#FFFF8094	An abort is no longer possible in the current system function status.
16#FFFF8098	The contents of the STRING specified under the parameter 'nameOfStation' do not respect the usual 'NameOfStation' conventions.
16#FFFF8099	An error occurred while the function was being processed.
16#FFFF809A	Setting the device name is not supported by the PROFINET interface.

16#FFFF80A0	In HW Config, the options "Obtain device names in another way" or "Obtain IP address in another way" are deselected for the respective communication interface. Both these options must be selected for the corresponding communication interface in HW Config so that the device name can be set.
16#FFFF80FF	There is no configuration information on a PROFINET interface at the logical address. To remedy this, load a project with a configuration for the PROFINET interface into the device.
16#FFFF80C1	There are currently no resources available to execute this system function. The user program can call this function until this return value no longer appears.
16#FFFF80C2	A system function call for the relevant PROFINET interface is already in progress. The user program can call this function until this return value no longer appears.

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of the logical diagnostic address. Specifying this is optional. With INPUT, the logical diagnostic address is in the input range. With OUTPUT, the logical diagnostic address is in the output range.

logicalAddressPnInterface

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical diagnostic address of the PROFINET interface, or the logical diagnostic address of the associated ports on the PROFINET interface, from which the active device name is to be set.

NameOfStation

Direction:	Input parameter
Data type:	STRING
Parameter index:	3

Specification of the station name. According to the standard, 240 characters are permitted.

storeNameOfStationPermanent (optional)

Direction:	Input parameter
Data type:	EnumYesNo
Parameter index:	4
System default:	NO

EnumYesNo

NO (91)	No
YES (173)	Yes

With 'storeNameOfStationPermanent := NO', the device name is effective until the next change and/or restart; after a restart, the device name is deleted and returns an empty string. With 'storeNameOfStationPermanent := YES', the device name is retained even following a restart.

nextCommand (optional)

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	5
System default:	IMMEDIATELY

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. With IMMEDIATELY, an initial call starts the asynchronous processing of the system function. The system function is to be called until it reports that it has been completed (OK or error indications).

With WHEN_COMMAND_DONE, the transition takes place when the command is finished. With ABORT_CURRENT_COMMAND and the same 'commandId' as in the previous function call with IMMEDIATELY, the active function is aborted.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	6

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

This parameter specifies a 'CommandId' of type 'commandIdType' that is unique from all others in the system. The user can make the system generate a 'syncCommandId' by calling _getSyncCommandId. The same syncCommandId is used for subsequent calls.

2.2.2.12 _synchronizeDpInterfaces

This function synchronizes the DP master interface with the DP slave interface. This requires a corresponding interface configuration.

The function is complete when the interfaces are synchronous or if synchronization is not possible. This function can be called multiple times (using the same 'commandId') to scan the execution status.

This function can only be active once at any given time. Multiple starts (multiple calls using a different 'commandId') cause an error message.

If the operating mode is switched from RUN to STOP while the function is being executed, the function is aborted. After the transition from STOP to RUN, the function can be called again.

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Return value: DINT

Description of the return values:

Asynchronous call: (in addition to return values for synchronous call)	
16#00007000	DP synchronization started: Return value signals that the function has been started.
Synchronous call:	
16#00007010	DP interfaces synchronized: Return value signals that the interfaces are synchronized. The internal state machine is advanced to DP_INTERFACES_SYNCHRONIZED.

Note

With PROFIBUS, the function must be started when RUN is reached; with PROFINET, the call for PROFINET can be made during startup. With PROFINET, the call only functions when the system variable StateOfDpInterfaceSynchronization is at DP_CLOCK_DETECTED when the function is called.

16#00007080	DP synchronization active: Return value signals that the function is still being executed.
16#FFFF805x	DP synchronization not enabled: Return value signals that synchronization was ended unsuccessfully. The internal state machine is advanced to WAIT_FOR_DP_CLOCK. Details x: 0 This error is set if the _enableDpInterfaceSynchronizationMode function was not called with parameter MASTER_SLAVE_ALARMMESSAGES_1. 1 This error is set if the interfaces could not be synchronized. One possible cause is interference in the clock signal. 2 Temporary internal synchronization problem. The user state machine must be advanced.

16#FFFF8080	DP synchronization not enabled: It is not permissible to call the function in the current system state. The internal state machine is not advanced.
16#FFFF8081	DP_Synchronization already active: Return value signals that the function has already been called with another 'commandId'. The new call was aborted or rejected. The internal state machine is not advanced.
16#FFFF8082	Function cannot be called synchronously due to resource bottleneck: The return value signals that the function cannot be called synchronously at the moment for resource reasons (no semaphore available). The internal state machine is not advanced.
16#FFFF8083	Required alarm messages are not enabled: The function has been called, although the alarm messages required for sequential control have not been activated or the C230-2 has not been configured as an isochronous slave (see also _enableDpInterfaceSynchronizationMode function). The internal state machine is not advanced.
16#FFFF8090	DP synchronization not supported: Return value signals that the function is not supported by the platform. The internal state machine is not advanced.

Parameters:

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	1

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and the function cannot be completed.

With WHEN_COMMAND_DONE, the transition takes place when the command is finished. With ABORT_CURRENT_COMMAND and the same 'commandId' as in the previous function call with IMMEDIATELY, the active function is aborted.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	2

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status.

2.2.2.13 _udpAddMulticastGroupMembership

This system function is used to join a multicast group on a selected Ethernet interface. A maximum of three multicast groups can be created on an Ethernet interface. These can occupy the same port or different ports.

The function can only be called of from motion tasks or the background task.

Note

In the case of 'UDP Multicast' communications, the '_multicastPort' must correspond to the 'destinationPort' for function _udpSend.

Return value: DINT

Description of the return values:

16#00000000	Communication on this port completed without errors
16#00007001	Communication on this port started
16#00007002	Communication on this port active
16#00007003	Communication on this port aborted
16#FFFFFFFF	Resource bottleneck
16#FFFFFFFD	Internal error.
16#FFFFFFFC	Receive function already active on this port
16#FFFFFFFB	Reception errors
16#FFFFFFF9	Error occurred while sending
16#FFFFFFF8	Incorrect parameter
16#FFFFFFF7	Result not available
16#FFFFFFF6	Multicast IP address not valid
16#FFFFFFF5	IP address not valid
16#FFFFFFF4	Multicast IP address already entered
16#FFFFFFF3	Multicast IP address could not be entered
16#FFFFFFF2	No available resources for entering the multicast IP address in the Ethernet controller

Parameters:

multicastIPAddress

Direction:	Input parameter
Data type:	ARRAY [0..3] OF USINT
Parameter index:	1

IP address for multicasts

interfaceIPAddress

Direction:	Input parameter
Data type:	ARRAY [0..3] OF USINT
Parameter index:	2

IP address of multicast interface

multicastPort

Direction:	Input parameter
Data type:	UINT
Parameter index:	3

Port number of multicast socket

multicastTTL (optional)

Direction:	Input parameter
Data type:	UINT
Parameter index:	4
System default:	1

Time To Live for multicast telegrams

multicastLOOP (optional)

Direction:	Input parameter
Data type:	UINT
Parameter index:	5
System default:	1

Copies of the multicast telegrams to itself if value is greater than 0. The parameter applies to all multicast groups that are set on one interface.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	6

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies the command transition. At present, only synchronous execution is supported.

2.2.2.14 _udpDropMulticastGroupMembership

This system function is used to exit a multicast group on a selected Ethernet interface.

The function can only be called of from motion tasks or the background task.

Return value: DINT

Description of the return values:

16#00000000	Communication on this port completed without errors
16#00007001	Communication on this port started
16#00007002	Communication on this port active
16#00007003	Communication on this port aborted
16#FFFFFFFF	Resource bottleneck
16#FFFFFFFD	Internal error
16#FFFFFFFC	Receive function already active on this port
16#FFFFFFFB	Reception errors
16#FFFFFFF9	Error while sending
16#FFFFFFF8	Incorrect parameter
16#FFFFFFF7	Result not available
16#FFFFFFF6	Multicast IP address not valid
16#FFFFFFF5	IP address not valid
16#FFFFFFF4	Multicast IP address already entered
16#FFFFFFF3	Multicast IP address could not be entered
16#FFFFFFF2	No available resources for entering the multicast IP address in the Ethernet controller
16#FFFFFFF1	Multicast IP address does not exist on the controller.
16#FFFFFFF0	Error when enabling the multicast IP address in the Ethernet controller.

Parameters:

multicastIPAddress

Direction:	Input parameter
Data type:	ARRAY [0..3] OF USINT
Parameter index:	1

IP address for multicasts

interfaceIPAddress

Direction:	Input parameter
Data type:	ARRAY [0..3] OF USINT
Parameter index:	2

IP address of multicast interface

multicastPort

Direction:	Input parameter
Data type:	UINT
Parameter index:	3

Port number of multicast socket

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	4

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies the command transition. At present only synchronous execution is supported.

2.2.3 Interface handling - Status

2.2.3.1 _getActiveDpSlaveAddress

This system function reads the active communication address of the specified interface.

Additional references

More detailed information on this system function can be found in

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the SIMOTION online help

Return value: StructRetDpSlaveAddress

StructRetDpSlaveAddress
Display structure of the system function _getActiveDpSlaveAddress

Structure	Name	Data type	Unit
functionResult	Result of the function execution	DINT	-
dpSlaveAddress	DP slave address	SINT	-

Description of the return values:

Return value "functionResult" has the following meaning:

16#00000000	Function terminated successfully.
16#FFFF8090	Invalid address, outside of valid range/invalid DP segment adapter.
16#FFFF8091	Invalid DP slave address.
16#FFFF80C1	Resources for implementing the functionality are occupied.

Parameters:

logicalAddressCommunicationAdapter

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

Specifies the logical diagnostic address of the interface.

2.2.3.2 _getActiveNameOfStation

Note

Applies from SIMOTION version V4.4

Use the '_getPnNameOfStation' system function instead of '_getActiveNameOfStation' when you create a new SIMOTION project. This system function supplies the effective 'NameOfStation' name of the PROFINET interface. This originates from the configuration or was set with the DCP service using 'TcpIp' or by means of the _setNameOfStation system function.

The system function reads the currently active NameOfStation. This originates from the configuration or was set with the DCP service using 'TcpIp' or by means of the _setNameOfStation system function.

A maximum of 239 characters is returned. According to the standard, 240 characters are permitted, but this is not currently supported by SIMOTION.

If NameOfStation is not available, a string with a length of 0 is returned. In pnPortNumber, 16#FFFFFFFF is always returned. In pnSlotNumber, 16#FFFFFFFF is always returned.

Additional references

More detailed information on this system function can be found in

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Return value: StructRetDeviceNameOfStation

StructRetDeviceNameOfStation
Return values for _getPnInterfacePortNeighbour and _getActiveNameOfStation

Structure	Name	Data type	Unit
functionResult	Result of the execution of the system function	DINT	-
nameOfStation	nameOfStation of own device (_getActivNameOfStation) or neighboring device (_getPnInterfacePortNeighbour)	STRING	-
pnPortNumber	pnPortNumber of the neighboring device, with which this is connected to the local device.	UDINT	-
pnSlotNumber	pnSlotNumber of the neighboring device, with which this is connected to the local device.	UDINT	-

Description of the return values:

functionResult

The return value returns the job status and name.

16#00000000	System function completed successfully. NameOfStation is contained in the return value.
16#00007000	The system function is in inactive status. The functionality associated with the system function is not active. An initial call with REQUEST_TRUE is required.
16#00007001	The system function is started with REQUEST_TRUE. A new call in the user program until completion of the system function is required with REQUEST_FALSE. This return value is reported only for asynchronous execution.
16#00007002	The system function is still running. A new call in the user program until completion of the system function is required with REQUEST_FALSE. This return value is reported only for asynchronous execution.
16#00007003	System function aborted successfully. If an abort is not possible, the system function returns 16#FFFF8094.
16#FFFF8090	The specified logical diagnostic address is invalid.
16#FFFF8091	No valid logical diagnostic address for a PROFINET interface or a port of a PROFINET interface.
16#FFFF8092	PROFINET interface is configured, but not available. Can only occur if the PROFINET interface is inserted in SIMOTION via a module.
16#FFFF8094	An abort is no longer possible in the current system function status.
16#FFFF8099	Internal error. Function terminated.
16#FFFF809A	The PROFINET interface does not support this service. Function terminated.
16#FFFF80A0	The "Obtain device names in another way" option is deselected in HW Config. Function terminated.
16#FFFF80C1	There are currently no resources available to perform this system function. The user program can continue to call this system function until this return value no longer appears.
16#FFFF80CF	A system function call for the relevant PROFINET interface is already in progress. The user program can continue to call this system function until this return value no longer appears.

nameOfStation

Device name of relevant PROFINET interface. A maximum of 239 characters is returned. According to the standard, 240 characters are permitted, but this is not currently supported by SIMOTION.

If NameOfStation is not available, a string with a length of 0 is returned.

pnPortNumber

16#FFFFFFFF is always returned.

pnSlotNumber

16#FFFFFFFF is always returned.

Parameters:

logicalAddressPnInterface

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

Specifies the logical diagnostic address of the PROFINET interface, or the logical diagnostic addresses of the associated ports on the PROFINET interface, from which the active device name is to be read.

requestMode

Direction:	Input parameter
Data type:	EnumReqSysFunctMode
Parameter index:	2

EnumReqSysFunctMode

REQUEST_TRUE (0)	Start functionality
REQUEST_FALSE (1)	Status query of functionality started with REQUEST_TRUE
REQUEST_ABORT (2)	Abort of the function execution

REQUEST_TRUE starts the functionality. REQUEST_FALSE scans the status of the functionality started with REQUEST_TRUE. REQUEST_ABORT aborts the function execution.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	3

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

This parameter specifies a 'CommandId' of type 'commandIdType' that is unique from all others in the system. The user can make the system generate a 'syncCommandId' by calling _getSyncCommandId. The same 'syncCommandId' is used for subsequent calls.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	5

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and the function cannot be completed.

With WHEN_COMMAND_DONE, the transition takes place when the command is finished. With ABORT_CURRENT_COMMAND and the same 'commandId' as in the previous function call with IMMEDIATELY, the active function is aborted.

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	6
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

The ioId parameter indicates whether logicalAddressPnInterface of the Profinet interface is an input address or an output address.

2.2.3.3 _getDoIndexNumberFromLogAddress

This function determines the DO index to a logical address from the current configuration. This DO index makes it possible to read the corresponding DO ID from parameter 978 (see PROFIdrive profile).

The function always determines a DO index, also for non-PROFIdrive drives, because it cannot always be determined whether a PROFIdrive drive is involved (e.g. configuration with GSD).

Additional references

Further information is available at:

- SIMOTION Communication System Manual
- or in the online help

Return value: StructRetGetDoIndexNumberFromLogAddress

StructRetGetDoIndexNumberFromLogAddress
Return parameter for _getDoIndexNumberFromLogAddress

Structure	Name	Data type	Unit
functionResult	Return value of the job	DINT	-
DoIndexNumber	DO index	UDINT	-

The return value consists of 'functionResult' and 'DoIndexNumber'.

Description of the return values:

functionResult

'functionResult' reports the job status.

16#00000000	Job completed without errors.
16#FFFF8090	Job aborted. Specified logical base address invalid.
16#FFFF8091	Job aborted. The _getDoIndexNumberFromLogAddress function cannot reach the specified logical address.
16#FFFF809F	Internal error, job aborted. Function not executable.

DoIndexNumber

'DoIndexNumber' reports the determined DO Index.

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of the logical address. With INPUT, the logical address is in the input range. With OUTPUT, the logical address is in the output range.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies a configured logical IO address.

2.2.3.4 _getDpStationAddressFromLogDiagnosticAddress

This function returns the station number or device number of a PROFIBUS DP slave or PROFINET device on the basis of its diagnostic address.

Note

For PROFIBUS, 'dpStationAddress' is returned with the actual DP slave address. Provided the DP slave address in the user program has not been changed using _setDpSlaveAddress or _activateDpSlaveAddress, the effective DP slave address is identical to the configured address.

For PROFINET 'dpStationAddress' is returned with the actual device number.

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Return value: StructRetGetDpStationAddressFromLogDiagnosticAddress

EnumDpSegmentId

NO_DP_SEGMENT (1)	No DP segment
DP_1 (2)	DP PROFIBUS interface 1 - external
DP_2 (3)	DP PROFIBUS interface 2 - external
DP_3 (4)	DP PROFIBUS interface 3 - internal
IE_1 (5)	Industrial Ethernet interface 1
IE_2 (6)	Industrial Ethernet interface 2
PN_1 (7)	PROFINET IO interface 1
PN_2 (8)	PROFINET IO interface 2

StructRetGetDpStationAddressFromLogDiagnosticAddress
Return parameter for _getDPStationAddressFromLogDiagnosticAddress

Structure	Name	Data type	Unit
functionResult	Total job return value	DINT	-
DpStationAddress	DP station address with SegmentId	StructDpStationAddressType	-

StructDpStationAddressType
Structure for transfer parameter dpStationAddress

Structure	Name	Data type	Unit
dpSegmentId	DP segment	EnumDpSegmentId	-
dpSlaveAddress	Station address in the DP segment	DINT	-

Note

IE_1 (5) and IE_2 (6) are not currently used by SIMOTION.

Description of the return values:

functionResult

16#00000000	OK - System function completed successfully.
16#00007000	Functionality is not active, only for asynchronous execution.
16#00007001	Functionality initiated, only for asynchronous execution.
16#00007002	Functionality active, only for asynchronous execution.
16#FFFF8090	No valid PROFIBUS or PROFINET segment.
16#FFFF8091	No configuration present for the specified DP station address or device number.
16#FFFF8098	The automated status of the system function terminated with timeout.
16#FFFF8099	Internal error, terminate function.
16#FFFF80C3	Temporary resource bottleneck.
16#FFFF809D	Abort successful.
16#FFFF8092	An asynchronous job (IMMEDIATELY) is active. Before a new job is output (e.g. with WHEN_COMMAND_DONE), the current asynchronous job must be ended.
16#FFFF80CF	CommandId not unique.

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of the diagnostic address of the distributed station or IO device or I device. For INPUT, the diagnostic address of the distributed station or IO device or I device is in the input range. For OUTPUT, the diagnostic address of the distributed station or IO device or I device is in the output range. Diagnostic addresses are always of type INPUT.

logDiagnosticAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Diagnostic address of the DP slave or IO device or I device

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	3

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies the command transition. With IMMEDIATELY, the transition takes place immediately after the command is issued. With WHEN_COMMAND_DONE, the transition takes place when the command has been executed. With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	4

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status.

2.2.3.5 _getGeoAddressFromLogAddress

This function determines the corresponding geographical address from a logical address.

Additional references

Further information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Return value: StructRetGetGeoAddressFromLogAddress

StructRetGetGeoAddressFromLogAddress
Return parameter for _getGeoAddressFromLogAddress

Structure	Name	Data type	Unit
functionResult	Return value of the job	DINT	-
area	Area identifier	UDINT	-
rack1	Rack/station number	UDINT	-
rack2	Master system ID	UDINT	-
slot	Slot number	UDINT	-
subSlot	Subslot number	UDINT	-
subAddress	Difference to the base address	UDINT	-

The return value consists of 'functionResult', 'AREA', 'RACK1', 'RACK2', 'SLOT', 'SUBSLOT' and 'SUBADDR'.

Description of the return values:

functionResult

'functionResult' reports the job status.

16#00000000	Job completed without errors.
16#FFFF8090	Job aborted. Specified logical base address invalid.
16#FFFF8091	Internal error, job aborted. The _getGeoAddressFromLogAddress function cannot reach the specified logical address.
16#FFFF819F	Internal error, job aborted. Function not executable.

area

'AREA' reports the area identifier.

The area identifier indicates how the other return values RACK1, RACK2, SLOT, SUBSLOT and SUBADDR are to be interpreted.

AREA = 1 : PBUS

- RACK1 -> Module rack no.
- RACK2 -> Not relevant, always 0
- SLOT -> Slot no.
- SUBSLOT -> Not relevant, always 0
- SUBADDR -> Difference between logical address and base address

AREA = 2 : PROFIBUS DP

- RACK1 -> Station number
- RACK2 -> Master system ID
- SLOT -> Slot no. inside a station
- SUBSLOT -> Not relevant, always 0
- SUBADDR -> Difference between logical address and base address

AREA = 7 : ONBOARD

- RACK1 -> Not relevant, always 0
- RACK2 -> Not relevant, always 0
- SLOT -> Slot no. of the CPU
- SUBSLOT -> Slot no. of a sub-module within a SLOT
- SUBADDR -> Difference between logical address and base address

AREA = 8 : PROFINET IO

- RACK1 -> Device number, number of the PROFINET IO device containing the relevant subslot
- RACK2 -> PROFINET IO system ID, number of the IO system containing the PROFINET IO device with the relevant subslot; value range 100 - 115
- SLOT -> Slot number in the PROFINET IO device containing the relevant subslot
- SUBSLOT -> Subslot number of the subslot containing the logical address of the 'logAddress' input parameter
- SUBADDR -> Difference between the logical address of the 'logAddress' input parameter and the logical base address of the relevant subslot

'RACK1' reports module rack/station number.

'RACK2' reports the master system-Id.

'SLOT' reports the slot number.

'SUBSLOT' reports the subslot number.

'SUBADDR' reports the difference to the basic address.

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of the logical address. With INPUT, the logical address is in the input range. With OUTPUT, the logical address is in the output range.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies a logical IO address or a diagnostic address.

2.2.3.6 _getIPConfig

This function determines the IP configuration of a selected Ethernet interface.

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Return value: StructRetIPConfig

StructRetIPConfig
Return parameter for _getIPConfig

Structure	Name	Data type	Unit
functionResult	Result	UDINT	-
ipAddress	Current IP address of the Ethernet interface	ARRAY [0..5] OF USINT	-
subnetMask	Current subnet screen of the Ethernet interface	ARRAY [0..5] OF USINT	-
gatewayAddress	Address of the default router in the network	ARRAY [0..5] OF USINT	-

Result, IP address, subnet screen form, and address of the default router

Description of the return values:

Return value "functionResult" has the following meaning:

16#00000000 - IP_CONF_RLST_READY The function has been completed successfully. The IP configuration is available without error.

16#FFFF8090 - IP_CONF_WRONG_PARAM The function has been called with incorrect parameters.

16#FFFF8099 - IP_CONF_INTERNAL_ERROR An internal error has occurred while executing the function.

16#FFFF80C1 - IP_CONF_NO_RESSOURCES No resources are available to supply the IP configuration.

16#FFFF80C3 - IP_CONF_NOT_AVAILABLE The function has been called for a module that does not have an Ethernet interface.

Parameters:

ethernetInterface

Direction:	Input parameter
Data type:	EnumInterfaceID
Parameter index:	1

EnumInterfaceID

IE_01 (0)	Ethernet interface 1: All platforms
IE_02 (1)	Ethernet interface 2: All platforms

Ethernet interface for which the configuration is to be read.

2.2.3.7 _getLogDiagnosticAddressFromDpStationAddress

This function returns the diagnostic address of a PROFIBUS DP slave or PROFINET device on the basis of its station number.

Note

For PROFIBUS, 'dpStationAddress' is specified as the actual DP slave address. Provided the DP slave address in the user program has not been changed using '_setDpSlaveAddress' or '_activateDpSlaveAddress', the configured and actual DP slave addresses will be identical.

For PROFINET, 'dpStationAddress' should be used to specify the device number of the relevant IO device.

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the SIMOTION online help

Return value: StructRetGetLogDiagnosticAddressFromStationAddress

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

StructRetGetLogDiagnosticAddressFromStationAddress
Return parameter for _getLogDiagnosticAddressFromDPStationAddress

Structure	Name	Data type	Unit
functionResult	Total job return value	DINT	-
ioIdType	IO type	EnumIoIdType	-
logDiagnosticAddress	Logical diagnostic address of DP slave	DINT	-

Description of the return values:

Return value "functionResult" has the following meaning:

16#00000000	System function successfully completed
16#00007000	Functionality is not active, only for asynchronous execution.
16#00007001	Functionality initiated, only for asynchronous execution.
16#00007002	Functionality active, only for asynchronous execution.
16#FFFF8090	No valid PROFIBUS or PROFINET segment.
16#FFFF8091	No configuration present for the specified DP station address or device number.
16#FFFF8098	Functionality terminated with timeout
16#FFFF8099	Internal error, terminate function.
16#FFFF80C3	Temporary resource bottleneck.
16#FFFF809D	Abort successful.

16#FFFF8092	An asynchronous job (IMMEDIATELY) is active. Before a new job is output (e.g. with WHEN_COMMAND_DONE), the current asynchronous job must be ended.
16#FFFF80CF	CommandId not unique.

Parameters:

dpStationAddress

Direction:	Input parameter
Data type:	StructDpStationAddressType
Parameter index:	1

EnumDpSegmentId

NO_DP_SEGMENT (1)	No DP segment
DP_1 (2)	DP PROFIBUS interface 1 - external
DP_2 (3)	DP PROFIBUS interface 2 - external
DP_3 (4)	DP PROFIBUS interface 3 - internal
IE_1 (5)	Industrial Ethernet interface 1
IE_2 (6)	Industrial Ethernet interface 2
PN_1 (7)	PROFINET IO interface 1
PN_2 (8)	PROFINET IO interface 2

StructDpStationAddressType
Structure for transfer parameter dpStationAddress

Structure	Name	Data type	Unit
dpSegmentId	DP segment	EnumDpSegmentId	-
dpSlaveAddress	Station address in the DP segment	DINT	-

Specifies the station address of the DP slave with Dp SegmentId. IE_1 (5) and IE_2 (6) are not currently used by SIMOTION.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	3

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies the command transition. With IMMEDIATELY, the transition takes place immediately after the command is issued. With WHEN_COMMAND_DONE, the transition takes place when the command has been executed. With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	4

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status.

2.2.3.8 _getNextLogAddress

All configured logical addresses of a segment can be determined using this function.

Return value: StructRetGetNextLogAddress

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

EnumYesNo

NO (91)	No
YES (173)	Yes

StructRetGetNextLogAddress
Return parameter for _getNextLogAddress

Structure	Name	Data type	Unit
functionResult	Return value of the job	DINT	-
ioIdType	IO type	EnumIoIdType	-
diagnosticType	Information on whether address is a diagnostics address	EnumYesNo	-
logicalAddress	Logical address	UDINT	-
length	Length of the logical address	UDINT	-
stationAddress	Station number/rack	UDINT	-
slot	Slot number	UDINT	-
subSlot	Subslot number	UDINT	-
nextIndex	Index to the next logical address	DINT	-

The return value consists of 'functionResult', 'ioIdType', 'diagnosticType', 'logicalAddress', 'length', 'stationAddress', 'slot', 'subslot' and 'nextIndex'.

Description of the return values:

functionResult

reports the job status.

16#00000000	Job completed without errors. No other addresses are available.
16#00000001	Job completed without errors. Further addresses are available. - With a new call, the function can determine the next logical address with the value returned in the return value 'nextIndex'. The return value 'nextIndex' serves here as new input value for 'index'.
16#FFFF8090	Job aborted. The specified index is invalid. No logical address could be assigned.
16#FFFF8091	Job aborted. IO segment was not found (check area and mastersystemId).
16#FFFF8092	Job aborted. The IO segment does not contain any configured addresses.

16#FFFF809F	Internal error, job aborted. Function not executable.

ioIdType

reports the Input/output assignment of the logical address.

Note

For mixed modules, the input and output addresses are reported separately.

diagnosticType

reports whether the address is a diagnostic address.

logicalAddress

reports the logical address of the slot or subslot.

length

reports the length of the logical address.

stationAddress

reports the station number/rack number.

slot

reports the slot number.

subslot

reports the subslot number.

nextIndex

reports the index of the next logical address. The function can be called again with this index, depending on the 'functionResult'.

Parameters:

index

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

The index can be used to address a logical address within an IO segment. The index = 0 always points to the first logical address of an IO segment. The 'nextIndex' passed as return value is used as new 'index' in the subsequent call and addresses the subsequent logical address.

area

Direction:	Input parameter
Data type:	UDINT
Parameter index:	2

Specifies the area identifier of an IO segment. This value can be determined using system function _getSegmentIdentification. AREA = 1 : PBUS AREA = 2: PROFIBUS DP AREA = 7: ONBOARD AREA = 8: PROFINET IO

mastersystemId

Direction:	Input parameter
Data type:	UDINT
Parameter index:	3

Specifies the master system ID of an IO segment. This value can be determined using system function _getSegmentIdentification.

2.2.3.9 _getPnInterfacePortNeighbour

Note

Applies from SIMOTION version V4.4

Use the '_getPnPortNeighbour' system function instead of '_getPnInterfacePortNeighbour' when you create a new SIMOTION project. This system function supplies the name of the PROFINET device connected to the port of the PROFINET interface.

This system function reads information from the PN device connected to the port.

If no identifiable neighbor device, or no device at all, is connected to the relevant port, an empty string will be returned in 'nameOfStation' and 16#FFFFFFFF in 'pnPortNumber'.

If a project-wide unique device name has not been assigned to the connected PN device, the hardware address (MAC address) is returned. The MAC address is output in the notation 11:22:33:44:55:66:77.

If an identifiable neighbor device has been found, a string with a length other than 0 will be returned in 'nameOfStation' and a value 0 < x < 255 will be returned in 'pnPortNumber'.

For this system function, only exactly one function call per port is permitted simultaneously.

Additional references

More detailed information on this system function can be found in

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the SIMOTION online help

Return value: StructRetDeviceNameOfStation

StructRetDeviceNameOfStation
Return values for _getPnInterfacePortNeighbour and _getActiveNameOfStation

Structure	Name	Data type	Unit
functionResult	Result of the execution of the system function	DINT	-
nameOfStation	nameOfStation of own device (_getActivNameOfStation) or neighboring device (_getPnInterfacePortNeighbour)	STRING	-
pnPortNumber	pnPortNumber of the neighboring device, with which this is connected to the local device.	UDINT	-
pnSlotNumber	pnSlotNumber of the neighboring device, with which this is connected to the local device.	UDINT	-

The return value returns the job status and name.

Description of the return values:

functionResult

16#00000000	System function completed successfully. The device name and port number of the relevant neighbor device will be returned if it can be identified.
16#00007000	The system function is in inactive status. The functionality of the system function is not active. An initial call with REQUEST_TRUE is required.
16#00007001	The system function is started with REQUEST_TRUE. A new call in the user program until completion of the system function is required with REQUEST_FALSE. This return value is reported only for asynchronous execution.
16#00007002	The system function is still running. A new call in the user program until completion of the system function is required with REQUEST_FALSE. This return value is reported only for asynchronous execution.

16#00007003	System function aborted successfully. If an abort is not possible, the system function returns the value 16#FFFF8094.
16#FFFF8090	The specified logical diagnostic address is invalid.
16#FFFF8091	No valid logical diagnostic address for a PROFINET interface port.
16#FFFF8092	PROFINET interface is configured, but not available. Can only occur if the PROFINET interface is inserted in SIMOTION via a module.
16#FFFF8093	The STRING returned under 'nameOfStation' is too long. Only 235 characters are permitted. According to the standard, 240 characters are permitted, but this is not currently supported by SIMOTION.
16#FFFF8094	An abort is no longer possible in the current system function status.
16#FFFF8099	Internal error. Function terminated.
16#FFFF809A	The PROFINET interface does not support this service. Function terminated.
16#FFFF809D	Several neighbors are connected to the port addressed by 'logicalAddressOfPnPort'. Function terminated.
16#FFFF80C1	There are currently no resources available to execute this system function. The user program can call this function until this return value no longer appears.
16#FFFF80CF	A system function call for the relevant PROFINET interface port is already in progress. The user program can call this function until this return value no longer appears.

nameOfStation

Device name of neighbor device

pnPortNumber

Port number of neighbor device used connect to the local device. If no identifiable neighbor device, or no device at all, is connected to the relevant port,

- an empty string will be returned in 'nameOfStation'
- and 16#FFFFFFFF in 'pnPortNumber'.

If an identifiable neighbor device has been found,

- a string with a length other than 0 will be returned in 'nameOfStation' and
- a value of 0 < x < 255 will be returned in 'pnPortNumber'.

pnSlotNumber

Slot number of neighbor device containing the neighbor port. Only returned if the neighbor device can be identified and contains several slots with ports. If a slot number is not available, 16#FFFFFFFF is returned. For this system function, only exactly one function call per port is permitted simultaneously.

Parameters:

logicalAddressOfPnInterfacePort

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

Specifies the logical diagnostic address of the port used to determine the device name, port number, and slot number of its neighbor.

requestMode

Direction:	Input parameter
Data type:	EnumReqSysFunctMode
Parameter index:	2

EnumReqSysFunctMode

REQUEST_TRUE (0)	Start functionality
REQUEST_FALSE (1)	Status query of functionality started with REQUEST_TRUE
REQUEST_ABORT (2)	Abort of the function execution

REQUEST_TRUE starts the functionality. REQUEST_FALSE scans the status of the functionality started with REQUEST_TRUE. REQUEST_ABORT aborts the function execution.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	3

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND. The same CommandId is used for subsequent calls.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	5

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and the function cannot be completed.

With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	6
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

The ioId parameter indicates whether logicalAddressOfPnInterfacePort of the Profinet interface is an input address or an output address.

2.2.3.10 _getPnIpConfig

This function determines the IP configuration of a selected Ethernet interface.

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Return value: StructRetPnIpConfig

StructRetPnIpConfig
Return parameter for _getPnIpConfig

Structure	Name	Data type	Unit
functionResult	Result	UDINT	-
ipAddress	Current IP address of the Ethernet interface	ARRAY [0..3] OF USINT	-
subnetMask	Current subnet screen of the Ethernet interface	ARRAY [0..3] OF USINT	-
gatewayAddress	Address of the default router in the network	ARRAY [0..3] OF USINT	-

Result, IP address, subnet screen form, and address of the default router

Description of the return values:

functionResult

has the following meaning:

16#00000000	The active IP configuration was read successfully.
16#00007001	First call, the trigger to read the IP configuration OK (only when command is issued asynchronously). Must be repeated in the next program cycle.
16#00007002	Intermediate call, the reading of the IP configuration is still active. Repeat in the next program cycle where necessary until the IP configuration has been read successfully or canceled (only when the command is issued asynchronously).
16#00007003	Reading of IP configuration aborted (only when command is issued asynchronously).
16#FFFF8090	The specified logical diagnostic address is invalid.
16#FFFF8091	No valid logical diagnostic address for a PROFINET interface or a port of a PROFINET interface.
16#FFFF8092	PROFINET interface is configured, but not available. Can only occur if the PROFINET interface is inserted in SIMOTION as a module.
16#FFFF8094	An abort is no longer possible in the current system function status.
16#FFFF8099	An error occurred while the function was being processed.
16#FFFF809A	Setting the device name is not supported by the PROFINET interface.

16#FFFF80A0	In HW Config, the options "Obtain device names in another way" or "Obtain IP address in another way" are deselected for the respective communication interface. Select both of these options for the corresponding communication interface in HW Config. Function terminated.
16#FFFF80FF	There is no configuration information on a PROFINET interface at the logical diagnostic address. To remedy this, please load a project with a configuration for the PROFINET interface into the device.
16#FFFF80C1	There are currently no resources available to execute this system function. The user program can call this function until this return value no longer appears.

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Specifies whether the diagnostic address in the logicalAddressPnInterface parameter is an input or output address.

logicalAddressPnInterface

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

You must specify a valid diagnostic address of a PROFINET interface/port of a PROFINET interface, from which the IP configuration is to be read. Both Ethernet interfaces with PROFINET basic services (PN-IE) and PN-IO interfaces of a SIMOTION device can be addressed via this address.

nextCommand (optional)

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	3
System default:	IMMEDIATELY

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). IMMEDIATELY means: The functionality is executed asynchronously. With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	4

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND. The same CommandId is used for subsequent calls.

2.2.3.11 _getPnNameOfStation

Note

Applies from SIMOTION version V4.4

Use the '_getPnNameOfStation' system function instead of '_getActiveNameOfStation' when you create a new SIMOTION project. This system function supplies the effective 'NameOfStation' name of the PROFINET interface. This originates from the configuration or was set with the DCP service using 'TcpIp' or by means of the _setNameOfStation system function.

If NameOfStation is not available, a string with a length of 0 is returned. In 'pnPortNumber', 16#FFFFFFFF is then returned. In 'pnSlotNumber', 16#FFFFFFFF is then returned.

Additional references

More detailed information on this system function can be found in

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Return value: StructRetDeviceNameOfStation

StructRetDeviceNameOfStation
Return values for _getPnInterfacePortNeighbour and _getActiveNameOfStation

Structure	Name	Data type	Unit
functionResult	Result of the execution of the system function	DINT	-
nameOfStation	nameOfStation of own device (_getActivNameOfStation) or neighboring device (_getPnInterfacePortNeighbour)	STRING	-
pnPortNumber	pnPortNumber of the neighboring device, with which this is connected to the local device.	UDINT	-
pnSlotNumber	pnSlotNumber of the neighboring device, with which this is connected to the local device.	UDINT	-

Description of the return values:

functionResult

The return value returns the job status and name.

16#00000000	System function completed successfully. The device name was read successfully and is available in the return value.
16#00007001	Initial call, the trigger to read the device name is still active. Repetition necessary in the next program cycle until reading has been completed successfully or aborted (only when command is issued asynchronously).
16#00007002	Intermediate call, the reading of the device name is still active. Repetition necessary in the next program cycle until the reading of the device name has been completed successfully or aborted (only when command is issued asynchronously).
16#00007003	Reading the device name aborted. If an abort is not possible, the system function returns 16#FFFF8094.
16#FFFF8090	The specified logical diagnostic address is invalid.
16#FFFF8091	No valid logical diagnostic address for an interface or port of a PROFINET interface.
16#FFFF8092	PROFINET interface is configured, but not available. Can only occur if the PROFINET interface is inserted in SIMOTION via a module.
16#FFFF8094	An abort is no longer possible in the current system function status.
16#FFFF8099	An error occurred while the function was being processed.
16#FFFF809A	Readout of device name is not supported by the PROFINET interface.
16#FFFF80FF	There is no configuration information on a PROFINET interface at the logical diagnostic address. To remedy this, please load a project with a configuration for the PROFINET interface into the device.
16#FFFF80C1	There are currently no resources available to perform this system function. The user program can continue to call this system function until this return value no longer appears.
16#FFFF80C2	A system function call for the relevant PROFINET interface is already in progress. The user program can call this function until this return value no longer appears.

nameOfStation

Device name of relevant PROFINET interface.

If NameOfStation is not available, a string with a length of 0 is returned.

pnPortNumber

16#FFFFFFFF is always returned.

pnSlotNumber

16#FFFFFFFF is always returned.

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

The iold parameter indicates whether logicalAddressPnInterface of the Profinet interface is an input address or an output address.

logicalAddressPnInterface

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical diagnostic address of the PROFINET interface, or the logical diagnostic addresses of the associated ports on the PROFINET interface, from which the active device name is to be read.

nextCommand (optional)

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	3
System default:	IMMEDIATELY

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. With IMMEDIATELY, an initial call starts the asynchronous processing of the system function. The system function is to be called until it reports that it has been completed (OK or error indications).

With WHEN_COMMAND_DONE, the transition takes place when the command is finished. With ABORT_CURRENT_COMMAND and the same 'commandId' as in the previous function call with IMMEDIATELY, the active function is aborted.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	4

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

This parameter specifies a 'CommandId' of type 'commandIdType' that is unique from all others in the system. The user can make the system generate a 'syncCommandId' by calling _getSyncCommandId. The same 'syncCommandId' is used for subsequent calls.

2.2.3.12 _getPnPortNeighbour

Note

Applies from SIMOTION version V4.4

Use the '_getPnPortNeighbour' system function instead of '_getPnInterfacePortNeighbour' when you create a new SIMOTION project. This system function supplies the name of the PROFINET device connected to the port of the PROFINET interface.

If no identifiable neighbor device, or no device at all, is connected to the relevant port, an empty string will be returned in 'nameOfStation' and the value 16#FFFFFFFF in 'pnPortNumber' and 'pnSlotNumber'.

If an identifiable neighbor device has been found, a string with a length other than 0 will be returned in 'nameOfStation' and a value 0 < x < 255 will be returned in 'pnPortNumber'.

For this system function, only exactly one function call per port is permitted simultaneously.

Additional references

More detailed information on this system function can be found in

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the SIMOTION online help

Return value: StructRetDeviceNameOfStation

StructRetDeviceNameOfStation
Return values for _getPnInterfacePortNeighbour and _getActiveNameOfStation

Structure	Name	Data type	Unit
functionResult	Result of the execution of the system function	DINT	-
nameOfStation	nameOfStation of own device (_getActivNameOfStation) or neighboring device (_getPnInterfacePortNeighbour)	STRING	-
pnPortNumber	pnPortNumber of the neighboring device, with which this is connected to the local device.	UDINT	-
pnSlotNumber	pnSlotNumber of the neighboring device, with which this is connected to the local device.	UDINT	-

The return value returns the job status and name.

Description of the return values:

functionResult

16#00000000	System function completed successfully. The device name and port number of the relevant neighbor device will be returned if it can be identified.
16#00007001	Initial call, the trigger to read out the device name is still active. Repetition necessary in the next program cycle (only when command is issued asynchronously).
16#00007002	Intermediate call, the readout of the neighboring device is still running. A new call in the user program is required until completion of the system function. This return value is reported only for asynchronous execution.
16#00007003	System function aborted successfully. If an abort is not possible, the system function returns the value 16#FFFF8094.
16#FFFF8090	The specified logical diagnostic address is invalid.
16#FFFF8091	No valid logical diagnostic address for a PROFINET interface port.
16#FFFF8092	PROFINET interface is configured, but not available. Can only occur if the PROFINET interface is inserted in SIMOTION via a module.
16#FFFF8093	The STRING returned under 'nameOfStation' is too long. According to the standard, only 240 characters are permitted.
16#FFFF8094	An abort is no longer possible in the current system function status.
16#FFFF8098	The device name returned under 'pnNameOfStation' is invalid, e.g. illegal characters.
16#FFFF8099	Internal error. Function terminated.
16#FFFF809A	The PROFINET interface does not support this service. Function terminated.
16#FFFF809B	The system function cannot be aborted. Function terminated.
16#FFFF809C	There is an unidentifiable neighboring device which is returning invalid data for the slot number, port number, and/or the device name. Function terminated.
16#FFFF809D	Several neighbors are connected to the port addressed by 'logicalAddressOfPnPort'. Function terminated.
16#FFFF80A2	No port neighbor was found on the port addressed by 'logicalAddressOfPnPort'. Function terminated.
16#FFFF80A4	A port neighbor was found on the port addressed by 'logicalAddressOfPnPort', which has an empty or invalid name. Function terminated.
16#FFFF80FF	There is no configuration information on a PROFINET interface at the logical diagnostic address. To remedy this, load a project with a configuration for the PROFINET interface into the device.
16#FFFF80C1	There are currently no resources available to execute this system function. The user program can call this function until this return value no longer appears.
16#FFFF80C2	An instance of the system function with identical call parameters is already running. The user program can call this function until this return value no longer appears.

nameOfStation

Device name of neighbor device

pnPortNumber

Port number of neighbor device used connect to the local device. If no identifiable neighbor device, or no device at all, is connected to the relevant port,

- an empty string will be returned in 'nameOfStation'
- and 16#FFFFFFFF in 'pnPortNumber'.

If an identifiable neighbor device has been found,

- a string with a length other than 0 will be returned in 'nameOfStation' and
- a value of 0 < x < 255 will be returned in 'pnPortNumber'.

pnSlotNumber

Slot number of neighbor device containing the neighbor port. Only returned if the neighbor device can be identified and contains several slots with ports. If a slot number is not available, 16#FFFFFFFF is returned. For this system function, only exactly one function call per port is permitted simultaneously.

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

The 'ioId' parameter indicates whether 'logicalAddressOfPnInterfacePort' of the PROFINET interface is an input address or an output address.

logicalAddressOfPnPort

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical diagnostic address of the port used to determine the device name, port number, and slot number of its neighbor.

nextCommand (optional)

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	3
System default:	IMMEDIATELY

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. With IMMEDIATELY, an initial call starts the asynchronous processing of the system function. The system function is to be called until it reports that it has been completed (OK or error indications).

With WHEN_COMMAND_DONE, the transition takes place when the command is finished. With ABORT_CURRENT_COMMAND and the same 'commandId' as in the previous function call with IMMEDIATELY, the active function is aborted.

With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	4

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND. The same CommandId is used for subsequent calls.

2.2.3.13 _getSegmentIdentification

This function determines all available and configured IO segments.

Additional references

Additional information can be found as follows:

- SIMOTION C Operating Instructions
- SIMOTION D4x5 Commissioning and Hardware Installation Manual
- SIMOTION D4x5-2 Commissioning and Hardware Installation Manual
- or in the SIMOTION online help

Return value: StructRetGetSegmentIdentification

EnumDpSegmentId

NO_DP_SEGMENT (1)	No DP segment
DP_1 (2)	DP PROFIBUS interface 1 - external
DP_2 (3)	DP PROFIBUS interface 2 - external
DP_3 (4)	DP PROFIBUS interface 3 - internal
IE_1 (5)	Industrial Ethernet interface 1
IE_2 (6)	Industrial Ethernet interface 2
PN_1 (7)	PROFINET IO interface 1
PN_2 (8)	PROFINET IO interface 2

StructRetGetSegmentIdentification
Return parameter for _getSegmentIdentification

Structure	Name	Data type	Unit
functionResult	Return value of the job	DINT	-
numberOfSegments	Number of available and configured segments	UDINT	-
area	Area identifier	ARRAY [0..9] OF UDINT	-
dpSegmentId	Segment ID	ARRAY [0..9] OF EnumDpSegmentId	-
mastersystemId	Master system ID	ARRAY [0..9] OF UDINT	-
diagnosticAddress	Diagnostic address of the segment	ARRAY [0..9] OF UDINT	-

The return value consists of 'functionResult', 'numberOfSegments', 'area', 'dpSegmentId', 'mastersystemId', and 'diagnosticAddress'.

Description of the return values:

functionResult

'functionResult' reports the job status.

16#00000000	Job completed without errors.
16#FFFF809F	Internal error, job aborted. Function not executable.

numberOfSegments

'numberOfSegments' reports the number of available and configured IO segments.

area

'area' reports the area identifier.

AREA = 1 : PBUS AREA = 2: PROFIBUS DP AREA = 7: ONBOARD AREA = 8: PROFINET IO

dpSegmentId

'dpSegmentId' reports module rack/station number.

NO_DP_SEGMENT = 1 : No DP segment (e.g., PBUS, ONBOARD) DP_1 = 2: PROFIBUS DP interface 1 DP_2 = 3: PROFIBUS DP interface 2 DP_3 = 4: PROFIBUS DP interface Integrated PN_1 = 7: PROFINET IO interface 1 PN_2 = 8 : PROFINET IO interface 2

Note

The interfaces IE_1 and IE_2 are not IO segments and are not supported.

'mastersystemId' reports the master system ID.

Note

PROFIBUS and ONBOARD have no master system ID. It is assigned the value 0.

'diagnosticAddress' reports the diagnostic address of the IO segment.

Note

PROFIBUS and ONBOARD do not have their own diagnostic address. It is assigned the value INVALID_ADDRESS.

Note

For modules with two PROFINET interfaces, the integrated PN interface and the optional PN interface is always reported with PN_1 and PN_2, respectively.

2.2.3.14 _getStateOfAllDpSlaves

This function returns a list of status data for all existing DP slaves. This function only provides information for DP slaves with PROFIBUS DP.

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Return value: StructRetGetStateOfAllDpSlaves

EnumStateOfDpSlave

INACTIVE (0)	Slave/IO device deactivated by program
IN_OPERATION (1)	Input/output data exchange with slave/IO device
DATA_EXCHANGE_INACTIVE (2)	No input/output data exchange with slave/IO device
NOT_PRESENT (3)	Slave/IO device is not available (no supply voltage, no connection to bus, etc.)

StructRetGetStateOfAllDpSlaves
Return parameter for _getStateOfAllDPSlaves

Structure	Name	Data type	Unit
functionResult	Total job return value	DINT	-
ActualStateOfAllDpSlaves	Status data of the available DP slaves	ARRAY [0..127] OF EnumStateOfDpSlave	-

Description of the return values:

16#00000000	Okay - system function completed successfully
16#00007000	Functionality not active
16#00007001	Functionality initiated
16#00007002	Functionality active
16#FFFF8090	No valid logical address for DP master interface
16#FFFF8091	Not a valid DP segment
16#FFFF8092	A new call of the function with these parameters is not permitted
16#FFFF8093	Not a valid DP station
16#FFFF8098	Functionality terminated with timeout
16#FFFF8099	Internal error, function terminated
16#FFFF80C1	Function already started, new call with parameter 'ReqGetStateOfAllDPSlavesStateMode=TRUE'
16#FFFF80C3	Temporary resource bottleneck
16#FFFF80CF	CommandId not unique
16#FFFF809E	Abort of function not allowed
16#FFFF809D	Abort successful

Describes the state of the requested DP slave:

INACTIVE means the slave is deactivated by the program. IN_OPERATION means input/output data exchange takes place via the slave.

DATA_EXCHANGE_INACTIVE means the station is disturbed (the head-end of the station or one of its modules is disturbed).

Possible causes:

- Fault in station head-end
  - No data exchange with station head-end or modules in the station is possible.
- Fault on a module in the station
  - Data exchange with station head-end or modules in the station is possible. One or more diagnostics of station head-end or modules in the station are pending. The user program can receive additional information about the causes by reading the diagnostics of the station head-end or the individual modules.

NOT_PRESENT means the slave is not available (no supply voltage, no connection to bus, etc.).

Parameters:

logicalDiagnosticAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

Logical diagnostic address of DP master interface

ReqGetStateOfAllDpSlavesStateMode

Direction:	Input parameter
Data type:	EnumReqActDeactGetStateMode
Parameter index:	2

EnumReqActDeactGetStateMode

REQUEST_TRUE (0)	Start functionality
REQUEST_FALSE (1)	Query status with REQUEST_TRUE
REQUEST_ABORT (2)	Abort of the function execution

REQUEST_TRUE starts the functionality. REQUEST_FALSE scans the status of the functionality started with REQUEST_TRUE. REQUEST_ABORT aborts the function execution.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	3

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and the function cannot be completed.

With WHEN_COMMAND_DONE, the transition takes place when the command has been executed. With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	4

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status.

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	6
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of the diagnostic address of the distributed station. For INPUT, the diagnostic address of the distributed station is in the input range. For OUTPUT, the diagnostic address of the distributed station is in the output range. Diagnostic addresses are always of type INPUT.

2.2.3.15 _getStateOfAllDpStations

The system function determines the status of the communication to cyclical communications partners (device - PROFINET, slave - PROFIBUS, sender or recipient controller-controller internode communication - PROFINET).

The system function _getStateOfAllDpStations supplies the overall status of the stations configured for the intended interface (PROFIBUS, PROFINET IO) where this information is available. This status is formed from the status of the individual modules of the station.

If the system function _getStateOfAllDpStations returns the overall status DISTURBED, _getStateOfIO can be used to query the status of the slots/sub-slots or _getStateOfSingleDpSlave for diagnosing the communication interface.

The user program calls the system function the first time with 'RequestMode = REQUEST_TRUE'. This call returns a maximum of 10 results. In the 'functionResult', 0x00000001 signals that other status options are available.

The user program then calls the system function with the identifier 'RequestMode = REQUEST_FALSE'. The call is repeated until all status values have been read out. This is signaled with 'functionResult' 0x00000000.

Note

Where further function calls are made with 'RequestMode = REQUEST_FALSE', the same CommandID must be used as when calling with 'REQUEST_TRUE'. Unless the same CommandId is used, the function will continue to be restarted and only the first 10 status values will be displayed.

If the process of reading out the status values is to be aborted, the system function must be called with 'RequestMode = REQUEST_ABORT'.

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- Communication System Manual
- or in the online help

Return value: StructRetDeviceGetStateOfAllDpStations

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

EnumStateOfDpStation

NO_VALID_STATE (0)	Default value or a valid state could not be determined
INACTIVE (1)	Slave/IO device deactivated by program
IN_OPERATION (2)	Input/output data exchange with slave/IO device
DISTURBED (3)	DP slave / IO device is faulty (diagnostics pending, modules faulty ...)
NOT_PRESENT (4)	The diagnostics, input/output address of one or all modules in a station cannot be accessed by the controller. (Possible causes: Missing power supply of the module, missing connection to the bus, station failure, for example, caused by faulty or removed module in an ET200.)

StructRetDeviceGetStateOfAllDpStations
Return value for _getStateOfAllDpStations

Structure	Name	Data type	Unit
functionResult	Result	DINT	-
NumberResultsDpStations	Number of states that are signaled in actualStateOfDpStations	UDINT	-
StructStateOfDpStations	Status data of the available DP slaves / IO devices	ARRAY [0..9] OF StructStateOfDpStations	-

StructStateOfDpStations

Return parameters of the individual devices

Structure	Name	Data type	Unit
ioIdType	IO type	EnumIoIdType	-
logicalDiagnosticAddressDpStations	Diagnostics address of the PROFIBUS slave or IO device	DINT	-
actualStateOfDpStation	Status of the relevant PROFIBUS slave or IO device	EnumStateOfDpStation	-

Description of the return values: Return value 'functionResult'

16#00000000	The system function has been executed. All information on the configured PROFIBUS DP slaves or PROFINET IO devices is reported.
16#00000001	Ok - even more status values for PROFINET IO devices or PROFIBUS DP slaves are available. The system function must be called again with 'REQUEST_FALSE' if the other status values are to be read out.
16#00007000	Functionality is not active, only for asynchronous execution.
16#00007001	Functionality initiated, only possible for asynchronous execution.
16#00007002	Functionality active, only possible for asynchronous execution.
16#00007003	System function aborted successfully. If an abort is not possible, the system function returns the value 16#FFFF8094.
16#FFFF8090	The specified logical diagnostic address is invalid. Address is outside of value range for the logical diagnostic addresses.
16#FFFF8091	Specified logical diagnostic address is not an address for a PROFIBUS DP / PROFINET segment.
16#FFFF8092	The relevant PROFIBUS DP / PROFINET segment does not contain any configured PROFIBUS DP stations or PROFINET IO devices.
16#FFFF8093	The status values readout is not supported by the relevant PROFIBUS DP / PROFINET segment. The status values can only be read out by a PROFIBUS DP master interface or a PROFINET IO controller of a PROFINET interface.
16#FFFF8094	An abort is no longer possible in the current system function status.
16#FFFF8097	No status can be provided for the logical diagnostic address.
16#FFFF8099	Internal error, function terminated.
16#FFFF809E	Job aborted. Attempt to abort a non-active function.
16#FFFF80C1	There are currently no resources available to execute this system function. The user program can continue to call this system function until this return value no longer appears.
16#FFFF80C2	A re-querying of the PROFIBUS DP / PROFINET segment with REQUEST_TRUE was rejected because a function call for querying the status values of the relevant PROFINET / PROFIBUS DP segment is still in progress with an identical 'CommandId'. The user program can call this function until this return value no longer appears.

Return value 'NumberResultsDpStations'

'NumberResultsDpStations' supplies the number of result elements in the 'StructStateOfDpStations' array. If 'NumberResultsDpStations' is less than 10, the system function enters the value NO_VALID_STATE the logical diagnostic address 16#00007FFE (invalid diagnostic address) and in 'actualStateOfDpStation' for those array elements that are larger than 'NumberResultsDpStations'.

Return value 'actualStateOfDpStation'

Status of the relevant PROFIBUS DP slave or PROFINET IO device. NO_VALID_STATE system default or no valid state could be determined. INACTIVE means the slave or the IO device is deactivated by the program. IN_OPERATION means that input/output data exchange with slave or IO device is in progress. DISTURBED means the station is disturbed (head-end of station or a module in the station).

Possible causes:

- Fault in station head-end,
  - i.e. no data exchange with station head-end or modules in the station is possible.
- Fault on a module in the station.
- Data exchange with station head-end or modules in the station is possible.
  - One or more diagnostics of station head-end or modules in the station are pending.
- The user program can receive additional information about the causes by reading the diagnostics of the station head-end or the individual modules.

NOT_PRESENT means the diagnostics, input/output address of one or all modules in a station cannot be accessed by the controller (possible causes: Missing power supply of the module, missing connection to the bus, station failure, for example, caused by faulty or removed module in an ET200). Data exchange with the station or the modules in the station is not possible (system-internal functionality, user program, engineering system).

Parameters:

logicalDiagnosticAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

The logical diagnostic address of a PROFIBUS DP master interface or an IO controller's PROFINET interface should be entered under the 'logicalDiagnosticAddress' parameter.

Unless all elements are filled out in the 'actualStateOfDpStation' ARRAY, then the return value 16#00007FFE (invalid diagnostic address) is output. This means the content of 'actualStateOfDpStation' is irrelevant.

requestMode

Direction:	Input parameter
Data type:	EnumReqSysFunctMode
Parameter index:	2

EnumReqSysFunctMode

REQUEST_TRUE (0)	Start functionality
REQUEST_FALSE (1)	Status query of functionality started with REQUEST_TRUE
REQUEST_ABORT (2)	Abort of the function execution

REQUEST_TRUE starts the functionality. REQUEST_FALSE queries the status with the started functionality REQUEST_TRUE. REQUEST_ABORT aborts the function execution.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	3

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and cannot be completed. If the system function reports a display, the internal processing will be finished (other than for nextCommand, ABORT_CURRENT_COMMAND) and a restart of the system function possible with REQUEST_TRUE.

With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	4

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

This parameter specifies a 'commandId' of type 'commandIdType' that is unique from all others in the system for the purpose of tracking the command status. A 'commandId' is valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND. The system can generate the parameter 'syncCommandId' by calling the system function _getSyncCommandId. The same 'syncCommandId' is to be used for subsequent calls.

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	6
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of the diagnostic address of the communication interface (PROFIBUS DP/ PROFINET). For INPUT, the diagnostic address of the communication interface (PROFIBUS DP/ PROFINET) is in the input range. For OUTPUT, the diagnostic address of the communication interface (PROFIBUS DP/PROFINET) is in the output range. Diagnostic addresses are always of type INPUT.

2.2.3.16 _getStateOfDiagnosticDataCommand

This function supplies the status of the '_readDiagnosticData' function.

Return value: StructRetDeviceCommandState

EnumDeviceCommandIdState

ACTIVE (4)	Job active (asynchronous command)
NOT_EXISTENT (94)	Job completed or does not exist yet
WAITING (154)	Job active (synchronous command)

StructRetDeviceCommandState
Return parameters for device commands

Structure	Name	Data type	Unit
functionResult	Return value of function	DINT	-
commandIdState	State of job	EnumDeviceCommandIdState	-

The return value consists of 'functionResult' and 'commandStateId'.'functionResult' formally indicates job completion with value 0.'commandStateId' indicates the job status:

-ACTIVE	Job active (synchronous command)
-WAITING	Job active (asynchronous command)
-NOT_EXISTENT	Job completed or does not exist yet

Parameters:

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	1

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Specifies the 'commandId' of the data transfer command whose status is to be returned.

2.2.3.17 _getStateOfDpSlave

This function supplies the information whether the PROFIBUS DP slave or the PROFINET IO device is activated or deactivated.

Note that for evaluating the _getStateOfDpSlave system function result, the 'functionResult' and 'commandIdState' return values are related to each other and are evaluated together.

If the 'functionResult' return parameter has the value 0, a single call of the system function completed successfully. The current status of the slave/device can be derived from the 'commandIdState' return parameter and represents the device status associated with the logical address when the system function was executed.

For the asynchronous call of the _getStateOfDpSlave function, this relationship means that a single call was successful but a subsequent call to determine the status is required.

Example

1st-nth Asynchronous start:

functionResult = 0, commandIdState = FUNCTION_IS_ACTIVE(28674) Meaning: Although the function was successfully called asynchronously, the device status could not yet be determined.

N+1 asynchronous call:

functionResult = 0, commandIdState = ACTIVE(1) / INACTIVE(2) / IN_ACTIVATION(3) / IN_DEACTIVATION(4) Meaning: The function was successfully called asynchronously, the device status could be determined. No further subsequent call is required.

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- SIMOTION Communication System Manual
- or in the online help

Return value: StructRetDeviceGetStateOfDpSlave

EnumDeviceStateOfDpSlave

NO_RESSOURCES_AVAILABLE (-32573)	No further resources available (16#FFFF80c3)
ACTIVE (1)	DP station activated (16#00000001)
INACTIVE (2)	DP station deactivated (16#00000002)
IN_ACTIVATION (3)	Activating DP station (16#00000003)
IN_DEACTIVATION (4)	Deactivating DP station (16#00000004)
ACTIVATED_NO_ALARM (5)	Internal value - not relevant for user (16#00000005)
DEACTIVATED_NO_ALARM (6)	Internal value - not relevant for user (16#00000006)
FUNCTION_IS_WAITING_FOR_ACTIVATION (28672)	Function is waiting for activation to complete (16#00007000)
FUNCTION_IS_ACTIVATED (28673)	Function is activated (16#00007001)
FUNCTION_IS_ACTIVE (28674)	Function is active (16#00007002)

StructRetDeviceGetStateOfDpSlave

Return parameters for device commands

Structure	Name	Data type	Unit
functionResult	Return value of function	DINT	-
commandIdState	State of job	EnumDeviceStateOfDpSlave	-

Description of the return values:

16#00000000	Abort job. No jobs are now being executed for the logical address.
16#00007003	Abort job. The job has already been accepted, but its execution could not continue as feedback is expected from the system.
The return value reports the job status.	
16#FFFF8090	Specified logical base address invalid.
16#FFFF8091	No valid DP adapter segment, job aborted: Function not executed, all resources are released.
16#FFFF8093	Not a valid DP station.
16#FFFF8098	Functionality terminated after timeout.
16#FFFF8099	Internal error: Function terminated.

Parameters:

logicalAddressOfDpStation

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

Specifying the logical diagnostic address of the DP station or IO device or I device.

reqActDeactGetStateMode

Direction:	Input parameter
Data type:	EnumReqActDeactGetStateMode
Parameter index:	2

EnumReqActDeactGetStateMode

REQUEST_TRUE (0)	Start functionality
REQUEST_FALSE (1)	Query status with REQUEST_TRUE
REQUEST_ABORT (2)	Abort of the function execution

REQUEST_TRUE starts the functionality. REQUEST_FALSE scans the status of the functionality started with REQUEST_TRUE. REQUEST_ABORT aborts the function execution.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	3

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and the function cannot be completed.

With WHEN_COMMAND_DONE, the transition takes place when the command is finished. With ABORT_CURRENT_COMMAND and the same 'commandId' as in the previous function call with IMMEDIATELY, the active function is aborted.

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	4
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of the logical diagnostic address of the distributed station or IO device or I device. For INPUT, the logical diagnostic address of the distributed station, IO device, or I device is in the input range. For OUTPUT, the logical diagnostic address of the distributed station, IO device, or I device is in the output range.

Note

Diagnostic addresses are always of type INPUT.

2.2.3.18 _getStateOfIO

This system function supplies the status of the device, the module or the submodule. This is addressed with the "Logical address", "Input/output identification" and "Scope" parameters.

The user program calls the system function the first time with 'RequestMode = REQUEST_TRUE'. This call returns a maximum of 10 results. In the 'functionResult', 0x00000001 signals that other status values are available.

The user program then calls the system function with the identifier 'RequestMode = REQUEST_FALSE'. The call is repeated until all status values have been read out. This is signaled with 'functionResult' 0x00000000.

Note

Where further function calls are made with 'RequestMode = REQUEST_FALSE', the same CommandID must be used as when calling with 'REQUEST_TRUE'. Unless the same CommandId is used, the function will continue to be restarted and only the first 10 status values will be displayed.

If the process of reading out the status values is to be aborted, the system function must be called with 'RequestMode = REQUEST_ABORT'.

Example:

Configuration of the I device in HW Config

Transfer address: 300

Diagnostics address of the interface: 16378

Via the diagnostics address of the PN interface:

Example

```
myRetStructRetDeviceGetStateOfIO_1 :=_getStateOfIO(
 logicalAddress := 16378
 ,ioId := INPUT
 ,scope := OBJECT
 ,requestMode := REQUEST_TRUE
 ,nextCommand := when_command_done
 ,commandId := _getcommandid());
```

Via the transfer address of the I device:

Example

```
myRetStructRetDeviceGetStateOfIO_1 :=_getStateOfIO(
 logicalAddress := 300
 ,ioId := INPUT
 ,scope := OBJECT
 ,requestMode := REQUEST_TRUE
 ,nextCommand := when_command_done
 ,commandId := _getcommandid());
```

Return value: StructRetDeviceGetStateOfIO

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

EnumStateOfIO

NO_VALID_STATE (0)	Default value or a valid status could not be determined
INACTIVE (1)	PROFIBUS DP slave / PROFINET IO device deactivated by program
IN_OPERATION (2)	Data exchange performed with PROFIBUS DP slave, PROFINET IO device, module or submodule
DISTURBED (3)	PROFIBUS DP slave / PROFINET IO device, module or submodule is faulty (diagnostics pending, modules faulty ...)
NOT_PRESENT (4)	PROFIBUS DP slave / PROFINET IO device, module or submodule is not present (missing supply voltage, missing connection to the bus ...)

EnumStateOfMaintenanceOfIO

MAINTENANCE_NO_INFO (0)	No information
MAINTENANCE_REQUIRED (1)	Maintenance needed soon
MAINTENANCE_DEMANDED (2)	Maintenance required

StructRetDeviceGetStateOfIO
Return value for _getStateOfIO

Structure	Name	Data type	Unit
functionResult	Result	DINT	-
NumberOfResults	Number of states that are signaled in actualStateOfDpStations	UDINT	-
stateOfIO	Status for the PROFIBUS DP slave, PROFINET IO device, module or submodule	ARRAY [0..9] OF StructStateOfIO	-

StructStateOfIO
Return value of a unit

Structure	Name	Data type	Unit
logicalAddress	Logical address of the PROFIBUS DP slave, PROFINET IO device, module or submodule	DINT	-
ioIdType	IO type	EnumIoIdType	-
actualStateOfDpStation	Status for the PROFIBUS DP slave, PROFINET IO device, module or submodule	EnumStateOfIO	-
stateOfMaintenace	Status for maintenance of the PROFIBUS DP slave, PROFINET IO device, module or submodule objects	EnumStateOfMaintenanceOfIO	-

Description of the return values:

Return value 'functionResult'

16#00000000	The system function has been executed. All information about the configured modules / submodules is reported.
16#00000001	OK - other status values for objects are present. The system function must be called again with 'REQUEST_FALSE' if the other status values are to be read out.
16#00007000	Functionality is not active, only for asynchronous execution.
16#00007001	Functionality initiated, only possible for asynchronous execution.
16#00007002	Functionality active, only possible for asynchronous execution.
16#00007003	System function aborted successfully. If an abort is not possible, the system function returns the value 16#FFFF8094.
16#FFFF8090	The specified logical diagnostic address is invalid. Address is outside of value range for the logical diagnostic addresses.
16#FFFF8091	Specified logical diagnostic address is not an address for a PROFIBUS DP / PROFINET segment.

16#FFFF8092	Depending on the logical address and the input/output assignment, the relevant PROFIBUS DP / PROFINET segment does not contain any configured PROFIBUS DP stations or PROFINET IO devices or input or output addresses.
16#FFFF8093	The status values readout is not supported by the relevant PROFIBUS DP / PROFINET segment. The status values can only be read out by a PROFIBUS DP master interface or a PROFINET IO controller of a PROFINET interface.
16#FFFF8094	An abort is no longer possible in the current system function status.
16#FFFF8097	No status can be provided for the logical diagnostic address.
16#FFFF8092	The relevant PROFIBUS DP / PROFINET segment does not contain any configured PROFIBUS stations or IO devices.
16#FFFF8093	The status values readout is not supported by the relevant PROFIBUS DP / PROFINET segment. The status values can only be read out by a PROFIBUS DP master interface or a PROFINET IO controller of a PROFINET interface.
16#FFFF8094	An abort is no longer possible in the current system function status.
16#FFFF8097	No status can be provided for the logical diagnostic address.
16#FFFF8099	Internal error, function terminated.
16#FFFF809E	Job aborted. Attempt to abort a non-active function.
16#FFFF80C1	There are currently no resources available to execute this system function. The user program can continue to call this system function until this return value no longer appears.
16#FFFF80C2	A re-querying of the PROFIBUS DP / PROFINET segment with REQUEST_TRUE was rejected because a function call for querying the status values of the relevant PROFINET / PROFIBUS DP segment is still in progress with an identical 'CommandId'. The user program can call this function until this return value no longer appears.

Return value 'stateOfIO'

Logical address of the associated PROFIBUS DP slave, IO device, module or submodule.

Input/output identification for the logical address.

Status.

NO_VALID_STATE means no default value or valid state could be determined.

INACTIVE means the slave or IO device is deactivated by the program.

IN_OPERATION means that input/output data exchange with slave or IO device is in progress.

DISTURBED means the station is disturbed (head-end of station or a module in the station).

Possible causes:

- Fault on station head-end, so no data exchange with station head-end or modules in the station.
- Fault on a module in the station. Data exchange with station head-end or modules in the station is possible. One or more diagnostics of station head-end or modules in the station are pending.
- The user program can receive additional information about the causes by reading the diagnostics of the station head-end or the individual modules.

NOT_PRESENT means the diagnostics, input/output address of one or all modules in a station cannot be accessed by SIMOTION (possible causes: Missing supply voltage, missing connection to the bus, station has failed, e.g. station does not report, module has an internal error or has been removed, for example for an ET200S, etc.)

Parameters:

logicalAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

The logical address of a PROFIBUS master interface, or of a PROFINET interface of an IO controller, a module or a submodule should be entered under the 'logicalAddress' parameter.

Unless all elements are filled out in the 'actualStateOfDpStation' ARRAY, then the return value 16#00007FFE (invalid diagnostic address) is output. This means the content of 'actualStateOfDpStation' is irrelevant.

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	2
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of the logical address With INPUT, the logical address is in the input range. With OUTPUT, the logical address is in the output range.

scope (optional)

Direction:	Input parameter
Data type:	EnumScopeOfIO
Parameter index:	3
System default:	OBJECT

EnumScopeOfIO

OBJECT (0)	Status of all elements of the object referenced by its logical address.
CONNECTED (1)	Status of all elements connected to the addressed object.

Scope of the logical address

For OBJECT, the status of all submodules, which are to be referenced to an object via the logical address, is returned. The following are designated in a representative role as the object: PROFINET IO interface, PROFINET IO device, PROFIBUS DP interface or PROFIBUS DP slave (DP station).

For PROFINET IO / PROFINET IO device, the status of all submodules addressed via PROFINET IO (device) is reported, for example, the status of the submodules or the status of all ports. For PROFIBUS DP interface or PROFIBUS DP slave, the status of the submodules is also returned.

For CONNECTED, the logical address is that of a distributed station. For CONNECTED, the status of all modules of the distributed station is output.

For PROFINET IO interface, the status of all PROFINET IO devices is returned. For PROFINET IO device, the status of all modules is returned. For PROFIBUS DP interface, the status of all PROFIBUS DP slaves is output. For PROFIBUS DP slave, the status of all modules is output.

requestMode

Direction:	Input parameter
Data type:	EnumReqSysFunctMode
Parameter index:	4

EnumReqSysFunctMode

REQUEST_TRUE (0)	Start functionality
REQUEST_FALSE (1)	Status query of functionality started with REQUEST_TRUE
REQUEST_ABORT (2)	Abort of the function execution

REQUEST_TRUE starts the functionality. REQUEST_FALSE queries the status with the started functionality REQUEST_TRUE. REQUEST_ABORT aborts the function execution.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	5

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and the function cannot be completed.

With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	6

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

This parameter specifies a 'commandId' of type 'commandIdType' that is unique from all others in the system for the purpose of tracking the command status. A 'commandId' is valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND. The system can generate the parameter 'syncCommandId' by calling the system function _getSyncCommandId. The same 'syncCommandId' is to be used for subsequent calls.

2.2.3.19 _getStateOfProcessInterruptCommand

This function supplies the status of the '_sendProcessInterrupt' function.

Return value: StructRetDeviceCommandState

EnumDeviceCommandIdState

ACTIVE (4)	Job active (asynchronous command)
NOT_EXISTENT (94)	Job completed or does not exist yet
WAITING (154)	Job active (synchronous command)

StructRetDeviceCommandState
Return parameters for device commands

Structure	Name	Data type	Unit
functionResult	Return value of function	DINT	-
commandIdState	State of job	EnumDeviceCommandIdState	-

The return value consists of 'functionResult' and 'commandStateId'.'functionResult' formally indicates job completion with value 0.'commandStateId' indicates the job status:

-ACTIVE	Job active (synchronous command)
-NOT_EXISTENT	Job completed or does not exist yet
-WAITING	Job active (asynchronous command)

Parameters:

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	1

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Specifies the 'commandId' of the data transfer command whose status is to be returned.

2.2.3.20 _getStateOfRecordCommand

The function provides the status of a command to transfer a data set within the following system functions:

- _readRecord
- _writeRecord
- _readDriveFaults
- _readDriveParameterDescription
- _readDriveMultiParameterDescription
- _readDriveParameter
- _readDriveMultiParameter
- _writeDriveParameter
- _writeDriveMultiParameter

Return value: StructRetDeviceCommandState

EnumDeviceCommandIdState

ACTIVE (4)	Job active (asynchronous command)
NOT_EXISTENT (94)	Job completed or does not exist yet
WAITING (154)	Job active (synchronous command)

StructRetDeviceCommandState
Return parameters for device commands

Structure	Name	Data type	Unit
functionResult	Return value of function	DINT	-
commandIdState	State of job	EnumDeviceCommandIdState	-

The return value consists of 'functionResult' and 'commandStateId'.'functionResult' formally indicates job completion with value 0.'commandStateId' indicates the job status:

-ACTIVE	Job active (asynchronous command)
-NOT_EXISTENT	Job completed or does not exist yet
-WAITING	Job active (synchronous command).

Parameters:

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	1

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Specifies the 'commandId' of the data transfer command whose status is to be returned.

2.2.3.21 _getStateOfSingleDpSlave

Note

Do not use the _getStateOfSingleDpSlave function anymore as it is obsolete. Use the _getStateOfIO function instead.

The function determines the status of the communication to a cyclical communications partner (device - PROFINET, slave - PROFIBUS, sender or recipient controller-controller internode communication - PROFINET).

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- Communication System Manual
- or in the online help

Return value: StructRetGetStateOfSingleDpSlave

EnumStateOfDpSlave

INACTIVE (0)	Slave/IO device deactivated by program
IN_OPERATION (1)	Input/output data exchange with slave/IO device
DATA_EXCHANGE_INACTIVE (2)	No input/output data exchange with slave/IO device
NOT_PRESENT (3)	Slave/IO device is not available (no supply voltage, no connection to bus, etc.)

StructRetGetStateOfSingleDpSlave
Return parameter for _getStateOfSingleDPSlave

Structure	Name	Data type	Unit
functionResult	Total job return value	DINT	-
ActualStateOfSingleDpSlave	Status data of the individual DP slave	EnumStateOfDpSlave	-

Describes the state of the requested DP slave:

INACTIVE means the slave or IO device is deactivated by the program. IN_OPERATION means that input/output data exchange with slave or IO device is in progress.

DATA_EXCHANGE_INACTIVE means the station is disturbed (the head-end of the station or one of its modules is disturbed).

Possible causes:

- Fault in station head-end
  - No data exchange with station head-end or modules in the station is possible.
- Fault on a module in the station
  - Data exchange with station head-end or modules in the station is possible. One or more diagnostics of station head-end or modules in the station are pending. The user program can receive additional information about the causes by reading the diagnostics of the station head-end or the individual modules.

NOT_PRESENT means the slave or IO device is not available (no supply voltage, no connection to bus, etc.).

Description of the return values:

16#00000000	System function completed successfully.
16#00007000	Functionality is not active, only for asynchronous execution.
16#00007001	Functionality initiated, only for asynchronous execution.
16#00007002	Functionality active, only for asynchronous execution.
16#FFFF8090	No valid PROFIBUS or PROFINET segment.
16#FFFF8091	No configuration present for the specified DP station address or device number.
16#FFFF8092	A new call of the function with these parameters is not permitted.
16#FFFF8093	No valid logical diagnostic address of a DP station or an IO device.
16#FFFF8098	The automated status of the system function terminated with timeout.
16#FFFF8099	Internal error, terminate function.
16#FFFF80C1	Function already started, new call with parameter 'ReqGetStateOfSingleDPSlaveStateMode=TRUE'.
16#FFFF80C3	Temporary resource bottleneck.
16#FFFF80CF	CommandId not unique.
16#FFFF809E	Abort of function not allowed.
16#FFFF809D	Abort successful.

Parameters:

logicalDiagnosticAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

Diagnostic address of the communications partner or the receive connection of the internode communication.

Note

Several connections are possible at the receiver side of the controller-controller internode communication. Each connection is represented as communications partner.

ReqGetStateOfSingleDPSlaveStateMode

Direction:	Input parameter
Data type:	EnumReqActDeactGetStateMode
Parameter index:	2

EnumReqActDeactGetStateMode

REQUEST_TRUE (0)	Start functionality
REQUEST_FALSE (1)	Query status with REQUEST_TRUE
REQUEST_ABORT (2)	Abort of the function execution

REQUEST_TRUE starts the functionality. REQUEST_FALSE scans the status of the functionality started with REQUEST_TRUE. REQUEST_ABORT aborts the function execution.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	3

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and the function cannot be completed.

With WHEN_COMMAND_DONE, the transition takes place when the command has been executed. With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	4

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status.

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	6
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of the logical diagnostic address of the distributed station. For INPUT, the logical diagnostic address of the distributed station is in the input range. For OUTPUT, the logical diagnostic address of the distributed station is in the output range. Diagnostic addresses can be of type INPUT or OUTPUT.

2.2.3.22 _GetStateOfXCommand

This function supplies the user with the current status of the XSend or XReceive job.

Additional references

Additional information is available at:

- SIMOTION Basic Functions Function Manual
- or in the SIMOTION online help

Return value: StructRetXCommandState

EnumXCommandIdState

ACTIVE (4)	Job active (asynchronous command)
NOT_EXISTENT (94)	Job completed or does not exist yet
WAITING (154)	Job active (synchronous command)

StructRetXCommandState
Output parameter for _GetStateOfCommand

Structure	Name	Data type	Unit
functionResult	Return value of function	DINT	-
commandIdState	State of job	EnumXCommandIdStat e	-

Return value of function:

If 'functionResult'=0, then 'commandIdState' is valid. ACTIVE means the job is currently underway (asynchronous command). NOT_EXISTENT means the job is already complete or does not exist yet. WAITING means the job execution has not yet commenced (synchronous command).

Parameters:

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	1

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status.

2.2.3.23 _getStationType

This function can be used to determine the device ID and vendor ID for a DP station (DP slave, IO device, etc.).

Return value: StructRetGetStationType

StructRetGetStationType
Return parameter for _getStationType

Structure	Name	Data type	Unit
functionResult	Return value of the job	DINT	-
deviceId	Device ID of the station	UDINT	-
vendorId	Vendor ID of the station	UDINT	-

The return value consists of 'functionResult,' 'deviceId,' and 'vendorId'.

Description of the return values:

functionResult

'functionResult' reports the job status.

16#00000000	Job completed without errors.
16#FFFF8090	Job aborted. The specified station address could not be found.
16#FFFF8091	Job aborted. Segment was not found (check 'area' and 'mastersystemId').
16#FFFF809F	Internal error, job aborted. Function not executable.

deviceId

'deviceId' reports the device ID of a station.

Note

Device IDs are assigned by the PROFIBUS user organization and must be requested for each module type by the manufacturer. Please ask the manufacturer about the appropriate device ID for your module. With I/O bus and ONBOARD, there is no device ID. The value 0 is returned.

vendorId

'vendorId' reports the vendor ID of a station.

Note

A list with the manufacturer's IDs can be obtained from the PROFIBUS user organization at www.profibus.com (I&M Support). With PROFIBUS, I/O bus and ONBOARD, there is no vendor ID. The value 0 is returned.

Parameters:

area

Direction:	Input parameter
Data type:	UDINT
Parameter index:	1

Specifies the area identifier of a segment. This value can be determined using system function _getSegmentIdentification. AREA = 1 : PBUS AREA = 2: PROFIBUS DP AREA = 7: ONBOARD AREA = 8: PROFINET IO

mastersystemId

Direction:	Input parameter
Data type:	UDINT
Parameter index:	2

Specifies the master system ID of a segment. This value can be determined using system function _getSegmentIdentification.

stationAddress

Direction:	Input parameter
Data type:	UDINT
Parameter index:	3

Specifies a station address/rack address within a segment. This value can be determined using system function _getNextLogAddress.

2.3 drives

2.3.1 SIMODRIVE

2.3.1.1 _abortAllReadWriteDriveParameterJobs

This function aborts all parameter jobs currently being processed. The term "parameter job" refers to the functions _readDrive(Multi)ParameterDescription, _readDrive(Multi)Parameter, _writeDrive(Multi)Parameter and _readDriveFaults.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual
  Section Deleting _readDrive and _writeDrive jobs
- or in the SIMOTION online help

Return value: DINT

Description of the return values:

16#00000000	Job completed without errors.
16#FFFF819F	Error, job aborted. Function not executable.
16#FFFF81C5	Error, can be repeated in the next program cycle. Function could not be executed. This can happen for the following reasons: - The function _abortAllReadWriteDriveParameterJobs is already being executed. - A parameter job is currently being called from another task.

2.3.1.2 _POSMOA_control

The _POSMOA_control function block can be used to control the connected SIMODRIVE POSMO A in positioning mode (p700=1). The functions are as follows: initializing, homing, jogging, program execution, MDI, fault acknowledgment, automatic single block mode/automatic.

Hardware/software requirements:

SIMOTION V4.1 and higher. Hardware version POSMO A 75W: O and higher. Software version POSMO A 75W: As of V3.0

Hardware version POSMO A 300 W: G and higher. Software version POSMO A 300W: As of V3.0

SIMOTION <= V4.0 Hardware version POSMO A 75W: J and higher. Software version POSMO A 75W: from V1.3

Hardware version POSMO A 300 W: A and higher. Software version POSMO A 300W: As of V1.4

Restrictions: The functions of software version POSMO A V1.3 are supported.

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see documentation, e.g. creating variables, etc.). An executable application example can also be found in "SIMOTION Utilities & Applications".
"SIMOTION Utilities & Applications" is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myPOSMOA_control : _POSMOA_control; // instance of FB (_POSMOA_control)
myPOSMOA_control(
pkwIn := myPkwIn, // [IN] byte array device (ARRAY[0..7] OF BYTE)
pzdIn := myPzdIn, // [IN] byte array device (ARRAY[0..3] OF BYTE)
enable := myEnable, // [IN] Sets the drive into the errory condition (BOOL)
homing := myHoming, // [IN] Sets the reference point (BOOL)
releaseBrake := myReleaseBrake, // [IN] release the brake (BOOL)
jog1 := myJog1, // [IN] Select jogging 1 (anticlockwise) (BOOL)
jog2 := myJog2, // [IN] Select jogging 2 (clockwise) (BOOL)
jogOverride := myJogOverride, // [IN] speed override in per cent, jogging (INT)
start := myStart, // [IN] 0->1: start traversing block (block number) (BOOL)
singleBlock := mySingleBlock, // [IN] TRUE: single block mode active (BOOL)
enableRdIn := myEnableRdIn, // [IN] TRUE: the following program block can be start (BOOL)
extBlockChange := myExtBlockChange, // [IN] TRUE: the acktive block is canceled and the next block
will started (BOOL)
noStopIntermediate := myNoStopIntermediate, // [IN] =1: No i-Stop  =0: i-Stop (BOOL)
noStop := myNoStop, // [IN] =1: No Stop =0: Stop (BOOL)
resetError := myResetError, // [IN] reset Error from drive (BOOL)
blockNumber := myBlockNumber, // [IN] traversing block numbers 3...27 (BYTE)
veloOverride := myVeloOverride, // [IN] velocityOverride [%] for traversing movement (INT)
setStartInformation := mySetStartInformation, // [IN] Bitcombination, which is transfered to drive
(BYTE)
mdiMode := myMdiMode, // [IN] =1: MDI relative, =0: MDI absolut (BOOL)
mdiVelocity := myMdiVelocity, // [IN] velocity, MDI traversing (0-100%) (INT)
mdiAcceleration := myMdiAcceleration, // [IN] Acceleration, MDI traversing (0-100%) (INT)
mdiPosition := myMdiPosition, // [IN] Target Position, MDI traversing (REAL)
reqControl := myReqControl, // [IN] TRUE: POSMO A is controled by automation device (BOOL)
busy := myBusy, // [IN_OUT] Coordination of the drive FB's (BOOL)
pkwOut := myPkwOut, // [IN_OUT] byte array device (ARRAY[0..7] OF BYTE)
pzdOut := myPzdOut // [IN_OUT] byte array device (ARRAY[0..3] OF BYTE)
);
```

2.3.1.3 _POSMOA_nControl

The _POSMOA_nControl function block can be used to control the connected SIMODRIVE POSMO A in speed setpoint mode (p700=2).

Hardware/software requirements:

SIMOTION V4.1 and higher. Hardware version POSMO A 75W: O and higher. Software version POSMO A 75W: As of V3.0

Hardware version POSMO A 300 W: G and higher. Software version POSMO A 300W: As of V3.0

SIMOTION <=V4.0 Hardware version POSMO A 75W : J and higher. Software version POSMO A 75W: from V1.3

Hardware version POSMO A 300 W: A and higher. Software version POSMO A 300W: As of V1.4

Restrictions: The functions of software version POSMO A V1.3 are supported.

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can also be found in "SIMOTION Utilities & Applications".
"SIMOTION Utilities & Applications" is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myPOSMOA_nControl : _POSMOA_nControl; // instance of FB (_POSMOA_nControl)
myPOSMOA_nControl(
init := myInit, // [IN] TRUE: set drive "ready for switch-on" refer bit1, bit2 control word (BOOL)
enable := myEnable, // [IN] TRUE: set drive "ready to operation" refer bit0 in control word (BOOL)
releaseBrake := myReleaseBrake, // [IN] TRUE: open holding brake (BOOL)
accelTime := myAccelTime, // [IN] duration time for ramp-up and ramp-down (DINT)
releaseRamp := myReleaseRamp, // [IN] TRUE: enable/release ramp (BOOL)
startRamp := myStartRamp, // [IN] TRUE: start ramp (BOOL)
setpoint := mySetpoint, // [IN] setpoint for speed (INT)
releaseSetpoint := myReleaseSetpoint, // [IN] TRUE: enable/release setpoint (BOOL)
resetError := myResetError, // [IN] TRUE: quit drive error (BOOL)
reqControl := myReqControl, // [IN] TRUE: request control from automation device (BOOL)
pkwIn := myPkwIn, // [IN] byte array device (ARRAY[0..7] OF BYTE)
pzdIn := myPzdIn, // [IN] byte array device (ARRAY[0..3] OF BYTE)
busy := myBusy, // [IN_OUT] Coordination of the drive FB's (BOOL)
pkwOut := myPkwOut, // [IN_OUT] byte array device (ARRAY[0..7] OF BYTE)
pzdOut := myPzdOut // [IN_OUT] byte array device (ARRAY[0..3] OF BYTE)
);
```

2.3.1.4 _POSMOA_rwAllParameter

The _POSMOA_rwAllParameter function block can be used to read and write the parameter set of the connected SIMODRIVE POSMO A.

Hardware/software requirements:

SIMOTION V4.1 and higher. Hardware version POSMO A 75W: O and higher. Software version POSMO A 75W: As of V3.0

Hardware version POSMO A 300 W: G and higher. Software version POSMO A 300W: As of V3.0

SIMOTION <=V4.0 Hardware version POSMO A 75W : J and higher. Software version POSMO A 75W: from V1.3

Hardware version POSMO A 300 W: A and higher. Software version POSMO A 300W: As of V1.4

Restrictions: The functions of software version POSMO A V1.3 are supported.

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can also be found in "SIMOTION Utilities & Applications".
"SIMOTION Utilities & Applications" is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myPOSMOA_rwAllParameter : _POSMOA_rwAllParameter; // instance of FB (_POSMOA_rwAllParameter)
myPOSMOA_rwAllParameter(
pkwIn := myPkwIn, // [IN] Transfers I/O Inputs of POSMO A to FB (ARRAY[0..7] OF BYTE)
execRead := myExecRead, // [IN] = 0/1: read all parameters from POSMO A (BOOL)
execWrite := myExecWrite, // [IN] = 0/1: transfer all paramters into POSMO A (BOOL)
posMode := myPosMode, // [IN] TRUE : posmo a in positioning mode (BOOL)
pkwOut := myPkwOut, // [IN_OUT] Prepared FB data for I/O variable for the I/O outputs (ARRAY[0..7] OF
BYTE)
allParaPosmoA := myPOSMOA_params, // [IN_OUT] data structure for all parameters of the POSMO A
(Struct_POSMOA_params)
busy := myBusy // [IN_OUT] Coordination of the drive FBs (BOOL)
);

// **********************
// Struct_POSMOA_params
// **********************
// myPOSMOA_params : Struct_POSMOA_params;
myPOSMOA_params.p1 := 0.0; // [IN]     Linear/rotary axis
myPOSMOA_params.p2 := 10.0; // [IN]     Travel per gear revolution
myPOSMOA_params.p3 := 126.5625; // [IN]     Gear reduction factor
myPOSMOA_params.p4 := 0; // [IN]     Unit of measure
myPOSMOA_params.p5 := 0.0; // [IN]     Position at home position
myPOSMOA_params.p6 := -200000.0; // [IN]     Start of software limit switch
myPOSMOA_params.p7 := 200000.0; // [IN]     End of software limit switch
myPOSMOA_params.p8 := 3000.0; // [IN]     Maximum rotation speed
myPOSMOA_params.p9 := 10; // [IN]     Ramp-up time
myPOSMOA_params.p10 := 237; // [IN]     Maximum velocity
myPOSMOA_params.p11 := 2.0; // [IN]     Target area
myPOSMOA_params.p12 := 20000.0; // [IN]     Maximum following error
myPOSMOA_params.p13 := 50; // [IN]     Monitoring time
myPOSMOA_params.p14 := 20000.0; // [IN]     Zero speed area
myPOSMOA_params.p15 := 0.0; // [IN]     Backlash on reversal compensation
myPOSMOA_params.p16 := 9.0; // [IN]     Maximum overcurrent
myPOSMOA_params.p17 := 20; // [IN]     P-gain of speed controller
myPOSMOA_params.p18 := 22; // [IN]     Integral time of speed controller
myPOSMOA_params.p19 := 1.0; // [IN]     KV factor
myPOSMOA_params.p20 := 0.3; // [IN]     Current setpoint smoothing
myPOSMOA_params.p21 := 2.0; // [IN]     Rotation speed setpoint smoothing
myPOSMOA_params.p22 := 1000.0; // [IN]     Maximum acceleration
myPOSMOA_params.p23 := 0; // [IN]     Jerk time constant
myPOSMOA_params.p24 := 100; // [IN]     Override
myPOSMOA_params.p25 := 100; // [IN]     Acceleration override
myPOSMOA_params.p26 := 20; // [IN]     Rotation speed override for jogging
myPOSMOA_params.p27 := 50; // [IN]     Acceleration override for jogging
myPOSMOA_params.p28 := 9.0; // [IN]     Maximum current
myPOSMOA_params.p29 := 12000; // [IN]     Electronics temperature tolerance time
myPOSMOA_params.p30 := 0; // [IN]     Interference suppression
myPOSMOA_params.p31 := 0; // [IN]     Terminal 1 function
myPOSMOA_params.p32 := 0; // [IN]     Terminal 2 function
myPOSMOA_params.p33 := 0; // [IN]     Address for measurement output 1
myPOSMOA_params.p34 := 7; // [IN]     Shift factor for measurement output 1
myPOSMOA_params.p35 := 128; // [IN]     Offset for measurement output 1
myPOSMOA_params.p36 := 0; // [IN]     Address for measurement output 2
myPOSMOA_params.p37 := 0; // [IN]     Shift factor for measurement output 2
myPOSMOA_params.p38 := 128; // [IN]     Offset for measurement output 2
myPOSMOA_params.p39 := 0.0; // [IN]     Position setpoint
```

```
myPOSMOA_params.p40 := 0.0; // [IN]     Actual position value
myPOSMOA_params.p41 := 0.0; // [IN]     Speed setpoint
myPOSMOA_params.p43 := 0.0; // [IN]     Current setpoint
myPOSMOA_params.p48 := 0; // [IN_OUT] Current traversing block number
myPOSMOA_params.p49 := 0; // [IN_OUT] Subsequent block number
myPOSMOA_params.p50 := 0; // [IN]     Velocity setpoint
myPOSMOA_params.p54 := 5; // [IN]     P-gain of speed controller zero speed
myPOSMOA_params.p55 := 0.0; // [IN]     Signal position
myPOSMOA_params.p56 := 0; // [IN]     Operating position
myPOSMOA_params.p57 := 100; // [IN]     P-gain of stop controller zero speed (HW Version F and higher)
myPOSMOA_params.p58 := 100; // [IN]     Holding brake release time
myPOSMOA_params.p59 := 10.0; // [IN]     Holding brake closure speed
myPOSMOA_params.p60 := 400; // [IN]     Holding brake deceleration time
myPOSMOA_params.p61 := 100; // [IN]     Holding brake controller disable time
myPOSMOA_params.p62 := 0.0; // [IN]     Measuring position
myPOSMOA_params.p80 := myP80; // [IN] Traversing blocks 1 to 27 See Table 2-5 (ARRAY[1..28] OF
Array_POSMOA_prgCtrlInfo )
myPOSMOA_params.p81 := [28(0.0)]; // [IN]  Target position for traversing blocks 1 to 27
myPOSMOA_params.p82 := [28(100)]; // [IN]  Velocity or rotation speed for traversing blocks 1 to 27
myPOSMOA_params.p83 := [28(100)]; // [IN]  Acceleration for traversing blocks 1 to 27
myPOSMOA_params.p84 := [28(0)]; // [IN]  Timer value for traversing blocks 1 to 27
myPOSMOA_params.p85 := [28(0.0)]; // [IN]  Signaling position for trav. blocks 1 to 27
myPOSMOA_params.p86 := [28(0)]; // [IN]  SMStart MMStart for trav. blocks 1 to 27
myPOSMOA_params.p87 := [28(0)]; // [IN]  MMStop MMPos for traversing blocks 1 to 27
myPOSMOA_params.p99 := [13,18,23,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]; // [IN] Program management(see
Distributed Positioning Motor on PROFIBUS-DP user manual)
myPOSMOA_params.p100 := 0; // [IN]  Control word simulation
myPOSMOA_params.p101 := [10(0)]; // [IN]  Blocks 1 to 10 of the data structure for the POSMO A
parameters
myPOSMOA_params.p700 := 2; // [IN]  operating mode (1=variable speed drive, 2=positioning mode)
myPOSMOA_params.p701 := 0; // [IN]  telegram substitution (0=deactivated, 1=activated)
myPOSMOA_params.p880 := 4096.0; // [IN]  standardisation speed (speed after gear on setpoint speed of
4096 decimal)
myPOSMOA_params.p1426 := 100.0; // [IN]  tolerance range of actual speed
myPOSMOA_params.p1427 := 0; // [IN]  delay time (ms) for signal "run-up finished"
```

2.3.1.5 _POSMOA_rwParameter

The _POSMOA_rwParameter function block can be used to assign parameters for the connected SIMODRIVE POSMO A. The functions are as follows: reading parameters, writing parameters, loading factory settings, saving parameters.

Hardware/software requirements:

SIMOTION V4.1 and higher. Hardware version POSMO A 75W: O and higher. Software version POSMO A 75W: As of V3.0

Hardware version POSMO A 300 W: G and higher. Software version POSMO A 300W: As of V3.0

SIMOTION <=V4.0 Hardware version POSMO A 75W : J and higher. Software version POSMO A 75W: from V1.3

Hardware version POSMO A 300 W: A and higher. Software version POSMO A 300W: As of V1.4

Restrictions: The functions of software version POSMO A V1.3 are supported.

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can also be found in "SIMOTION Utilities & Applications".
"SIMOTION Utilities & Applications" is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myPOSMOA_rwParameter : _POSMOA_rwParameter; // instance of FB (_POSMOA_rwParameter)
myPOSMOA_rwParameter(
pkwIn := myPkwIn, // [IN] byte array device (ARRAY[0..7] OF BYTE)
enableRead := myEnableRead, // [IN] =1: cyclically read parameters =0/1 read parameters once (BOOL)
execWrite := myExecWrite, // [IN] =0/1: write into parameters (BOOL)
paraNumber := myParaNumber, // [IN] Parameter number which is to be read or write into (UINT)
subindex := mySubindex, // [IN] Subindex = set to 0 for parameters without subindex (USINT)
setDefault := mySetDefault, // [IN] =0/1: load factory default to drive (BOOL)
saveParameter := mySaveParameter, // [IN] =0/1: save parameter (BOOL)
busy := myBusy, // [IN_OUT] Coordination of the drive FBÂ´s (BOOL)
pkwOut := myPkwOut, // [IN_OUT] byte array device (ARRAY[0..7] OF BYTE)
realValue := myRealValue, // [IN_OUT] write: -> value to be written read: -> read value (REAL)
dintValue := myDintValue // [IN_OUT] write: -> value to be written read: -> read value (DINT)
);
```

2.3.1.6 _readDriveFaults

This function enables the current fault buffer entry in the drive to be read.

Note

- Up to 16 parameter jobs can be executed in parallel (the term parameter job also includes the functions _readDrive(Multi)ParameterDescription, _readDrive(Multi)Parameter, and _writeDrive(Multi)Parameter).
- Only one parameter job is permitted per planned DP station. A new job to the same DP station is rejected with the error 16#FFFF81C7.
- If the system functions _readRecord and _writeRecord are used in addition to the parameter jobs, you must ensure that they are not to be executed simultaneously by a DP station. Reason: The same communication services are used.
- If a parameter job (only asynchronous calls) is no longer to be processed, then it must be aborted with the ABORT_CURRENT_COMMAND transition condition. Otherwise no new job can be issued.

Parameters can be read out with the following data sets:

- Data set 47 is always read out for PROFIBUS (external or integrated), regardless of whether the function is called with a valid (0>=doId<=254) or invalid 'doId' (doId=255). If the passed 'doID' is invalid, it will be determined by the system itself. This requires the specification of the logical address of the axis.
- Two data sets are available for PROFINET:
  - Base Mode Parameter Access - local (data set 0xB02E)

Note

Determining the doID for SIMOTION versions V4.2 and older

For the SIMOTION versions <= V4.2, the automatic determination of 'dold' (if 'dold' is not specified or is invalid) does not take place for all device types in a completely profile-compliant manner. This affects drives which do not support the optional parameter P978 (e.g. SINAMICS G120), where the system in this case determines a 'dold' that is too small by 1 and the wrong DO is therefore accessed. To work around this error, transfer a correct, valid 'dold' during the call-up.

Note

MAP (module access point) address allocation with PROFINET

SIMOTION V4.1 SP2: MAP must always be at subslot 1 (HW Config configuration). The MAP submodule is the DO proxy and hosts the alarm channel and one or more PAPs (parameter access points).

Base Mode Parameter Access - global (data set 0xB02F)

This data set is used if a valid 'dold' (0>=doId<=254) is entered. Any valid MAP or address can be specified, as assignment only takes place on the basis of the 'dold'.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual
- or in the SIMOTION online help.

Return value: StructRetReadDriveFaults

StructRetReadDriveFaults
Return parameter for _readDriveFaults

Structure	Name	Data type	Unit
functionResult	Total job return value	DINT	-
parameterResult	Parameter-specific return value	DINT	-
dataLength	Length of data read	UDINT	-
data	Data read from the fault buffer	ARRAY [0..7] OF UINT	-

Description of the return values:

The return value consists of 'functionResult', 'parameterResult', 'dataLength', and 'data[8]'.

'functionResult'

Reports the status of the total job plus errors during the transfer and internal errors.

16#00000000	Reading of current fault buffer entry okay, Job completed with positive result. Job okay.
16#00007001	Must be repeated in the next program cycle. Initial call, initiation of reading of current fault buffer entry okay (only when command is issued asynchronously).
16#00007002	Must be repeated in the next program cycle. Intermediate call, reading of current fault buffer entry is still active (only when command is issued asynchronously).
16#00007003	Reading the current fault buffer entry aborted.
16#FFFF8190	Internal error, job aborted. Specified logical base address invalid: No assignment is available in SDBs, or there is no base address.

16#FFFF8191	Internal error, job aborted. The _readDriveFaults function cannot reach the specified logical base address.
16#FFFF8192	Internal error, job aborted. Error in response identifier.
16#FFFF819D	Internal error, job aborted. An I slave/I device interface is unable to issue a parameter job to the higher-level master/controller.
16#FFFF819E	Internal error, job aborted. Attempt to abort a non-active function.
16#FFFF819F	Internal error, job aborted. Function not executable.
16#FFFF81C3	Internal error, can be repeated in the next program cycle. Required resources are presently occupied: - in the _readDriveFaults function - In the module
16#FFFF81C5	Internal error, can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF81C7	Internal error, can be repeated in the next program cycle. Another parameter job has already been issued to the DP station. - based on the user program - from a system-internal component (e.g. TO axis executes parameter comparison after station recovery)
16#FFFF81CF	Internal error, can be repeated in the next program cycle. Another parameter job call is currently active under this 'commandId'.
16#FFFF8290	Corresponds to the error message 16#FFFF8190.
16#FFFF8291	Corresponds to the error message 16#FFFF8191.
16#FFFF82A2	Error during data set transfer, job aborted. Error in Layer2: - Station failure - Timeout
16#FFFF82A3	Error during data set transfer, job aborted. Error in user interface/user: - Protocol error - Station failure - Timeout
16#FFFF82A4-A7	Error during data set transfer, job aborted. Error numbers according to IEC 61158 reserved - Query the cause at the module's manufacturer
16#FFFF82A8	Error during data set transfer, job aborted. Error because of version conflict.
16#FFFF82A9	Error during data set transfer, job aborted. Function not supported by the module.
16#FFFF82AA-AF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82B0	Error during data set transfer, job aborted. Parameter jobs are not supported by the addressed module.
16#FFFF82B2	Error during data set transfer, job aborted. Module reports access to an invalid slot/subslot.
16#FFFF82B3	Error during data set transfer, job aborted. Module reports type conflict.

16#FFFF82B4	Error during data set transfer, job aborted. Module reports access to an invalid area.
16#FFFF82B5	Error during data set transfer, can be repeated in the next program cycle. The system function cannot be executed due to an internal operating state of the module.
16#FFFF82B6	Error during data set transfer, job aborted. Module rejects access.
16#FFFF82B7	Error during data set transfer, job aborted. The job could not be dispatched due to a job telegram error.
16#FFFF82B8	Error during data set transfer, job aborted. Module reports an invalid parameter.
16#FFFF82B9	Error during data set transfer, job aborted. Module reports an invalid type.
16#FFFF82BA-BF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82C0	Error during data set transfer, can be repeated in the next program cycle. The module lists the data set, but the module does not yet have any read data.
16#FFFF82C2	Error during data set transfer, command can be repeated immediately. The module is currently executing the maximum possible jobs for one CPU.
16#FFFF82C3	Error during data set transfer, command can be repeated immediately. Required resources are presently occupied: - In the _readRecord function - In the module
16#FFFF82C4	Error during data set transfer, command can be repeated immediately. Communication errors: - Parity error - SW Ready not set - Error in block length administration - Checksum error on CPU side - Checksum error on module side
16#FFFF82C5	Error during data set transfer, Can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF82C6	Error during data set transfer, can be repeated in the next program cycle. Data set transfer has been aborted due to priority class abort (restart or background).

'parameterResult'

Reports parameter-specific errors.

16#00000000	Parameter access completed without errors.
16#FFFF8000	Parameter error, job aborted. The selected fault buffer does not exist.
16#FFFF8011	Parameter error, job aborted. Job cannot be executed due to operating state.
16#FFFF8019	Parameter error, job aborted. Access to a non-existing axis or an invalid drive object.
16#FFFF80xx	Parameter error, job aborted. Manufacturer-specific error: To determine the exact meaning of the error value 'xx' (20-FF), consult the associated manufacturer-dependent device manual.

'dataLength'

reports the length of the data read.

'data[8]'

contains the read data (word array, maximum of 8 words).

Parameters:

iold (optional)

Direction:	Input parameter
Data type:	EnumIoldType
Parameter index:	1
System default:	INPUT

EnumIoldType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of logical base address of the drive. With INPUT, the logical address of the drive is in the input range. With OUTPUT, the logical address of the drive is in the output range. Diagnostic addresses are always of type INPUT.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical base address of the drive.

If the optional 'dold' parameter is also used, any station address (preferably the diagnostic address of the station) can be specified.

With PROFINET, parameter access occurs via the MAP (module access point) of a DO. As an alternative to the logical base address of the drive, specification of the diagnostic address of the associated MAP is recommended.

doId (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	3
System default:	255

Specifies the DO-ID to address a specific DO directly. 0 to 254 are permitted. Under the following conditions, the DO-ID may not be specified, or specified with an invalid value (>254): - Access via the DO-ID is not supported by the DP slave / I/O device (P978 not implemented). - Data set 0xB02F is not supported (PROFINET only). - Access is to be performed via the Parameter Access Point of a DO (PROFINET only).

faultType (optional)

Direction:	Input parameter
Data type:	EnumDriveFaultsType
Parameter index:	4
System default:	DRIVE

EnumDriveFaultsType

DRIVE (0)	Drive-specific fault buffer
USER (1)	User-specific fault buffer
SAFETY (2)	Safety-relevant fault buffer

Specification of the fault buffer type. 'faultType' can be used to assign and read out drive- and user-specific parameters. DRIVE reads out the drive-specific fault buffer (parameter 947). USER reads out the user-specific fault buffer (parameter 945). Fault codes of the drive (P947) can be assigned user-specific faults.

SAFETY reads out the fault buffer (parameter 9747) relevant for the safety functionality. The reason for the safety fault can also be read via drive parameter r9749.

Note

The fault buffer in P9747 relevant for safety is reserved for SINAMICS modules. Other modules could have saved different content in P9747.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	5

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies the command transition. With IMMEDIATELY, the transition takes place immediately after the command is issued (asynchronous call). With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	6

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND.

2.3.1.7 _readDriveMultiParameter

This function enables up to 39 parameters to be read.

Note

- Up to 16 parameter jobs can be executed in parallel (the term parameter job also includes the functions _readDriveParameter,_readDrive(Multi)ParameterDescription, _writeDrive(Multi)Parameter, and _readDriveFaults).
- Only one parameter job is permitted per planned DP station. A new job to the same DP station is rejected with the error 16#FFFF81C7.
- If the system functions _readRecord and _writeRecord are used in addition to the parameter jobs, you must ensure that they are not to be executed simultaneously by a DP station. Reason: The same communication services are used.
- If a parameter job (only asynchronous calls) is no longer to be processed, then it must be aborted with the ABORT_CURRENT_COMMAND transition condition. Otherwise no new job can be issued.

Parameters can be read out with the following data sets:

- Data set 47 is always read out for PROFIBUS (external or integrated), regardless of whether the function is called with a valid (0>=doId<=254) or invalid 'doId' (doId=255). If the passed 'doID' is invalid, it will be determined by the system itself. This requires the specification of the logical address of the axis.
- Two data sets are available for PROFINET:
  - Base Mode Parameter Access - local (data set 0xB02E)

Note

Determining the doID for SIMOTION versions V4.2 and older

For the SIMOTION versions <= V4.2, the automatic determination of 'dold' (if 'dold' is not specified or is invalid) does not take place for all device types in a completely profile-compliant manner. This affects drives which do not support the optional parameter P978 (e.g. SINAMICS G120), where the system in this case determines a 'dold' that is too small by 1 and the wrong DO is therefore accessed. To work around this error, transfer a correct, valid 'dold' during the call-up.

Note

MAP (module access point) address allocation with PROFINET

SIMOTION V4.1 SP2: MAP must always be at subslot 1 (HW Config configuration). The MAP submodule is the DO proxy and hosts the alarm channel and one or more PAPs (parameter access points).

Base Mode Parameter Access - global (data set 0xB02F)

This data set is used if a valid 'dold' (0>=doId<=254) is entered. Any valid MAP or address can be specified, as assignment only takes place on the basis of the 'dold'.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual
- or in the SIMOTION online help.

Return value: StructRetReadDriveMultiParameter

StructRetReadDriveMultiParameter
Return parameter for _readDriveMultiParameter

Structure	Name	Data type	Unit
functionResult	Total job return value	DINT	-
parameterResultErrorExistent	A parameter-specific fault is present in the return value	BOOL	-
parameterResult	Parameter-specific return value	ARRAY [0..38] OF DINT	-
dataType	Data type of the read parameter	ARRAY [0..38] OF UDINT	-
dataLength	Length of data read	ARRAY [0..38] OF UDINT	-
data	Data read from the drive (big endian)	ARRAY [0..233] OF BYTE	-

Description of the return values:

The return value consists of 'functionResult', 'parameterResultErrorExistent', 'parameterResult[39]', 'dataType[39]', 'dataLength[39]' and 'data[233]'.

'functionResult'

Reports the status of the total job plus errors during the transfer and internal errors.

16#00000000	Reading of parameters okay, job completed with positive result. Job okay.
16#00007001	Must be repeated in the next program cycle. Initial call, initiation of reading of parameters okay (only when command is issued asynchronously).

16#00007002	Must be repeated in the next program cycle. Intermediate call, reading of parameter description is still active (only when command is issued asynchronously).
16#00007003	Reading of parameters aborted.
16#FFFF8190	Internal error, job aborted. Specified logical base address invalid: No assignment is available in SDBs, or there is no base address.
16#FFFF8191	Internal error, job aborted. The _readDriveMultiParameter function cannot reach the specified logical base address.
16#FFFF8192	Internal error, job aborted. Error in response identifier.
16#FFFF8193	Internal error, job aborted. The number of parameters to be read is not permissible.
16#FFFF819D	Internal error, job aborted. An I slave/I device interface is unable to issue a parameter job to the higher-level master/controller.
16#FFFF819E	Internal error, job aborted. Attempt to abort a non-active function.
16#FFFF819F	Internal error, job aborted. Function not executable.
16#FFFF81C3	Internal error, can be repeated in the next program cycle. Required resources are presently occupied: - In the _readDriveMultiParameter function - In the module
16#FFFF81C5	Internal error, can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF81C7	Internal error, can be repeated in the next program cycle. Another parameter job has already been issued to the DP station. - based on the user program - from a system-internal component (e.g. TO axis executes parameter comparison after station recovery)
16#FFFF81CF	Internal error, can be repeated in the next program cycle. Another parameter job call is currently active under this 'commandId'.
16#FFFF8290	Corresponds to the error message 16#FFFF8190.
16#FFFF8291	Corresponds to the error message 16#FFFF8191.
16#FFFF82A2	Error during data set transfer, job aborted. Error in Layer2: - Station failure - Timeout
16#FFFF82A3	Error during data set transfer, job aborted. Error in user interface/user: - Protocol error - Station failure - Timeout
16#FFFF82A4-A7	Error during data set transfer, job aborted. Error numbers according to IEC 61158 reserved - Query the cause at the module's manufacturer
16#FFFF82A8	Error during data set transfer, job aborted. Error because of version conflict.

16#FFFF82A9	Error during data set transfer, job aborted. Function not supported by the module.
16#FFFF82AA-AF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82B0	Error during data set transfer, job aborted. Parameter jobs are not supported by the addressed module.
16#FFFF82B2	Error during data set transfer, job aborted. Module reports access to an invalid slot/subslot.
16#FFFF82B3	Error during data set transfer, job aborted. Module reports type conflict.
16#FFFF82B4	Error during data set transfer, job aborted. Module reports access to an invalid area.
16#FFFF82B5	Error during data set transfer, can be repeated in the next program cycle. The system function cannot be executed due to an internal operating state of the module.
16#FFFF82B6	Error during data set transfer, job aborted. Module rejects access.
16#FFFF82B7	Error during data set transfer, job aborted. The job could not be dispatched due to a job error.
16#FFFF82B8	Error during data set transfer, job aborted. Module reports an invalid parameter.
16#FFFF82B9	Error during data set transfer, job aborted. Module reports an invalid type.
16#FFFF82BA-BF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82C0	Error during data set transfer, can be repeated in the next program cycle. The module lists the data set, but the module does not yet have any read data.
16#FFFF82C2	Error during data set transfer, command can be repeated immediately. The module is currently executing the maximum possible jobs for one CPU.
16#FFFF82C3	Error during data set transfer, command can be repeated immediately. Required resources are presently occupied: - In the _readRecord function - In the module
16#FFFF82C4	Error during data set transfer, command can be repeated immediately. Communication errors: - Parity error - SW Ready not set - Error in block length administration - Checksum error on CPU side - Checksum error on module side
16#FFFF82C5	Error during data set transfer, Can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF82C6	Error during data set transfer, can be repeated in the next program cycle. Data set transfer has been aborted due to priority class abort (restart or background).

'parameterResultErrorExistent'

Returns TRUE when an error is reported in at least one 'parameterResult'.

'parameterResult'

Reports parameter-specific errors.

16#00000000	Parameter access completed without errors.
16#FFFF8000	Parameter error, job aborted. Access to a non-existent parameter.
16#FFFF8003	Parameter error, job aborted. Access to a non-existent subindex.
16#FFFF8004	Parameter error, job aborted. Access with subindex to non-indexed parameter (not an array parameter).
16#FFFF800B	Parameter error, job aborted. Change access without parameter change rights.
16#FFFF8011	Parameter error, job aborted. Job cannot be executed due to operating state.
16#FFFF8015	Parameter error, job aborted. The length of the current response exceeds the maximum transferable length. The following errors can also be the cause - The DP station provides fewer parameters than requested - The DP station does not support multi-parameter jobs
16#FFFF8016	Parameter error, job aborted. Illegal or unsupported value for attribute, number of elements, parameter number, or subindex, or a combination of these.
16#FFFF8017	Parameter error, job aborted. Illegal or unsupported parameter format.
16#FFFF8019	Parameter error, job aborted. Access to a non-existing axis or an invalid drive object.
16#FFFF80xx	Parameter error, job aborted. Manufacturer-specific error: To determine the exact meaning of the error value 'xx' (20-FF), consult the associated manufacturer-dependent device manual.

'dataType[39]'

returns the data type for each parameter.

Note

Further information on the structure of a data type can be found:

- In the SIMOTION Communication System Manual or
- At www.profibus.com, in the documentation "Profile Guidelines, Part 2: Data Types, Programming Languages and Platforms".

'dataLength[39]'

reports the length of the data read for each parameter.

'data[233]'

contains the read data (byte array, maximum of 233 bytes).

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of logical base address of the drive. With INPUT, the logical address of the drive is in the input range. With OUTPUT, the logical address of the drive is in the output range. Diagnostic addresses are always of type INPUT.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical base address of the drive.

If the optional 'doId' parameter is also used, any station address (preferably the diagnostic address of the station) can be specified.

With PROFINET, parameter access occurs via the MAP (module access point) of a DO. As an alternative to the logical base address of the drive, specification of the diagnostic address of the associated MAP is recommended.

doId (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	3
System default:	255

Specifies the DO-ID to address a specific DO directly. 0 to 254 are permitted.

The DO-ID can be unspecified or specified as invalid (>254) under the following conditions:

- Access via the DO-ID is not supported by the DP slave/IO device (P978 is not implemented).
- Data set 0xB02F is not supported (PROFINET only).
- Access is to occur via the parameter access point of a DO (PROFINET only).

numberOfParameters

Direction:	Input parameter
Data type:	UDINT
Parameter index:	4

Number of parameters to be read. 1 to 39 permitted.

parameterNumber

Direction:	Input parameter
Data type:	ARRAY [0..38] OF UDINT
Parameter index:	5

Specifies the parameters from which the values are to be read. 1 to 65535 permitted.

numberOfElements (optional)

Direction:	Input parameter
Data type:	ARRAY [0..38] OF UDINT
Parameter index:	6
System default:	[0..38] = 0

Number of elements that are to be read from a parameter. Single parameter -> No need to specify elements. Array parameter -> 1 element under subindex. Array parameter -> 2 to n elements starting from subindex.

subIndex (optional)

Direction:	Input parameter
Data type:	ARRAY [0..38] OF UDINT
Parameter index:	7
System default:	[0..38] = 0

Specifies the subindex that addresses the first array element of an array parameter. 0 to 65535 permitted. Single parameter -> No need to specify elements.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	8

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies the command transition. With IMMEDIATELY, the transition takes place immediately after the command is issued (asynchronous call). With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	9

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND.

2.3.1.8 _readDriveMultiParameterDescription

This function enables up to four parameter descriptions to be read. The interpretation of the parameter descriptions is contained in the PROFIdrive profile.

Note

- Up to 16 parameter jobs can be executed in parallel (the term parameter job also includes the functions _readDriveParameterDescription, _readDrive(Multi)Parameter, _writeDrive(Multi)Parameter, and _readDriveFaults).
- Only one parameter job is permitted per planned DP station. A new job to the same DP station is rejected with the error 16#FFFF81C7.
- If the system functions _readRecord and _writeRecord are used in addition to the parameter jobs, you must ensure that they are not to be executed simultaneously by a DP station. Reason: The same communication services are used.
- If a parameter job (only asynchronous calls) is no longer to be processed, then it must be aborted with the ABORT_CURRENT_COMMAND transition condition. Otherwise no new job can be issued.

Parameters can be read out with the following data sets:

- Data set 47 is always read out for PROFIBUS (external or integrated), regardless of whether the function is called with a valid (0>=doId<=254) or invalid 'doId' (doId=255). If the passed 'doID' is invalid, it will be determined by the system itself. This requires the specification of the logical address of the axis.
- Two data sets are available for PROFINET:
  - Base Mode Parameter Access - local (data set 0xB02E)

Note

Determining the doID for SIMOTION versions V4.2 and older

For the SIMOTION versions <= V4.2, the automatic determination of 'dold' (if 'dold' is not specified or is invalid) does not take place for all device types in a completely profile-compliant manner. This affects drives which do not support the optional parameter P978 (e.g. SINAMICS G120), where the system in this case determines a 'dold' that is too small by 1 and the wrong DO is therefore accessed. To work around this error, transfer a correct, valid 'dold' during the call-up.

Note

MAP (module access point) address allocation with PROFINET

SIMOTION V4.1 SP2: MAP must always be at subslot 1 (HW Config configuration). The MAP submodule is the DO proxy and hosts the alarm channel and one or more PAPs (parameter access points).

Base Mode Parameter Access - global (data set 0xB02F)

This data set is used if a valid 'dold' (0>=doId<=254) is entered. Any valid MAP or address can be specified, as assignment only takes place on the basis of the 'dold'.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual,
- Section PROFIdrive-specific data types,
- Section Rules for SIMOTION _writeDrive.../_readDrive... commands
- or in the SIMOTION online help.

Return value: StructRetReadDriveMultiParameterDescription

StructRetReadDriveMultiParameterDescription
Return parameter for _readDriveMultiParameterDescription

Structure	Name	Data type	Unit
functionResult	Total job return value	DINT	-
parameterResultErrorExistent	A parameter-specific fault is present in the return value	BOOL	-
parameterResult	Parameter-specific return value	ARRAY [0..3] OF DINT	-
dataType	Parameter data type	ARRAY [0..3] OF USINT	-
numberOfElements	Number of parameter elements	ARRAY [0..3] OF UINT	-
factor	Normalization factor	ARRAY [0..3] OF REAL	-
variableIndex	Variable index	ARRAY [0..3] OF USINT	-
conversionIndex	Conversion index	ARRAY [0..3] OF SINT	-
pzdReference	Process data reference parameter	ARRAY [0..3] OF UINT	-
pzdNormalization	Process data normalization	ARRAY [0..3] OF UINT	-

Description of the return values:

The return value consists of 'functionResult', 'parameterResultErrorExistent', 'parameterResult[4]', 'dataType[4]', 'numberOfElements[4]', 'factor[4]', 'variableIndex[4]', 'conversionIndex[4]', 'pzdReference[4]',and 'pzdNormalization[4]'.

'functionResult'

Reports the status of the total job plus errors during the transfer and internal errors.

16#00000000	Read parameter descriptions okay, job completed with positive result. Job okay.
16#00007001	Must be repeated in the next program cycle. Initial call, initiation of reading of parameter descriptions okay (only when command is issued asynchronously).
16#00007002	Must be repeated in the next program cycle. Intermediate call, reading of parameter descriptions is still active (only when command is issued asynchronously).
16#00007003	Reading of parameter descriptions aborted.
16#FFFF8190	Internal error, job aborted. Specified logical base address invalid: No assignment is available in SDBs, or there is no base address.
16#FFFF8191	Internal error, job aborted. The _readDriveMultiParameterDescription function cannot reach the specified logical base address.
16#FFFF8192	Internal error, job aborted. Error in response identifier.
16#FFFF8193	Internal error, job aborted. The number of parameters to be read is not permissible.
16#FFFF819D	Internal error, job aborted. An I slave/I device interface is unable to issue a parameter job to the higher-level master/controller.
16#FFFF819E	Internal error, job aborted. Attempt to abort a non-active function.
16#FFFF819F	Internal error, job aborted. Function not executable.
16#FFFF81C3	Internal error, can be repeated in the next program cycle. Required resources are presently occupied: - In the _readDriveMultiParameterDescription function - In the module
16#FFFF81C5	Internal error, can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF81C7	Internal error, can be repeated in the next program cycle. Another parameter job has already been issued to the DP station. - based on the user program - from a system-internal component (e.g. TO axis executes parameter comparison after station recovery)
16#FFFF81CF	Internal error, can be repeated in the next program cycle. Another parameter job call is currently active under this 'commandId'.
16#FFFF8290	Corresponds to the error message 16#FFFF8190.
16#FFFF8291	Corresponds to the error message 16#FFFF8191.

16#FFFF82A2	Error during data set transfer, job aborted. Error in Layer2: - Station failure - Timeout
16#FFFF82A3	Error during data set transfer, job aborted. Error in user interface/user: - Protocol error - Station failure - Timeout
16#FFFF82A4-A7	Error during data set transfer, job aborted. Error numbers according to IEC 61158 reserved - Query the cause at the module's manufacturer
16#FFFF82A8	Error during data set transfer, job aborted. Error because of version conflict.
16#FFFF82A9	Error during data set transfer, job aborted. Function not supported by the module.
16#FFFF82AA-AF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82B0	Error during data set transfer, job aborted. Parameter jobs are not supported by the addressed module.
16#FFFF82B2	Error during data set transfer, job aborted. Module reports access to an invalid slot/subslot.
16#FFFF82B3	Error during data set transfer, job aborted. Module reports type conflict.
16#FFFF82B4	Error during data set transfer, job aborted. Module reports access to an invalid area.
16#FFFF82B5	Error during data set transfer, can be repeated in the next program cycle. The system function cannot be executed due to an internal operating state of the module.
16#FFFF82B6	Error during data set transfer, job aborted. Module rejects access.
16#FFFF82B7	Error during data set transfer, job aborted. The job could not be dispatched due to a job error.
16#FFFF82B8	Error during data set transfer, job aborted. Module reports an invalid parameter.
16#FFFF82B9	Error during data set transfer, job aborted. Module reports an invalid type.
16#FFFF82BA-BF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82C0	Error during data set transfer, can be repeated in the next program cycle. The module lists the data set, but the module does not yet have any read data.
16#FFFF82C2	Error during data set transfer, command can be repeated immediately. The module is currently executing the maximum possible jobs for one CPU.
16#FFFF82C3	Error during data set transfer, command can be repeated immediately. Required resources are presently occupied: - In the _readRecord function - In the module

16#FFFF82C4	Error during data set transfer, command can be repeated immediately. Communication errors: - Parity error - SW Ready not set - Error in block length administration - Checksum error on CPU side - Checksum error on module side
16#FFFF82C5	Error during data set transfer, Can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF82C6	Error during data set transfer, can be repeated in the next program cycle. Data set transfer has been aborted due to priority class abort (restart or background).

'parameterResultErrorExistent'

Returns TRUE when an error is reported in at least one 'parameterResult'.

'parameterResult'

Reports parameter-specific errors.

16#00000000	Parameter access completed without errors.
16#FFFF8000	Parameter error, job aborted. Access to a non-existent parameter.
16#FFFF8009	Parameter error, job aborted. Access to a non-existent description (parameter does exist).
16#FFFF8011	Parameter error, job aborted. Job cannot be executed due to operating state.
16#FFFF8019	Parameter error, job aborted. Access to a non-existing axis or an invalid drive object.
16#FFFF80xx	Parameter error, job aborted. Manufacturer-specific error: To determine the exact meaning of the error value 'xx' (20-FF), consult the associated manufacturer-dependent device manual.

'dataType'

returns the parameter data type (for the coding, see PROFIdrive profile).

Note

Further information on the structure of a data type can be found:

- In the SIMOTION Communication System Manual or
- At www.profibus.com, in the documentation "Profile Guidelines, Part 2: Data Types, Programming Languages and Platforms".

'numberOfElements'

supplies the number of elements of an array parameter. The string length is specified here for parameters of data type "String".

'factor'

supplies the normalization factor that converts the (internal) value into an (externally) displayable quantity (see PROFIdrive profile).

'variableIndex'

returns the variable index and displays a fixed code for the physical quantity of the parameter value (see PROFIdrive profile).

'conversionIndex'

supplies the fixed code of the conversion index (conversion factor + offset) of a parameter (see PROFIdrive profile).

'pzdReference'

returns the physical reference value for calculating a physical value of a normalized variable (see PROFIdrive profile).

'pzdNormalization'

supplies the bit to which the physical reference value refers (see PROFIdrive profile).

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of logical base address of the drive. With INPUT, the logical address of the drive is in the input range. With OUTPUT, the logical address of the drive is in the output range. Diagnostic addresses are always of type INPUT.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical base address of the drive.

If the optional 'dold' parameter is also used, any station address (preferably the diagnostic address of the station) can be specified.

With PROFINET, parameter access occurs via the MAP (module access point) of a DO. As an alternative to the logical base address of the drive, specification of the diagnostic address of the associated MAP is recommended.

doId (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	3
System default:	255

Specifies the DO-ID to address a specific DO directly. 0 to 254 are permitted. Under the following conditions, the DO-ID may not be specified, or specified with an invalid value (>254): - Access via the DO-ID is not supported by the DP slave / I/O device (P978 not implemented). - Data set 0xB02F is not supported (PROFINET only). - Access is to be performed via the Parameter Access Point of a DO (PROFINET only).

numberOfParameters

Direction:	Input parameter
Data type:	UDINT
Parameter index:	4

Number of parameter descriptions to be read. 1 to 4 permitted.

parameterNumber

Direction:	Input parameter
Data type:	ARRAY [0..3] OF UDINT
Parameter index:	5

Specifies the parameters from which the description is to be read. 1 to 65535 permitted.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	6

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies the command transition. With IMMEDIATELY, the transition takes place immediately after the command is issued (asynchronous call). With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	7

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND.

2.3.1.9 _readDriveParameter

This function enables a drive parameter to be read.

Note

- Up to 16 parameter jobs can be executed in parallel (the term parameter job also includes the functions _readDriveMultiParameter, _readDrive(Multi)ParameterDescription, _writeDrive(Multi)Parameter, and _readDriveFaults).
- Only one parameter job is permitted per planned DP station. A new job to the same DP station is rejected with the error 16#FFFF81C7.
- If the system functions _readRecord and _writeRecord are used in addition to the parameter jobs, you must ensure that they are not to be executed simultaneously by a DP station. Reason: The same communication services are used.
- If a parameter job (only asynchronous calls) is no longer to be processed, then it must be aborted with the ABORT_CURRENT_COMMAND transition condition. Otherwise no new job can be issued.

Parameters can be read out with the following data sets:

- Data set 47 is always read out for PROFIBUS (external or integrated), regardless of whether the function is called with a valid (0>=doId<=254) or invalid 'doId' (doId=255). If the passed 'doId' is invalid, it will be determined by the system itself. This requires the specification of the logical address of the axis.
- Two data sets are available for PROFINET:
  - Base Mode Parameter Access - local (data set 0xB02E)

Note

Determining the doID for SIMOTION versions V4.2 and older

For the SIMOTION versions <= V4.2, the automatic determination of 'dold' (if 'dold' is not specified or is invalid) does not take place for all device types in a completely profile-compliant manner. This affects drives which do not support the optional parameter P978 (e.g. SINAMICS G120), where the system in this case determines a 'dold' that is too small by 1 and the wrong DO is therefore accessed. To work around this error, transfer a correct, valid 'dold' during the call-up.

Note

MAP (module access point) address allocation with PROFINET

SIMOTION V4.1 SP2: MAP must always be at subslot 1 (HW Config configuration). The MAP submodule is the DO proxy and hosts the alarm channel and one or more PAPs (parameter access points).

Base Mode Parameter Access - global (data set 0xB02F)

This data set is used if a valid 'dold' (0>=doId<=254) is entered. Any valid MAP or address can be specified, as assignment only takes place on the basis of the 'dold'.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual,
  Section PROFIdrive-specific data types,
  Section Rules for SIMOTION _writeDrive.../_readDrive... commands
- or in the SIMOTION online help.

Return value: StructRetReadDriveParameter

StructRetReadDriveParameter
Return parameter for _readDriveParameter

Structure	Name	Data type	Unit
functionResult	Total job return value	DINT	-
parameterResult	Parameter-specific return value	DINT	-
dataType	Data type of the read parameter	UDINT	-
dataLength	Length of read data in bytes	UDINT	-
data	Data read by the drive (big endian)	ARRAY [0..233] OF BYTE	-

Description of the return values:

The return value consists of 'functionResult', 'parameterResult', 'dataType', 'dataLength' and 'data[234]'.

'functionResult'

Reports the status of the total job plus errors during the transfer and internal errors.

16#00000000	Reading of parameters okay, job completed with positive result. Job okay.
16#00007001	Must be repeated in the next program cycle. Initial call, initiation of reading of parameter description okay (only when command is issued asynchronously).

16#00007002	Must be repeated in the next program cycle. Intermediate call, reading of parameter description is still active (only when command is issued asynchronously).
16#00007003	Reading of parameters aborted. Job aborted.
16#FFFF8190	Internal error, job aborted. Specified logical base address invalid: No assignment is available in SDBs, or there is no base address.
16#FFFF8191	Internal error, job aborted. The _readDriveParameter function cannot reach the specified logical base address.
16#FFFF8192	Internal error, job aborted. Error in response identifier.
16#FFFF819D	Internal error, job aborted. An I slave/I device interface is unable to issue a parameter job to the higher-level master/controller.
16#FFFF819E	Internal error, job aborted. Attempt to abort a non-active function.
16#FFFF819F	Internal error, job aborted. Function not executable.
16#FFFF81C3	Internal error, can be repeated in the next program cycle. Required resources are presently occupied: - In the _readDriveParameter function - In the module
16#FFFF81C5	Internal error, can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF81C7	Internal error, can be repeated in the next program cycle. Another parameter job has already been issued to the DP station. - based on the user program - from a system-internal component (e.g. TO axis executes parameter comparison after station recovery)
16#FFFF81CF	Internal error, can be repeated in the next program cycle. Another parameter job call is currently active under this 'commandId'.
16#FFFF8290	Corresponds to the error message 16#FFFF8190.
16#FFFF8291	Corresponds to the error message 16#FFFF8191.
16#FFFF82A2	Error during data set transfer, job aborted. Error in Layer2: - Station failure - Timeout
16#FFFF82A3	Error during data set transfer, job aborted. Error in user interface/user: - Protocol error - Station failure - Timeout
16#FFFF82A4-A7	Error during data set transfer, job aborted. Error numbers according to IEC 61158 reserved - Query the cause at the module's manufacturer
16#FFFF82A8	Error during data set transfer, job aborted. Error because of version conflict.
16#FFFF82A9	Error during data set transfer, job aborted. Function not supported by the module.

16#FFFF82AA-AF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82B0	Error during data set transfer, job aborted. Parameter jobs are not supported by the addressed module.
16#FFFF82B2	Error during data set transfer, job aborted. Module reports access to an invalid slot/subslot.
16#FFFF82B3	Error during data set transfer, job aborted. Module reports type conflict.
16#FFFF82B4	Error during data set transfer, job aborted. Module reports access to an invalid area.
16#FFFF82B5	Error during data set transfer, can be repeated in the next program cycle. The system function cannot be executed due to an internal operating state of the module.
16#FFFF82B6	Error during data set transfer, job aborted. Module rejects access.
16#FFFF82B7	Error during data set transfer, job aborted. The job could not be dispatched due to a job error.
16#FFFF82B8	Error during data set transfer, job aborted. Module reports an invalid parameter.
16#FFFF82B9	Error during data set transfer, job aborted. Module reports an invalid type.
16#FFFF82BA-BF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82C0	Error during data set transfer, can be repeated in the next program cycle. The module lists the data set, but the module does not yet have any read data.
16#FFFF82C2	Error during data set transfer, command can be repeated immediately. The module is currently executing the maximum possible jobs for one CPU.
16#FFFF82C3	Error during data set transfer, command can be repeated immediately. Required resources are presently occupied: - In the _readRecord function - In the module
16#FFFF82C4	Error during data set transfer, command can be repeated immediately. Communication errors: - Parity error - SW Ready not set - Error in block length administration - Checksum error on CPU side - Checksum error on module side
16#FFFF82C5	Error during data set transfer, Can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF82C6	Error during data set transfer, can be repeated in the next program cycle. Data set transfer has been aborted due to priority class abort (restart or background).

'parameterResult'

Reports parameter-specific errors.

16#00000000	Parameter access completed without errors.

16#FFFF8000	Parameter error, job aborted. Access to a non-existent parameter.
16#FFFF8003	Parameter error, job aborted. Access to a non-existent subindex.
16#FFFF8004	Parameter error, job aborted. Access with subindex to non-indexed parameter (not an array parameter).
16#FFFF8011	Parameter error, job aborted. Job cannot be executed due to operating state.
16#FFFF8015	Parameter error, job aborted. The length of the current response exceeds the maximum transferable length.
16#FFFF8016	Parameter error, job aborted. Illegal or unsupported value for attribute, number of elements, parameter number, or subindex, or a combination of these.
16#FFFF8019	Parameter error, job aborted. Access to a non-existing axis or an invalid drive object.
16#FFFF80xx	Parameter error, job aborted. Manufacturer-specific error: To determine the exact meaning of the error value 'xx' (20-FF), consult the associated manufacturer-dependent device manual.

'dataType'

returns the data type of the read parameter.

Note

Further information on the structure of a data type can be found:

- In the SIMOTION Communication System Manual or
- At www.profibus.com, in the documentation "Profile Guidelines, Part 2: Data Types, Programming Languages and Platforms".

'dataLength'

reports the length of the data read.

'data[234]'

contains the read data (byte array, maximum of 234 bytes).

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of logical base address of the drive. With INPUT, the logical address of the drive is in the input range. With OUTPUT, the logical address of the drive is in the output range. Diagnostic addresses are always of type INPUT.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical base address of the drive.

If the optional 'doId' parameter is also used, any station address (preferably the diagnostic address of the station) can be specified.

With PROFINET, parameter access occurs via the MAP (module access point) of a DO. As an alternative to the logical base address of the drive, specification of the diagnostic address of the associated MAP is recommended.

doId (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	3
System default:	255

Specifies the DO-ID to address a specific DO directly. 0 to 254 are permitted.

The DO-ID can be unspecified or specified as invalid (>254) under the following conditions:

- Access via the DO-ID is not supported by the DP slave/IO device (P978 is not implemented).
- Data set 0xB02F is not supported (PROFINET only).
- Access is to occur via the parameter access point of a DO (PROFINET only).

parameterNumber

Direction:	Input parameter
Data type:	UDINT
Parameter index:	4

Specifies the parameter from which the values are to be read. 1 to 65535 permitted.

numberOfElements (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	5
System default:	0

Number of elements to be read from a parameter. Single parameter -> No need to specify elements. Array parameter -> 1 element under subindex. Array parameter -> 2 to n elements starting from subindex.

subIndex (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	6
System default:	0

Specifies the subindex that addresses the first array element of an array parameter, setting between 0 and 65535. Single parameter -> No need to specify elements.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	7

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies the command transition. With IMMEDIATELY, the transition takes place immediately after the command is issued (asynchronous call). With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	8

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND.

2.3.1.10 _readDriveParameterDescription

This function enables the descriptive data of a drive parameter to be read out. The interpretation of the parameter description is contained in the PROFIdrive profile.

Note

- Up to 16 parameter jobs can be executed in parallel (the term parameter job also includes the functions _readDriveParameterDescription, _readDrive(Multi)Parameter, _writeDrive(Multi)Parameter, and _readDriveFaults).
- Only one parameter job is permitted per planned DP station. A new job to the same DP station is rejected with the error 16#FFFF81C7.
- If the system functions _readRecord and _writeRecord are used in addition to the parameter jobs, you must ensure that they are not to be executed simultaneously by a DP station. Reason: The same communication services are used.
- If a parameter job (only asynchronous calls) is no longer to be processed, then it must be aborted with the ABORT_CURRENT_COMMAND transition condition. Otherwise no new job can be issued.

Parameters can be read out with the following data sets:

- Data set 47 is always read out for PROFIBUS (external or integrated), regardless of whether the function is called with a valid (0>=doId<=254) or invalid 'doId' (doId=255). If the passed 'doID' is invalid, it will be determined by the system itself. This requires the specification of the logical address of the axis.
- Two data sets are available for PROFINET:
  - Base Mode Parameter Access - local (data set 0xB02E)

Note

MAP (module access point) address allocation with PROFINET

SIMOTION V4.1 SP2: MAP must always be at subslot 1 (HW Config configuration). The MAP submodule is the DO proxy and hosts the alarm channel and one or more PAPs (parameter access points).

Base Mode Parameter Access - global (data set 0xB02F)

This data set is used if a valid 'dold' (0>=doId<=254) is entered. Any valid MAP or address can be specified, as assignment only takes place on the basis of the 'dold'.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual,
- Section PROFIdrive-specific data types,
- Section Rules for SIMOTION _writeDrive.../_readDrive... commands
- or in the SIMOTION online help.

Return value: StructRetReadDriveParameterDescription

StructRetReadDriveParameterDescription
Return parameter for _readDriveParameterDescription

Structure	Name	Data type	Unit
functionResult	Total job return value	DINT	-
parameterResult	Parameter-specific return value	DINT	-
dataType	Parameter data type	USINT	-
numberOfElements	Number of parameter elements	UINT	-
factor	Normalization factor	REAL	-
variableIndex	Variable index	USINT	-
conversionIndex	Conversion index	SINT	-
pzdReference	Process data reference parameter	UINT	-
pzdNormalization	Process data normalization	UINT	-

The return value consists of 'functionResult', 'parameterResult', 'dataType', 'numberOfElements', 'factor', 'variableIndex', 'conversionIndex', 'pzdReference', and 'pzdNormalization'.

'functionResult'

Reports the status of the total job plus errors during the transfer and internal errors.

16#00000000	Reading of parameter description okay, job completed with positive result. Job okay.

16#00007001	Must be repeated in the next program cycle. Initial call, initiation of reading of parameter description okay (only when command is issued asynchronously).
16#00007002	Must be repeated in the next program cycle. Intermediate call, reading of parameter description is still active (only when command is issued asynchronously).
16#00007003	Reading of parameter description aborted. Job aborted.
16#FFFF8190	Internal error, job aborted. Specified logical base address invalid: No assignment is available in SDBs, or there is no base address.
16#FFFF8191	Internal error, job aborted. The _readDriveParameterDescription function cannot reach the specified logical base address.
16#FFFF8192	Internal error, job aborted. Error in response identifier.
16#FFFF819D	Internal error, job aborted. An I slave/I device interface is unable to issue a parameter job to the higher-level master/controller.
16#FFFF819E	Internal error, job aborted. Attempt to abort a non-active function.
16#FFFF819F	Internal error, job aborted. Function not executable.
16#FFFF81C3	Internal error, can be repeated in the next program cycle. Required resources are presently occupied: - In the _readDriveParameterDescription function - In the module
16#FFFF81C5	Internal error, can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF81C7	Internal error, can be repeated in the next program cycle. Another parameter job has already been issued to the DP station. - based on the user program - from a system-internal component (e.g. TO axis executes parameter comparison after station recovery)
16#FFFF81CF	Internal error, can be repeated in the next program cycle. Another parameter job call is currently active under this 'commandId'.
16#FFFF8290	Corresponds to the error message 16#FFFF8190.
16#FFFF8291	Corresponds to the error message 16#FFFF8191.
16#FFFF82A2	Error during data set transfer, job aborted. Error in Layer2: - Station failure - Timeout
16#FFFF82A3	Error during data set transfer, job aborted. Error in user interface/user: - Protocol error - Station failure - Timeout
16#FFFF82A4-A7	Error during data set transfer, job aborted. Error numbers according to IEC 61158 reserved - Query the cause at the module's manufacturer

16#FFFF82A8	Error during data set transfer, job aborted. Error because of version conflict.
16#FFFF82A9	Error during data set transfer, job aborted. Function not supported by the module.
16#FFFF82AA-AF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82B0	Error during data set transfer, job aborted. Parameter jobs are not supported by the addressed module.
16#FFFF82B2	Error during data set transfer, job aborted. Module reports access to an invalid slot/subslot.
16#FFFF82B3	Error during data set transfer, job aborted. Module reports type conflict.
16#FFFF82B4	Error during data set transfer, job aborted. Module reports access to an invalid area.
16#FFFF82B5	Error during data set transfer, can be repeated in the next program cycle. The system function cannot be executed due to an internal operating state of the module.
16#FFFF82B6	Error during data set transfer, job aborted. Module rejects access.
16#FFFF82B7	Error during data set transfer, job aborted. The job could not be dispatched due to a job error.
16#FFFF82B8	Error during data set transfer, job aborted. Module reports an invalid parameter.
16#FFFF82B9	Error during data set transfer, job aborted. Module reports an invalid type.
16#FFFF82BA-BF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82C0	Error during data set transfer, can be repeated in the next program cycle. The module lists the data set, but the module does not yet have any read data.
16#FFFF82C2	Error during data set transfer, command can be repeated immediately. The module is currently executing the maximum possible jobs for one CPU.
16#FFFF82C3	Error during data set transfer, command can be repeated immediately. Required resources are presently occupied: - In the _readRecord function - In the module
16#FFFF82C4	Error during data set transfer, command can be repeated immediately. Communication errors: - Parity error - SW Ready not set - Error in block length administration - Checksum error on CPU side - Checksum error on module side
16#FFFF82C5	Error during data set transfer, Can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF82C6	Error during data set transfer, can be repeated in the next program cycle. Data set transfer has been aborted due to priority class abort (restart or background).

'parameterResult'

Reports parameter-specific errors.

16#00000000	Parameter access completed without errors.
16#FFFF8000	Parameter error, job aborted. Access to a non-existent parameter.
16#FFFF8009	Parameter error, job aborted. Access to a non-existent description (parameter does exist).
16#FFFF8011	Parameter error, job aborted. Job cannot be executed due to operating state.
16#FFFF8019	Parameter error, job aborted. Access to a non-existing axis or an invalid drive object.
16#FFFF80xx	Parameter error, job aborted. Manufacturer-specific error: To determine the exact meaning of the error value 'xx' (20-FF), consult the associated manufacturer-dependent device manual.

'dataType'

returns the parameter data type (for the coding, see PROFIdrive profile).

Note

Further information on the structure of a data type can be found:

- In the SIMOTION Communication System Manual or
- At www.profibus.com, in the documentation "Profile Guidelines Part 2: Data Types, Programming Languages and Platforms".

'numberOfElements'

supplies the number of elements of an array parameter. The string length is specified here for parameters of data type "String length".

'factor'

supplies the normalization factor that converts the (internal) value into an (externally) displayable quantity (see PROFIdrive profile).

'variableIndex'

returns the variable index and displays a fixed code for the physical quantity of the parameter value (see PROFIdrive profile).

'conversionIndex'

supplies the fixed code of the conversion index (conversion factor + offset) of a parameter (see PROFIdrive profile).

'pzdReference'

returns the physical reference value for calculating a physical value of a normalized variable (see PROFIdrive profile).

'pzdNormalization'

supplies the bit to which the physical reference value refers (see PROFIdrive profile).

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of logical base address of the drive. With INPUT, the logical address of the drive is in the input range. With OUTPUT, the logical address of the drive is in the output range. Diagnostic addresses are always of type INPUT.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical base address of the drive.

If the optional 'doId' parameter is also used, any station address (preferably the diagnostic address of the station) can be specified.

With PROFINET, parameter access occurs via the MAP (module access point) of a DO. As an alternative to the logical base address of the drive, specification of the diagnostic address of the associated MAP is recommended.

doId (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	3
System default:	255

Specifies the DO-ID to address a specific DO directly. 0 to 254 are permitted.

The DO-ID can be unspecified or specified as invalid (>254) under the following conditions:

- Access via the DO-ID is not supported by the DP slave/IO device (P978 is not implemented).
- Data set 0xB02F is not supported (PROFINET only).
- Access is to occur via the parameter access point of a DO (PROFINET only).

parameterNumber

Direction:	Input parameter
Data type:	UDINT
Parameter index:	4

Specifies the parameter from which the description is to be read. 1 to 65535 permitted.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	5

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies the command transition. With IMMEDIATELY, the transition takes place immediately after the command is issued (asynchronous call). With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	6

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND.

2.3.1.11 _RWPAR_cyclic

The _RWPAR_cyclic function block can be used to read or write the individual parameters from a drive. For this purpose, the relevant drive must support data exchange via the PIV interface.

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can also be found in "SIMOTION Utilities & Applications".
"SIMOTION Utilities & Applications" is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myRWPAR_cyclic : _RWPAR_cyclic; // instance of FB (_RWPAR_cyclic)
myRWPAR_cyclic(
periIn := myPeriIn, // [IN] byte array device (ARRAY[0..7] OF BYTE)
slotNumber := mySlotNumber, // [IN] number of PKW-Slot (UINT)
execute := myExecute, // [IN] starts order with rising edge (BOOL)
paraNumber := myParaNumber, // [IN] Parameter number which is to be read or write into (UINT)
index := myIndex, // [IN] Subindex = set to 0 for parameters without subindex (USINT)
sendParaValue := mySendParaValue, // [IN] value of parameter to send (DWORD)
jobIdentification := myJobIdentification // [IN] job identification (SINT)
);
```

2.3.1.12 _writeDriveMultiParameter

This function enables up to 23 parameters to be written.

Exactly 23 parameters can only be written once every parameter only has a format with the maximum data length = 2 (e.g. WORD). If other formats with larger data lengths are written, the maximum number is reduced. The procedure is the same when reading parameters. A maximum of 39 parameters can be read if all parameters have the data length <= 2.

More information is provided in:

Profile Drive Technology PROFIdrive, Technical Specification for PROFIBUS and PROFINET, Version 4.1, May 2006 - Section 7.6.2.4, see Formulas

Note

- Up to 16 parameter jobs can be executed in parallel (the term parameter job also includes the functions _writeDriveParameter, _readDrive(Multi)ParameterDescription, _readDrive(Multi)Parameter, and _readDriveFaults).
- Only one parameter job is permitted per planned DP station. A new job to the same DP station is rejected with the error 16#FFFF81C7.
- If the system functions _readRecord and _writeRecord are used in addition to the parameter jobs, you must ensure that they are not to be executed simultaneously by a DP station. Reason: The same communication services are being used.
- If a parameter job (only asynchronous calls) is no longer to be processed, then it must be aborted with the ABORT_CURRENT_COMMAND transition condition. Otherwise no new job can be issued.

Parameters can be read out with the following data sets:

- Data set 47 is always read out for PROFIBUS (external or integrated), regardless of whether the function is called with a valid (0>=doId<=254) or invalid 'doId' (doId=255). If the passed 'doId' is invalid, it will be determined by the system itself. This requires the specification of the logical address of the axis.
- Two data sets are available for PROFINET:
  - Base Mode Parameter Access - local (data set 0xB02E)

Note

Determining the doID for SIMOTION versions V4.2 and older

For the SIMOTION versions <= V4.2, the automatic determination of 'dold' (if 'dold' is not specified or is invalid) does not take place for all device types in a completely profile-compliant manner. This affects drives which do not support the optional parameter P978 (e.g. SINAMICS G120), where the system in this case determines a 'dold' that is too small by 1 and the wrong DO is therefore accessed. To work around this error, transfer a correct, valid 'dold' during the call-up.

Note

MAP (module access point) address allocation with PROFINET

SIMOTION V4.1 SP2: MAP must always be at subslot 1 (HW Config configuration). The MAP submodule is the DO proxy and hosts the alarm channel and one or more PAPs (parameter access points).

Base Mode Parameter Access - global (data set 0xB02F)

This data set is used if a valid 'dold' (0>=doId<=254) is entered. Any valid MAP or address can be specified, as assignment only takes place on the basis of the 'dold'.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual,
- Section PROFIdrive-specific data types,
- Section Rules for SIMOTION _writeDrive.../_readDrive... commands
- or in the SIMOTION online help.

Return value: StructRetWriteDriveMultiParameter

StructRetWriteDriveMultiParameter
Return parameter for _writeDriveMultiParameter

Structure	Name	Data type	Unit
functionResult	Total job return value	DINT	-
parameterResultErrorExistent	A parameter-specific fault is present in the return value	BOOL	-
parameterResult	Parameter-specific return value	ARRAY [0..22] OF DINT	-

Description of the return values:

The return value consists of 'functionResult', 'parameterResultErrorExistent' and 'parameterResult[23]'.

'functionResult'

Reports the status of the total job plus errors during the transfer and internal errors.

16#00000000	Writing of parameters okay, job completed with positive result. Job okay.
16#00007001	Must be repeated in the next program cycle. Initial call, initiation of writing of parameters okay (only when command is issued asynchronously).
16#00007002	Must be repeated in the next program cycle. Intermediate call, writing of parameters is still active (only when command is issued asynchronously).
16#00007003	Writing of parameters aborted.

16#FFFF8110	Internal error, job aborted. The sum of the lengths specified by the user exceeds the maximum transferable length.
16#FFFF8111	Internal error, job aborted. The length specified by the user does not match the length calculated from the data type and number of elements in the case of at least one parameter.
16#FFFF8112	Internal error, job aborted. A valid format could not be found for at least one parameter.
16#FFFF8190	Internal error, job aborted. Specified logical base address invalid: No assignment is available in SDBs, or there is no base address.
16#FFFF8191	Internal error, job aborted. The _writeDriveMultiParameter function cannot reach the specified logical base address.
16#FFFF8192	Internal error, job aborted. Error in response identifier.
16#FFFF8193	Internal error, job aborted. The number of parameters to be read is not permissible.
16#FFFF819D	Internal error, job aborted. An I slave/I device interface is unable to issue a parameter job to the higher-level master/controller.
16#FFFF819E	Internal error, job aborted. Attempt to abort a non-active function.
16#FFFF819F	Internal error, job aborted. Function not executable.
16#FFFF81C3	Internal error, can be repeated in the next program cycle. Required resources are presently occupied: - In the _writeDriveMultiParameter function - In the module
16#FFFF81C5	Internal error, can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF81C7	Internal error, can be repeated in the next program cycle. Another parameter job has already been issued to the DP station. - based on the user program - from a system-internal component (e.g. TO axis executes parameter comparison after station recovery)
16#FFFF81CF	Internal error, can be repeated in the next program cycle. Another parameter job call is currently active under this 'commandId'.
16#FFFF8290	Corresponds to the error message 16#FFFF8190.
16#FFFF8291	Corresponds to the error message 16#FFFF8191.
16#FFFF82A2	Error during data set transfer, job aborted. Error in Layer2: - Station failure - Timeout
16#FFFF82A3	Error during data set transfer, job aborted. Error in user interface/user: - Protocol error - Station failure - Timeout

16#FFFF82A4-A7	Error during data set transfer, job aborted. Error numbers according to IEC 61158 reserved - Query the cause at the module's manufacturer
16#FFFF82A8	Error during data set transfer, job aborted. Error because of version conflict.
16#FFFF82A9	Error during data set transfer, job aborted. Function not supported by the module.
16#FFFF82AA-AF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82B0	Error during data set transfer, job aborted. Parameter jobs are not supported by the addressed module.
16#FFFF82B2	Error during data set transfer, job aborted. Module reports access to an invalid slot/subslot.
16#FFFF82B3	Error during data set transfer, job aborted. Module reports type conflict.
16#FFFF82B4	Error during data set transfer, job aborted. Module reports access to an invalid area.
16#FFFF82B5	Error during data set transfer, can be repeated in the next program cycle. The system function cannot be executed due to an internal operating state of the module.
16#FFFF82B6	Error during data set transfer, job aborted. Module rejects access.
16#FFFF82B7	Error during data set transfer, job aborted. The job could not be dispatched due to a job error.
16#FFFF82B8	Error during data set transfer, job aborted. Module reports an invalid parameter.
16#FFFF82B9	Error during data set transfer, job aborted. Module reports an invalid type.
16#FFFF82BA-BF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82C1	Error during data set transfer, command can be repeated immediately. Data of the preceding write job on the module for the same data set has not yet been processed by the module.
16#FFFF82C2	Error during data set transfer, command can be repeated immediately. The module is currently executing the maximum possible jobs for one CPU.
16#FFFF82C3	Error during data set transfer, command can be repeated immediately. Required resources are presently occupied: - In the _writeRecord function - In the module
16#FFFF82C4	Error during data set transfer, command can be repeated immediately. Communication errors: - Parity error - SW Ready not set - Error in block length administration - Checksum error on CPU side - Checksum error on module side
16#FFFF82C5	Error during data set transfer, can be repeated in the next program cycle. Distributed I/O not available.

16#FFFF82C6	Error during data set transfer, can be repeated in the next program cycle. Data set transfer has been aborted due to priority class abort (restart or background).

'parameterResultErrorExistent'

Returns TRUE when an error is reported in at least one 'parameterResult'.

'parameterResult'

Reports parameter-specific errors.

16#00000000	Parameter access completed without errors.
16#FFFF8000	Parameter error, job aborted. Access to a non-existent parameter.
16#FFFF8001	Parameter error, job aborted. Change access to a parameter that cannot be modified.
16#FFFF8002	Parameter error, job aborted. Change access with value outside value limits.
16#FFFF8003	Parameter error, job aborted. Access to a non-existent subindex.
16#FFFF8004	Parameter error, job aborted. Access with subindex to non-indexed parameter (not an array parameter).
16#FFFF8005	Parameter error, job aborted. Modification access with value that does not match the parameter data type.
16#FFFF8006	Parameter error, job aborted. Modification access with value other than 0 where this is not allowed.
16#FFFF800B	Parameter error, job aborted. Change access without parameter change rights.
16#FFFF8011	Parameter error, job aborted. Job cannot be executed due to operating state.
16#FFFF8014	Parameter error, job aborted. Modification access with value that is within value limits, but illegal for some other sustainable reason (parameter with defined individual values).
16#FFFF8016	Parameter error, job aborted. Illegal or unsupported value for attribute, number of elements, parameter number, or subindex, or a combination of these.
16#FFFF8017	Parameter error, job aborted. Illegal format.
16#FFFF8018	Parameter error, job aborted. Number of values of parameter data does not match the number of elements in the parameter address.
16#FFFF8019	Parameter error, job aborted. Access to a non-existing axis or an invalid drive object.
16#FFFF80xx	Parameter error, job aborted. Manufacturer-specific error: To determine the exact meaning of the error value 'xx' (20-FF), consult the associated manufacturer-dependent device manual.

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of logical base address of the drive. With INPUT, the logical address of the drive is in the input range. With OUTPUT, the logical address of the drive is in the output range. Diagnostic addresses are always of type INPUT.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical base address of the drive.

If the optional 'doId' parameter is also used, any station address (preferably the diagnostic address of the station) can be specified.

With PROFINET, parameter access occurs via the MAP (module access point) of a DO. As an alternative to the logical base address of the drive, specification of the diagnostic address of the associated MAP is recommended.

doId (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	3
System default:	255

Specifies the DO-ID to address a specific DO directly. 0 to 254 are permitted.

The DO-ID can be unspecified or specified as invalid (>254) under the following conditions:

- Access via the DO-ID is not supported by the DP slave/IO device (P978 is not implemented).
- Data set 0xB02F is not supported (PROFINET only).
- Access is to occur via the parameter access point of a DO (PROFINET only).

numberOfParameters

Direction:	Input parameter
Data type:	UDINT
Parameter index:	4

Number of parameters to be written. 1 to 23 permitted.

parameterNumber

Direction:	Input parameter
Data type:	ARRAY [0..22] OF UDINT
Parameter index:	5

Specifies the parameters from which the values are to be changed. 1 to 65535 permitted.

numberOfElements (optional)

Direction:	Input parameter
Data type:	ARRAY [0..22] OF UDINT
Parameter index:	6
System default:	[0..22] = 0

Number of elements to be modified from a parameter. Single parameter -> No need to specify elements. Array parameter -> 1 element under subindex. Array parameter -> 2 to n elements starting from subindex.

subIndex (optional)

Direction:	Input parameter
Data type:	ARRAY [0..22] OF UDINT
Parameter index:	7
System default:	[0..22] = 0

Specifies the subindex that addresses the first array element of an array parameter. 0 to 65535 permitted. Single parameter -> No need to specify elements.

dataType (optional)

Direction:	Input parameter
Data type:	ARRAY [0..22] OF UDINT
Parameter index:	8
System default:	[0..22] = 0

Specifies the data type of the parameter to be written.

Note

Further information on the structure of a data type can be found:

- In the SIMOTION Communication System Manual or
- At www.profibus.com, in the documentation "Profile Guidelines, Part 2: Data Types, Programming Languages and Platforms".

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	9

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies the command transition. With IMMEDIATELY, the transition takes place immediately after the command is issued (asynchronous call). With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	10

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND.

dataLength

Direction:	Input parameter
Data type:	ARRAY [0..22] OF UDINT
Parameter index:	12

Specifies the length of the data to be written (maximum 228 bytes in total).

data

Direction:	Input parameter
Data type:	ARRAY [0..227] OF BYTE
Parameter index:	13

Source area for the user data to be written. The user data must be stored in big endian format.

2.3.1.13 _writeDriveParameter

This function enables a drive parameter to be written.

Note

- Up to 16 parameter jobs can be executed in parallel (the term parameter job also includes the functions _writeDriveMultiParameter, _readDrive(Multi)ParameterDescription, _readDrive(Multi)Parameter and _readDriveFaults).
- Only one parameter job is permitted per planned DP station. A new job to the same DP station is rejected with the error 16#FFFF81C7.
- If the system functions _readRecord and _writeRecord are used in addition to the parameter jobs, you must ensure that they are not to be executed simultaneously by a DP station. Reason: The same communication services are used.
- If a parameter job (only asynchronous calls) is no longer to be processed, then it must be aborted with the ABORT_CURRENT_COMMAND transition condition. Otherwise no new job can be issued.

Parameters can be read out with the following data sets:

- Data set 47 is always read out for PROFIBUS (external or integrated), regardless of whether the function is called with a valid (0>=doId<=254) or invalid 'doId' (doId=255). If the passed 'doId' is invalid, it will be determined by the system itself. This requires the specification of the logical address of the axis.
- Two data sets are available for PROFINET:
  - Base Mode Parameter Access - local (data set 0xB02E)

Note

Determining the doID for SIMOTION versions V4.2 and older

For the SIMOTION versions <= V4.2, the automatic determination of 'dold' (if 'dold' is not specified or is invalid) does not take place for all device types in a completely profile-compliant manner. This affects drives which do not support the optional parameter P978 (e.g. SINAMICS G120), where the system in this case determines a 'dold' that is too small by 1 and the wrong DO is therefore accessed. To work around this error, transfer a correct, valid 'dold' during the call-up.

Note

MAP (module access point) address allocation with PROFINET

SIMOTION V4.1 SP2: MAP must always be at subslot 1 (HW Config configuration). The MAP submodule is the DO proxy and hosts the alarm channel and one or more PAPs (parameter access points).

Base Mode Parameter Access - global (data set 0xB02F)

This data set is used if a valid 'dold' (0>=doId<=254) is entered. Any valid MAP or address can be specified, as assignment only takes place on the basis of the 'dold'.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual,
- Section PROFIdrive-specific data types,
- Section Rules for SIMOTION _writeDrive.../_readDrive... commands
- or in the SIMOTION online help.

Return value: StructRetWriteDriveParameter

StructRetWriteDriveParameter
Return parameter for _writeDriveParameter

Structure	Name	Data type	Unit
functionResult	Total job return value	DINT	-
parameterResult	Parameter-specific return value	DINT	-

Description of the return values:

The return value consists of 'functionResult' and 'parameterResult'.

'functionResult'

Reports the status of the total job plus errors during the transfer and internal errors.

16#00000000	Writing of parameters okay, job completed with positive result. Job okay.
16#00007001	Must be repeated in the next program cycle. Initial call, initiation of reading of parameter description okay (only when command is issued asynchronously).
16#00007002	Must be repeated in the next program cycle. Intermediate call, reading of parameter description is still active (only when command is issued asynchronously).
16#00007003	Writing of parameters aborted.
16#FFFF8110	Internal error, job aborted. The length specified by the user exceeds the maximum transferable length.

16#FFFF8111	Internal error, job aborted The length specified by the user does not match the length calculated from the data type and number of elements.
16#FFFF8112	Internal error, job aborted. No valid format could be found.
16#FFFF8190	Internal error, job aborted. Specified logical base address invalid: No assignment is available in SDBs, or there is no base address.
16#FFFF8191	Internal error, job aborted. The _writeDriveParameter function cannot reach the specified logical base address.
16#FFFF8192	Internal error, job aborted. Error in response identifier.
16#FFFF819D	Internal error, job aborted. An I slave/I device interface is unable to issue a parameter job to the higher-level master/controller.
16#FFFF819E	Internal error, job aborted. Attempt to abort a non-active function.
16#FFFF819F	Internal error, job aborted. Function not executable.
16#FFFF81C3	Internal error, can be repeated in the next program cycle. Required resources are presently occupied: - In the _writeDriveParameter function - In the module
16#FFFF81C5	Internal error, can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF81C7	Internal error, can be repeated in the next program cycle. Another parameter job has already been issued to the DP station. - based on the user program - from a system-internal component (e.g. TO axis executes parameter comparison after station recovery)
16#FFFF81CF	Internal error, can be repeated in the next program cycle. Another parameter job call is currently active under this 'commandId'.
16#FFFF8290	Corresponds to the error message 16#FFFF8190.
16#FFFF8291	Corresponds to the error message 16#FFFF8191.
16#FFFF82A2	Error during data set transfer, job aborted. Error in Layer2: - Station failure - Timeout
16#FFFF82A3	Error during data set transfer, job aborted. Error in user interface/user: - Protocol error - Station failure - Timeout
16#FFFF82A4-A7	Error during data set transfer, job aborted. Error numbers according to IEC 61158 reserved - Query the cause at the module's manufacturer
16#FFFF82A8	Error during data set transfer, job aborted. Error because of version conflict.
16#FFFF82A9	Error during data set transfer, job aborted. Function not supported by the module.

16#FFFF82AA-AF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82B0	Error during data set transfer, job aborted. Parameter jobs are not supported by the addressed module.
16#FFFF82B2	Error during data set transfer, job aborted. Module reports access to an invalid slot/subslot.
16#FFFF82B3	Error during data set transfer, job aborted. Module reports type conflict.
16#FFFF82B4	Error during data set transfer, job aborted. Module reports access to an invalid area.
16#FFFF82B5	Error during data set transfer, can be repeated in the next program cycle. The system function cannot be executed due to an internal operating state of the module.
16#FFFF82B6	Error during data set transfer, job aborted. Module rejects access.
16#FFFF82B7	Error during data set transfer, job aborted. The job could not be dispatched due to a job error.
16#FFFF82B8	Error during data set transfer, job aborted. Module reports an invalid parameter.
16#FFFF82B9	Error during data set transfer, job aborted. Module reports an invalid type.
16#FFFF82BA-BF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82C1	Error during data set transfer, command can be repeated immediately. Data of the preceding write job on the module for the same data set has not yet been processed by the module.
16#FFFF82C2	Error during data set transfer, command can be repeated immediately. The module is currently executing the maximum possible jobs for one CPU.
16#FFFF82C3	Error during data set transfer, command can be repeated immediately. Required resources are presently occupied: - In the _writeRecord function - In the module
16#FFFF82C4	Error during data set transfer, command can be repeated immediately. Communication errors: - Parity error - SW Ready not set - Error in block length administration - Checksum error on CPU side - Checksum error on module side
16#FFFF82C5	Error during data set transfer, can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF82C6	Error during data set transfer, can be repeated in the next program cycle. Data set transfer has been aborted due to priority class abort (restart or background).

'parameterResult'

Reports parameter-specific errors.

16#00000000	Parameter access completed without errors.

16#FFFF8000	Parameter error, job aborted. Access to a non-existent parameter.
16#FFFF8001	Parameter error, job aborted. Change access to a parameter that cannot be modified.
16#FFFF8002	Parameter error, job aborted. Change access with value outside value limits.
16#FFFF8003	Parameter error, job aborted. Access to a non-existent subindex.
16#FFFF8004	Parameter error, job aborted. Access with subindex to non-indexed parameter (not an array parameter).
16#FFFF8005	Parameter error, job aborted. Modification access with value that does not match the parameter data type.
16#FFFF8006	Parameter error, job aborted. Modification access with value other than 0 where this is not allowed.
16#FFFF800B	Parameter error, job aborted. Change access without parameter change rights.
16#FFFF8011	Parameter error, job aborted. Job cannot be executed due to operating state.
16#FFFF8014	Parameter error, job aborted. Modification access with value that is within value limits, but illegal for some other sustainable reason (parameter with defined individual values).
16#FFFF8016	Parameter error, job aborted. Illegal or unsupported value for attribute, number of elements, parameter number, or subindex, or a combination of these.
16#FFFF8017	Parameter error, job aborted. Illegal format.
16#FFFF8018	Parameter error, job aborted. Number of values of parameter data does not match the number of elements in the parameter address.
16#FFFF8019	Parameter error, job aborted. Access to a non-existing axis or an invalid drive object.
16#FFFF80xx	Parameter error, job aborted. Manufacturer-specific error: To determine the exact meaning of the error value 'xx' (20-FF), consult the associated manufacturer-dependent device manual.

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of logical base address of the drive. With INPUT, the logical address of the drive is in the input range. With OUTPUT, the logical address of the drive is in the output range. Diagnostic addresses are always of type INPUT.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical base address of the drive.

If the optional 'doId' parameter is also used, any station address (preferably the diagnostic address of the station) can be specified.

With PROFINET, parameter access occurs via the MAP (module access point) of a DO. As an alternative to the logical base address of the drive, specification of the diagnostic address of the associated MAP is recommended.

doId (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	3
System default:	255

Specifies the DO-ID to address a specific DO directly. 0 to 254 are permitted. Under the following conditions, the DO-ID may not be specified, or specified with an invalid value (>254): - Access via the DO-ID is not supported by the DP slave / I/O device (P978 not implemented). - Data set 0xB02F is not supported (PROFINET only). - Access is to be performed via the Parameter Access Point of a DO (PROFINET only).

parameterNumber

Direction:	Input parameter
Data type:	UDINT
Parameter index:	4

Specifies the parameter numbers from which the values are to be written. 1 to 65535 permitted.

numberOfElements (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	5
System default:	0

Number of elements that are to be written from a parameter. Single parameter -> No need to specify elements. Array parameter -> 1 element under subindex. Array parameter -> 2 to n elements starting from subindex.

subIndex (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	6
System default:	0

Specifies the subindex that addresses the first array element of the parameter. 0 to 65535 permitted.

dataType (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	7
System default:	0

Specifies the data type of the parameter to be written.

Note

Further information on the structure of a data type can be found:

- In the SIMOTION Communication System Manual or
- At www.profibus.com, in the documentation "Profile Guidelines, Part 2: Data Types, Programming Languages and Platforms".

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	8

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies the command transition. With IMMEDIATELY, the transition takes place immediately after the command is issued (asynchronous call). With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	9

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND.

dataLength

Direction:	Input parameter
Data type:	UDINT
Parameter index:	11

Specifies the length of the data to be written (maximum 228 bytes).

data

Direction:	Input parameter
Data type:	ARRAY [0..227] OF BYTE
Parameter index:	12

Source area for the user data to be written. The user data must be stored in big endian format.

2.3.2 SIMOVERT

2.3.2.1 _abortAllReadWriteDriveParameterJobs

This function aborts all parameter jobs currently being processed. The term "parameter job" refers to the functions _readDrive(Multi)ParameterDescription, _readDrive(Multi)Parameter, _writeDrive(Multi)Parameter and _readDriveFaults.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual
  Section Deleting _readDrive and _writeDrive jobs
- or in the SIMOTION online help

Return value: DINT

Description of the return values:

16#00000000	Job completed without errors.
16#FFFF819F	Error, job aborted. Function not executable.
16#FFFF81C5	Error, can be repeated in the next program cycle. Function could not be executed. This can happen for the following reasons: - The function _abortAllReadWriteDriveParameterJobs is already being executed. - A parameter job is currently being called from another task.

2.3.2.2 _readDriveFaults

This function enables the current fault buffer entry in the drive to be read.

Note

- Up to 16 parameter jobs can be executed in parallel (the term parameter job also includes the functions _readDrive(Multi)ParameterDescription, _readDrive(Multi)Parameter, and _writeDrive(Multi)Parameter).
- Only one parameter job is permitted per planned DP station. A new job to the same DP station is rejected with the error 16#FFFF81C7.
- If the system functions _readRecord and _writeRecord are used in addition to the parameter jobs, you must ensure that they are not to be executed simultaneously by a DP station. Reason: The same communication services are used.
- If a parameter job (only asynchronous calls) is no longer to be processed, then it must be aborted with the ABORT_CURRENT_COMMAND transition condition. Otherwise no new job can be issued.

Parameters can be read out with the following data sets:

- Data set 47 is always read out for PROFIBUS (external or integrated), regardless of whether the function is called with a valid (0>=doId<=254) or invalid 'doId' (doId=255). If the passed 'doID' is invalid, it will be determined by the system itself. This requires the specification of the logical address of the axis.
- Two data sets are available for PROFINET:
  - Base Mode Parameter Access - local (data set 0xB02E)

Note

Determining the doID for SIMOTION versions V4.2 and older

For the SIMOTION versions <= V4.2, the automatic determination of 'dold' (if 'dold' is not specified or is invalid) does not take place for all device types in a completely profile-compliant manner. This affects drives which do not support the optional parameter P978 (e.g. SINAMICS G120), where the system in this case determines a 'dold' that is too small by 1 and the wrong DO is therefore accessed. To work around this error, transfer a correct, valid 'dold' during the call-up.

Note

MAP (module access point) address allocation with PROFINET

SIMOTION V4.1 SP2: MAP must always be at subslot 1 (HW Config configuration). The MAP submodule is the DO proxy and hosts the alarm channel and one or more PAPs (parameter access points).

Base Mode Parameter Access - global (data set 0xB02F)

This data set is used if a valid 'dold' (0>=doId<=254) is entered. Any valid MAP or address can be specified, as assignment only takes place on the basis of the 'dold'.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual
- or in the SIMOTION online help.

Return value: StructRetReadDriveFaults

StructRetReadDriveFaults
Return parameter for _readDriveFaults

Structure	Name	Data type	Unit
functionResult	Total job return value	DINT	-
parameterResult	Parameter-specific return value	DINT	-
dataLength	Length of data read	UDINT	-
data	Data read from the fault buffer	ARRAY [0..7] OF UINT	-

Description of the return values:

The return value consists of 'functionResult', 'parameterResult', 'dataLength', and 'data[8]'.

'functionResult'

Reports the status of the total job plus errors during the transfer and internal errors.

16#00000000	Reading of current fault buffer entry okay, Job completed with positive result. Job okay.
16#00007001	Must be repeated in the next program cycle. Initial call, initiation of reading of current fault buffer entry okay (only when command is issued asynchronously).
16#00007002	Must be repeated in the next program cycle. Intermediate call, reading of current fault buffer entry is still active (only when command is issued asynchronously).
16#00007003	Reading the current fault buffer entry aborted.
16#FFFF8190	Internal error, job aborted. Specified logical base address invalid: No assignment is available in SDBs, or there is no base address.

16#FFFF8191	Internal error, job aborted. The _readDriveFaults function cannot reach the specified logical base address.
16#FFFF8192	Internal error, job aborted. Error in response identifier.
16#FFFF819D	Internal error, job aborted. An I slave/I device interface is unable to issue a parameter job to the higher-level master/controller.
16#FFFF819E	Internal error, job aborted. Attempt to abort a non-active function.
16#FFFF819F	Internal error, job aborted. Function not executable.
16#FFFF81C3	Internal error, can be repeated in the next program cycle. Required resources are presently occupied: - in the _readDriveFaults function - In the module
16#FFFF81C5	Internal error, can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF81C7	Internal error, can be repeated in the next program cycle. Another parameter job has already been issued to the DP station. - based on the user program - from a system-internal component (e.g. TO axis executes parameter comparison after station recovery)
16#FFFF81CF	Internal error, can be repeated in the next program cycle. Another parameter job call is currently active under this 'commandId'.
16#FFFF8290	Corresponds to the error message 16#FFFF8190.
16#FFFF8291	Corresponds to the error message 16#FFFF8191.
16#FFFF82A2	Error during data set transfer, job aborted. Error in Layer2: - Station failure - Timeout
16#FFFF82A3	Error during data set transfer, job aborted. Error in user interface/user: - Protocol error - Station failure - Timeout
16#FFFF82A4-A7	Error during data set transfer, job aborted. Error numbers according to IEC 61158 reserved - Query the cause at the module's manufacturer
16#FFFF82A8	Error during data set transfer, job aborted. Error because of version conflict.
16#FFFF82A9	Error during data set transfer, job aborted. Function not supported by the module.
16#FFFF82AA-AF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82B0	Error during data set transfer, job aborted. Parameter jobs are not supported by the addressed module.
16#FFFF82B2	Error during data set transfer, job aborted. Module reports access to an invalid slot/subslot.
16#FFFF82B3	Error during data set transfer, job aborted. Module reports type conflict.

16#FFFF82B4	Error during data set transfer, job aborted. Module reports access to an invalid area.
16#FFFF82B5	Error during data set transfer, can be repeated in the next program cycle. The system function cannot be executed due to an internal operating state of the module.
16#FFFF82B6	Error during data set transfer, job aborted. Module rejects access.
16#FFFF82B7	Error during data set transfer, job aborted. The job could not be dispatched due to a job telegram error.
16#FFFF82B8	Error during data set transfer, job aborted. Module reports an invalid parameter.
16#FFFF82B9	Error during data set transfer, job aborted. Module reports an invalid type.
16#FFFF82BA-BF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82C0	Error during data set transfer, can be repeated in the next program cycle. The module lists the data set, but the module does not yet have any read data.
16#FFFF82C2	Error during data set transfer, command can be repeated immediately. The module is currently executing the maximum possible jobs for one CPU.
16#FFFF82C3	Error during data set transfer, command can be repeated immediately. Required resources are presently occupied: - In the _readRecord function - In the module
16#FFFF82C4	Error during data set transfer, command can be repeated immediately. Communication errors: - Parity error - SW Ready not set - Error in block length administration - Checksum error on CPU side - Checksum error on module side
16#FFFF82C5	Error during data set transfer, Can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF82C6	Error during data set transfer, can be repeated in the next program cycle. Data set transfer has been aborted due to priority class abort (restart or background).

'parameterResult'

Reports parameter-specific errors.

16#00000000	Parameter access completed without errors.
16#FFFF8000	Parameter error, job aborted. The selected fault buffer does not exist.
16#FFFF8011	Parameter error, job aborted. Job cannot be executed due to operating state.
16#FFFF8019	Parameter error, job aborted. Access to a non-existing axis or an invalid drive object.
16#FFFF80xx	Parameter error, job aborted. Manufacturer-specific error: To determine the exact meaning of the error value 'xx' (20-FF), consult the associated manufacturer-dependent device manual.

'dataLength'

reports the length of the data read.

'data[8]'

contains the read data (word array, maximum of 8 words).

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of logical base address of the drive. With INPUT, the logical address of the drive is in the input range. With OUTPUT, the logical address of the drive is in the output range. Diagnostic addresses are always of type INPUT.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical base address of the drive.

If the optional 'doId' parameter is also used, any station address (preferably the diagnostic address of the station) can be specified.

With PROFINET, parameter access occurs via the MAP (module access point) of a DO. As an alternative to the logical base address of the drive, specification of the diagnostic address of the associated MAP is recommended.

doId (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	3
System default:	255

Specifies the DO-ID to address a specific DO directly. 0 to 254 are permitted. Under the following conditions, the DO-ID may not be specified, or specified with an invalid value (>254): - Access via the DO-ID is not supported by the DP slave / I/O device (P978 not implemented). - Data set 0xB02F is not supported (PROFINET only). - Access is to be performed via the Parameter Access Point of a DO (PROFINET only).

faultType (optional)

Direction:	Input parameter
Data type:	EnumDriveFaultsType
Parameter index:	4
System default:	DRIVE

EnumDriveFaultsType

DRIVE (0)	Drive-specific fault buffer
USER (1)	User-specific fault buffer
SAFETY (2)	Safety-relevant fault buffer

Specification of the fault buffer type. 'faultType' can be used to assign and read out drive- and user-specific parameters. DRIVE reads out the drive-specific fault buffer (parameter 947). USER reads out the user-specific fault buffer (parameter 945). Fault codes of the drive (P947) can be assigned user-specific faults.

SAFETY reads out the fault buffer (parameter 9747) relevant for the safety functionality. The reason for the safety fault can also be read via drive parameter r9749.

Note

The fault buffer in P9747 relevant for safety is reserved for SINAMICS modules. Other modules could have saved different content in P9747.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	5

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies the command transition. With IMMEDIATELY, the transition takes place immediately after the command is issued (asynchronous call). With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	6

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND.

2.3.2.3 _readDriveMultiParameter

This function enables up to 39 parameters to be read.

Note

- Up to 16 parameter jobs can be executed in parallel (the term parameter job also includes the functions _readDriveParameter,_readDrive(Multi)ParameterDescription, _writeDrive(Multi)Parameter, and _readDriveFaults).
- Only one parameter job is permitted per planned DP station. A new job to the same DP station is rejected with the error 16#FFFF81C7.
- If the system functions _readRecord and _writeRecord are used in addition to the parameter jobs, you must ensure that they are not to be executed simultaneously by a DP station. Reason: The same communication services are used.
- If a parameter job (only asynchronous calls) is no longer to be processed, then it must be aborted with the ABORT_CURRENT_COMMAND transition condition. Otherwise no new job can be issued.

Parameters can be read out with the following data sets:

- Data set 47 is always read out for PROFIBUS (external or integrated), regardless of whether the function is called with a valid (0>=doId<=254) or invalid 'doId' (doId=255). If the passed 'doID' is invalid, it will be determined by the system itself. This requires the specification of the logical address of the axis.
- Two data sets are available for PROFINET:
  - Base Mode Parameter Access - local (data set 0xB02E)

Note

Determining the doID for SIMOTION versions V4.2 and older

For the SIMOTION versions <= V4.2, the automatic determination of 'dold' (if 'dold' is not specified or is invalid) does not take place for all device types in a completely profile-compliant manner. This affects drives which do not support the optional parameter P978 (e.g. SINAMICS G120), where the system in this case determines a 'dold' that is too small by 1 and the wrong DO is therefore accessed. To work around this error, transfer a correct, valid 'dold' during the call-up.

Note

MAP (module access point) address allocation with PROFINET

SIMOTION V4.1 SP2: MAP must always be at subslot 1 (HW Config configuration). The MAP submodule is the DO proxy and hosts the alarm channel and one or more PAPs (parameter access points).

Base Mode Parameter Access - global (data set 0xB02F)

This data set is used if a valid 'dold' (0>=doId<=254) is entered. Any valid MAP or address can be specified, as assignment only takes place on the basis of the 'dold'.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual
- or in the SIMOTION online help.

Return value: StructRetReadDriveMultiParameter

StructRetReadDriveMultiParameter
Return parameter for _readDriveMultiParameter

Structure	Name	Data type	Unit
functionResult	Total job return value	DINT	-
parameterResultErrorExistent	A parameter-specific fault is present in the return value	BOOL	-
parameterResult	Parameter-specific return value	ARRAY [0..38] OF DINT	-
dataType	Data type of the read parameter	ARRAY [0..38] OF UDINT	-
dataLength	Length of data read	ARRAY [0..38] OF UDINT	-
data	Data read from the drive (big endian)	ARRAY [0..233] OF BYTE	-

Description of the return values:

The return value consists of 'functionResult', 'parameterResultErrorExistent', 'parameterResult[39]', 'dataType[39]', 'dataLength[39]' and 'data[233]'.

'functionResult'

Reports the status of the total job plus errors during the transfer and internal errors.

16#00000000	Reading of parameters okay, job completed with positive result. Job okay.
16#00007001	Must be repeated in the next program cycle. Initial call, initiation of reading of parameters okay (only when command is issued asynchronously).

16#00007002	Must be repeated in the next program cycle. Intermediate call, reading of parameter description is still active (only when command is issued asynchronously).
16#00007003	Reading of parameters aborted.
16#FFFF8190	Internal error, job aborted. Specified logical base address invalid: No assignment is available in SDBs, or there is no base address.
16#FFFF8191	Internal error, job aborted. The _readDriveMultiParameter function cannot reach the specified logical base address.
16#FFFF8192	Internal error, job aborted. Error in response identifier.
16#FFFF8193	Internal error, job aborted. The number of parameters to be read is not permissible.
16#FFFF819D	Internal error, job aborted. An I slave/I device interface is unable to issue a parameter job to the higher-level master/controller.
16#FFFF819E	Internal error, job aborted. Attempt to abort a non-active function.
16#FFFF819F	Internal error, job aborted. Function not executable.
16#FFFF81C3	Internal error, can be repeated in the next program cycle. Required resources are presently occupied: - In the _readDriveMultiParameter function - In the module
16#FFFF81C5	Internal error, can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF81C7	Internal error, can be repeated in the next program cycle. Another parameter job has already been issued to the DP station. - based on the user program - from a system-internal component (e.g. TO axis executes parameter comparison after station recovery)
16#FFFF81CF	Internal error, can be repeated in the next program cycle. Another parameter job call is currently active under this 'commandId'.
16#FFFF8290	Corresponds to the error message 16#FFFF8190.
16#FFFF8291	Corresponds to the error message 16#FFFF8191.
16#FFFF82A2	Error during data set transfer, job aborted. Error in Layer2: - Station failure - Timeout
16#FFFF82A3	Error during data set transfer, job aborted. Error in user interface/user: - Protocol error - Station failure - Timeout
16#FFFF82A4-A7	Error during data set transfer, job aborted. Error numbers according to IEC 61158 reserved - Query the cause at the module's manufacturer
16#FFFF82A8	Error during data set transfer, job aborted. Error because of version conflict.

16#FFFF82A9	Error during data set transfer, job aborted. Function not supported by the module.
16#FFFF82AA-AF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82B0	Error during data set transfer, job aborted. Parameter jobs are not supported by the addressed module.
16#FFFF82B2	Error during data set transfer, job aborted. Module reports access to an invalid slot/subslot.
16#FFFF82B3	Error during data set transfer, job aborted. Module reports type conflict.
16#FFFF82B4	Error during data set transfer, job aborted. Module reports access to an invalid area.
16#FFFF82B5	Error during data set transfer, can be repeated in the next program cycle. The system function cannot be executed due to an internal operating state of the module.
16#FFFF82B6	Error during data set transfer, job aborted. Module rejects access.
16#FFFF82B7	Error during data set transfer, job aborted. The job could not be dispatched due to a job error.
16#FFFF82B8	Error during data set transfer, job aborted. Module reports an invalid parameter.
16#FFFF82B9	Error during data set transfer, job aborted. Module reports an invalid type.
16#FFFF82BA-BF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82C0	Error during data set transfer, can be repeated in the next program cycle. The module lists the data set, but the module does not yet have any read data.
16#FFFF82C2	Error during data set transfer, command can be repeated immediately. The module is currently executing the maximum possible jobs for one CPU.
16#FFFF82C3	Error during data set transfer, command can be repeated immediately. Required resources are presently occupied: - In the _readRecord function - In the module
16#FFFF82C4	Error during data set transfer, command can be repeated immediately. Communication errors: - Parity error - SW Ready not set - Error in block length administration - Checksum error on CPU side - Checksum error on module side
16#FFFF82C5	Error during data set transfer, Can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF82C6	Error during data set transfer, can be repeated in the next program cycle. Data set transfer has been aborted due to priority class abort (restart or background).

'parameterResultErrorExistent'

Returns TRUE when an error is reported in at least one 'parameterResult'.

'parameterResult'

Reports parameter-specific errors.

16#00000000	Parameter access completed without errors.
16#FFFF8000	Parameter error, job aborted. Access to a non-existent parameter.
16#FFFF8003	Parameter error, job aborted. Access to a non-existent subindex.
16#FFFF8004	Parameter error, job aborted. Access with subindex to non-indexed parameter (not an array parameter).
16#FFFF800B	Parameter error, job aborted. Change access without parameter change rights.
16#FFFF8011	Parameter error, job aborted. Job cannot be executed due to operating state.
16#FFFF8015	Parameter error, job aborted. The length of the current response exceeds the maximum transferable length. The following errors can also be the cause - The DP station provides fewer parameters than requested - The DP station does not support multi-parameter jobs
16#FFFF8016	Parameter error, job aborted. Illegal or unsupported value for attribute, number of elements, parameter number, or subindex, or a combination of these.
16#FFFF8017	Parameter error, job aborted. Illegal or unsupported parameter format.
16#FFFF8019	Parameter error, job aborted. Access to a non-existing axis or an invalid drive object.
16#FFFF80xx	Parameter error, job aborted. Manufacturer-specific error: To determine the exact meaning of the error value 'xx' (20-FF), consult the associated manufacturer-dependent device manual.

'dataType[39]'

returns the data type for each parameter.

Note

Further information on the structure of a data type can be found:

- In the SIMOTION Communication System Manual or
- At www.profibus.com, in the documentation "Profile Guidelines, Part 2: Data Types, Programming Languages and Platforms".

'dataLength[39]'

reports the length of the data read for each parameter.

'data[233]'

contains the read data (byte array, maximum of 233 bytes).

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of logical base address of the drive. With INPUT, the logical address of the drive is in the input range. With OUTPUT, the logical address of the drive is in the output range. Diagnostic addresses are always of type INPUT.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical base address of the drive.

If the optional 'doId' parameter is also used, any station address (preferably the diagnostic address of the station) can be specified.

With PROFINET, parameter access occurs via the MAP (module access point) of a DO. As an alternative to the logical base address of the drive, specification of the diagnostic address of the associated MAP is recommended.

doId (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	3
System default:	255

Specifies the DO-ID to address a specific DO directly. 0 to 254 are permitted.

The DO-ID can be unspecified or specified as invalid (>254) under the following conditions:

- Access via the DO-ID is not supported by the DP slave/IO device (P978 is not implemented).
- Data set 0xB02F is not supported (PROFINET only).
- Access is to occur via the parameter access point of a DO (PROFINET only).

numberOfParameters

Direction:	Input parameter
Data type:	UDINT
Parameter index:	4

Number of parameters to be read. 1 to 39 permitted.

parameterNumber

Direction:	Input parameter
Data type:	ARRAY [0..38] OF UDINT
Parameter index:	5

Specifies the parameters from which the values are to be read. 1 to 65535 permitted.

numberOfElements (optional)

Direction:	Input parameter
Data type:	ARRAY [0..38] OF UDINT
Parameter index:	6
System default:	[0..38] = 0

Number of elements that are to be read from a parameter. Single parameter -> No need to specify elements. Array parameter -> 1 element under subindex. Array parameter -> 2 to n elements starting from subindex.

subIndex (optional)

Direction:	Input parameter
Data type:	ARRAY [0..38] OF UDINT
Parameter index:	7
System default:	[0..38] = 0

Specifies the subindex that addresses the first array element of an array parameter. 0 to 65535 permitted. Single parameter -> No need to specify elements.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	8

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies the command transition. With IMMEDIATELY, the transition takes place immediately after the command is issued (asynchronous call). With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	9

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND.

2.3.2.4 _readDriveMultiParameterDescription

This function enables up to four parameter descriptions to be read. The interpretation of the parameter descriptions is contained in the PROFIdrive profile.

Note

- Up to 16 parameter jobs can be executed in parallel (the term parameter job also includes the functions _readDriveParameterDescription, _readDrive(Multi)Parameter, _writeDrive(Multi)Parameter, and _readDriveFaults).
- Only one parameter job is permitted per planned DP station. A new job to the same DP station is rejected with the error 16#FFFF81C7.
- If the system functions _readRecord and _writeRecord are used in addition to the parameter jobs, you must ensure that they are not to be executed simultaneously by a DP station. Reason: The same communication services are used.
- If a parameter job (only asynchronous calls) is no longer to be processed, then it must be aborted with the ABORT_CURRENT_COMMAND transition condition. Otherwise no new job can be issued.

Parameters can be read out with the following data sets:

- Data set 47 is always read out for PROFIBUS (external or integrated), regardless of whether the function is called with a valid (0>=doId<=254) or invalid 'doId' (doId=255). If the passed 'doID' is invalid, it will be determined by the system itself. This requires the specification of the logical address of the axis.
- Two data sets are available for PROFINET:
  - Base Mode Parameter Access - local (data set 0xB02E)

Note

Determining the doID for SIMOTION versions V4.2 and older

For the SIMOTION versions <= V4.2, the automatic determination of 'dold' (if 'dold' is not specified or is invalid) does not take place for all device types in a completely profile-compliant manner. This affects drives which do not support the optional parameter P978 (e.g. SINAMICS G120), where the system in this case determines a 'dold' that is too small by 1 and the wrong DO is therefore accessed. To work around this error, transfer a correct, valid 'dold' during the call-up.

Note

MAP (module access point) address allocation with PROFINET

SIMOTION V4.1 SP2: MAP must always be at subslot 1 (HW Config configuration). The MAP submodule is the DO proxy and hosts the alarm channel and one or more PAPs (parameter access points).

Base Mode Parameter Access - global (data set 0xB02F)

This data set is used if a valid 'dold' (0>=doId<=254) is entered. Any valid MAP or address can be specified, as assignment only takes place on the basis of the 'dold'.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual,
- Section PROFIdrive-specific data types,
- Section Rules for SIMOTION _writeDrive.../_readDrive... commands
- or in the SIMOTION online help.

Return value: StructRetReadDriveMultiParameterDescription

StructRetReadDriveMultiParameterDescription
Return parameter for _readDriveMultiParameterDescription

Structure	Name	Data type	Unit
functionResult	Total job return value	DINT	-
parameterResultErrorExistent	A parameter-specific fault is present in the return value	BOOL	-
parameterResult	Parameter-specific return value	ARRAY [0..3] OF DINT	-
dataType	Parameter data type	ARRAY [0..3] OF USINT	-
numberOfElements	Number of parameter elements	ARRAY [0..3] OF UINT	-
factor	Normalization factor	ARRAY [0..3] OF REAL	-
variableIndex	Variable index	ARRAY [0..3] OF USINT	-
conversionIndex	Conversion index	ARRAY [0..3] OF SINT	-
pzdReference	Process data reference parameter	ARRAY [0..3] OF UINT	-
pzdNormalization	Process data normalization	ARRAY [0..3] OF UINT	-

Description of the return values:

The return value consists of 'functionResult', 'parameterResultErrorExistent', 'parameterResult[4]', 'dataType[4]', 'numberOfElements[4]', 'factor[4]', 'variableIndex[4]', 'conversionIndex[4]', 'pzdReference[4]',and 'pzdNormalization[4]'.

'functionResult'

Reports the status of the total job plus errors during the transfer and internal errors.

16#00000000	Read parameter descriptions okay, job completed with positive result. Job okay.
16#00007001	Must be repeated in the next program cycle. Initial call, initiation of reading of parameter descriptions okay (only when command is issued asynchronously).
16#00007002	Must be repeated in the next program cycle. Intermediate call, reading of parameter descriptions is still active (only when command is issued asynchronously).
16#00007003	Reading of parameter descriptions aborted.
16#FFFF8190	Internal error, job aborted. Specified logical base address invalid: No assignment is available in SDBs, or there is no base address.
16#FFFF8191	Internal error, job aborted. The _readDriveMultiParameterDescription function cannot reach the specified logical base address.
16#FFFF8192	Internal error, job aborted. Error in response identifier.
16#FFFF8193	Internal error, job aborted. The number of parameters to be read is not permissible.
16#FFFF819D	Internal error, job aborted. An I slave/I device interface is unable to issue a parameter job to the higher-level master/controller.
16#FFFF819E	Internal error, job aborted. Attempt to abort a non-active function.
16#FFFF819F	Internal error, job aborted. Function not executable.
16#FFFF81C3	Internal error, can be repeated in the next program cycle. Required resources are presently occupied: - In the _readDriveMultiParameterDescription function - In the module
16#FFFF81C5	Internal error, can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF81C7	Internal error, can be repeated in the next program cycle. Another parameter job has already been issued to the DP station. - based on the user program - from a system-internal component (e.g. TO axis executes parameter comparison after station recovery)
16#FFFF81CF	Internal error, can be repeated in the next program cycle. Another parameter job call is currently active under this 'commandId'.
16#FFFF8290	Corresponds to the error message 16#FFFF8190.
16#FFFF8291	Corresponds to the error message 16#FFFF8191.

16#FFFF82A2	Error during data set transfer, job aborted. Error in Layer2: - Station failure - Timeout
16#FFFF82A3	Error during data set transfer, job aborted. Error in user interface/user: - Protocol error - Station failure - Timeout
16#FFFF82A4-A7	Error during data set transfer, job aborted. Error numbers according to IEC 61158 reserved - Query the cause at the module's manufacturer
16#FFFF82A8	Error during data set transfer, job aborted. Error because of version conflict.
16#FFFF82A9	Error during data set transfer, job aborted. Function not supported by the module.
16#FFFF82AA-AF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82B0	Error during data set transfer, job aborted. Parameter jobs are not supported by the addressed module.
16#FFFF82B2	Error during data set transfer, job aborted. Module reports access to an invalid slot/subslot.
16#FFFF82B3	Error during data set transfer, job aborted. Module reports type conflict.
16#FFFF82B4	Error during data set transfer, job aborted. Module reports access to an invalid area.
16#FFFF82B5	Error during data set transfer, can be repeated in the next program cycle. The system function cannot be executed due to an internal operating state of the module.
16#FFFF82B6	Error during data set transfer, job aborted. Module rejects access.
16#FFFF82B7	Error during data set transfer, job aborted. The job could not be dispatched due to a job error.
16#FFFF82B8	Error during data set transfer, job aborted. Module reports an invalid parameter.
16#FFFF82B9	Error during data set transfer, job aborted. Module reports an invalid type.
16#FFFF82BA-BF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82C0	Error during data set transfer, can be repeated in the next program cycle. The module lists the data set, but the module does not yet have any read data.
16#FFFF82C2	Error during data set transfer, command can be repeated immediately. The module is currently executing the maximum possible jobs for one CPU.
16#FFFF82C3	Error during data set transfer, command can be repeated immediately. Required resources are presently occupied: - In the _readRecord function - In the module

16#FFFF82C4	Error during data set transfer, command can be repeated immediately. Communication errors: - Parity error - SW Ready not set - Error in block length administration - Checksum error on CPU side - Checksum error on module side
16#FFFF82C5	Error during data set transfer, Can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF82C6	Error during data set transfer, can be repeated in the next program cycle. Data set transfer has been aborted due to priority class abort (restart or background).

'parameterResultErrorExistent'

Returns TRUE when an error is reported in at least one 'parameterResult'.

'parameterResult'

Reports parameter-specific errors.

16#00000000	Parameter access completed without errors.
16#FFFF8000	Parameter error, job aborted. Access to a non-existent parameter.
16#FFFF8009	Parameter error, job aborted. Access to a non-existent description (parameter does exist).
16#FFFF8011	Parameter error, job aborted. Job cannot be executed due to operating state.
16#FFFF8019	Parameter error, job aborted. Access to a non-existing axis or an invalid drive object.
16#FFFF80xx	Parameter error, job aborted. Manufacturer-specific error: To determine the exact meaning of the error value 'xx' (20-FF), consult the associated manufacturer-dependent device manual.

'dataType'

returns the parameter data type (for the coding, see PROFIdrive profile).

Note

Further information on the structure of a data type can be found:

- In the SIMOTION Communication System Manual or
- At www.profibus.com, in the documentation "Profile Guidelines, Part 2: Data Types, Programming Languages and Platforms".

'numberOfElements'

supplies the number of elements of an array parameter. The string length is specified here for parameters of data type "String".

'factor'

supplies the normalization factor that converts the (internal) value into an (externally) displayable quantity (see PROFIdrive profile).

'variableIndex'

returns the variable index and displays a fixed code for the physical quantity of the parameter value (see PROFIdrive profile).

'conversionIndex'

supplies the fixed code of the conversion index (conversion factor + offset) of a parameter (see PROFIdrive profile).

'pzdReference'

returns the physical reference value for calculating a physical value of a normalized variable (see PROFIdrive profile).

'pzdNormalization'

supplies the bit to which the physical reference value refers (see PROFIdrive profile).

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of logical base address of the drive. With INPUT, the logical address of the drive is in the input range. With OUTPUT, the logical address of the drive is in the output range. Diagnostic addresses are always of type INPUT.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical base address of the drive.

If the optional 'dold' parameter is also used, any station address (preferably the diagnostic address of the station) can be specified.

With PROFINET, parameter access occurs via the MAP (module access point) of a DO. As an alternative to the logical base address of the drive, specification of the diagnostic address of the associated MAP is recommended.

doId (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	3
System default:	255

Specifies the DO-ID to address a specific DO directly. 0 to 254 are permitted. Under the following conditions, the DO-ID may not be specified, or specified with an invalid value (>254): - Access via the DO-ID is not supported by the DP slave / I/O device (P978 not implemented). - Data set 0xB02F is not supported (PROFINET only). - Access is to be performed via the Parameter Access Point of a DO (PROFINET only).

numberOfParameters

Direction:	Input parameter
Data type:	UDINT
Parameter index:	4

Number of parameter descriptions to be read. 1 to 4 permitted.

parameterNumber

Direction:	Input parameter
Data type:	ARRAY [0..3] OF UDINT
Parameter index:	5

Specifies the parameters from which the description is to be read. 1 to 65535 permitted.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	6

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies the command transition. With IMMEDIATELY, the transition takes place immediately after the command is issued (asynchronous call). With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	7

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND.

2.3.2.5 _readDriveParameter

This function enables a drive parameter to be read.

Note

- Up to 16 parameter jobs can be executed in parallel (the term parameter job also includes the functions _readDriveMultiParameter, _readDrive(Multi)ParameterDescription, _writeDrive(Multi)Parameter, and _readDriveFaults).
- Only one parameter job is permitted per planned DP station. A new job to the same DP station is rejected with the error 16#FFFF81C7.
- If the system functions _readRecord and _writeRecord are used in addition to the parameter jobs, you must ensure that they are not to be executed simultaneously by a DP station. Reason: The same communication services are used.
- If a parameter job (only asynchronous calls) is no longer to be processed, then it must be aborted with the ABORT_CURRENT_COMMAND transition condition. Otherwise no new job can be issued.

Parameters can be read out with the following data sets:

- Data set 47 is always read out for PROFIBUS (external or integrated), regardless of whether the function is called with a valid (0>=doId<=254) or invalid 'doId' (doId=255). If the passed 'doId' is invalid, it will be determined by the system itself. This requires the specification of the logical address of the axis.
- Two data sets are available for PROFINET:
  - Base Mode Parameter Access - local (data set 0xB02E)

Note

Determining the doID for SIMOTION versions V4.2 and older

For the SIMOTION versions <= V4.2, the automatic determination of 'dold' (if 'dold' is not specified or is invalid) does not take place for all device types in a completely profile-compliant manner. This affects drives which do not support the optional parameter P978 (e.g. SINAMICS G120), where the system in this case determines a 'dold' that is too small by 1 and the wrong DO is therefore accessed. To work around this error, transfer a correct, valid 'dold' during the call-up.

Note

MAP (module access point) address allocation with PROFINET

SIMOTION V4.1 SP2: MAP must always be at subslot 1 (HW Config configuration). The MAP submodule is the DO proxy and hosts the alarm channel and one or more PAPs (parameter access points).

Base Mode Parameter Access - global (data set 0xB02F)

This data set is used if a valid 'dold' (0>=doId<=254) is entered. Any valid MAP or address can be specified, as assignment only takes place on the basis of the 'dold'.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual,
  Section PROFIdrive-specific data types,
  Section Rules for SIMOTION _writeDrive.../_readDrive... commands
- or in the SIMOTION online help.

Return value: StructRetReadDriveParameter

StructRetReadDriveParameter
Return parameter for _readDriveParameter

Structure	Name	Data type	Unit
functionResult	Total job return value	DINT	-
parameterResult	Parameter-specific return value	DINT	-
dataType	Data type of the read parameter	UDINT	-
dataLength	Length of read data in bytes	UDINT	-
data	Data read by the drive (big endian)	ARRAY [0..233] OF BYTE	-

Description of the return values:

The return value consists of 'functionResult', 'parameterResult', 'dataType', 'dataLength' and 'data[234]'.

'functionResult'

Reports the status of the total job plus errors during the transfer and internal errors.

16#00000000	Reading of parameters okay, job completed with positive result. Job okay.
16#00007001	Must be repeated in the next program cycle. Initial call, initiation of reading of parameter description okay (only when command is issued asynchronously).

16#00007002	Must be repeated in the next program cycle. Intermediate call, reading of parameter description is still active (only when command is issued asynchronously).
16#00007003	Reading of parameters aborted. Job aborted.
16#FFFF8190	Internal error, job aborted. Specified logical base address invalid: No assignment is available in SDBs, or there is no base address.
16#FFFF8191	Internal error, job aborted. The _readDriveParameter function cannot reach the specified logical base address.
16#FFFF8192	Internal error, job aborted. Error in response identifier.
16#FFFF819D	Internal error, job aborted. An I slave/I device interface is unable to issue a parameter job to the higher-level master/controller.
16#FFFF819E	Internal error, job aborted. Attempt to abort a non-active function.
16#FFFF819F	Internal error, job aborted. Function not executable.
16#FFFF81C3	Internal error, can be repeated in the next program cycle. Required resources are presently occupied: - In the _readDriveParameter function - In the module
16#FFFF81C5	Internal error, can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF81C7	Internal error, can be repeated in the next program cycle. Another parameter job has already been issued to the DP station. - based on the user program - from a system-internal component (e.g. TO axis executes parameter comparison after station recovery)
16#FFFF81CF	Internal error, can be repeated in the next program cycle. Another parameter job call is currently active under this 'commandId'.
16#FFFF8290	Corresponds to the error message 16#FFFF8190.
16#FFFF8291	Corresponds to the error message 16#FFFF8191.
16#FFFF82A2	Error during data set transfer, job aborted. Error in Layer2: - Station failure - Timeout
16#FFFF82A3	Error during data set transfer, job aborted. Error in user interface/user: - Protocol error - Station failure - Timeout
16#FFFF82A4-A7	Error during data set transfer, job aborted. Error numbers according to IEC 61158 reserved - Query the cause at the module's manufacturer
16#FFFF82A8	Error during data set transfer, job aborted. Error because of version conflict.
16#FFFF82A9	Error during data set transfer, job aborted. Function not supported by the module.

16#FFFF82AA-AF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82B0	Error during data set transfer, job aborted. Parameter jobs are not supported by the addressed module.
16#FFFF82B2	Error during data set transfer, job aborted. Module reports access to an invalid slot/subslot.
16#FFFF82B3	Error during data set transfer, job aborted. Module reports type conflict.
16#FFFF82B4	Error during data set transfer, job aborted. Module reports access to an invalid area.
16#FFFF82B5	Error during data set transfer, can be repeated in the next program cycle. The system function cannot be executed due to an internal operating state of the module.
16#FFFF82B6	Error during data set transfer, job aborted. Module rejects access.
16#FFFF82B7	Error during data set transfer, job aborted. The job could not be dispatched due to a job error.
16#FFFF82B8	Error during data set transfer, job aborted. Module reports an invalid parameter.
16#FFFF82B9	Error during data set transfer, job aborted. Module reports an invalid type.
16#FFFF82BA-BF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82C0	Error during data set transfer, can be repeated in the next program cycle. The module lists the data set, but the module does not yet have any read data.
16#FFFF82C2	Error during data set transfer, command can be repeated immediately. The module is currently executing the maximum possible jobs for one CPU.
16#FFFF82C3	Error during data set transfer, command can be repeated immediately. Required resources are presently occupied: - In the _readRecord function - In the module
16#FFFF82C4	Error during data set transfer, command can be repeated immediately. Communication errors: - Parity error - SW Ready not set - Error in block length administration - Checksum error on CPU side - Checksum error on module side
16#FFFF82C5	Error during data set transfer, Can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF82C6	Error during data set transfer, can be repeated in the next program cycle. Data set transfer has been aborted due to priority class abort (restart or background).

'parameterResult'

Reports parameter-specific errors.

16#00000000	Parameter access completed without errors.

16#FFFF8000	Parameter error, job aborted. Access to a non-existent parameter.
16#FFFF8003	Parameter error, job aborted. Access to a non-existent subindex.
16#FFFF8004	Parameter error, job aborted. Access with subindex to non-indexed parameter (not an array parameter).
16#FFFF8011	Parameter error, job aborted. Job cannot be executed due to operating state.
16#FFFF8015	Parameter error, job aborted. The length of the current response exceeds the maximum transferable length.
16#FFFF8016	Parameter error, job aborted. Illegal or unsupported value for attribute, number of elements, parameter number, or subindex, or a combination of these.
16#FFFF8019	Parameter error, job aborted. Access to a non-existing axis or an invalid drive object.
16#FFFF80xx	Parameter error, job aborted. Manufacturer-specific error: To determine the exact meaning of the error value 'xx' (20-FF), consult the associated manufacturer-dependent device manual.

'dataType'

returns the data type of the read parameter.

Note

Further information on the structure of a data type can be found:

- In the SIMOTION Communication System Manual or
- At www.profibus.com, in the documentation "Profile Guidelines, Part 2: Data Types, Programming Languages and Platforms".

'dataLength'

reports the length of the data read.

'data[234]'

contains the read data (byte array, maximum of 234 bytes).

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of logical base address of the drive. With INPUT, the logical address of the drive is in the input range. With OUTPUT, the logical address of the drive is in the output range. Diagnostic addresses are always of type INPUT.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical base address of the drive.

If the optional 'doId' parameter is also used, any station address (preferably the diagnostic address of the station) can be specified.

With PROFINET, parameter access occurs via the MAP (module access point) of a DO. As an alternative to the logical base address of the drive, specification of the diagnostic address of the associated MAP is recommended.

doId (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	3
System default:	255

Specifies the DO-ID to address a specific DO directly. 0 to 254 are permitted.

The DO-ID can be unspecified or specified as invalid (>254) under the following conditions:

- Access via the DO-ID is not supported by the DP slave/IO device (P978 is not implemented).
- Data set 0xB02F is not supported (PROFINET only).
- Access is to occur via the parameter access point of a DO (PROFINET only).

parameterNumber

Direction:	Input parameter
Data type:	UDINT
Parameter index:	4

Specifies the parameter from which the values are to be read. 1 to 65535 permitted.

numberOfElements (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	5
System default:	0

Number of elements to be read from a parameter. Single parameter -> No need to specify elements. Array parameter -> 1 element under subindex. Array parameter -> 2 to n elements starting from subindex.

subIndex (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	6
System default:	0

Specifies the subindex that addresses the first array element of an array parameter, setting between 0 and 65535. Single parameter -> No need to specify elements.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	7

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies the command transition. With IMMEDIATELY, the transition takes place immediately after the command is issued (asynchronous call). With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	8

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND.

2.3.2.6 _readDriveParameterDescription

This function enables the descriptive data of a drive parameter to be read out. The interpretation of the parameter description is contained in the PROFIdrive profile.

Note

- Up to 16 parameter jobs can be executed in parallel (the term parameter job also includes the functions _readDriveParameterDescription, _readDrive(Multi)Parameter, _writeDrive(Multi)Parameter, and _readDriveFaults).
- Only one parameter job is permitted per planned DP station. A new job to the same DP station is rejected with the error 16#FFFF81C7.
- If the system functions _readRecord and _writeRecord are used in addition to the parameter jobs, you must ensure that they are not to be executed simultaneously by a DP station. Reason: The same communication services are used.
- If a parameter job (only asynchronous calls) is no longer to be processed, then it must be aborted with the ABORT_CURRENT_COMMAND transition condition. Otherwise no new job can be issued.

Parameters can be read out with the following data sets:

- Data set 47 is always read out for PROFIBUS (external or integrated), regardless of whether the function is called with a valid (0>=doId<=254) or invalid 'doId' (doId=255). If the passed 'doID' is invalid, it will be determined by the system itself. This requires the specification of the logical address of the axis.
- Two data sets are available for PROFINET:
  - Base Mode Parameter Access - local (data set 0xB02E)

Note

MAP (module access point) address allocation with PROFINET

SIMOTION V4.1 SP2: MAP must always be at subslot 1 (HW Config configuration). The MAP submodule is the DO proxy and hosts the alarm channel and one or more PAPs (parameter access points).

Base Mode Parameter Access - global (data set 0xB02F)

This data set is used if a valid 'dold' (0>=doId<=254) is entered. Any valid MAP or address can be specified, as assignment only takes place on the basis of the 'dold'.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual,
- Section PROFIdrive-specific data types,
- Section Rules for SIMOTION _writeDrive.../_readDrive... commands
- or in the SIMOTION online help.

Return value: StructRetReadDriveParameterDescription

StructRetReadDriveParameterDescription
Return parameter for _readDriveParameterDescription

Structure	Name	Data type	Unit
functionResult	Total job return value	DINT	-
parameterResult	Parameter-specific return value	DINT	-
dataType	Parameter data type	USINT	-
numberOfElements	Number of parameter elements	UINT	-
factor	Normalization factor	REAL	-
variableIndex	Variable index	USINT	-
conversionIndex	Conversion index	SINT	-
pzdReference	Process data reference parameter	UINT	-
pzdNormalization	Process data normalization	UINT	-

The return value consists of 'functionResult', 'parameterResult', 'dataType', 'numberOfElements', 'factor', 'variableIndex', 'conversionIndex', 'pzdReference', and 'pzdNormalization'.

'functionResult'

Reports the status of the total job plus errors during the transfer and internal errors.

16#00000000	Reading of parameter description okay, job completed with positive result. Job okay.

16#00007001	Must be repeated in the next program cycle. Initial call, initiation of reading of parameter description okay (only when command is issued asynchronously).
16#00007002	Must be repeated in the next program cycle. Intermediate call, reading of parameter description is still active (only when command is issued asynchronously).
16#00007003	Reading of parameter description aborted. Job aborted.
16#FFFF8190	Internal error, job aborted. Specified logical base address invalid: No assignment is available in SDBs, or there is no base address.
16#FFFF8191	Internal error, job aborted. The _readDriveParameterDescription function cannot reach the specified logical base address.
16#FFFF8192	Internal error, job aborted. Error in response identifier.
16#FFFF819D	Internal error, job aborted. An I slave/I device interface is unable to issue a parameter job to the higher-level master/controller.
16#FFFF819E	Internal error, job aborted. Attempt to abort a non-active function.
16#FFFF819F	Internal error, job aborted. Function not executable.
16#FFFF81C3	Internal error, can be repeated in the next program cycle. Required resources are presently occupied: - In the _readDriveParameterDescription function - In the module
16#FFFF81C5	Internal error, can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF81C7	Internal error, can be repeated in the next program cycle. Another parameter job has already been issued to the DP station. - based on the user program - from a system-internal component (e.g. TO axis executes parameter comparison after station recovery)
16#FFFF81CF	Internal error, can be repeated in the next program cycle. Another parameter job call is currently active under this 'commandId'.
16#FFFF8290	Corresponds to the error message 16#FFFF8190.
16#FFFF8291	Corresponds to the error message 16#FFFF8191.
16#FFFF82A2	Error during data set transfer, job aborted. Error in Layer2: - Station failure - Timeout
16#FFFF82A3	Error during data set transfer, job aborted. Error in user interface/user: - Protocol error - Station failure - Timeout
16#FFFF82A4-A7	Error during data set transfer, job aborted. Error numbers according to IEC 61158 reserved - Query the cause at the module's manufacturer

16#FFFF82A8	Error during data set transfer, job aborted. Error because of version conflict.
16#FFFF82A9	Error during data set transfer, job aborted. Function not supported by the module.
16#FFFF82AA-AF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82B0	Error during data set transfer, job aborted. Parameter jobs are not supported by the addressed module.
16#FFFF82B2	Error during data set transfer, job aborted. Module reports access to an invalid slot/subslot.
16#FFFF82B3	Error during data set transfer, job aborted. Module reports type conflict.
16#FFFF82B4	Error during data set transfer, job aborted. Module reports access to an invalid area.
16#FFFF82B5	Error during data set transfer, can be repeated in the next program cycle. The system function cannot be executed due to an internal operating state of the module.
16#FFFF82B6	Error during data set transfer, job aborted. Module rejects access.
16#FFFF82B7	Error during data set transfer, job aborted. The job could not be dispatched due to a job error.
16#FFFF82B8	Error during data set transfer, job aborted. Module reports an invalid parameter.
16#FFFF82B9	Error during data set transfer, job aborted. Module reports an invalid type.
16#FFFF82BA-BF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82C0	Error during data set transfer, can be repeated in the next program cycle. The module lists the data set, but the module does not yet have any read data.
16#FFFF82C2	Error during data set transfer, command can be repeated immediately. The module is currently executing the maximum possible jobs for one CPU.
16#FFFF82C3	Error during data set transfer, command can be repeated immediately. Required resources are presently occupied: - In the _readRecord function - In the module
16#FFFF82C4	Error during data set transfer, command can be repeated immediately. Communication errors: - Parity error - SW Ready not set - Error in block length administration - Checksum error on CPU side - Checksum error on module side
16#FFFF82C5	Error during data set transfer, Can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF82C6	Error during data set transfer, can be repeated in the next program cycle. Data set transfer has been aborted due to priority class abort (restart or background).

'parameterResult'

Reports parameter-specific errors.

16#00000000	Parameter access completed without errors.
16#FFFF8000	Parameter error, job aborted. Access to a non-existent parameter.
16#FFFF8009	Parameter error, job aborted. Access to a non-existent description (parameter does exist).
16#FFFF8011	Parameter error, job aborted. Job cannot be executed due to operating state.
16#FFFF8019	Parameter error, job aborted. Access to a non-existing axis or an invalid drive object.
16#FFFF80xx	Parameter error, job aborted. Manufacturer-specific error: To determine the exact meaning of the error value 'xx' (20-FF), consult the associated manufacturer-dependent device manual.

'dataType'

returns the parameter data type (for the coding, see PROFIdrive profile).

Note

Further information on the structure of a data type can be found:

- In the SIMOTION Communication System Manual or
- At www.profibus.com, in the documentation "Profile Guidelines Part 2: Data Types, Programming Languages and Platforms".

'numberOfElements'

supplies the number of elements of an array parameter. The string length is specified here for parameters of data type "String length".

'factor'

supplies the normalization factor that converts the (internal) value into an (externally) displayable quantity (see PROFIdrive profile).

'variableIndex'

returns the variable index and displays a fixed code for the physical quantity of the parameter value (see PROFIdrive profile).

'conversionIndex'

supplies the fixed code of the conversion index (conversion factor + offset) of a parameter (see PROFIdrive profile).

'pzdReference'

returns the physical reference value for calculating a physical value of a normalized variable (see PROFIdrive profile).

'pzdNormalization'

supplies the bit to which the physical reference value refers (see PROFIdrive profile).

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of logical base address of the drive. With INPUT, the logical address of the drive is in the input range. With OUTPUT, the logical address of the drive is in the output range. Diagnostic addresses are always of type INPUT.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical base address of the drive.

If the optional 'doId' parameter is also used, any station address (preferably the diagnostic address of the station) can be specified.

With PROFINET, parameter access occurs via the MAP (module access point) of a DO. As an alternative to the logical base address of the drive, specification of the diagnostic address of the associated MAP is recommended.

doId (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	3
System default:	255

Specifies the DO-ID to address a specific DO directly. 0 to 254 are permitted.

The DO-ID can be unspecified or specified as invalid (>254) under the following conditions:

- Access via the DO-ID is not supported by the DP slave/IO device (P978 is not implemented).
- Data set 0xB02F is not supported (PROFINET only).
- Access is to occur via the parameter access point of a DO (PROFINET only).

parameterNumber

Direction:	Input parameter
Data type:	UDINT
Parameter index:	4

Specifies the parameter from which the description is to be read. 1 to 65535 permitted.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	5

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies the command transition. With IMMEDIATELY, the transition takes place immediately after the command is issued (asynchronous call). With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	6

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND.

2.3.2.7 _RWPAR_cyclic

The _RWPAR_cyclic function block can be used to read or write the individual parameters from a drive. For this purpose, the relevant drive must support data exchange via the PIV interface.

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can also be found in "SIMOTION Utilities & Applications".
"SIMOTION Utilities & Applications" is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myRWPAR_cyclic : _RWPAR_cyclic; // instance of FB (_RWPAR_cyclic)
myRWPAR_cyclic(
periIn := myPeriIn, // [IN] byte array device (ARRAY[0..7] OF BYTE)
slotNumber := mySlotNumber, // [IN] number of PKW-Slot (UINT)
execute := myExecute, // [IN] starts order with rising edge (BOOL)
paraNumber := myParaNumber, // [IN] Parameter number which is to be read or write into (UINT)
index := myIndex, // [IN] Subindex = set to 0 for parameters without subindex (USINT)
sendParaValue := mySendParaValue, // [IN] value of parameter to send (DWORD)
jobIdentification := myJobIdentification // [IN] job identification (SINT)
);
```

2.3.2.8 _writeDriveMultiParameter

This function enables up to 23 parameters to be written.

Exactly 23 parameters can only be written once every parameter only has a format with the maximum data length = 2 (e.g. WORD). If other formats with larger data lengths are written, the maximum number is reduced. The procedure is the same when reading parameters. A maximum of 39 parameters can be read if all parameters have the data length <= 2.

More information is provided in:

Profile Drive Technology PROFIdrive, Technical Specification for PROFIBUS and PROFINET, Version 4.1, May 2006 - Section 7.6.2.4, see Formulas

Note

- Up to 16 parameter jobs can be executed in parallel (the term parameter job also includes the functions _writeDriveParameter, _readDrive(Multi)ParameterDescription, _readDrive(Multi)Parameter, and _readDriveFaults).
- Only one parameter job is permitted per planned DP station. A new job to the same DP station is rejected with the error 16#FFFF81C7.
- If the system functions _readRecord and _writeRecord are used in addition to the parameter jobs, you must ensure that they are not to be executed simultaneously by a DP station. Reason: The same communication services are being used.
- If a parameter job (only asynchronous calls) is no longer to be processed, then it must be aborted with the ABORT_CURRENT_COMMAND transition condition. Otherwise no new job can be issued.

Parameters can be read out with the following data sets:

- Data set 47 is always read out for PROFIBUS (external or integrated), regardless of whether the function is called with a valid (0>=doId<=254) or invalid 'doId' (doId=255). If the passed 'doId' is invalid, it will be determined by the system itself. This requires the specification of the logical address of the axis.
- Two data sets are available for PROFINET:
  - Base Mode Parameter Access - local (data set 0xB02E)

Note

Determining the doID for SIMOTION versions V4.2 and older

For the SIMOTION versions <= V4.2, the automatic determination of 'doId' (if 'doId' is not specified or is invalid) does not take place for all device types in a completely profile-compliant manner. This affects drives which do not support the optional parameter P978 (e.g. SINAMICS G120), where the system in this case determines a 'doId' that is too small by 1 and the wrong DO is therefore accessed. To work around this error, transfer a correct, valid 'doId' during the call-up.

Note

MAP (module access point) address allocation with PROFINET

SIMOTION V4.1 SP2: MAP must always be at subslot 1 (HW Config configuration). The MAP submodule is the DO proxy and hosts the alarm channel and one or more PAPs (parameter access points).

Base Mode Parameter Access - global (data set 0xB02F)

This data set is used if a valid 'dold' (0>=doId<=254) is entered. Any valid MAP or address can be specified, as assignment only takes place on the basis of the 'dold'.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual,
- Section PROFIdrive-specific data types,
- Section Rules for SIMOTION _writeDrive.../_readDrive... commands
- or in the SIMOTION online help.

Return value: StructRetWriteDriveMultiParameter

StructRetWriteDriveMultiParameter
Return parameter for _writeDriveMultiParameter

Structure	Name	Data type	Unit
functionResult	Total job return value	DINT	-
parameterResultErrorExistent	A parameter-specific fault is present in the return value	BOOL	-
parameterResult	Parameter-specific return value	ARRAY [0..22] OF DINT	-

Description of the return values:

The return value consists of 'functionResult', 'parameterResultErrorExistent' and 'parameterResult[23]'.

'functionResult'

Reports the status of the total job plus errors during the transfer and internal errors.

16#00000000	Writing of parameters okay, job completed with positive result. Job okay.
16#00007001	Must be repeated in the next program cycle. Initial call, initiation of writing of parameters okay (only when command is issued asynchronously).
16#00007002	Must be repeated in the next program cycle. Intermediate call, writing of parameters is still active (only when command is issued asynchronously).
16#00007003	Writing of parameters aborted.

16#FFFF8110	Internal error, job aborted. The sum of the lengths specified by the user exceeds the maximum transferable length.
16#FFFF8111	Internal error, job aborted. The length specified by the user does not match the length calculated from the data type and number of elements in the case of at least one parameter.
16#FFFF8112	Internal error, job aborted. A valid format could not be found for at least one parameter.
16#FFFF8190	Internal error, job aborted. Specified logical base address invalid: No assignment is available in SDBs, or there is no base address.
16#FFFF8191	Internal error, job aborted. The _writeDriveMultiParameter function cannot reach the specified logical base address.
16#FFFF8192	Internal error, job aborted. Error in response identifier.
16#FFFF8193	Internal error, job aborted. The number of parameters to be read is not permissible.
16#FFFF819D	Internal error, job aborted. An I slave/I device interface is unable to issue a parameter job to the higher-level master/controller.
16#FFFF819E	Internal error, job aborted. Attempt to abort a non-active function.
16#FFFF819F	Internal error, job aborted. Function not executable.
16#FFFF81C3	Internal error, can be repeated in the next program cycle. Required resources are presently occupied: - In the _writeDriveMultiParameter function - In the module
16#FFFF81C5	Internal error, can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF81C7	Internal error, can be repeated in the next program cycle. Another parameter job has already been issued to the DP station. - based on the user program - from a system-internal component (e.g. TO axis executes parameter comparison after station recovery)
16#FFFF81CF	Internal error, can be repeated in the next program cycle. Another parameter job call is currently active under this 'commandId'.
16#FFFF8290	Corresponds to the error message 16#FFFF8190.
16#FFFF8291	Corresponds to the error message 16#FFFF8191.
16#FFFF82A2	Error during data set transfer, job aborted. Error in Layer2: - Station failure - Timeout
16#FFFF82A3	Error during data set transfer, job aborted. Error in user interface/user: - Protocol error - Station failure - Timeout

16#FFFF82A4-A7	Error during data set transfer, job aborted. Error numbers according to IEC 61158 reserved - Query the cause at the module's manufacturer
16#FFFF82A8	Error during data set transfer, job aborted. Error because of version conflict.
16#FFFF82A9	Error during data set transfer, job aborted. Function not supported by the module.
16#FFFF82AA-AF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82B0	Error during data set transfer, job aborted. Parameter jobs are not supported by the addressed module.
16#FFFF82B2	Error during data set transfer, job aborted. Module reports access to an invalid slot/subslot.
16#FFFF82B3	Error during data set transfer, job aborted. Module reports type conflict.
16#FFFF82B4	Error during data set transfer, job aborted. Module reports access to an invalid area.
16#FFFF82B5	Error during data set transfer, can be repeated in the next program cycle. The system function cannot be executed due to an internal operating state of the module.
16#FFFF82B6	Error during data set transfer, job aborted. Module rejects access.
16#FFFF82B7	Error during data set transfer, job aborted. The job could not be dispatched due to a job error.
16#FFFF82B8	Error during data set transfer, job aborted. Module reports an invalid parameter.
16#FFFF82B9	Error during data set transfer, job aborted. Module reports an invalid type.
16#FFFF82BA-BF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82C1	Error during data set transfer, command can be repeated immediately. Data of the preceding write job on the module for the same data set has not yet been processed by the module.
16#FFFF82C2	Error during data set transfer, command can be repeated immediately. The module is currently executing the maximum possible jobs for one CPU.
16#FFFF82C3	Error during data set transfer, command can be repeated immediately. Required resources are presently occupied: - In the _writeRecord function - In the module
16#FFFF82C4	Error during data set transfer, command can be repeated immediately. Communication errors: - Parity error - SW Ready not set - Error in block length administration - Checksum error on CPU side - Checksum error on module side
16#FFFF82C5	Error during data set transfer, can be repeated in the next program cycle. Distributed I/O not available.

16#FFFF82C6	Error during data set transfer, can be repeated in the next program cycle. Data set transfer has been aborted due to priority class abort (restart or background).

'parameterResultErrorExistent'

Returns TRUE when an error is reported in at least one 'parameterResult'.

'parameterResult'

Reports parameter-specific errors.

16#00000000	Parameter access completed without errors.
16#FFFF8000	Parameter error, job aborted. Access to a non-existent parameter.
16#FFFF8001	Parameter error, job aborted. Change access to a parameter that cannot be modified.
16#FFFF8002	Parameter error, job aborted. Change access with value outside value limits.
16#FFFF8003	Parameter error, job aborted. Access to a non-existent subindex.
16#FFFF8004	Parameter error, job aborted. Access with subindex to non-indexed parameter (not an array parameter).
16#FFFF8005	Parameter error, job aborted. Modification access with value that does not match the parameter data type.
16#FFFF8006	Parameter error, job aborted. Modification access with value other than 0 where this is not allowed.
16#FFFF800B	Parameter error, job aborted. Change access without parameter change rights.
16#FFFF8011	Parameter error, job aborted. Job cannot be executed due to operating state.
16#FFFF8014	Parameter error, job aborted. Modification access with value that is within value limits, but illegal for some other sustainable reason (parameter with defined individual values).
16#FFFF8016	Parameter error, job aborted. Illegal or unsupported value for attribute, number of elements, parameter number, or subindex, or a combination of these.
16#FFFF8017	Parameter error, job aborted. Illegal format.
16#FFFF8018	Parameter error, job aborted. Number of values of parameter data does not match the number of elements in the parameter address.
16#FFFF8019	Parameter error, job aborted. Access to a non-existing axis or an invalid drive object.
16#FFFF80xx	Parameter error, job aborted. Manufacturer-specific error: To determine the exact meaning of the error value 'xx' (20-FF), consult the associated manufacturer-dependent device manual.

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of logical base address of the drive. With INPUT, the logical address of the drive is in the input range. With OUTPUT, the logical address of the drive is in the output range. Diagnostic addresses are always of type INPUT.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical base address of the drive.

If the optional 'doId' parameter is also used, any station address (preferably the diagnostic address of the station) can be specified.

With PROFINET, parameter access occurs via the MAP (module access point) of a DO. As an alternative to the logical base address of the drive, specification of the diagnostic address of the associated MAP is recommended.

doId (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	3
System default:	255

Specifies the DO-ID to address a specific DO directly. 0 to 254 are permitted.

The DO-ID can be unspecified or specified as invalid (>254) under the following conditions:

- Access via the DO-ID is not supported by the DP slave/IO device (P978 is not implemented).
- Data set 0xB02F is not supported (PROFINET only).
- Access is to occur via the parameter access point of a DO (PROFINET only).

numberOfParameters

Direction:	Input parameter
Data type:	UDINT
Parameter index:	4

Number of parameters to be written. 1 to 23 permitted.

parameterNumber

Direction:	Input parameter
Data type:	ARRAY [0..22] OF UDINT
Parameter index:	5

Specifies the parameters from which the values are to be changed. 1 to 65535 permitted.

numberOfElements (optional)

Direction:	Input parameter
Data type:	ARRAY [0..22] OF UDINT
Parameter index:	6
System default:	[0..22] = 0

Number of elements to be modified from a parameter. Single parameter -> No need to specify elements. Array parameter -> 1 element under subindex. Array parameter -> 2 to n elements starting from subindex.

subIndex (optional)

Direction:	Input parameter
Data type:	ARRAY [0..22] OF UDINT
Parameter index:	7
System default:	[0..22] = 0

Specifies the subindex that addresses the first array element of an array parameter. 0 to 65535 permitted. Single parameter -> No need to specify elements.

dataType (optional)

Direction:	Input parameter
Data type:	ARRAY [0..22] OF UDINT
Parameter index:	8
System default:	[0..22] = 0

Specifies the data type of the parameter to be written.

Note

Further information on the structure of a data type can be found:

- In the SIMOTION Communication System Manual or
- At www.profibus.com, in the documentation "Profile Guidelines, Part 2: Data Types, Programming Languages and Platforms".

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	9

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies the command transition. With IMMEDIATELY, the transition takes place immediately after the command is issued (asynchronous call). With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	10

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND.

dataLength

Direction:	Input parameter
Data type:	ARRAY [0..22] OF UDINT
Parameter index:	12

Specifies the length of the data to be written (maximum 228 bytes in total).

data

Direction:	Input parameter
Data type:	ARRAY [0..227] OF BYTE
Parameter index:	13

Source area for the user data to be written. The user data must be stored in big endian format.

2.3.2.9 _writeDriveParameter

This function enables a drive parameter to be written.

Note

- Up to 16 parameter jobs can be executed in parallel (the term parameter job also includes the functions _writeDriveMultiParameter, _readDrive(Multi)ParameterDescription, _readDrive(Multi)Parameter and _readDriveFaults).
- Only one parameter job is permitted per planned DP station. A new job to the same DP station is rejected with the error 16#FFFF81C7.
- If the system functions _readRecord and _writeRecord are used in addition to the parameter jobs, you must ensure that they are not to be executed simultaneously by a DP station. Reason: The same communication services are used.
- If a parameter job (only asynchronous calls) is no longer to be processed, then it must be aborted with the ABORT_CURRENT_COMMAND transition condition. Otherwise no new job can be issued.

Parameters can be read out with the following data sets:

- Data set 47 is always read out for PROFIBUS (external or integrated), regardless of whether the function is called with a valid (0>=doId<=254) or invalid 'doId' (doId=255). If the passed 'doId' is invalid, it will be determined by the system itself. This requires the specification of the logical address of the axis.
- Two data sets are available for PROFINET:
  - Base Mode Parameter Access - local (data set 0xB02E)

Note

Determining the doID for SIMOTION versions V4.2 and older

For the SIMOTION versions <= V4.2, the automatic determination of 'dold' (if 'dold' is not specified or is invalid) does not take place for all device types in a completely profile-compliant manner. This affects drives which do not support the optional parameter P978 (e.g. SINAMICS G120), where the system in this case determines a 'dold' that is too small by 1 and the wrong DO is therefore accessed. To work around this error, transfer a correct, valid 'dold' during the call-up.

Note

MAP (module access point) address allocation with PROFINET

SIMOTION V4.1 SP2: MAP must always be at subslot 1 (HW Config configuration). The MAP submodule is the DO proxy and hosts the alarm channel and one or more PAPs (parameter access points).

Base Mode Parameter Access - global (data set 0xB02F)

This data set is used if a valid 'dold' (0>=doId<=254) is entered. Any valid MAP or address can be specified, as assignment only takes place on the basis of the 'dold'.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual,
- Section PROFIdrive-specific data types,
- Section Rules for SIMOTION _writeDrive.../_readDrive... commands
- or in the SIMOTION online help.

Return value: StructRetWriteDriveParameter

StructRetWriteDriveParameter
Return parameter for _writeDriveParameter

Structure	Name	Data type	Unit
functionResult	Total job return value	DINT	-
parameterResult	Parameter-specific return value	DINT	-

Description of the return values:

The return value consists of 'functionResult' and 'parameterResult'.

'functionResult'

Reports the status of the total job plus errors during the transfer and internal errors.

16#00000000	Writing of parameters okay, job completed with positive result. Job okay.
16#00007001	Must be repeated in the next program cycle. Initial call, initiation of reading of parameter description okay (only when command is issued asynchronously).
16#00007002	Must be repeated in the next program cycle. Intermediate call, reading of parameter description is still active (only when command is issued asynchronously).
16#00007003	Writing of parameters aborted.
16#FFFF8110	Internal error, job aborted. The length specified by the user exceeds the maximum transferable length.

16#FFFF8111	Internal error, job aborted The length specified by the user does not match the length calculated from the data type and number of elements.
16#FFFF8112	Internal error, job aborted. No valid format could be found.
16#FFFF8190	Internal error, job aborted. Specified logical base address invalid: No assignment is available in SDBs, or there is no base address.
16#FFFF8191	Internal error, job aborted. The _writeDriveParameter function cannot reach the specified logical base address.
16#FFFF8192	Internal error, job aborted. Error in response identifier.
16#FFFF819D	Internal error, job aborted. An I slave/I device interface is unable to issue a parameter job to the higher-level master/controller.
16#FFFF819E	Internal error, job aborted. Attempt to abort a non-active function.
16#FFFF819F	Internal error, job aborted. Function not executable.
16#FFFF81C3	Internal error, can be repeated in the next program cycle. Required resources are presently occupied: - In the _writeDriveParameter function - In the module
16#FFFF81C5	Internal error, can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF81C7	Internal error, can be repeated in the next program cycle. Another parameter job has already been issued to the DP station. - based on the user program - from a system-internal component (e.g. TO axis executes parameter comparison after station recovery)
16#FFFF81CF	Internal error, can be repeated in the next program cycle. Another parameter job call is currently active under this 'commandId'.
16#FFFF8290	Corresponds to the error message 16#FFFF8190.
16#FFFF8291	Corresponds to the error message 16#FFFF8191.
16#FFFF82A2	Error during data set transfer, job aborted. Error in Layer2: - Station failure - Timeout
16#FFFF82A3	Error during data set transfer, job aborted. Error in user interface/user: - Protocol error - Station failure - Timeout
16#FFFF82A4-A7	Error during data set transfer, job aborted. Error numbers according to IEC 61158 reserved - Query the cause at the module's manufacturer
16#FFFF82A8	Error during data set transfer, job aborted. Error because of version conflict.
16#FFFF82A9	Error during data set transfer, job aborted. Function not supported by the module.

16#FFFF82AA-AF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82B0	Error during data set transfer, job aborted. Parameter jobs are not supported by the addressed module.
16#FFFF82B2	Error during data set transfer, job aborted. Module reports access to an invalid slot/subslot.
16#FFFF82B3	Error during data set transfer, job aborted. Module reports type conflict.
16#FFFF82B4	Error during data set transfer, job aborted. Module reports access to an invalid area.
16#FFFF82B5	Error during data set transfer, can be repeated in the next program cycle. The system function cannot be executed due to an internal operating state of the module.
16#FFFF82B6	Error during data set transfer, job aborted. Module rejects access.
16#FFFF82B7	Error during data set transfer, job aborted. The job could not be dispatched due to a job error.
16#FFFF82B8	Error during data set transfer, job aborted. Module reports an invalid parameter.
16#FFFF82B9	Error during data set transfer, job aborted. Module reports an invalid type.
16#FFFF82BA-BF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82C1	Error during data set transfer, command can be repeated immediately. Data of the preceding write job on the module for the same data set has not yet been processed by the module.
16#FFFF82C2	Error during data set transfer, command can be repeated immediately. The module is currently executing the maximum possible jobs for one CPU.
16#FFFF82C3	Error during data set transfer, command can be repeated immediately. Required resources are presently occupied: - In the _writeRecord function - In the module
16#FFFF82C4	Error during data set transfer, command can be repeated immediately. Communication errors: - Parity error - SW Ready not set - Error in block length administration - Checksum error on CPU side - Checksum error on module side
16#FFFF82C5	Error during data set transfer, can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF82C6	Error during data set transfer, can be repeated in the next program cycle. Data set transfer has been aborted due to priority class abort (restart or background).

'parameterResult'

Reports parameter-specific errors.

16#00000000	Parameter access completed without errors.

16#FFFF8000	Parameter error, job aborted. Access to a non-existent parameter.
16#FFFF8001	Parameter error, job aborted. Change access to a parameter that cannot be modified.
16#FFFF8002	Parameter error, job aborted. Change access with value outside value limits.
16#FFFF8003	Parameter error, job aborted. Access to a non-existent subindex.
16#FFFF8004	Parameter error, job aborted. Access with subindex to non-indexed parameter (not an array parameter).
16#FFFF8005	Parameter error, job aborted. Modification access with value that does not match the parameter data type.
16#FFFF8006	Parameter error, job aborted. Modification access with value other than 0 where this is not allowed.
16#FFFF800B	Parameter error, job aborted. Change access without parameter change rights.
16#FFFF8011	Parameter error, job aborted. Job cannot be executed due to operating state.
16#FFFF8014	Parameter error, job aborted. Modification access with value that is within value limits, but illegal for some other sustainable reason (parameter with defined individual values).
16#FFFF8016	Parameter error, job aborted. Illegal or unsupported value for attribute, number of elements, parameter number, or subindex, or a combination of these.
16#FFFF8017	Parameter error, job aborted. Illegal format.
16#FFFF8018	Parameter error, job aborted. Number of values of parameter data does not match the number of elements in the parameter address.
16#FFFF8019	Parameter error, job aborted. Access to a non-existing axis or an invalid drive object.
16#FFFF80xx	Parameter error, job aborted. Manufacturer-specific error: To determine the exact meaning of the error value 'xx' (20-FF), consult the associated manufacturer-dependent device manual.

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of logical base address of the drive. With INPUT, the logical address of the drive is in the input range. With OUTPUT, the logical address of the drive is in the output range. Diagnostic addresses are always of type INPUT.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical base address of the drive.

If the optional 'doId' parameter is also used, any station address (preferably the diagnostic address of the station) can be specified.

With PROFINET, parameter access occurs via the MAP (module access point) of a DO. As an alternative to the logical base address of the drive, specification of the diagnostic address of the associated MAP is recommended.

doId (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	3
System default:	255

Specifies the DO-ID to address a specific DO directly. 0 to 254 are permitted. Under the following conditions, the DO-ID may not be specified, or specified with an invalid value (>254): - Access via the DO-ID is not supported by the DP slave / I/O device (P978 not implemented). - Data set 0xB02F is not supported (PROFINET only). - Access is to be performed via the Parameter Access Point of a DO (PROFINET only).

parameterNumber

Direction:	Input parameter
Data type:	UDINT
Parameter index:	4

Specifies the parameter numbers from which the values are to be written. 1 to 65535 permitted.

numberOfElements (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	5
System default:	0

Number of elements that are to be written from a parameter. Single parameter -> No need to specify elements. Array parameter -> 1 element under subindex. Array parameter -> 2 to n elements starting from subindex.

subIndex (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	6
System default:	0

Specifies the subindex that addresses the first array element of the parameter. 0 to 65535 permitted.

dataType (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	7
System default:	0

Specifies the data type of the parameter to be written.

Note

Further information on the structure of a data type can be found:

- In the SIMOTION Communication System Manual or
- At www.profibus.com, in the documentation "Profile Guidelines, Part 2: Data Types, Programming Languages and Platforms".

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	8

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies the command transition. With IMMEDIATELY, the transition takes place immediately after the command is issued (asynchronous call). With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	9

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND.

dataLength

Direction:	Input parameter
Data type:	UDINT
Parameter index:	11

Specifies the length of the data to be written (maximum 228 bytes).

data

Direction:	Input parameter
Data type:	ARRAY [0..227] OF BYTE
Parameter index:	12

Source area for the user data to be written. The user data must be stored in big endian format.

2.3.3 SINAMICS

2.3.3.1 _abortAllReadWriteDriveParameterJobs

This function aborts all parameter jobs currently being processed. The term "parameter job" refers to the functions _readDrive(Multi)ParameterDescription, _readDrive(Multi)Parameter, _writeDrive(Multi)Parameter and _readDriveFaults.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual
  Section Deleting _readDrive and _writeDrive jobs
- or in the SIMOTION online help

Return value: DINT

Description of the return values:

16#00000000	Job completed without errors.
16#FFFF819F	Error, job aborted. Function not executable.
16#FFFF81C5	Error, can be repeated in the next program cycle. Function could not be executed. This can happen for the following reasons: - The function _abortAllReadWriteDriveParameterJobs is already being executed. - A parameter job is currently being called from another task.

2.3.3.2 _LineModule_control

<Reference> <ALinks> <ALink> <ALinkKeyword>FB_LineModule_control_Description_11930635</ ALinkKeyword> <ALinkText>Standard function for SINAMICS S120 Line Modules: Section Description</ALinkText> </ALink> <ALink> <ALinkKeyword>FB_LineModule_control_Parameterizing_Addressing_11942027</ALinkKeyword> <ALinkText>Standard function for SINAMICS S120 Line Modules: Section Parameterizing/ addressing</ALinkText> </ALink> <ALink> <ALinkKeyword>FB_LineModule_control_Programming_11953419</ALinkKeyword> <ALinkText>SStandard function for SINAMICS S120 Line Modules: Section Programming</ ALinkText> </ALink> <ALink> <ALinkKeyword>FB_LineModule_control_Application_example_11964811</ALinkKeyword> <ALinkText>Standard function for SINAMICS S120 Line Modules: Section Application example/ ALinkText> </ALink> <ALink> <ALinkKeyword>FB_LineModule_control_Appendix_11976203</ ALinkKeyword> <ALinkText>Standard function for SINAMICS S120 Line Modules: Section Appendix/ ALinkText> </ALink> </ALinks>

Note

More detailed information on this function block can be found in the description:
Function Manual SIMOTION Standard Function for SINAMICS S120 Line Modules
or in the online help

</Reference>

2.3.3.3 _readDriveFaults

This function enables the current fault buffer entry in the drive to be read.

Note

- Up to 16 parameter jobs can be executed in parallel (the term parameter job also includes the functions _readDrive(Multi)ParameterDescription, _readDrive(Multi)Parameter, and _writeDrive(Multi)Parameter).
- Only one parameter job is permitted per planned DP station. A new job to the same DP station is rejected with the error 16#FFFF81C7.
- If the system functions _readRecord and _writeRecord are used in addition to the parameter jobs, you must ensure that they are not to be executed simultaneously by a DP station. Reason: The same communication services are used.
- If a parameter job (only asynchronous calls) is no longer to be processed, then it must be aborted with the ABORT_CURRENT_COMMAND transition condition. Otherwise no new job can be issued.

Parameters can be read out with the following data sets:

- Data set 47 is always read out for PROFIBUS (external or integrated), regardless of whether the function is called with a valid (0>=doId<=254) or invalid 'doId' (doId=255). If the passed 'doID' is invalid, it will be determined by the system itself. This requires the specification of the logical address of the axis.
- Two data sets are available for PROFINET:
  - Base Mode Parameter Access - local (data set 0xB02E)

Note

Determining the doID for SIMOTION versions V4.2 and older

For the SIMOTION versions <= V4.2, the automatic determination of 'dold' (if 'dold' is not specified or is invalid) does not take place for all device types in a completely profile-compliant manner. This affects drives which do not support the optional parameter P978 (e.g. SINAMICS G120), where the system in this case determines a 'dold' that is too small by 1 and the wrong DO is therefore accessed. To work around this error, transfer a correct, valid 'dold' during the call-up.

Note

MAP (module access point) address allocation with PROFINET

SIMOTION V4.1 SP2: MAP must always be at subslot 1 (HW Config configuration). The MAP submodule is the DO proxy and hosts the alarm channel and one or more PAPs (parameter access points).

Base Mode Parameter Access - global (data set 0xB02F)

This data set is used if a valid 'dold' (0>=doId<=254) is entered. Any valid MAP or address can be specified, as assignment only takes place on the basis of the 'dold'.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual
- or in the SIMOTION online help.

Return value: StructRetReadDriveFaults

StructRetReadDriveFaults
Return parameter for _readDriveFaults

Structure	Name	Data type	Unit
functionResult	Total job return value	DINT	-
parameterResult	Parameter-specific return value	DINT	-
dataLength	Length of data read	UDINT	-
data	Data read from the fault buffer	ARRAY [0..7] OF UINT	-

Description of the return values:

The return value consists of 'functionResult', 'parameterResult', 'dataLength', and 'data[8]'.

'functionResult'

Reports the status of the total job plus errors during the transfer and internal errors.

16#00000000	Reading of current fault buffer entry okay, Job completed with positive result. Job okay.
16#00007001	Must be repeated in the next program cycle. Initial call, initiation of reading of current fault buffer entry okay (only when command is issued asynchronously).
16#00007002	Must be repeated in the next program cycle. Intermediate call, reading of current fault buffer entry is still active (only when command is issued asynchronously).
16#00007003	Reading the current fault buffer entry aborted.
16#FFFF8190	Internal error, job aborted. Specified logical base address invalid: No assignment is available in SDBs, or there is no base address.

16#FFFF8191	Internal error, job aborted. The _readDriveFaults function cannot reach the specified logical base address.
16#FFFF8192	Internal error, job aborted. Error in response identifier.
16#FFFF819D	Internal error, job aborted. An I slave/I device interface is unable to issue a parameter job to the higher-level master/controller.
16#FFFF819E	Internal error, job aborted. Attempt to abort a non-active function.
16#FFFF819F	Internal error, job aborted. Function not executable.
16#FFFF81C3	Internal error, can be repeated in the next program cycle. Required resources are presently occupied: - in the _readDriveFaults function - In the module
16#FFFF81C5	Internal error, can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF81C7	Internal error, can be repeated in the next program cycle. Another parameter job has already been issued to the DP station. - based on the user program - from a system-internal component (e.g. TO axis executes parameter comparison after station recovery)
16#FFFF81CF	Internal error, can be repeated in the next program cycle. Another parameter job call is currently active under this 'commandId'.
16#FFFF8290	Corresponds to the error message 16#FFFF8190.
16#FFFF8291	Corresponds to the error message 16#FFFF8191.
16#FFFF82A2	Error during data set transfer, job aborted. Error in Layer2: - Station failure - Timeout
16#FFFF82A3	Error during data set transfer, job aborted. Error in user interface/user: - Protocol error - Station failure - Timeout
16#FFFF82A4-A7	Error during data set transfer, job aborted. Error numbers according to IEC 61158 reserved - Query the cause at the module's manufacturer
16#FFFF82A8	Error during data set transfer, job aborted. Error because of version conflict.
16#FFFF82A9	Error during data set transfer, job aborted. Function not supported by the module.
16#FFFF82AA-AF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82B0	Error during data set transfer, job aborted. Parameter jobs are not supported by the addressed module.
16#FFFF82B2	Error during data set transfer, job aborted. Module reports access to an invalid slot/subslot.
16#FFFF82B3	Error during data set transfer, job aborted. Module reports type conflict.

16#FFFF82B4	Error during data set transfer, job aborted. Module reports access to an invalid area.
16#FFFF82B5	Error during data set transfer, can be repeated in the next program cycle. The system function cannot be executed due to an internal operating state of the module.
16#FFFF82B6	Error during data set transfer, job aborted. Module rejects access.
16#FFFF82B7	Error during data set transfer, job aborted. The job could not be dispatched due to a job telegram error.
16#FFFF82B8	Error during data set transfer, job aborted. Module reports an invalid parameter.
16#FFFF82B9	Error during data set transfer, job aborted. Module reports an invalid type.
16#FFFF82BA-BF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82C0	Error during data set transfer, can be repeated in the next program cycle. The module lists the data set, but the module does not yet have any read data.
16#FFFF82C2	Error during data set transfer, command can be repeated immediately. The module is currently executing the maximum possible jobs for one CPU.
16#FFFF82C3	Error during data set transfer, command can be repeated immediately. Required resources are presently occupied: - In the _readRecord function - In the module
16#FFFF82C4	Error during data set transfer, command can be repeated immediately. Communication errors: - Parity error - SW Ready not set - Error in block length administration - Checksum error on CPU side - Checksum error on module side
16#FFFF82C5	Error during data set transfer, Can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF82C6	Error during data set transfer, can be repeated in the next program cycle. Data set transfer has been aborted due to priority class abort (restart or background).

'parameterResult'

Reports parameter-specific errors.

16#00000000	Parameter access completed without errors.
16#FFFF8000	Parameter error, job aborted. The selected fault buffer does not exist.
16#FFFF8011	Parameter error, job aborted. Job cannot be executed due to operating state.
16#FFFF8019	Parameter error, job aborted. Access to a non-existing axis or an invalid drive object.
16#FFFF80xx	Parameter error, job aborted. Manufacturer-specific error: To determine the exact meaning of the error value 'xx' (20-FF), consult the associated manufacturer-dependent device manual.

'dataLength'

reports the length of the data read.

'data[8]'

contains the read data (word array, maximum of 8 words).

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of logical base address of the drive. With INPUT, the logical address of the drive is in the input range. With OUTPUT, the logical address of the drive is in the output range. Diagnostic addresses are always of type INPUT.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical base address of the drive.

If the optional 'doId' parameter is also used, any station address (preferably the diagnostic address of the station) can be specified.

With PROFINET, parameter access occurs via the MAP (module access point) of a DO. As an alternative to the logical base address of the drive, specification of the diagnostic address of the associated MAP is recommended.

doId (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	3
System default:	255

Specifies the DO-ID to address a specific DO directly. 0 to 254 are permitted. Under the following conditions, the DO-ID may not be specified, or specified with an invalid value (>254): - Access via the DO-ID is not supported by the DP slave / I/O device (P978 not implemented). - Data set 0xB02F is not supported (PROFINET only). - Access is to be performed via the Parameter Access Point of a DO (PROFINET only).

faultType (optional)

Direction:	Input parameter
Data type:	EnumDriveFaultsType
Parameter index:	4
System default:	DRIVE

EnumDriveFaultsType

DRIVE (0)	Drive-specific fault buffer
USER (1)	User-specific fault buffer
SAFETY (2)	Safety-relevant fault buffer

Specification of the fault buffer type. 'faultType' can be used to assign and read out drive- and user-specific parameters. DRIVE reads out the drive-specific fault buffer (parameter 947). USER reads out the user-specific fault buffer (parameter 945). Fault codes of the drive (P947) can be assigned user-specific faults.

SAFETY reads out the fault buffer (parameter 9747) relevant for the safety functionality. The reason for the safety fault can also be read via drive parameter r9749.

Note

The fault buffer in P9747 relevant for safety is reserved for SINAMICS modules. Other modules could have saved different content in P9747.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	5

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies the command transition. With IMMEDIATELY, the transition takes place immediately after the command is issued (asynchronous call). With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	6

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND.

2.3.3.4 _readDriveMultiParameter

This function enables up to 39 parameters to be read.

Note

- Up to 16 parameter jobs can be executed in parallel (the term parameter job also includes the functions _readDriveParameter,_readDrive(Multi)ParameterDescription, _writeDrive(Multi)Parameter, and _readDriveFaults).
- Only one parameter job is permitted per planned DP station. A new job to the same DP station is rejected with the error 16#FFFF81C7.
- If the system functions _readRecord and _writeRecord are used in addition to the parameter jobs, you must ensure that they are not to be executed simultaneously by a DP station. Reason: The same communication services are used.
- If a parameter job (only asynchronous calls) is no longer to be processed, then it must be aborted with the ABORT_CURRENT_COMMAND transition condition. Otherwise no new job can be issued.

Parameters can be read out with the following data sets:

- Data set 47 is always read out for PROFIBUS (external or integrated), regardless of whether the function is called with a valid (0>=doId<=254) or invalid 'doId' (doId=255). If the passed 'doID' is invalid, it will be determined by the system itself. This requires the specification of the logical address of the axis.
- Two data sets are available for PROFINET:
  - Base Mode Parameter Access - local (data set 0xB02E)

Note

Determining the doID for SIMOTION versions V4.2 and older

For the SIMOTION versions <= V4.2, the automatic determination of 'dold' (if 'dold' is not specified or is invalid) does not take place for all device types in a completely profile-compliant manner. This affects drives which do not support the optional parameter P978 (e.g. SINAMICS G120), where the system in this case determines a 'dold' that is too small by 1 and the wrong DO is therefore accessed. To work around this error, transfer a correct, valid 'dold' during the call-up.

Note

MAP (module access point) address allocation with PROFINET

SIMOTION V4.1 SP2: MAP must always be at subslot 1 (HW Config configuration). The MAP submodule is the DO proxy and hosts the alarm channel and one or more PAPs (parameter access points).

Base Mode Parameter Access - global (data set 0xB02F)

This data set is used if a valid 'dold' (0>=doId<=254) is entered. Any valid MAP or address can be specified, as assignment only takes place on the basis of the 'dold'.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual
- or in the SIMOTION online help.

Return value: StructRetReadDriveMultiParameter

StructRetReadDriveMultiParameter
Return parameter for _readDriveMultiParameter

Structure	Name	Data type	Unit
functionResult	Total job return value	DINT	-
parameterResultErrorExistent	A parameter-specific fault is present in the return value	BOOL	-
parameterResult	Parameter-specific return value	ARRAY [0..38] OF DINT	-
dataType	Data type of the read parameter	ARRAY [0..38] OF UDINT	-
dataLength	Length of data read	ARRAY [0..38] OF UDINT	-
data	Data read from the drive (big endian)	ARRAY [0..233] OF BYTE	-

Description of the return values:

The return value consists of 'functionResult', 'parameterResultErrorExistent', 'parameterResult[39]', 'dataType[39]', 'dataLength[39]' and 'data[233]'.

'functionResult'

Reports the status of the total job plus errors during the transfer and internal errors.

16#00000000	Reading of parameters okay, job completed with positive result. Job okay.
16#00007001	Must be repeated in the next program cycle. Initial call, initiation of reading of parameters okay (only when command is issued asynchronously).

16#00007002	Must be repeated in the next program cycle. Intermediate call, reading of parameter description is still active (only when command is issued asynchronously).
16#00007003	Reading of parameters aborted.
16#FFFF8190	Internal error, job aborted. Specified logical base address invalid: No assignment is available in SDBs, or there is no base address.
16#FFFF8191	Internal error, job aborted. The _readDriveMultiParameter function cannot reach the specified logical base address.
16#FFFF8192	Internal error, job aborted. Error in response identifier.
16#FFFF8193	Internal error, job aborted. The number of parameters to be read is not permissible.
16#FFFF819D	Internal error, job aborted. An I slave/I device interface is unable to issue a parameter job to the higher-level master/controller.
16#FFFF819E	Internal error, job aborted. Attempt to abort a non-active function.
16#FFFF819F	Internal error, job aborted. Function not executable.
16#FFFF81C3	Internal error, can be repeated in the next program cycle. Required resources are presently occupied: - In the _readDriveMultiParameter function - In the module
16#FFFF81C5	Internal error, can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF81C7	Internal error, can be repeated in the next program cycle. Another parameter job has already been issued to the DP station. - based on the user program - from a system-internal component (e.g. TO axis executes parameter comparison after station recovery)
16#FFFF81CF	Internal error, can be repeated in the next program cycle. Another parameter job call is currently active under this 'commandId'.
16#FFFF8290	Corresponds to the error message 16#FFFF8190.
16#FFFF8291	Corresponds to the error message 16#FFFF8191.
16#FFFF82A2	Error during data set transfer, job aborted. Error in Layer2: - Station failure - Timeout
16#FFFF82A3	Error during data set transfer, job aborted. Error in user interface/user: - Protocol error - Station failure - Timeout
16#FFFF82A4-A7	Error during data set transfer, job aborted. Error numbers according to IEC 61158 reserved - Query the cause at the module's manufacturer
16#FFFF82A8	Error during data set transfer, job aborted. Error because of version conflict.

16#FFFF82A9	Error during data set transfer, job aborted. Function not supported by the module.
16#FFFF82AA-AF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82B0	Error during data set transfer, job aborted. Parameter jobs are not supported by the addressed module.
16#FFFF82B2	Error during data set transfer, job aborted. Module reports access to an invalid slot/subslot.
16#FFFF82B3	Error during data set transfer, job aborted. Module reports type conflict.
16#FFFF82B4	Error during data set transfer, job aborted. Module reports access to an invalid area.
16#FFFF82B5	Error during data set transfer, can be repeated in the next program cycle. The system function cannot be executed due to an internal operating state of the module.
16#FFFF82B6	Error during data set transfer, job aborted. Module rejects access.
16#FFFF82B7	Error during data set transfer, job aborted. The job could not be dispatched due to a job error.
16#FFFF82B8	Error during data set transfer, job aborted. Module reports an invalid parameter.
16#FFFF82B9	Error during data set transfer, job aborted. Module reports an invalid type.
16#FFFF82BA-BF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82C0	Error during data set transfer, can be repeated in the next program cycle. The module lists the data set, but the module does not yet have any read data.
16#FFFF82C2	Error during data set transfer, command can be repeated immediately. The module is currently executing the maximum possible jobs for one CPU.
16#FFFF82C3	Error during data set transfer, command can be repeated immediately. Required resources are presently occupied: - In the _readRecord function - In the module
16#FFFF82C4	Error during data set transfer, command can be repeated immediately. Communication errors: - Parity error - SW Ready not set - Error in block length administration - Checksum error on CPU side - Checksum error on module side
16#FFFF82C5	Error during data set transfer, Can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF82C6	Error during data set transfer, can be repeated in the next program cycle. Data set transfer has been aborted due to priority class abort (restart or background).

'parameterResultErrorExistent'

Returns TRUE when an error is reported in at least one 'parameterResult'.

'parameterResult'

Reports parameter-specific errors.

16#00000000	Parameter access completed without errors.
16#FFFF8000	Parameter error, job aborted. Access to a non-existent parameter.
16#FFFF8003	Parameter error, job aborted. Access to a non-existent subindex.
16#FFFF8004	Parameter error, job aborted. Access with subindex to non-indexed parameter (not an array parameter).
16#FFFF800B	Parameter error, job aborted. Change access without parameter change rights.
16#FFFF8011	Parameter error, job aborted. Job cannot be executed due to operating state.
16#FFFF8015	Parameter error, job aborted. The length of the current response exceeds the maximum transferable length. The following errors can also be the cause - The DP station provides fewer parameters than requested - The DP station does not support multi-parameter jobs
16#FFFF8016	Parameter error, job aborted. Illegal or unsupported value for attribute, number of elements, parameter number, or subindex, or a combination of these.
16#FFFF8017	Parameter error, job aborted. Illegal or unsupported parameter format.
16#FFFF8019	Parameter error, job aborted. Access to a non-existing axis or an invalid drive object.
16#FFFF80xx	Parameter error, job aborted. Manufacturer-specific error: To determine the exact meaning of the error value 'xx' (20-FF), consult the associated manufacturer-dependent device manual.

'dataType[39]'

returns the data type for each parameter.

Note

Further information on the structure of a data type can be found:

- In the SIMOTION Communication System Manual or
- At www.profibus.com, in the documentation "Profile Guidelines, Part 2: Data Types, Programming Languages and Platforms".

'dataLength[39]'

reports the length of the data read for each parameter.

'data[233]'

contains the read data (byte array, maximum of 233 bytes).

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of logical base address of the drive. With INPUT, the logical address of the drive is in the input range. With OUTPUT, the logical address of the drive is in the output range. Diagnostic addresses are always of type INPUT.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical base address of the drive.

If the optional 'doId' parameter is also used, any station address (preferably the diagnostic address of the station) can be specified.

With PROFINET, parameter access occurs via the MAP (module access point) of a DO. As an alternative to the logical base address of the drive, specification of the diagnostic address of the associated MAP is recommended.

doId (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	3
System default:	255

Specifies the DO-ID to address a specific DO directly. 0 to 254 are permitted.

The DO-ID can be unspecified or specified as invalid (>254) under the following conditions:

- Access via the DO-ID is not supported by the DP slave/IO device (P978 is not implemented).
- Data set 0xB02F is not supported (PROFINET only).
- Access is to occur via the parameter access point of a DO (PROFINET only).

numberOfParameters

Direction:	Input parameter
Data type:	UDINT
Parameter index:	4

Number of parameters to be read. 1 to 39 permitted.

parameterNumber

Direction:	Input parameter
Data type:	ARRAY [0..38] OF UDINT
Parameter index:	5

Specifies the parameters from which the values are to be read. 1 to 65535 permitted.

numberOfElements (optional)

Direction:	Input parameter
Data type:	ARRAY [0..38] OF UDINT
Parameter index:	6
System default:	[0..38] = 0

Number of elements that are to be read from a parameter. Single parameter -> No need to specify elements. Array parameter -> 1 element under subindex. Array parameter -> 2 to n elements starting from subindex.

subIndex (optional)

Direction:	Input parameter
Data type:	ARRAY [0..38] OF UDINT
Parameter index:	7
System default:	[0..38] = 0

Specifies the subindex that addresses the first array element of an array parameter. 0 to 65535 permitted. Single parameter -> No need to specify elements.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	8

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies the command transition. With IMMEDIATELY, the transition takes place immediately after the command is issued (asynchronous call). With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	9

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND.

2.3.3.5 _readDriveMultiParameterDescription

This function enables up to four parameter descriptions to be read. The interpretation of the parameter descriptions is contained in the PROFIdrive profile.

Note

- Up to 16 parameter jobs can be executed in parallel (the term parameter job also includes the functions _readDriveParameterDescription, _readDrive(Multi)Parameter, _writeDrive(Multi)Parameter, and _readDriveFaults).
- Only one parameter job is permitted per planned DP station. A new job to the same DP station is rejected with the error 16#FFFF81C7.
- If the system functions _readRecord and _writeRecord are used in addition to the parameter jobs, you must ensure that they are not to be executed simultaneously by a DP station. Reason: The same communication services are used.
- If a parameter job (only asynchronous calls) is no longer to be processed, then it must be aborted with the ABORT_CURRENT_COMMAND transition condition. Otherwise no new job can be issued.

Parameters can be read out with the following data sets:

- Data set 47 is always read out for PROFIBUS (external or integrated), regardless of whether the function is called with a valid (0>=doId<=254) or invalid 'doId' (doId=255). If the passed 'doID' is invalid, it will be determined by the system itself. This requires the specification of the logical address of the axis.
- Two data sets are available for PROFINET:
  - Base Mode Parameter Access - local (data set 0xB02E)

Note

Determining the doID for SIMOTION versions V4.2 and older

For the SIMOTION versions <= V4.2, the automatic determination of 'dold' (if 'dold' is not specified or is invalid) does not take place for all device types in a completely profile-compliant manner. This affects drives which do not support the optional parameter P978 (e.g. SINAMICS G120), where the system in this case determines a 'dold' that is too small by 1 and the wrong DO is therefore accessed. To work around this error, transfer a correct, valid 'dold' during the call-up.

Note

MAP (module access point) address allocation with PROFINET

SIMOTION V4.1 SP2: MAP must always be at subslot 1 (HW Config configuration). The MAP submodule is the DO proxy and hosts the alarm channel and one or more PAPs (parameter access points).

Base Mode Parameter Access - global (data set 0xB02F)

This data set is used if a valid 'dold' (0>=doId<=254) is entered. Any valid MAP or address can be specified, as assignment only takes place on the basis of the 'dold'.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual,
- Section PROFIdrive-specific data types,
- Section Rules for SIMOTION _writeDrive.../_readDrive... commands
- or in the SIMOTION online help.

Return value: StructRetReadDriveMultiParameterDescription

StructRetReadDriveMultiParameterDescription
Return parameter for _readDriveMultiParameterDescription

Structure	Name	Data type	Unit
functionResult	Total job return value	DINT	-
parameterResultErrorExistent	A parameter-specific fault is present in the return value	BOOL	-
parameterResult	Parameter-specific return value	ARRAY [0..3] OF DINT	-
dataType	Parameter data type	ARRAY [0..3] OF USINT	-
numberOfElements	Number of parameter elements	ARRAY [0..3] OF UINT	-
factor	Normalization factor	ARRAY [0..3] OF REAL	-
variableIndex	Variable index	ARRAY [0..3] OF USINT	-
conversionIndex	Conversion index	ARRAY [0..3] OF SINT	-
pzdReference	Process data reference parameter	ARRAY [0..3] OF UINT	-
pzdNormalization	Process data normalization	ARRAY [0..3] OF UINT	-

Description of the return values:

The return value consists of 'functionResult', 'parameterResultErrorExistent', 'parameterResult[4]', 'dataType[4]', 'numberOfElements[4]', 'factor[4]', 'variableIndex[4]', 'conversionIndex[4]', 'pzdReference[4]',and 'pzdNormalization[4]'.

'functionResult'

Reports the status of the total job plus errors during the transfer and internal errors.

16#00000000	Read parameter descriptions okay, job completed with positive result. Job okay.
16#00007001	Must be repeated in the next program cycle. Initial call, initiation of reading of parameter descriptions okay (only when command is issued asynchronously).
16#00007002	Must be repeated in the next program cycle. Intermediate call, reading of parameter descriptions is still active (only when command is issued asynchronously).
16#00007003	Reading of parameter descriptions aborted.
16#FFFF8190	Internal error, job aborted. Specified logical base address invalid: No assignment is available in SDBs, or there is no base address.
16#FFFF8191	Internal error, job aborted. The _readDriveMultiParameterDescription function cannot reach the specified logical base address.
16#FFFF8192	Internal error, job aborted. Error in response identifier.
16#FFFF8193	Internal error, job aborted. The number of parameters to be read is not permissible.
16#FFFF819D	Internal error, job aborted. An I slave/I device interface is unable to issue a parameter job to the higher-level master/controller.
16#FFFF819E	Internal error, job aborted. Attempt to abort a non-active function.
16#FFFF819F	Internal error, job aborted. Function not executable.
16#FFFF81C3	Internal error, can be repeated in the next program cycle. Required resources are presently occupied: - In the _readDriveMultiParameterDescription function - In the module
16#FFFF81C5	Internal error, can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF81C7	Internal error, can be repeated in the next program cycle. Another parameter job has already been issued to the DP station. - based on the user program - from a system-internal component (e.g. TO axis executes parameter comparison after station recovery)
16#FFFF81CF	Internal error, can be repeated in the next program cycle. Another parameter job call is currently active under this 'commandId'.
16#FFFF8290	Corresponds to the error message 16#FFFF8190.
16#FFFF8291	Corresponds to the error message 16#FFFF8191.

16#FFFF82A2	Error during data set transfer, job aborted. Error in Layer2: - Station failure - Timeout
16#FFFF82A3	Error during data set transfer, job aborted. Error in user interface/user: - Protocol error - Station failure - Timeout
16#FFFF82A4-A7	Error during data set transfer, job aborted. Error numbers according to IEC 61158 reserved - Query the cause at the module's manufacturer
16#FFFF82A8	Error during data set transfer, job aborted. Error because of version conflict.
16#FFFF82A9	Error during data set transfer, job aborted. Function not supported by the module.
16#FFFF82AA-AF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82B0	Error during data set transfer, job aborted. Parameter jobs are not supported by the addressed module.
16#FFFF82B2	Error during data set transfer, job aborted. Module reports access to an invalid slot/subslot.
16#FFFF82B3	Error during data set transfer, job aborted. Module reports type conflict.
16#FFFF82B4	Error during data set transfer, job aborted. Module reports access to an invalid area.
16#FFFF82B5	Error during data set transfer, can be repeated in the next program cycle. The system function cannot be executed due to an internal operating state of the module.
16#FFFF82B6	Error during data set transfer, job aborted. Module rejects access.
16#FFFF82B7	Error during data set transfer, job aborted. The job could not be dispatched due to a job error.
16#FFFF82B8	Error during data set transfer, job aborted. Module reports an invalid parameter.
16#FFFF82B9	Error during data set transfer, job aborted. Module reports an invalid type.
16#FFFF82BA-BF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82C0	Error during data set transfer, can be repeated in the next program cycle. The module lists the data set, but the module does not yet have any read data.
16#FFFF82C2	Error during data set transfer, command can be repeated immediately. The module is currently executing the maximum possible jobs for one CPU.
16#FFFF82C3	Error during data set transfer, command can be repeated immediately. Required resources are presently occupied: - In the _readRecord function - In the module

16#FFFF82C4	Error during data set transfer, command can be repeated immediately. Communication errors: - Parity error - SW Ready not set - Error in block length administration - Checksum error on CPU side - Checksum error on module side
16#FFFF82C5	Error during data set transfer, Can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF82C6	Error during data set transfer, can be repeated in the next program cycle. Data set transfer has been aborted due to priority class abort (restart or background).

'parameterResultErrorExistent'

Returns TRUE when an error is reported in at least one 'parameterResult'.

'parameterResult'

Reports parameter-specific errors.

16#00000000	Parameter access completed without errors.
16#FFFF8000	Parameter error, job aborted. Access to a non-existent parameter.
16#FFFF8009	Parameter error, job aborted. Access to a non-existent description (parameter does exist).
16#FFFF8011	Parameter error, job aborted. Job cannot be executed due to operating state.
16#FFFF8019	Parameter error, job aborted. Access to a non-existing axis or an invalid drive object.
16#FFFF80xx	Parameter error, job aborted. Manufacturer-specific error: To determine the exact meaning of the error value 'xx' (20-FF), consult the associated manufacturer-dependent device manual.

'dataType'

returns the parameter data type (for the coding, see PROFIdrive profile).

Note

Further information on the structure of a data type can be found:

- In the SIMOTION Communication System Manual or
- At www.profibus.com, in the documentation "Profile Guidelines, Part 2: Data Types, Programming Languages and Platforms".

'numberOfElements'

supplies the number of elements of an array parameter. The string length is specified here for parameters of data type "String".

'factor'

supplies the normalization factor that converts the (internal) value into an (externally) displayable quantity (see PROFIdrive profile).

'variableIndex'

returns the variable index and displays a fixed code for the physical quantity of the parameter value (see PROFIdrive profile).

'conversionIndex'

supplies the fixed code of the conversion index (conversion factor + offset) of a parameter (see PROFIdrive profile).

'pzdReference'

returns the physical reference value for calculating a physical value of a normalized variable (see PROFIdrive profile).

'pzdNormalization'

supplies the bit to which the physical reference value refers (see PROFIdrive profile).

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of logical base address of the drive. With INPUT, the logical address of the drive is in the input range. With OUTPUT, the logical address of the drive is in the output range. Diagnostic addresses are always of type INPUT.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical base address of the drive.

If the optional 'doId' parameter is also used, any station address (preferably the diagnostic address of the station) can be specified.

With PROFINET, parameter access occurs via the MAP (module access point) of a DO. As an alternative to the logical base address of the drive, specification of the diagnostic address of the associated MAP is recommended.

doId (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	3
System default:	255

Specifies the DO-ID to address a specific DO directly. 0 to 254 are permitted. Under the following conditions, the DO-ID may not be specified, or specified with an invalid value (>254): - Access via the DO-ID is not supported by the DP slave / I/O device (P978 not implemented). - Data set 0xB02F is not supported (PROFINET only). - Access is to be performed via the Parameter Access Point of a DO (PROFINET only).

numberOfParameters

Direction:	Input parameter
Data type:	UDINT
Parameter index:	4

Number of parameter descriptions to be read. 1 to 4 permitted.

parameterNumber

Direction:	Input parameter
Data type:	ARRAY [0..3] OF UDINT
Parameter index:	5

Specifies the parameters from which the description is to be read. 1 to 65535 permitted.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	6

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies the command transition. With IMMEDIATELY, the transition takes place immediately after the command is issued (asynchronous call). With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	7

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND.

2.3.3.6 _readDriveParameter

This function enables a drive parameter to be read.

Note

- Up to 16 parameter jobs can be executed in parallel (the term parameter job also includes the functions _readDriveMultiParameter, _readDrive(Multi)ParameterDescription, _writeDrive(Multi)Parameter, and _readDriveFaults).
- Only one parameter job is permitted per planned DP station. A new job to the same DP station is rejected with the error 16#FFFF81C7.
- If the system functions _readRecord and _writeRecord are used in addition to the parameter jobs, you must ensure that they are not to be executed simultaneously by a DP station. Reason: The same communication services are used.
- If a parameter job (only asynchronous calls) is no longer to be processed, then it must be aborted with the ABORT_CURRENT_COMMAND transition condition. Otherwise no new job can be issued.

Parameters can be read out with the following data sets:

- Data set 47 is always read out for PROFIBUS (external or integrated), regardless of whether the function is called with a valid (0>=doId<=254) or invalid 'doId' (doId=255). If the passed 'doId' is invalid, it will be determined by the system itself. This requires the specification of the logical address of the axis.
- Two data sets are available for PROFINET:
  - Base Mode Parameter Access - local (data set 0xB02E)

Note

Determining the doID for SIMOTION versions V4.2 and older

For the SIMOTION versions <= V4.2, the automatic determination of 'dold' (if 'dold' is not specified or is invalid) does not take place for all device types in a completely profile-compliant manner. This affects drives which do not support the optional parameter P978 (e.g. SINAMICS G120), where the system in this case determines a 'dold' that is too small by 1 and the wrong DO is therefore accessed. To work around this error, transfer a correct, valid 'dold' during the call-up.

Note

MAP (module access point) address allocation with PROFINET

SIMOTION V4.1 SP2: MAP must always be at subslot 1 (HW Config configuration). The MAP submodule is the DO proxy and hosts the alarm channel and one or more PAPs (parameter access points).

Base Mode Parameter Access - global (data set 0xB02F)

This data set is used if a valid 'dold' (0>=doId<=254) is entered. Any valid MAP or address can be specified, as assignment only takes place on the basis of the 'dold'.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual,
  Section PROFIdrive-specific data types,
  Section Rules for SIMOTION _writeDrive.../_readDrive... commands
- or in the SIMOTION online help.

Return value: StructRetReadDriveParameter

StructRetReadDriveParameter
Return parameter for _readDriveParameter

Structure	Name	Data type	Unit
functionResult	Total job return value	DINT	-
parameterResult	Parameter-specific return value	DINT	-
dataType	Data type of the read parameter	UDINT	-
dataLength	Length of read data in bytes	UDINT	-
data	Data read by the drive (big endian)	ARRAY [0..233] OF BYTE	-

Description of the return values:

The return value consists of 'functionResult', 'parameterResult', 'dataType', 'dataLength' and 'data[234]'.

'functionResult'

Reports the status of the total job plus errors during the transfer and internal errors.

16#00000000	Reading of parameters okay, job completed with positive result. Job okay.
16#00007001	Must be repeated in the next program cycle. Initial call, initiation of reading of parameter description okay (only when command is issued asynchronously).

16#00007002	Must be repeated in the next program cycle. Intermediate call, reading of parameter description is still active (only when command is issued asynchronously).
16#00007003	Reading of parameters aborted. Job aborted.
16#FFFF8190	Internal error, job aborted. Specified logical base address invalid: No assignment is available in SDBs, or there is no base address.
16#FFFF8191	Internal error, job aborted. The _readDriveParameter function cannot reach the specified logical base address.
16#FFFF8192	Internal error, job aborted. Error in response identifier.
16#FFFF819D	Internal error, job aborted. An I slave/I device interface is unable to issue a parameter job to the higher-level master/controller.
16#FFFF819E	Internal error, job aborted. Attempt to abort a non-active function.
16#FFFF819F	Internal error, job aborted. Function not executable.
16#FFFF81C3	Internal error, can be repeated in the next program cycle. Required resources are presently occupied: - In the _readDriveParameter function - In the module
16#FFFF81C5	Internal error, can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF81C7	Internal error, can be repeated in the next program cycle. Another parameter job has already been issued to the DP station. - based on the user program - from a system-internal component (e.g. TO axis executes parameter comparison after station recovery)
16#FFFF81CF	Internal error, can be repeated in the next program cycle. Another parameter job call is currently active under this 'commandId'.
16#FFFF8290	Corresponds to the error message 16#FFFF8190.
16#FFFF8291	Corresponds to the error message 16#FFFF8191.
16#FFFF82A2	Error during data set transfer, job aborted. Error in Layer2: - Station failure - Timeout
16#FFFF82A3	Error during data set transfer, job aborted. Error in user interface/user: - Protocol error - Station failure - Timeout
16#FFFF82A4-A7	Error during data set transfer, job aborted. Error numbers according to IEC 61158 reserved - Query the cause at the module's manufacturer
16#FFFF82A8	Error during data set transfer, job aborted. Error because of version conflict.
16#FFFF82A9	Error during data set transfer, job aborted. Function not supported by the module.

16#FFFF82AA-AF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82B0	Error during data set transfer, job aborted. Parameter jobs are not supported by the addressed module.
16#FFFF82B2	Error during data set transfer, job aborted. Module reports access to an invalid slot/subslot.
16#FFFF82B3	Error during data set transfer, job aborted. Module reports type conflict.
16#FFFF82B4	Error during data set transfer, job aborted. Module reports access to an invalid area.
16#FFFF82B5	Error during data set transfer, can be repeated in the next program cycle. The system function cannot be executed due to an internal operating state of the module.
16#FFFF82B6	Error during data set transfer, job aborted. Module rejects access.
16#FFFF82B7	Error during data set transfer, job aborted. The job could not be dispatched due to a job error.
16#FFFF82B8	Error during data set transfer, job aborted. Module reports an invalid parameter.
16#FFFF82B9	Error during data set transfer, job aborted. Module reports an invalid type.
16#FFFF82BA-BF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82C0	Error during data set transfer, can be repeated in the next program cycle. The module lists the data set, but the module does not yet have any read data.
16#FFFF82C2	Error during data set transfer, command can be repeated immediately. The module is currently executing the maximum possible jobs for one CPU.
16#FFFF82C3	Error during data set transfer, command can be repeated immediately. Required resources are presently occupied: - In the _readRecord function - In the module
16#FFFF82C4	Error during data set transfer, command can be repeated immediately. Communication errors: - Parity error - SW Ready not set - Error in block length administration - Checksum error on CPU side - Checksum error on module side
16#FFFF82C5	Error during data set transfer, Can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF82C6	Error during data set transfer, can be repeated in the next program cycle. Data set transfer has been aborted due to priority class abort (restart or background).

'parameterResult'

Reports parameter-specific errors.

16#00000000	Parameter access completed without errors.

16#FFFF8000	Parameter error, job aborted. Access to a non-existent parameter.
16#FFFF8003	Parameter error, job aborted. Access to a non-existent subindex.
16#FFFF8004	Parameter error, job aborted. Access with subindex to non-indexed parameter (not an array parameter).
16#FFFF8011	Parameter error, job aborted. Job cannot be executed due to operating state.
16#FFFF8015	Parameter error, job aborted. The length of the current response exceeds the maximum transferable length.
16#FFFF8016	Parameter error, job aborted. Illegal or unsupported value for attribute, number of elements, parameter number, or subindex, or a combination of these.
16#FFFF8019	Parameter error, job aborted. Access to a non-existing axis or an invalid drive object.
16#FFFF80xx	Parameter error, job aborted. Manufacturer-specific error: To determine the exact meaning of the error value 'xx' (20-FF), consult the associated manufacturer-dependent device manual.

'dataType'

returns the data type of the read parameter.

Note

Further information on the structure of a data type can be found:

- In the SIMOTION Communication System Manual or
- At www.profibus.com, in the documentation "Profile Guidelines, Part 2: Data Types, Programming Languages and Platforms".

'dataLength'

reports the length of the data read.

'data[234]'

contains the read data (byte array, maximum of 234 bytes).

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of logical base address of the drive. With INPUT, the logical address of the drive is in the input range. With OUTPUT, the logical address of the drive is in the output range. Diagnostic addresses are always of type INPUT.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical base address of the drive.

If the optional 'doId' parameter is also used, any station address (preferably the diagnostic address of the station) can be specified.

With PROFINET, parameter access occurs via the MAP (module access point) of a DO. As an alternative to the logical base address of the drive, specification of the diagnostic address of the associated MAP is recommended.

doId (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	3
System default:	255

Specifies the DO-ID to address a specific DO directly. 0 to 254 are permitted.

The DO-ID can be unspecified or specified as invalid (>254) under the following conditions:

- Access via the DO-ID is not supported by the DP slave/IO device (P978 is not implemented).
- Data set 0xB02F is not supported (PROFINET only).
- Access is to occur via the parameter access point of a DO (PROFINET only).

parameterNumber

Direction:	Input parameter
Data type:	UDINT
Parameter index:	4

Specifies the parameter from which the values are to be read. 1 to 65535 permitted.

numberOfElements (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	5
System default:	0

Number of elements to be read from a parameter. Single parameter -> No need to specify elements. Array parameter -> 1 element under subindex. Array parameter -> 2 to n elements starting from subindex.

subIndex (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	6
System default:	0

Specifies the subindex that addresses the first array element of an array parameter, setting between 0 and 65535. Single parameter -> No need to specify elements.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	7

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies the command transition. With IMMEDIATELY, the transition takes place immediately after the command is issued (asynchronous call). With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	8

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND.

2.3.3.7 _readDriveParameterDescription

This function enables the descriptive data of a drive parameter to be read out. The interpretation of the parameter description is contained in the PROFIdrive profile.

Note

- Up to 16 parameter jobs can be executed in parallel (the term parameter job also includes the functions _readDriveParameterDescription, _readDrive(Multi)Parameter, _writeDrive(Multi)Parameter, and _readDriveFaults).
- Only one parameter job is permitted per planned DP station. A new job to the same DP station is rejected with the error 16#FFFF81C7.
- If the system functions _readRecord and _writeRecord are used in addition to the parameter jobs, you must ensure that they are not to be executed simultaneously by a DP station. Reason: The same communication services are used.
- If a parameter job (only asynchronous calls) is no longer to be processed, then it must be aborted with the ABORT_CURRENT_COMMAND transition condition. Otherwise no new job can be issued.

Parameters can be read out with the following data sets:

- Data set 47 is always read out for PROFIBUS (external or integrated), regardless of whether the function is called with a valid (0>=doId<=254) or invalid 'doId' (doId=255). If the passed 'doID' is invalid, it will be determined by the system itself. This requires the specification of the logical address of the axis.
- Two data sets are available for PROFINET:
  - Base Mode Parameter Access - local (data set 0xB02E)

Note

MAP (module access point) address allocation with PROFINET

SIMOTION V4.1 SP2: MAP must always be at subslot 1 (HW Config configuration). The MAP submodule is the DO proxy and hosts the alarm channel and one or more PAPs (parameter access points).

Base Mode Parameter Access - global (data set 0xB02F)

This data set is used if a valid 'dold' (0>=dold<=254) is entered. Any valid MAP or address can be specified, as assignment only takes place on the basis of the 'dold'.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual,
- Section PROFIdrive-specific data types,
- Section Rules for SIMOTION _writeDrive.../_readDrive... commands
- or in the SIMOTION online help.

Return value: StructRetReadDriveParameterDescription

StructRetReadDriveParameterDescription
Return parameter for _readDriveParameterDescription

Structure	Name	Data type	Unit
functionResult	Total job return value	DINT	-
parameterResult	Parameter-specific return value	DINT	-
dataType	Parameter data type	USINT	-
numberOfElements	Number of parameter elements	UINT	-
factor	Normalization factor	REAL	-
variableIndex	Variable index	USINT	-
conversionIndex	Conversion index	SINT	-
pzdReference	Process data reference parameter	UINT	-
pzdNormalization	Process data normalization	UINT	-

The return value consists of 'functionResult', 'parameterResult', 'dataType', 'numberOfElements', 'factor', 'variableIndex', 'conversionIndex', 'pzdReference', and 'pzdNormalization'.

'functionResult'

Reports the status of the total job plus errors during the transfer and internal errors.

16#00000000	Reading of parameter description okay, job completed with positive result. Job okay.

16#00007001	Must be repeated in the next program cycle. Initial call, initiation of reading of parameter description okay (only when command is issued asynchronously).
16#00007002	Must be repeated in the next program cycle. Intermediate call, reading of parameter description is still active (only when command is issued asynchronously).
16#00007003	Reading of parameter description aborted. Job aborted.
16#FFFF8190	Internal error, job aborted. Specified logical base address invalid: No assignment is available in SDBs, or there is no base address.
16#FFFF8191	Internal error, job aborted. The _readDriveParameterDescription function cannot reach the specified logical base address.
16#FFFF8192	Internal error, job aborted. Error in response identifier.
16#FFFF819D	Internal error, job aborted. An I slave/I device interface is unable to issue a parameter job to the higher-level master/controller.
16#FFFF819E	Internal error, job aborted. Attempt to abort a non-active function.
16#FFFF819F	Internal error, job aborted. Function not executable.
16#FFFF81C3	Internal error, can be repeated in the next program cycle. Required resources are presently occupied: - In the _readDriveParameterDescription function - In the module
16#FFFF81C5	Internal error, can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF81C7	Internal error, can be repeated in the next program cycle. Another parameter job has already been issued to the DP station. - based on the user program - from a system-internal component (e.g. TO axis executes parameter comparison after station recovery)
16#FFFF81CF	Internal error, can be repeated in the next program cycle. Another parameter job call is currently active under this 'commandId'.
16#FFFF8290	Corresponds to the error message 16#FFFF8190.
16#FFFF8291	Corresponds to the error message 16#FFFF8191.
16#FFFF82A2	Error during data set transfer, job aborted. Error in Layer2: - Station failure - Timeout
16#FFFF82A3	Error during data set transfer, job aborted. Error in user interface/user: - Protocol error - Station failure - Timeout
16#FFFF82A4-A7	Error during data set transfer, job aborted. Error numbers according to IEC 61158 reserved - Query the cause at the module's manufacturer

16#FFFF82A8	Error during data set transfer, job aborted. Error because of version conflict.
16#FFFF82A9	Error during data set transfer, job aborted. Function not supported by the module.
16#FFFF82AA-AF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82B0	Error during data set transfer, job aborted. Parameter jobs are not supported by the addressed module.
16#FFFF82B2	Error during data set transfer, job aborted. Module reports access to an invalid slot/subslot.
16#FFFF82B3	Error during data set transfer, job aborted. Module reports type conflict.
16#FFFF82B4	Error during data set transfer, job aborted. Module reports access to an invalid area.
16#FFFF82B5	Error during data set transfer, can be repeated in the next program cycle. The system function cannot be executed due to an internal operating state of the module.
16#FFFF82B6	Error during data set transfer, job aborted. Module rejects access.
16#FFFF82B7	Error during data set transfer, job aborted. The job could not be dispatched due to a job error.
16#FFFF82B8	Error during data set transfer, job aborted. Module reports an invalid parameter.
16#FFFF82B9	Error during data set transfer, job aborted. Module reports an invalid type.
16#FFFF82BA-BF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82C0	Error during data set transfer, can be repeated in the next program cycle. The module lists the data set, but the module does not yet have any read data.
16#FFFF82C2	Error during data set transfer, command can be repeated immediately. The module is currently executing the maximum possible jobs for one CPU.
16#FFFF82C3	Error during data set transfer, command can be repeated immediately. Required resources are presently occupied: - In the _readRecord function - In the module
16#FFFF82C4	Error during data set transfer, command can be repeated immediately. Communication errors: - Parity error - SW Ready not set - Error in block length administration - Checksum error on CPU side - Checksum error on module side
16#FFFF82C5	Error during data set transfer, Can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF82C6	Error during data set transfer, can be repeated in the next program cycle. Data set transfer has been aborted due to priority class abort (restart or background).

'parameterResult'

Reports parameter-specific errors.

16#00000000	Parameter access completed without errors.
16#FFFF8000	Parameter error, job aborted. Access to a non-existent parameter.
16#FFFF8009	Parameter error, job aborted. Access to a non-existent description (parameter does exist).
16#FFFF8011	Parameter error, job aborted. Job cannot be executed due to operating state.
16#FFFF8019	Parameter error, job aborted. Access to a non-existing axis or an invalid drive object.
16#FFFF80xx	Parameter error, job aborted. Manufacturer-specific error: To determine the exact meaning of the error value 'xx' (20-FF), consult the associated manufacturer-dependent device manual.

'dataType'

returns the parameter data type (for the coding, see PROFIdrive profile).

Note

Further information on the structure of a data type can be found:

- In the SIMOTION Communication System Manual or
- At www.profibus.com, in the documentation "Profile Guidelines Part 2: Data Types, Programming Languages and Platforms".

'numberOfElements'

supplies the number of elements of an array parameter. The string length is specified here for parameters of data type "String length".

'factor'

supplies the normalization factor that converts the (internal) value into an (externally) displayable quantity (see PROFIdrive profile).

'variableIndex'

returns the variable index and displays a fixed code for the physical quantity of the parameter value (see PROFIdrive profile).

'conversionIndex'

supplies the fixed code of the conversion index (conversion factor + offset) of a parameter (see PROFIdrive profile).

'pzdReference'

returns the physical reference value for calculating a physical value of a normalized variable (see PROFIdrive profile).

'pzdNormalization'

supplies the bit to which the physical reference value refers (see PROFIdrive profile).

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of logical base address of the drive. With INPUT, the logical address of the drive is in the input range. With OUTPUT, the logical address of the drive is in the output range. Diagnostic addresses are always of type INPUT.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical base address of the drive.

If the optional 'doId' parameter is also used, any station address (preferably the diagnostic address of the station) can be specified.

With PROFINET, parameter access occurs via the MAP (module access point) of a DO. As an alternative to the logical base address of the drive, specification of the diagnostic address of the associated MAP is recommended.

doId (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	3
System default:	255

Specifies the DO-ID to address a specific DO directly. 0 to 254 are permitted.

The DO-ID can be unspecified or specified as invalid (>254) under the following conditions:

- Access via the DO-ID is not supported by the DP slave/IO device (P978 is not implemented).
- Data set 0xB02F is not supported (PROFINET only).
- Access is to occur via the parameter access point of a DO (PROFINET only).

parameterNumber

Direction:	Input parameter
Data type:	UDINT
Parameter index:	4

Specifies the parameter from which the description is to be read. 1 to 65535 permitted.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	5

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies the command transition. With IMMEDIATELY, the transition takes place immediately after the command is issued (asynchronous call). With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	6

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND.

2.3.3.8 _resetDriveObjectFault

This function acknowledges an existing fault on a drive object. The function can be used on SINAMICS DO1 with the standard telegrams 390, 391, 392 (for short: 39x). Axis-type drive objects (SINAMICS DO-type servo, vector) cannot be accessed by this function and are only supported via the TO axis.

Additional references

Additional information is available at:

- SIMOTION Basic Functions Function Manual
- or in the SIMOTION online help.

Return value: DINT

The return value reports the job status.

Description of the return values:

16#00000000	Job completed without errors. The fault has been acknowledged on the drive object with the specified logical address.
16#00000001	Job completed without errors, but without effect. The drive object with the specified logical address did not have a fault.
16#FFFF8090	Job aborted. Specified logical base address invalid. Possible causes: - The logical address specified does not exist. - The logical address specified exists, but is not the address of a drive object.
16#FFFF8091	Job aborted. The function is not supported by the drive object with the specified logical address.
16#FFFF809F	Job aborted. Internal error, function cannot be executed.

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of the logical address. With INPUT, the logical address is in the input range. With OUTPUT, the logical address is in the output range.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical base address of the drive.

2.3.3.9 _setDriveObjectSTW

This function can be used to write bits in the control word of an output telegram to a drive object. The function can be used on SINAMICS DO1 on CU_STW in the standard telegrams 390, 391, 392 (for short: 39x). Axis-type drive objects (SINAMICS DO-type servo, vector) cannot be accessed by this function and are supported by _setAxisSTW on the TO axis.

Additional references

Additional information is available at:

- SIMOTION Basic Functions Function Manual
- or in the SIMOTION online help.

Return value: DINT

The return value reports the job status.

Description of the return values:

16#00000000	Job completed without errors. The bits from STW1BitSet selected via STW1BitMask have been written to the control word of the output telegram.
16#FFFF8090	Job aborted. Specified logical base address invalid. Possible causes: - The logical address specified does not exist. - The logical address specified is a diagnostic address. - The logical address specified exists, but is not the address of an output telegram for a drive object. - No telegram is configured for the drive object with the logical address specified. - A telegram is configured for the drive object with the logical address specified, but it contains no control word with which this system function is compatible. This may, for example, be telegram 370 for the SINAMICS infeed module or a free telegram for SINAMICS DO1 (P2079 has no 39x) or an axis-type drive object. - There is no entry in the FastIO configuration for the specified logical address. Please check this by exporting the SIMOTION controller's FastIO configuration (FastIO context menu). If the xml file created does not contain the logical address, please perform the transfer for the relevant SINAMICS CU in accordance with HW Config and reload the project. If this does not produce the desired result, delete the configuration information generated by SCOUT (FastIO context menu). Then perform the transfer in accordance with HW Config in the SINAMICS project for all Control Units containing a FastIO component (onboard I/O of Control Unit with standard telegram 39x or TM1x) before reloading the corrected project.
16#FFFF8091	Job aborted. The function is not supported by the drive object with the specified logical address. This may, for example, be a TM15 or TM17 IO module.
16#FFFF8093	Job aborted. STW1BitSet could not be written to the output telegram due to illegal bits in STW1BitMask.
16#FFFF809F	Job aborted. Function cannot be executed in V4.1 and V4.1 SP1.

Parameters:

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

Logical base address of the output telegram for the drive object.

STW1BitMask

Direction:	Input parameter
Data type:	UINT
Parameter index:	2

Bit mask for selecting the bits to be written in the control word of the output telegram. Bits that are operated autonomously in the connection to the SIMOTION drive object (such as those for synchronization, fault handling, etc.) cannot be accessed by this function.

In V4.1 SP2, accessible bits of CU_STW in telegram 39x of SINAMICS DO1: 16#007E

STW1BitSet

Direction:	Input parameter
Data type:	UINT
Parameter index:	3

Bit-coded value for writing in the control word of the output telegram. Only those bits selected by the STW1BitMask are affected.

2.3.3.10 _writeDriveMultiParameter

This function enables up to 23 parameters to be written.

Exactly 23 parameters can only be written once every parameter only has a format with the maximum data length = 2 (e.g. WORD). If other formats with larger data lengths are written, the maximum number is reduced. The procedure is the same when reading parameters. A maximum of 39 parameters can be read if all parameters have the data length <= 2.

More information is provided in:

Profile Drive Technology PROFIdrive, Technical Specification for PROFIBUS and PROFINET, Version 4.1, May 2006 - Section 7.6.2.4, see Formulas

---

Note

- Up to 16 parameter jobs can be executed in parallel (the term parameter job also includes the functions _writeDriveParameter, _readDrive(Multi)ParameterDescription, _readDrive(Multi)Parameter, and _readDriveFaults).
- Only one parameter job is permitted per planned DP station. A new job to the same DP station is rejected with the error 16#FFFF81C7.
- If the system functions _readRecord and _writeRecord are used in addition to the parameter jobs, you must ensure that they are not to be executed simultaneously by a DP station. Reason: The same communication services are being used.
- If a parameter job (only asynchronous calls) is no longer to be processed, then it must be aborted with the ABORT_CURRENT_COMMAND transition condition. Otherwise no new job can be issued.

Parameters can be read out with the following data sets:

- Data set 47 is always read out for PROFIBUS (external or integrated), regardless of whether the function is called with a valid (0>=doId<=254) or invalid 'doId' (doId=255). If the passed 'doId' is invalid, it will be determined by the system itself. This requires the specification of the logical address of the axis.
- Two data sets are available for PROFINET:
  - Base Mode Parameter Access - local (data set 0xB02E)

Note

Determining the doID for SIMOTION versions V4.2 and older

For the SIMOTION versions <= V4.2, the automatic determination of 'dold' (if 'dold' is not specified or is invalid) does not take place for all device types in a completely profile-compliant manner. This affects drives which do not support the optional parameter P978 (e.g. SINAMICS G120), where the system in this case determines a 'dold' that is too small by 1 and the wrong DO is therefore accessed. To work around this error, transfer a correct, valid 'dold' during the call-up.

Note

MAP (module access point) address allocation with PROFINET

SIMOTION V4.1 SP2: MAP must always be at subslot 1 (HW Config configuration). The MAP submodule is the DO proxy and hosts the alarm channel and one or more PAPs (parameter access points).

Base Mode Parameter Access - global (data set 0xB02F)

This data set is used if a valid 'dold' (0>=doId<=254) is entered. Any valid MAP or address can be specified, as assignment only takes place on the basis of the 'dold'.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual,
- Section PROFIdrive-specific data types,
- Section Rules for SIMOTION _writeDrive.../_readDrive... commands
- or in the SIMOTION online help.

Return value: StructRetWriteDriveMultiParameter

StructRetWriteDriveMultiParameter
Return parameter for _writeDriveMultiParameter

Structure	Name	Data type	Unit
functionResult	Total job return value	DINT	-
parameterResultErrorExistent	A parameter-specific fault is present in the return value	BOOL	-
parameterResult	Parameter-specific return value	ARRAY [0..22] OF DINT	-

Description of the return values:

The return value consists of 'functionResult', 'parameterResultErrorExistent' and 'parameterResult[23]'.

'functionResult'

Reports the status of the total job plus errors during the transfer and internal errors.

16#00000000	Writing of parameters okay, job completed with positive result. Job okay.
16#00007001	Must be repeated in the next program cycle. Initial call, initiation of writing of parameters okay (only when command is issued asynchronously).
16#00007002	Must be repeated in the next program cycle. Intermediate call, writing of parameters is still active (only when command is issued asynchronously).
16#00007003	Writing of parameters aborted.

16#FFFF8110	Internal error, job aborted. The sum of the lengths specified by the user exceeds the maximum transferable length.
16#FFFF8111	Internal error, job aborted. The length specified by the user does not match the length calculated from the data type and number of elements in the case of at least one parameter.
16#FFFF8112	Internal error, job aborted. A valid format could not be found for at least one parameter.
16#FFFF8190	Internal error, job aborted. Specified logical base address invalid: No assignment is available in SDBs, or there is no base address.
16#FFFF8191	Internal error, job aborted. The _writeDriveMultiParameter function cannot reach the specified logical base address.
16#FFFF8192	Internal error, job aborted. Error in response identifier.
16#FFFF8193	Internal error, job aborted. The number of parameters to be read is not permissible.
16#FFFF819D	Internal error, job aborted. An I slave/I device interface is unable to issue a parameter job to the higher-level master/controller.
16#FFFF819E	Internal error, job aborted. Attempt to abort a non-active function.
16#FFFF819F	Internal error, job aborted. Function not executable.
16#FFFF81C3	Internal error, can be repeated in the next program cycle. Required resources are presently occupied: - In the _writeDriveMultiParameter function - In the module
16#FFFF81C5	Internal error, can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF81C7	Internal error, can be repeated in the next program cycle. Another parameter job has already been issued to the DP station. - based on the user program - from a system-internal component (e.g. TO axis executes parameter comparison after station recovery)
16#FFFF81CF	Internal error, can be repeated in the next program cycle. Another parameter job call is currently active under this 'commandId'.
16#FFFF8290	Corresponds to the error message 16#FFFF8190.
16#FFFF8291	Corresponds to the error message 16#FFFF8191.
16#FFFF82A2	Error during data set transfer, job aborted. Error in Layer2: - Station failure - Timeout
16#FFFF82A3	Error during data set transfer, job aborted. Error in user interface/user: - Protocol error - Station failure - Timeout

16#FFFF82A4-A7	Error during data set transfer, job aborted. Error numbers according to IEC 61158 reserved - Query the cause at the module's manufacturer
16#FFFF82A8	Error during data set transfer, job aborted. Error because of version conflict.
16#FFFF82A9	Error during data set transfer, job aborted. Function not supported by the module.
16#FFFF82AA-AF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82B0	Error during data set transfer, job aborted. Parameter jobs are not supported by the addressed module.
16#FFFF82B2	Error during data set transfer, job aborted. Module reports access to an invalid slot/subslot.
16#FFFF82B3	Error during data set transfer, job aborted. Module reports type conflict.
16#FFFF82B4	Error during data set transfer, job aborted. Module reports access to an invalid area.
16#FFFF82B5	Error during data set transfer, can be repeated in the next program cycle. The system function cannot be executed due to an internal operating state of the module.
16#FFFF82B6	Error during data set transfer, job aborted. Module rejects access.
16#FFFF82B7	Error during data set transfer, job aborted. The job could not be dispatched due to a job error.
16#FFFF82B8	Error during data set transfer, job aborted. Module reports an invalid parameter.
16#FFFF82B9	Error during data set transfer, job aborted. Module reports an invalid type.
16#FFFF82BA-BF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82C1	Error during data set transfer, command can be repeated immediately. Data of the preceding write job on the module for the same data set has not yet been processed by the module.
16#FFFF82C2	Error during data set transfer, command can be repeated immediately. The module is currently executing the maximum possible jobs for one CPU.
16#FFFF82C3	Error during data set transfer, command can be repeated immediately. Required resources are presently occupied: - In the _writeRecord function - In the module
16#FFFF82C4	Error during data set transfer, command can be repeated immediately. Communication errors: - Parity error - SW Ready not set - Error in block length administration - Checksum error on CPU side - Checksum error on module side
16#FFFF82C5	Error during data set transfer, can be repeated in the next program cycle. Distributed I/O not available.

16#FFFF82C6	Error during data set transfer, can be repeated in the next program cycle. Data set transfer has been aborted due to priority class abort (restart or background).

'parameterResultErrorExistent'

Returns TRUE when an error is reported in at least one 'parameterResult'.

'parameterResult'

Reports parameter-specific errors.

16#00000000	Parameter access completed without errors.
16#FFFF8000	Parameter error, job aborted. Access to a non-existent parameter.
16#FFFF8001	Parameter error, job aborted. Change access to a parameter that cannot be modified.
16#FFFF8002	Parameter error, job aborted. Change access with value outside value limits.
16#FFFF8003	Parameter error, job aborted. Access to a non-existent subindex.
16#FFFF8004	Parameter error, job aborted. Access with subindex to non-indexed parameter (not an array parameter).
16#FFFF8005	Parameter error, job aborted. Modification access with value that does not match the parameter data type.
16#FFFF8006	Parameter error, job aborted. Modification access with value other than 0 where this is not allowed.
16#FFFF800B	Parameter error, job aborted. Change access without parameter change rights.
16#FFFF8011	Parameter error, job aborted. Job cannot be executed due to operating state.
16#FFFF8014	Parameter error, job aborted. Modification access with value that is within value limits, but illegal for some other sustainable reason (parameter with defined individual values).
16#FFFF8016	Parameter error, job aborted. Illegal or unsupported value for attribute, number of elements, parameter number, or subindex, or a combination of these.
16#FFFF8017	Parameter error, job aborted. Illegal format.
16#FFFF8018	Parameter error, job aborted. Number of values of parameter data does not match the number of elements in the parameter address.
16#FFFF8019	Parameter error, job aborted. Access to a non-existing axis or an invalid drive object.
16#FFFF80xx	Parameter error, job aborted. Manufacturer-specific error: To determine the exact meaning of the error value 'xx' (20-FF), consult the associated manufacturer-dependent device manual.

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of logical base address of the drive. With INPUT, the logical address of the drive is in the input range. With OUTPUT, the logical address of the drive is in the output range. Diagnostic addresses are always of type INPUT.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical base address of the drive.

If the optional 'doId' parameter is also used, any station address (preferably the diagnostic address of the station) can be specified.

With PROFINET, parameter access occurs via the MAP (module access point) of a DO. As an alternative to the logical base address of the drive, specification of the diagnostic address of the associated MAP is recommended.

doId (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	3
System default:	255

Specifies the DO-ID to address a specific DO directly. 0 to 254 are permitted.

The DO-ID can be unspecified or specified as invalid (>254) under the following conditions:

- Access via the DO-ID is not supported by the DP slave/IO device (P978 is not implemented).
- Data set 0xB02F is not supported (PROFINET only).
- Access is to occur via the parameter access point of a DO (PROFINET only).

numberOfParameters

Direction:	Input parameter
Data type:	UDINT
Parameter index:	4

Number of parameters to be written. 1 to 23 permitted.

parameterNumber

Direction:	Input parameter
Data type:	ARRAY [0..22] OF UDINT
Parameter index:	5

Specifies the parameters from which the values are to be changed. 1 to 65535 permitted.

numberOfElements (optional)

Direction:	Input parameter
Data type:	ARRAY [0..22] OF UDINT
Parameter index:	6
System default:	[0..22] = 0

Number of elements to be modified from a parameter. Single parameter -> No need to specify elements. Array parameter -> 1 element under subindex. Array parameter -> 2 to n elements starting from subindex.

subIndex (optional)

Direction:	Input parameter
Data type:	ARRAY [0..22] OF UDINT
Parameter index:	7
System default:	[0..22] = 0

Specifies the subindex that addresses the first array element of an array parameter. 0 to 65535 permitted. Single parameter -> No need to specify elements.

dataType (optional)

Direction:	Input parameter
Data type:	ARRAY [0..22] OF UDINT
Parameter index:	8
System default:	[0..22] = 0

Specifies the data type of the parameter to be written.

Note

Further information on the structure of a data type can be found:

- In the SIMOTION Communication System Manual or
- At www.profibus.com, in the documentation "Profile Guidelines, Part 2: Data Types, Programming Languages and Platforms".

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	9

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies the command transition. With IMMEDIATELY, the transition takes place immediately after the command is issued (asynchronous call). With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	10

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND.

dataLength

Direction:	Input parameter
Data type:	ARRAY [0..22] OF UDINT
Parameter index:	12

Specifies the length of the data to be written (maximum 228 bytes in total).

data

Direction:	Input parameter
Data type:	ARRAY [0..227] OF BYTE
Parameter index:	13

Source area for the user data to be written. The user data must be stored in big endian format.

2.3.3.11 _writeDriveParameter

This function enables a drive parameter to be written.

Note

- Up to 16 parameter jobs can be executed in parallel (the term parameter job also includes the functions _writeDriveMultiParameter, _readDrive(Multi)ParameterDescription, _readDrive(Multi)Parameter and _readDriveFaults).
- Only one parameter job is permitted per planned DP station. A new job to the same DP station is rejected with the error 16#FFFF81C7.
- If the system functions _readRecord and _writeRecord are used in addition to the parameter jobs, you must ensure that they are not to be executed simultaneously by a DP station. Reason: The same communication services are used.
- If a parameter job (only asynchronous calls) is no longer to be processed, then it must be aborted with the ABORT_CURRENT_COMMAND transition condition. Otherwise no new job can be issued.

Parameters can be read out with the following data sets:

- Data set 47 is always read out for PROFIBUS (external or integrated), regardless of whether the function is called with a valid (0>=doId<=254) or invalid 'doId' (doId=255). If the passed 'doId' is invalid, it will be determined by the system itself. This requires the specification of the logical address of the axis.
- Two data sets are available for PROFINET:
  - Base Mode Parameter Access - local (data set 0xB02E)

Note

Determining the doID for SIMOTION versions V4.2 and older

For the SIMOTION versions <= V4.2, the automatic determination of 'dold' (if 'dold' is not specified or is invalid) does not take place for all device types in a completely profile-compliant manner. This affects drives which do not support the optional parameter P978 (e.g. SINAMICS G120), where the system in this case determines a 'dold' that is too small by 1 and the wrong DO is therefore accessed. To work around this error, transfer a correct, valid 'dold' during the call-up.

Note

MAP (module access point) address allocation with PROFINET

SIMOTION V4.1 SP2: MAP must always be at subslot 1 (HW Config configuration). The MAP submodule is the DO proxy and hosts the alarm channel and one or more PAPs (parameter access points).

Base Mode Parameter Access - global (data set 0xB02F)

This data set is used if a valid 'dold' (0>=doId<=254) is entered. Any valid MAP or address can be specified, as assignment only takes place on the basis of the 'dold'.

Additional references

Additional information is available at:

- SIMOTION Communication System Manual,
- Section PROFIdrive-specific data types,
- Section Rules for SIMOTION _writeDrive.../_readDrive... commands
- or in the SIMOTION online help.

Return value: StructRetWriteDriveParameter

StructRetWriteDriveParameter
Return parameter for _writeDriveParameter

Structure	Name	Data type	Unit
functionResult	Total job return value	DINT	-
parameterResult	Parameter-specific return value	DINT	-

Description of the return values:

The return value consists of 'functionResult' and 'parameterResult'.

'functionResult'

Reports the status of the total job plus errors during the transfer and internal errors.

16#00000000	Writing of parameters okay, job completed with positive result. Job okay.
16#00007001	Must be repeated in the next program cycle. Initial call, initiation of reading of parameter description okay (only when command is issued asynchronously).
16#00007002	Must be repeated in the next program cycle. Intermediate call, reading of parameter description is still active (only when command is issued asynchronously).
16#00007003	Writing of parameters aborted.
16#FFFF8110	Internal error, job aborted. The length specified by the user exceeds the maximum transferable length.

16#FFFF8111	Internal error, job aborted The length specified by the user does not match the length calculated from the data type and number of elements.
16#FFFF8112	Internal error, job aborted. No valid format could be found.
16#FFFF8190	Internal error, job aborted. Specified logical base address invalid: No assignment is available in SDBs, or there is no base address.
16#FFFF8191	Internal error, job aborted. The _writeDriveParameter function cannot reach the specified logical base address.
16#FFFF8192	Internal error, job aborted. Error in response identifier.
16#FFFF819D	Internal error, job aborted. An I slave/I device interface is unable to issue a parameter job to the higher-level master/controller.
16#FFFF819E	Internal error, job aborted. Attempt to abort a non-active function.
16#FFFF819F	Internal error, job aborted. Function not executable.
16#FFFF81C3	Internal error, can be repeated in the next program cycle. Required resources are presently occupied: - In the _writeDriveParameter function - In the module
16#FFFF81C5	Internal error, can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF81C7	Internal error, can be repeated in the next program cycle. Another parameter job has already been issued to the DP station. - based on the user program - from a system-internal component (e.g. TO axis executes parameter comparison after station recovery)
16#FFFF81CF	Internal error, can be repeated in the next program cycle. Another parameter job call is currently active under this 'commandId'.
16#FFFF8290	Corresponds to the error message 16#FFFF8190.
16#FFFF8291	Corresponds to the error message 16#FFFF8191.
16#FFFF82A2	Error during data set transfer, job aborted. Error in Layer2: - Station failure - Timeout
16#FFFF82A3	Error during data set transfer, job aborted. Error in user interface/user: - Protocol error - Station failure - Timeout
16#FFFF82A4-A7	Error during data set transfer, job aborted. Error numbers according to IEC 61158 reserved - Query the cause at the module's manufacturer
16#FFFF82A8	Error during data set transfer, job aborted. Error because of version conflict.
16#FFFF82A9	Error during data set transfer, job aborted. Function not supported by the module.

16#FFFF82AA-AF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82B0	Error during data set transfer, job aborted. Parameter jobs are not supported by the addressed module.
16#FFFF82B2	Error during data set transfer, job aborted. Module reports access to an invalid slot/subslot.
16#FFFF82B3	Error during data set transfer, job aborted. Module reports type conflict.
16#FFFF82B4	Error during data set transfer, job aborted. Module reports access to an invalid area.
16#FFFF82B5	Error during data set transfer, can be repeated in the next program cycle. The system function cannot be executed due to an internal operating state of the module.
16#FFFF82B6	Error during data set transfer, job aborted. Module rejects access.
16#FFFF82B7	Error during data set transfer, job aborted. The job could not be dispatched due to a job error.
16#FFFF82B8	Error during data set transfer, job aborted. Module reports an invalid parameter.
16#FFFF82B9	Error during data set transfer, job aborted. Module reports an invalid type.
16#FFFF82BA-BF	Error during data set transfer, job aborted. Error numbers according to IEC 61158 manufacturer-specific - Query the cause at the module's manufacturer
16#FFFF82C1	Error during data set transfer, command can be repeated immediately. Data of the preceding write job on the module for the same data set has not yet been processed by the module.
16#FFFF82C2	Error during data set transfer, command can be repeated immediately. The module is currently executing the maximum possible jobs for one CPU.
16#FFFF82C3	Error during data set transfer, command can be repeated immediately. Required resources are presently occupied: - In the _writeRecord function - In the module
16#FFFF82C4	Error during data set transfer, command can be repeated immediately. Communication errors: - Parity error - SW Ready not set - Error in block length administration - Checksum error on CPU side - Checksum error on module side
16#FFFF82C5	Error during data set transfer, can be repeated in the next program cycle. Distributed I/O not available.
16#FFFF82C6	Error during data set transfer, can be repeated in the next program cycle. Data set transfer has been aborted due to priority class abort (restart or background).

'parameterResult'

Reports parameter-specific errors.

16#00000000	Parameter access completed without errors.

16#FFFF8000	Parameter error, job aborted. Access to a non-existent parameter.
16#FFFF8001	Parameter error, job aborted. Change access to a parameter that cannot be modified.
16#FFFF8002	Parameter error, job aborted. Change access with value outside value limits.
16#FFFF8003	Parameter error, job aborted. Access to a non-existent subindex.
16#FFFF8004	Parameter error, job aborted. Access with subindex to non-indexed parameter (not an array parameter).
16#FFFF8005	Parameter error, job aborted. Modification access with value that does not match the parameter data type.
16#FFFF8006	Parameter error, job aborted. Modification access with value other than 0 where this is not allowed.
16#FFFF800B	Parameter error, job aborted. Change access without parameter change rights.
16#FFFF8011	Parameter error, job aborted. Job cannot be executed due to operating state.
16#FFFF8014	Parameter error, job aborted. Modification access with value that is within value limits, but illegal for some other sustainable reason (parameter with defined individual values).
16#FFFF8016	Parameter error, job aborted. Illegal or unsupported value for attribute, number of elements, parameter number, or subindex, or a combination of these.
16#FFFF8017	Parameter error, job aborted. Illegal format.
16#FFFF8018	Parameter error, job aborted. Number of values of parameter data does not match the number of elements in the parameter address.
16#FFFF8019	Parameter error, job aborted. Access to a non-existing axis or an invalid drive object.
16#FFFF80xx	Parameter error, job aborted. Manufacturer-specific error: To determine the exact meaning of the error value 'xx' (20-FF), consult the associated manufacturer-dependent device manual.

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of logical base address of the drive. With INPUT, the logical address of the drive is in the input range. With OUTPUT, the logical address of the drive is in the output range. Diagnostic addresses are always of type INPUT.

logAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical base address of the drive.

If the optional 'doId' parameter is also used, any station address (preferably the diagnostic address of the station) can be specified.

With PROFINET, parameter access occurs via the MAP (module access point) of a DO. As an alternative to the logical base address of the drive, specification of the diagnostic address of the associated MAP is recommended.

doId (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	3
System default:	255

Specifies the DO-ID to address a specific DO directly. 0 to 254 are permitted. Under the following conditions, the DO-ID may not be specified, or specified with an invalid value (>254): - Access via the DO-ID is not supported by the DP slave / I/O device (P978 not implemented). - Data set 0xB02F is not supported (PROFINET only). - Access is to be performed via the Parameter Access Point of a DO (PROFINET only).

parameterNumber

Direction:	Input parameter
Data type:	UDINT
Parameter index:	4

Specifies the parameter numbers from which the values are to be written. 1 to 65535 permitted.

numberOfElements (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	5
System default:	0

Number of elements that are to be written from a parameter. Single parameter -> No need to specify elements. Array parameter -> 1 element under subindex. Array parameter -> 2 to n elements starting from subindex.

subIndex (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	6
System default:	0

Specifies the subindex that addresses the first array element of the parameter. 0 to 65535 permitted.

dataType (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	7
System default:	0

Specifies the data type of the parameter to be written.

Note

Further information on the structure of a data type can be found:

- In the SIMOTION Communication System Manual or
- At www.profibus.com, in the documentation "Profile Guidelines, Part 2: Data Types, Programming Languages and Platforms".

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	8

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies the command transition. With IMMEDIATELY, the transition takes place immediately after the command is issued (asynchronous call). With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	9

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND.

dataLength

Direction:	Input parameter
Data type:	UDINT
Parameter index:	11

Specifies the length of the data to be written (maximum 228 bytes).

data

Direction:	Input parameter
Data type:	ARRAY [0..227] OF BYTE
Parameter index:	12

Source area for the user data to be written. The user data must be stored in big endian format.

2.3.4 SIMATIC

2.3.4.1 _ET200S_FC_control

The _ET200S_FC_control function block can be used to exchange cyclical data with the ET200S frequency converter. When requested, parameters are read from or written to the ET200S frequency converter and errors acknowledged.

Note

The FB _ET200S_FC_control can only be used to control ET200S frequency converters with the MLFB 6SL3244-0SA00-1AA0.

The FB _ET200S_FC_control cannot be used for the ET200S frequency converter with the MLFB 6SL3244-0SA00-1AA1. Use standard telegram 1 here.

Hardware/software requirements:

ICU24 hardware version: 6SL3244-0SA00-1AA0 only. ICU24 software version: as of V1.02

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myET200S_FC_control : _ET200S_FC_control; // instance of FB (_ET200S_FC_control)
myET200S_FC_control(
execute := myExecute, // [IN] Anstoss Datentransfer (BOOL)
mode := myMode, // [IN] Voreinstellung (enum_ET200S_FC_mode)
pzdIn := myPzdIn, // [IN] Peripherieeingaenge (ARRAY[0..7] OF BYTE)
setpoint := mySetpoint, // [IN] Drehzahlsollwert (DINT)
ctrlDword := myCtrlDword, // [IN] Steuerwort (DWORD)
paraNumber := myParaNumber, // [IN] Parameternummer (UINT)
subIndex := mySubIndex, // [IN] Index (UINT)
paraValueWr := myParaValueWr // [IN] Parameternummer (REAL)
);
```

2.4 I/O modules

2.4.1 AS-Interface

2.4.1.1 _ASI_cmdInterface

You can use the _ASI_cmdInterface function block to easily and completely control the AS-Interface master behavior via your user program. The _ASI_cmdInterface FB transfers the commands and data to the AS-Interface master and transfers the provided response data. The calls for reading and writing of data sets are managed in the _ASI_cmdInterface FB.

Hardware/software requirements:

LINK 20E hardware version: 6GK1 415-2AA01 as of product version 08. Software version: as of V2.37

CP342-2P hardware version: 6GK7 343-2AH10-0XA0 after product version 03. Software version: as of V2.24

DP/AS-i LINK Advanced hardware version: 6GK1 415-2BA10 after product version 01. Software version: V1.0.5

IE/AS-i LINK PN IO hardware version: 6GK1 411-2BA10 after product version 01. Software version: V1.0.0

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myASI_cmdInterface : _ASI_cmdInterface; // instance of FB (_ASI_cmdInterface)
myASI_cmdInterface(
execute := myExecute, // [IN] Initiating a new request with rising edge (BOOL)
reset := myReset, // [IN] TRUE = reset/reinitialization with rising edge (BOOL)
periIn := myPeriIn, // [IN] Control information of the AS-Interface master for _ASI_cmdInterface FB
(see section "Addressing of the AS-Interface Masters (CP 343-2 P, DP/AS-Interface Link 20E)" (BYTE)
moduleAddress := myModuleAddress, // [IN] Module address from HW Config (DINT)
sendData := mySendData, // [IN] Data to be sent (command number, possibly parameters) (ARRAY[0..239]
OF USINT)
sendLength := mySendLength // [IN] Length of the data to be sent (depends on the command to be sent)
(UDINT)
);
```

2.4.1.2 _ASI_rdAsiMonDiagnostic

The _ASI_rdAsiMonDiagnostic FB is a function block for reading out the diagnostic information of the ASIsafe safety monitor. The diagnostic information can be used to determine the status of the enabling circuits of the safety monitor and the safety-oriented slaves. For this purpose, the _ASI_rdAsiMonDiagnostic FB continuously reads out and evaluates the diagnostic information provided by the safety monitor and made available in data structure Struct_ASI_diagAsiMon.

Hardware/software requirements:

LINK 20E hardware version: 6GK1 415-2AA01 as of product version 08. Software version: as of V2.37

CP342-2P hardware version: 6GK7 343-2AH10-0XA0 after product version 03. Software version: as of V2.24

DP/AS-i LINK Advanced hardware version: 6GK1 415-2BA10 after product version 01. Software version: V1.0.5

IE/AS-i LINK PN IO hardware version: 6GK1 411-2BA10 after product version 01. Software version: V1.0.0

ASIsafe Monitor 3RK1105-1BE04-2CA0 (enhanced safety monitor, 2 enabling circuits) 3RK1105-1BE04-0CA0 (safety monitor, 2 enabling circuits) 3RK1105-1AE04-2CA0 (enhanced safety monitor, 1 enabling circuit) 3RK1105-1AE04-0CA0 (safety monitor, 1 enabling circuit)

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myASI_rdAsiMonDiagnostic : _ASI_rdAsiMonDiagnostic; // instance of FB (_ASI_rdAsiMonDiagnostic)
myASI_rdAsiMonDiagnostic(
enable := myEnable, // [IN] TRUE = continuous read out of the diagnostic information, FALSE = the
initialization of the ASIsafe safety monitor is run through once (BOOL)
reset := myReset, // [IN] Rising edge = reset of FB _ASI_rdAsiMonDiagnostic (BOOL)
periIn0 := myPeriIn0, // [IN] I/O variable with bit address of the ASIsafe safety monitor (BOOL)
periIn1 := myPeriIn1, // [IN] I/O variable with bit address of the ASIsafe safety monitor (BOOL)
periIn2 := myPeriIn2, // [IN] I/O variable with bit address of the ASIsafe safety monitor (BOOL)
periIn3 := myPeriIn3, // [IN] I/O variable with bit address of the ASIsafe safety monitor (BOOL)
allDevices := myAllDevices, // [IN] Data format of the diagnostics data (depends on the ASIsafe safety
monitor version and the asimon configuration program) (BOOL)
asiSlaveError := myAsiSlaveError, // [IN] The parameter must be set (TRUE) when an AS-Interface slave
has failed. (BOOL)
asiPowerError := myAsiPowerError, // [IN] The parameter must be set (TRUE) when the AS-Interface power
supply has failed. (BOOL)
profibusError := myProfibusError, // [IN] The parameter must be set (TRUE) when the communication with
the AS-Interface master has been interrupted. (BOOL)
diagStop := myDiagStop, // [IN] TRUE = activation of the diagnosis stop functionality in FB
_ASI_rdAsiMonDiagnostic (corresponding to the asimon configuration program) (BOOL)
diagStopInvert := myDiagStopInvert, // [IN] TRUE = value on output parameter periOutDiagStop is output
inverted (corresponding to the asimon configuration program) (BOOL)
freezeAuto := myFreezeAuto, // [IN] TRUE = control of the diagnosis stop condition via FB
_ASI_rdAsiMonDiagnostic (BOOL)
freezeManual := myFreezeManual, // [IN] TRUE = manual activation of the diagnosis stop condition (BOOL)
unfreezeIfCh1Off := myUnfreezeIfCh1Off, // [IN] TRUE = diagnosis stop condition is also deactivated
(unfreeze) when the enabling circuit 1 has shut down (independent of the status in enabling circuit 2)
(BOOL)
unfreezeIfCh2Off := myUnfreezeIfCh2Off, // [IN] TRUE = diagnosis stop condition is also deactivated
(unfreeze) when the enabling circuit 2 has shut down (independent of the status in enabling circuit 1)
(BOOL)
unfreezeIfDeviceWait := myUnfreezeIfDeviceWait, // [IN] TRUE = the diagnosis stop condition is also
deactivated periodically (unfreeze) when it is detected in the running diagnostic sequence that at
least one device is in the "Wait" state (status = 2) (BOOL)
diagDataAsiMonitor := myASI_diagAsiMon, // [IN_OUT] Data structure for storing the diagnostic
information of the ASIsafe safety monitor (struct_ASI_diagAsiMon)
ch1Tripped := myCh1Tripped, // [IN_OUT] TRUE = status change from ON to OFF in enabling circuit 1
(BOOL)
ch2Tripped := myCh2Tripped // [IN_OUT] TRUE = status change from ON to OFF in enabling circuit 2 (BOOL)
);
```

2.4.2 Function modules - FM350-1

2.4.2.1 _FM3501_control2

The _FM3501_control2 function block can be used to control the module and to query the status of the FM 350-1.

Hardware/software requirements:

Hardware version: 6ES7 350-1AH02-0AE0 as of product version E02, 6ES7 350-1AH03-0AE0 as of product version E01. Software version: none

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myFM3501_control : _FM3501_control2;     // instance of FB (_FM3501_control2)
myFM3501_control(
enableSwGate := myEnableSwGate, // [IN] software gate (BOOL)
enableStopGate := myEnableStopGate, // [IN] gate stop (BOOL)
execResetOpError := myExecResetOpError, // [IN] operator error acknowledge (BOOL)
cntrRange := myCntrRange, // [IN] Range of counter value 0: -2exp31<x<2exp31, 1: 0< x< 2exp32 (BOOL)
setOutput0 := mySetOutput0 // [IN] set digital output 0
setOutput1 := mySetOutput1 // [IN] set digital output 1
periIn := myPeriIn, // [IN] peripheral input (ARRAY[0..15] OF BYTE)
data := myFM3501_fmData, // [IN_OUT] datastruct for FM 350-1 - counter module (Struct_FM3501_fmData)
setStartValue := mySetStartValue, // [IN_OUT] load direct (BOOL)
setPrepStartValue := mySetPrepStartValue, // [IN_OUT] load prepared (BOOL)
setCmpValue1 := mySetCmpValue1, // [IN_OUT] transfer comparator value 1 (BOOL)
setCmpValue2 := mySetCmpValue2, // [IN_OUT] transfer comparator value 2 (BOOL)
resetSyncState := myResetSyncState, // [IN_OUT] reset synchronisation bit (BOOL)
resetCntrState := myResetCntrState, // [IN_OUT] reset zero crossing bit (BOOL)
setParaOutput := mySetParaOutput // [IN_OUT]    set parameter for digital output 0/1
periOut := myPeriOut // [IN_OUT] peripheral output (ARRAY[0..15] OF BYTE)
);

// ***********************
// Struct_FM3501_fmData
// ***********************
// myFM3501_fmData : Struct_FM3501_fmData;
myFM3501_fmData.moduleAddress := 256; // [IN] module adress (wu)
myFM3501_fmData.loadValue1 := 0; // [IN] new load value (wu), count range -2exp31<x<2exp31
myFM3501_fmData.cmpValue1_1 := 0; // [IN] new comparator value 1 (wu), count range -2exp31<x<2exp31
myFM3501_fmData.cmpValue2_1 := 0; // [IN] new comparator value 2 (wu), count range -2exp31<x<2exp31
myFM3501_fmData.loadValue2 := 0; // [IN] new load value (wu), count range 0< x< 2exp32
myFM3501_fmData.cmpValue1_2 := 0; // [IN] new comparator value 1 (wu), count range 0< x< 2exp32
myFM3501_fmData.cmpValue2_2 := 0; // [IN] new comparator value 2 (wu), count range 0< x< 2exp32
myFM3501_fmData.enableTestMode := FALSE; // [IN] test free (internal use)
myFM3501_fmData.enableReverseSetting := FALSE; // [IN] enable set direction up (=forward) (wu)
myFM3501_fmData.enableForwardSetting := FALSE; // [IN] enable set direction dn (=backward) (wu)
myFM3501_fmData.enableOutput0 := FALSE; // [IN] control digital output DQ0 (wu)
myFM3501_fmData.enableOutput1 := FALSE; // [IN] control digital output DQ1 (wu)
myFM3501_fmData.configMeasOut0 := 0; // [IN] configuration data for digital output0 in operating mode
measuring
myFM3501_fmData.configCntrOut0 := 0; // [IN] configuration data for digital output0 in operating mode
counting
myFM3501_fmData.configCntrOut1 := 0; // [IN] configuration data for digital output1 in operating mode
counting
myFM3501_fmData.setCntrHyst := 0; // [IN] hysteresis in values of counter
myFM3501_fmData.setCntrPulse := 0; // [IN] duration of pulse [ms]
```

2.4.2.2 _FM3501_diagnostic

The _FM3501_diagnostic function block can be used to read out the complete diagnostic data from the FM 350-1.

Hardware/software requirements:

Hardware version: 6ES7 350-1AH02-0AE0 as of product version E02, 6ES7 350-1AH03-0AE0 as of product version E01. Software version: None. The following new functions of the FM350-1 module (order no.:6ES7 350-1AH03-0AE0) are not supported: - clock synchronism - measuring modes (frequency measurement, speed measurement, period duration measurement)

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myFM3501_diagnostic : _FM3501_diagnostic; // instance of FB (_FM3501_diagnostic)
myFM3501_diagnostic(
data := myFM3501_fmData, // [IN_OUT] datastruct for FM 350-1 - counter module (Struct_FM3501_fmData)
execute := myExecute // [IN_OUT] initiate diagnostic interrupt (BOOL)
);

// **********************
// Struct_FM3501_fmData
// **********************
// myFM3501_fmData : Struct_FM3501_fmData;
myFM3501_fmData.moduleAddress := 256; // [IN] module adress (wu)
myFM3501_fmData.loadValue1 := 0; // [IN] new load value (wu), count range -2exp31<x<2exp31
myFM3501_fmData.cmpValue1_1 := 0; // [IN] new comparator value 1 (wu), count range -2exp31<x<2exp31
myFM3501_fmData.cmpValue2_1 := 0; // [IN] new comparator value 2 (wu), count range -2exp31<x<2exp31
myFM3501_fmData.loadValue2 := 0; // [IN] new load value (wu), count range 0< x< 2exp32
myFM3501_fmData.cmpValue1_2 := 0; // [IN] new comparator value 1 (wu), count range 0< x< 2exp32
myFM3501_fmData.cmpValue2_2 := 0; // [IN] new comparator value 2 (wu), count range 0< x< 2exp32
myFM3501_fmData.enableTestMode := FALSE; // [IN] test free (internal use)
myFM3501_fmData.enableReverseSetting := FALSE; // [IN] enable set direction up (=forward) (wu)
myFM3501_fmData.enableForwardSetting := FALSE; // [IN] enable set direction dn (=backward) (wu)
myFM3501_fmData.enableOutput0 := FALSE; // [IN] control digital output DQ0 (wu)
myFM3501_fmData.enableOutput1 := FALSE; // [IN] control digital output DQ1 (wu)
```

2.4.3 Function modules - FM350-2

2.4.3.1 _FM3502_control

The _FM3502_control function block can be used to control the module and to query the status of the FM 350-2.

Hardware/software requirements:

Hardware version: 6ES7 350-2AH00-0AE0 as of product version E04. Software version: none

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myFM3502_control : _FM3502_control; // instance of FB (_FM3502_control)
myFM3502_control(
periIn := myPeriIn, // [IN] Transfers I/O Inputs of the FM to the FB (ARRAY[0..15] of BYTE)
periOut := myPeriOut, // [IN_OUT] Prepared data of the FB for the I/O outputs of the FM (ARRAY[0..15]
of BYTE)
data := myFM3502_fmData // [IN_OUT] Data structure whith channel-specific data (Struct_FM3502_fmData)
);

// ***************************************
// Struct_FM3502_wrJob   // write commands
// ***************************************
myFM3502_fmData.write.execJobNumber :=0; // [IN] number     (BYTE)

// ***************************************
// Struct_FM3502_rdJob   // read commands
// ***************************************
myFM3502_fmData.read.execJobNumber :=0; // [IN] number    (BYTE)

// ************************************************
// Struct_FM3502_controlSignals  // control signals
// ************************************************
myFM3502_fmData.control.enableOutput0 := FALSE; // [IN] control digital output DQ0  (BOOL)
myFM3502_fmData.control.enableOutput1 := FALSE; // [IN] control digital output DQ1  (BOOL)
myFM3502_fmData.control.enableOutput2 := FALSE; // [IN] control digital output DQ2  (BOOL)
myFM3502_fmData.control.enableOutput3 := FALSE; // [IN] control digital output DQ3  (BOOL)
myFM3502_fmData.control.enableOutput4 := FALSE; // [IN] control digital output DQ4  (BOOL)
myFM3502_fmData.control.enableOutput5 := FALSE; // [IN] control digital output DQ5  (BOOL)
myFM3502_fmData.control.enableOutput6 := FALSE; // [IN] control digital output DQ6  (BOOL)
myFM3502_fmData.control.enableOutput7 := FALSE; // [IN] control digital output DQ7  (BOOL)
myFM3502_fmData.control.setOutput0 := FALSE; // [IN] set digital output DQ0    (BOOL)
myFM3502_fmData.control.setOutput1 := FALSE; // [IN] set digital output DQ1    (BOOL)
myFM3502_fmData.control.setOutput2 := FALSE; // [IN] set digital output DQ2    (BOOL)
myFM3502_fmData.control.setOutput3 := FALSE; // [IN] set digital output DQ3    (BOOL)
myFM3502_fmData.control.setOutput4 := FALSE; // [IN] set digital output DQ4    (BOOL)
myFM3502_fmData.control.setOutput5 := FALSE; // [IN] set digital output DQ5    (BOOL)
myFM3502_fmData.control.setOutput6 := FALSE; // [IN] set digital output DQ6    (BOOL)
myFM3502_fmData.control.setOutput7 := FALSE; // [IN] set digital output DQ7    (BOOL)
myFM3502_fmData.control.enableSwGate0 := FALSE; // [IN] software gate 0            (BOOL)
myFM3502_fmData.control.enableSwGate1 := FALSE; // [IN] software gate 1            (BOOL)
myFM3502_fmData.control.enableSwGate2 := FALSE; // [IN] software gate 2            (BOOL)
myFM3502_fmData.control.enableSwGate3 := FALSE; // [IN] software gate 3            (BOOL)
myFM3502_fmData.control.enableSwGate4 := FALSE; // [IN] software gate 4            (BOOL)
myFM3502_fmData.control.enableSwGate5 := FALSE; // [IN] software gate 5            (BOOL)
myFM3502_fmData.control.enableSwGate6 := FALSE; // [IN] software gate 6            (BOOL)
myFM3502_fmData.control.enableSwGate7 := FALSE; // [IN] software gate 7            (BOOL)

// ***************************************
//  Struct_FM3502_fmData
// ***************************************
// myFM3502_fmData : Struct_FM3502_fmData;
myFM3502_fmData.moduleAddress :=256; // [IN] module address           (INT)
myFM3502_fmData.loadValue0 :=0; // [IN] load value counter 0           (DINT)
myFM3502_fmData.loadValue1 :=0; // [IN] load value counter 1           (DINT)
myFM3502_fmData.loadValue2 :=0; // [IN] load value counter 2           (DINT)
myFM3502_fmData.loadValue3 :=0; // [IN] load value counter 3           (DINT)
myFM3502_fmData.loadValue4 :=0; // [IN] load value counter 4           (DINT)
```

```
myFM3502_fmData.loadValue5 :=0; // [IN] load value counter 5            (DINT)
myFM3502_fmData.loadValue6 :=0; // [IN] load value counter 6            (DINT)
myFM3502_fmData.loadValue7 :=0; // [IN] load value counter 7            (DINT)
myFM3502_fmData.prepValue0 :=0; // [IN] load prepare value counter 0        (DINT)
myFM3502_fmData.prepValue1 :=0; // [IN] load prepare value counter 1        (DINT)
myFM3502_fmData.prepValue2 :=0; // [IN] load prepare value counter 2        (DINT)
myFM3502_fmData.prepValue3 :=0; // [IN] load prepare value counter 3        (DINT)
myFM3502_fmData.prepValue4 :=0; // [IN] load prepare value counter 4        (DINT)
myFM3502_fmData.prepValue5 :=0; // [IN] load prepare value counter 5        (DINT)
myFM3502_fmData.prepValue6 :=0; // [IN] load prepare value counter 6        (DINT)
myFM3502_fmData.prepValue7 :=0; // [IN] load prepare value counter 7        (DINT)
myFM3502_fmData.cmpValue0 :=0; // [IN] comparator value 0           (DINT)
myFM3502_fmData.cmpValue1 :=0; // [IN] comparator value 1           (DINT)
myFM3502_fmData.cmpValue2 :=0; // [IN] comparator value 2           (DINT)
myFM3502_fmData.cmpValue3 :=0; // [IN] comparator value 3           (DINT)
myFM3502_fmData.cmpValue4 :=0; // [IN] comparator value 4           (DINT)
myFM3502_fmData.cmpValue5 :=0; // [IN] comparator value 5           (DINT)
myFM3502_fmData.cmpValue6 :=0; // [IN] comparator value 6           (DINT)
myFM3502_fmData.cmpValue7 :=0; // [IN] comparator value 7           (DINT)
```

2.4.3.2 _FM3502_diagnostic

The _FM3502_diagnostic function block can be used to read out the complete diagnostic data from the FM 350-2.

Hardware/software requirements:

Hardware version: 6ES7 350-2AH00-0AE0 as of product version E04. Software version: none

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myFM3502_diagnostic : _FM3502_diagnostic; // instance of FB (_FM3502_diagnostic)
myFM3502_diagnostic(
enable := myEnable, // [IN] enable (BOOL)
data := myFM3502_fmData // [IN_OUT] Data structure whith channel-specific data (Struct_FM3502_fmData)
);

// ***************************************
// Struct_FM3502_wrJob   // write commands
// ***************************************
myFM3502_fmData.write.execJobNumber :=0; // [IN] number     (BYTE)

// ***************************************
// Struct_FM3502_rdJob   // read commands
// ***************************************
myFM3502_fmData.read.execJobNumber :=0; // [IN] number    (BYTE)

// ***********************************************
// Struct_FM3502_controlSignals  // control signals
// ***********************************************
myFM3502_fmData.control.enableOutput0 := FALSE; // [IN] control digital output DQ0  (BOOL)
myFM3502_fmData.control.enableOutput1 := FALSE; // [IN] control digital output DQ1  (BOOL)
myFM3502_fmData.control.enableOutput2 := FALSE; // [IN] control digital output DQ2  (BOOL)
myFM3502_fmData.control.enableOutput3 := FALSE; // [IN] control digital output DQ3  (BOOL)
myFM3502_fmData.control.enableOutput4 := FALSE; // [IN] control digital output DQ4  (BOOL)
myFM3502_fmData.control.enableOutput5 := FALSE; // [IN] control digital output DQ5  (BOOL)
myFM3502_fmData.control.enableOutput6 := FALSE; // [IN] control digital output DQ6  (BOOL)
myFM3502_fmData.control.enableOutput7 := FALSE; // [IN] control digital output DQ7  (BOOL)
myFM3502_fmData.control.setOutput0 := FALSE; // [IN] set digital output DQ0    (BOOL)
myFM3502_fmData.control.setOutput1 := FALSE; // [IN] set digital output DQ1    (BOOL)
myFM3502_fmData.control.setOutput2 := FALSE; // [IN] set digital output DQ2    (BOOL)
myFM3502_fmData.control.setOutput3 := FALSE; // [IN] set digital output DQ3    (BOOL)
myFM3502_fmData.control.setOutput4 := FALSE; // [IN] set digital output DQ4    (BOOL)
myFM3502_fmData.control.setOutput5 := FALSE; // [IN] set digital output DQ5    (BOOL)
myFM3502_fmData.control.setOutput6 := FALSE; // [IN] set digital output DQ6    (BOOL)
myFM3502_fmData.control.setOutput7 := FALSE; // [IN] set digital output DQ7    (BOOL)
myFM3502_fmData.control.enableSwGate0 := FALSE; // [IN] software gate 0            (BOOL)
myFM3502_fmData.control.enableSwGate1 := FALSE; // [IN] software gate 1            (BOOL)
myFM3502_fmData.control.enableSwGate2 := FALSE; // [IN] software gate 2            (BOOL)
myFM3502_fmData.control.enableSwGate3 := FALSE; // [IN] software gate 3            (BOOL)
myFM3502_fmData.control.enableSwGate4 := FALSE; // [IN] software gate 4            (BOOL)
myFM3502_fmData.control.enableSwGate5 := FALSE; // [IN] software gate 5            (BOOL)
myFM3502_fmData.control.enableSwGate6 := FALSE; // [IN] software gate 6            (BOOL)
myFM3502_fmData.control.enableSwGate7 := FALSE; // [IN] software gate 7            (BOOL)

// ***************************************
//  Struct_FM3502_fmData
// ***************************************
// myFM3502_fmData : Struct_FM3502_fmData;
myFM3502_fmData.moduleAddress :=256; // [IN] module address           (INT)
myFM3502_fmData.loadValue0 :=0; // [IN] load value counter 0           (DINT)
myFM3502_fmData.loadValue1 :=0; // [IN] load value counter 1           (DINT)
myFM3502_fmData.loadValue2 :=0; // [IN] load value counter 2           (DINT)
myFM3502_fmData.loadValue3 :=0; // [IN] load value counter 3           (DINT)
myFM3502_fmData.loadValue4 :=0; // [IN] load value counter 4           (DINT)
myFM3502_fmData.loadValue5 :=0; // [IN] load value counter 5           (DINT)
myFM3502_fmData.loadValue6 :=0; // [IN] load value counter 6           (DINT)
```

```
myFM3502_fmData.loadValue7 :=0; // [IN] load value counter 7            (DINT)
myFM3502_fmData.prepValue0 :=0; // [IN] load prepare value counter 0        (DINT)
myFM3502_fmData.prepValue1 :=0; // [IN] load prepare value counter 1        (DINT)
myFM3502_fmData.prepValue2 :=0; // [IN] load prepare value counter 2        (DINT)
myFM3502_fmData.prepValue3 :=0; // [IN] load prepare value counter 3        (DINT)
myFM3502_fmData.prepValue4 :=0; // [IN] load prepare value counter 4        (DINT)
myFM3502_fmData.prepValue5 :=0; // [IN] load prepare value counter 5        (DINT)
myFM3502_fmData.prepValue6 :=0; // [IN] load prepare value counter 6        (DINT)
myFM3502_fmData.prepValue7 :=0; // [IN] load prepare value counter 7        (DINT)
myFM3502_fmData.cmpValue0 :=0; // [IN] comparator value 0            (DINT)
myFM3502_fmData.cmpValue1 :=0; // [IN] comparator value 1            (DINT)
myFM3502_fmData.cmpValue2 :=0; // [IN] comparator value 2            (DINT)
myFM3502_fmData.cmpValue3 :=0; // [IN] comparator value 3            (DINT)
myFM3502_fmData.cmpValue4 :=0; // [IN] comparator value 4            (DINT)
myFM3502_fmData.cmpValue5 :=0; // [IN] comparator value 5            (DINT)
myFM3502_fmData.cmpValue6 :=0; // [IN] comparator value 6            (DINT)
myFM3502_fmData.cmpValue7 :=0; // [IN] comparator value 7            (DINT)
```

2.4.3.3 _FM3502_read

The _FM3502_read function block is used to read out the counter values and measured values of the FM 350-2.

Hardware/software requirements:

Hardware version: 6ES7 350-2AH00-0AE0 as of product version E04. Software version: none

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myFM3502_read : _FM3502_read; // instance of FB (_FM3502_read)
myFM3502_read(
periIn := myPeriIn, // [IN] Transfers I/O Inputs of the FM to the FB (ARRAY[0..15] of BYTE)
data := myFM3502_fmData // [IN_OUT] Data structure whith channel-specific data (Struct_FM3502_fmData)
);

// ************************************
// Struct_FM3502_wrJob   // write commands
// ************************************
myFM3502_fmData.write.execJobNumber :=0; // [IN] number     (BYTE)

// ************************************
// Struct_FM3502_rdJob   // read commands
// ************************************
myFM3502_fmData.read.execJobNumber :=0; // [IN] number    (BYTE)

// ***********************************************
// Struct_FM3502_controlSignals  // control signals
// ***********************************************
myFM3502_fmData.control.enableOutput0 := FALSE; // [IN] control digital output DQ0  (BOOL)
myFM3502_fmData.control.enableOutput1 := FALSE; // [IN] control digital output DQ1  (BOOL)
myFM3502_fmData.control.enableOutput2 := FALSE; // [IN] control digital output DQ2  (BOOL)
myFM3502_fmData.control.enableOutput3 := FALSE; // [IN] control digital output DQ3  (BOOL)
myFM3502_fmData.control.enableOutput4 := FALSE; // [IN] control digital output DQ4  (BOOL)
myFM3502_fmData.control.enableOutput5 := FALSE; // [IN] control digital output DQ5  (BOOL)
myFM3502_fmData.control.enableOutput6 := FALSE; // [IN] control digital output DQ6  (BOOL)
myFM3502_fmData.control.enableOutput7 := FALSE; // [IN] control digital output DQ7  (BOOL)
myFM3502_fmData.control.setOutput0 := FALSE; // [IN] set digital output DQ0    (BOOL)
myFM3502_fmData.control.setOutput1 := FALSE; // [IN] set digital output DQ1    (BOOL)
myFM3502_fmData.control.setOutput2 := FALSE; // [IN] set digital output DQ2    (BOOL)
myFM3502_fmData.control.setOutput3 := FALSE; // [IN] set digital output DQ3    (BOOL)
myFM3502_fmData.control.setOutput4 := FALSE; // [IN] set digital output DQ4    (BOOL)
myFM3502_fmData.control.setOutput5 := FALSE; // [IN] set digital output DQ5    (BOOL)
myFM3502_fmData.control.setOutput6 := FALSE; // [IN] set digital output DQ6    (BOOL)
myFM3502_fmData.control.setOutput7 := FALSE; // [IN] set digital output DQ7    (BOOL)
myFM3502_fmData.control.enableSwGate0 := FALSE; // [IN] software gate 0            (BOOL)
myFM3502_fmData.control.enableSwGate1 := FALSE; // [IN] software gate 1            (BOOL)
myFM3502_fmData.control.enableSwGate2 := FALSE; // [IN] software gate 2            (BOOL)
myFM3502_fmData.control.enableSwGate3 := FALSE; // [IN] software gate 3            (BOOL)
myFM3502_fmData.control.enableSwGate4 := FALSE; // [IN] software gate 4            (BOOL)
myFM3502_fmData.control.enableSwGate5 := FALSE; // [IN] software gate 5            (BOOL)
myFM3502_fmData.control.enableSwGate6 := FALSE; // [IN] software gate 6            (BOOL)
myFM3502_fmData.control.enableSwGate7 := FALSE; // [IN] software gate 7            (BOOL)

// ************************************
//  Struct_FM3502_fmData
// ************************************
// myFM3502_fmData : Struct_FM3502_fmData;
myFM3502_fmData.moduleAddress :=256; // [IN] module address            (INT)
myFM3502_fmData.loadValue0 :=0; // [IN] load value counter 0            (DINT)
myFM3502_fmData.loadValue1 :=0; // [IN] load value counter 1            (DINT)
myFM3502_fmData.loadValue2 :=0; // [IN] load value counter 2            (DINT)
myFM3502_fmData.loadValue3 :=0; // [IN] load value counter 3            (DINT)
myFM3502_fmData.loadValue4 :=0; // [IN] load value counter 4            (DINT)
myFM3502_fmData.loadValue5 :=0; // [IN] load value counter 5            (DINT)
myFM3502_fmData.loadValue6 :=0; // [IN] load value counter 6            (DINT)
```

```
myFM3502_fmData.loadValue7 :=0; // [IN] load value counter 7            (DINT)
myFM3502_fmData.prepValue0 :=0; // [IN] load prepare value counter 0        (DINT)
myFM3502_fmData.prepValue1 :=0; // [IN] load prepare value counter 1        (DINT)
myFM3502_fmData.prepValue2 :=0; // [IN] load prepare value counter 2        (DINT)
myFM3502_fmData.prepValue3 :=0; // [IN] load prepare value counter 3        (DINT)
myFM3502_fmData.prepValue4 :=0; // [IN] load prepare value counter 4        (DINT)
myFM3502_fmData.prepValue5 :=0; // [IN] load prepare value counter 5        (DINT)
myFM3502_fmData.prepValue6 :=0; // [IN] load prepare value counter 6        (DINT)
myFM3502_fmData.prepValue7 :=0; // [IN] load prepare value counter 7        (DINT)
myFM3502_fmData.cmpValue0 :=0; // [IN] comparator value 0            (DINT)
myFM3502_fmData.cmpValue1 :=0; // [IN] comparator value 1            (DINT)
myFM3502_fmData.cmpValue2 :=0; // [IN] comparator value 2            (DINT)
myFM3502_fmData.cmpValue3 :=0; // [IN] comparator value 3            (DINT)
myFM3502_fmData.cmpValue4 :=0; // [IN] comparator value 4            (DINT)
myFM3502_fmData.cmpValue5 :=0; // [IN] comparator value 5            (DINT)
myFM3502_fmData.cmpValue6 :=0; // [IN] comparator value 6            (DINT)
myFM3502_fmData.cmpValue7 :=0; // [IN] comparator value 7            (DINT)
```

2.4.3.4 _FM3502_write

The _FM3502_write function block executes write jobs (e.g. loading counter values and comparison values) to the FM 350-2.

Hardware/software requirements:

Hardware version: 6ES7 350-2AH00-0AE0 as of product version E04. Software version: none

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myFM3502_write : _FM3502_write; // instance of FB (_FM3502_write)
myFM3502_write(
periIn := myPeriIn, // [IN] Transfers I/O Inputs of the FM to the FB (ARRAY[0..15] of BYTE)
periOut := myPeriOut, // [IN_OUT] Prepared data of the FB for the I/O outputs of the FM (ARRAY[0..15]
of BYTE)
data := myFM3502_fmData // [IN_OUT] Data structure whith channel-specific data (Struct_FM3502_fmData)
);

// ***************************************
// Struct_FM3502_wrJob   // write commands
// ***************************************
myFM3502_fmData.write.execJobNumber :=0; // [IN] number     (BYTE)

// ***************************************
// Struct_FM3502_rdJob   // read commands
// ***************************************
myFM3502_fmData.read.execJobNumber :=0; // [IN] number    (BYTE)

// ************************************************
// Struct_FM3502_controlSignals  // control signals
// ************************************************
myFM3502_fmData.control.enableOutput0 := FALSE; // [IN] control digital output DQ0  (BOOL)
myFM3502_fmData.control.enableOutput1 := FALSE; // [IN] control digital output DQ1  (BOOL)
myFM3502_fmData.control.enableOutput2 := FALSE; // [IN] control digital output DQ2  (BOOL)
myFM3502_fmData.control.enableOutput3 := FALSE; // [IN] control digital output DQ3  (BOOL)
myFM3502_fmData.control.enableOutput4 := FALSE; // [IN] control digital output DQ4  (BOOL)
myFM3502_fmData.control.enableOutput5 := FALSE; // [IN] control digital output DQ5  (BOOL)
myFM3502_fmData.control.enableOutput6 := FALSE; // [IN] control digital output DQ6  (BOOL)
myFM3502_fmData.control.enableOutput7 := FALSE; // [IN] control digital output DQ7  (BOOL)
myFM3502_fmData.control.setOutput0 := FALSE; // [IN] set digital output DQ0    (BOOL)
myFM3502_fmData.control.setOutput1 := FALSE; // [IN] set digital output DQ1    (BOOL)
myFM3502_fmData.control.setOutput2 := FALSE; // [IN] set digital output DQ2    (BOOL)
myFM3502_fmData.control.setOutput3 := FALSE; // [IN] set digital output DQ3    (BOOL)
myFM3502_fmData.control.setOutput4 := FALSE; // [IN] set digital output DQ4    (BOOL)
myFM3502_fmData.control.setOutput5 := FALSE; // [IN] set digital output DQ5    (BOOL)
myFM3502_fmData.control.setOutput6 := FALSE; // [IN] set digital output DQ6    (BOOL)
myFM3502_fmData.control.setOutput7 := FALSE; // [IN] set digital output DQ7    (BOOL)
myFM3502_fmData.control.enableSwGate0 := FALSE; // [IN] software gate 0            (BOOL)
myFM3502_fmData.control.enableSwGate1 := FALSE; // [IN] software gate 1            (BOOL)
myFM3502_fmData.control.enableSwGate2 := FALSE; // [IN] software gate 2            (BOOL)
myFM3502_fmData.control.enableSwGate3 := FALSE; // [IN] software gate 3            (BOOL)
myFM3502_fmData.control.enableSwGate4 := FALSE; // [IN] software gate 4            (BOOL)
myFM3502_fmData.control.enableSwGate5 := FALSE; // [IN] software gate 5            (BOOL)
myFM3502_fmData.control.enableSwGate6 := FALSE; // [IN] software gate 6            (BOOL)
myFM3502_fmData.control.enableSwGate7 := FALSE; // [IN] software gate 7            (BOOL)

// ***************************************
//  Struct_FM3502_fmData
// ***************************************
// myFM3502_fmData : Struct_FM3502_fmData ;
myFM3502_fmData.moduleAddress :=256; // [IN] module address           (INT)
myFM3502_fmData.loadValue0 :=0; // [IN] load value counter 0           (DINT)
myFM3502_fmData.loadValue1 :=0; // [IN] load value counter 1           (DINT)
myFM3502_fmData.loadValue2 :=0; // [IN] load value counter 2           (DINT)
myFM3502_fmData.loadValue3 :=0; // [IN] load value counter 3           (DINT)
myFM3502_fmData.loadValue4 :=0; // [IN] load value counter 4           (DINT)
```

```
myFM3502_fmData.loadValue5 :=0; // [IN] load value counter 5           (DINT)
myFM3502_fmData.loadValue6 :=0; // [IN] load value counter 6           (DINT)
myFM3502_fmData.loadValue7 :=0; // [IN] load value counter 7           (DINT)
myFM3502_fmData.prepValue0 :=0; // [IN] load prepare value counter 0        (DINT)
myFM3502_fmData.prepValue1 :=0; // [IN] load prepare value counter 1        (DINT)
myFM3502_fmData.prepValue2 :=0; // [IN] load prepare value counter 2        (DINT)
myFM3502_fmData.prepValue3 :=0; // [IN] load prepare value counter 3        (DINT)
myFM3502_fmData.prepValue4 :=0; // [IN] load prepare value counter 4        (DINT)
myFM3502_fmData.prepValue5 :=0; // [IN] load prepare value counter 5        (DINT)
myFM3502_fmData.prepValue6 :=0; // [IN] load prepare value counter 6        (DINT)
myFM3502_fmData.prepValue7 :=0; // [IN] load prepare value counter 7        (DINT)
myFM3502_fmData.cmpValue0 :=0; // [IN] comparator value 0           (DINT)
myFM3502_fmData.cmpValue1 :=0; // [IN] comparator value 1           (DINT)
myFM3502_fmData.cmpValue2 :=0; // [IN] comparator value 2           (DINT)
myFM3502_fmData.cmpValue3 :=0; // [IN] comparator value 3           (DINT)
myFM3502_fmData.cmpValue4 :=0; // [IN] comparator value 4           (DINT)
myFM3502_fmData.cmpValue5 :=0; // [IN] comparator value 5           (DINT)
myFM3502_fmData.cmpValue6 :=0; // [IN] comparator value 6           (DINT)
myFM3502_fmData.cmpValue7 :=0; // [IN] comparator value 7           (DINT)
```

2.4.4 Function modules - FM352

2.4.4.1 _FM352_control

With the function block _FM352_control you can write the control signals and read out the checkback signals from the FM 352 and write and read the parameters of the FM 352.

Hardware/software requirements:

Hardware version: 6ES7 352-1AH01-0AE0 as of product version E08. Software version: None. The online functions of the parameter assignment tool in STEP 7 HW Config can only be used for diagnostic purposes (read-only access to the module). Write access (control function) has no effect. The parameters set by the program can be read out using the parameter assignment tool.

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myFM352_control : _FM352_control; // instance of FB (_FM352_control)
myFM352_control(
periIn := myPeriIn, // [IN] Transfers I/O Inputs of the FM to the FB (ARRAY[0..15] OF BYTE)
periOut := myPeriOut, // [IN_OUT] Prepared data of the FB for the I/O outputs of the FM (ARRAY[0..15]
OF BYTE)
ctrlData := myFM352_ctrlData, // [IN_OUT] Data structure whith channel-specific data
(Struct_FM352_ctrlData)
paraData := myFM352_paraData // [IN_OUT] Data structure with machine data and output cam data
(Struct_FM352_paraData)
);

// ************************
// Struct_FM352_ctrlData
// ************************
// myFM352_ctrlData : Struct_FM352_ctrlData;
myFM352_ctrlData.moduleAddress := 256; // [IN] Module address
myFM352_ctrlData.FmType := FALSE; // [IN] Type of FM: 0=FM352, 1=FM452 or FM352 >= V5.0
myFM352_ctrlData.enableSimPositive := FALSE; // [IN] 1=Simulation in positive direction (plus)
myFM352_ctrlData.enableSimNegative := FALSE; // [IN] 1=Simulation in negative direction (minus)
myFM352_ctrlData.enableOutputCam := FALSE; // [IN] 1=Enable cam processing
myFM352_ctrlData.enableTrack0Counter := FALSE; // [IN] 1=Enable counter function of cam track 0
myFM352_ctrlData.enableTrack1Counter := FALSE; // [IN] 1=Enable counter function of cam track 1
myFM352_ctrlData.enableTrack := 16#00; // [IN] Enable cam tracks (Bit0=track0)
myFM352_ctrlData.floatActValue := FALSE; // [IN] 1=Set actual value on-the-fly executed
myFM352_ctrlData.enableEdgeDetection := FALSE; // [IN] 1=Edge acquisition on
myFM352_ctrlData.enableSimulation := FALSE; // [IN] 1=Simulation on
myFM352_ctrlData.enableLenMeasuring := FALSE; // [IN] 1=Length measurement on
myFM352_ctrlData.execRetrigRefPoint := FALSE; // [IN] 1=Retrigger reference point on
myFM352_ctrlData.switchOffSwLimit := FALSE; // [IN] 1=Software limit switch off
myFM352_ctrlData.execWrMachineData := FALSE; // [IN] 1=Write machine data
myFM352_ctrlData.execWrActivateMData := FALSE; // [IN] 1=Enable machine date
myFM352_ctrlData.execWrActValueRevoke := FALSE; // [IN] 1=Undo set actual value
myFM352_ctrlData.execWrOutputCamData1 := FALSE; // [IN] 1=Write cam data 1 (cams 0..15)
myFM352_ctrlData.execWrOutputCamData2 := FALSE; // [IN] 1=Write cam data 2 (cams 16..31)
myFM352_ctrlData.execWrOutputCamData3 := FALSE; // [IN] 1=Write cam data 3 (cams 32..47)
myFM352_ctrlData.execWrOutputCamData4 := FALSE; // [IN] 1=Write cam data 4 (cams 48..63)
myFM352_ctrlData.execWrOutputCamData5 := FALSE; // [IN] 1=Write cam data 5 (cams 64..79)
myFM352_ctrlData.execWrOutputCamData6 := FALSE; // [IN] 1=Write cam data 6 (cams 80..95)
myFM352_ctrlData.execWrOutputCamData7 := FALSE; // [IN] 1=Write cam data 7 (cams 96..111)
myFM352_ctrlData.execWrOutputCamData8 := FALSE; // [IN] 1=Write cam data 8 (cams 112..127)
myFM352_ctrlData.execWrSetRefPoint := FALSE; // [IN] 1=Set reference point
myFM352_ctrlData.execWrActValue := FALSE; // [IN] 1=Set actual value
myFM352_ctrlData.execWrActValSetOnTheFly := FALSE; // [IN] 1=set actual value on-the-fly
myFM352_ctrlData.execWrZeroOffset := FALSE; // [IN] 1=Set zero offset
myFM352_ctrlData.execWrOutputCamEdge1 := FALSE; // [IN] 1=Change 1 cam (cam edges)
myFM352_ctrlData.execWrOutputCamEdge16 := FALSE; // [IN] 1=Change 16 cams (high speed cam changing)
myFM352_ctrlData.execRdMachineData := FALSE; // [IN] 1=Read machine data
myFM352_ctrlData.execRdOutputCamData1 := FALSE; // [IN] 1=Read cam data 1 (cams 0..15)
myFM352_ctrlData.execRdOutputCamData2 := FALSE; // [IN] 1=Read cam data 2 (cams 16..31)
myFM352_ctrlData.execRdOutputCamData3 := FALSE; // [IN] 1=Read cam data 3 (cams 32..47)
myFM352_ctrlData.execRdOutputCamData4 := FALSE; // [IN] 1=Read cam data 4 (cams 48..63)
myFM352_ctrlData.execRdOutputCamData5 := FALSE; // [IN] 1=Read cam data 5 (cams 64..79)
myFM352_ctrlData.execRdOutputCamData6 := FALSE; // [IN] 1=Read cam data 6 (cams 80..95)
myFM352_ctrlData.execRdOutputCamData7 := FALSE; // [IN] 1=Read cam data 7 (cams 96..111)
myFM352_ctrlData.execRdOutputCamData8 := FALSE; // [IN] 1=Read cam data 8 (cams 112..127)
myFM352_ctrlData.execRdMeasValue := FALSE; // [IN] 1=Read measurement values
```

```
myFM352_ctrlData.execRdCntrValueTrack := FALSE; // [IN] 1=Read count track values
myFM352_ctrlData.execRdActPosition := FALSE; // [IN] 1=Read position and track data
myFM352_ctrlData.execRdEncValue := FALSE; // [IN] 1=Read encoder value / counter reading
myFM352_ctrlData.execRdOutputCamData := FALSE; // [IN] 1=Read cam and track signals
myFM352_ctrlData.zeroOffset := 0; // [IN] Value for zero offset
myFM352_ctrlData.actValue := 0; // [IN] Coordinate for actual value
myFM352_ctrlData.actValueSetOnTheFly := 0; // [IN] Coordinate for actual value on-the-fly
myFM352_ctrlData.refPoint := 0; // [IN] Coordinate for reference point
myFM352_ctrlData.outputCamNumber := 0; // [IN] Change cam edges: Cam number
myFM352_ctrlData.beginOutputCam := 0; // [IN] Cam start
myFM352_ctrlData.endOutputCam := 0; // [IN] Cam end

// *************************
// Struct_FM352_outCamCtrl
// *************************
// first outputcam
myFM352_ctrlData.outputCam[1].number := 16#00; // [IN] Number of the cam to be changed
myFM352_ctrlData.outputCam[1].setForceDirection := FALSE; // [IN] 1=Change effective direction of the
cam
myFM352_ctrlData.outputCam[1].setBegin := FALSE; // [IN] 1=Change cam start to the value begin
myFM352_ctrlData.outputCam[1].setEnd := FALSE; // [IN] 1=Change cam end / cam time to the value end
myFM352_ctrlData.outputCam[1].setActuationTime := FALSE; // [IN] 1=Change lead time to the value of
actuationTime
myFM352_ctrlData.outputCam[1].deactivate := FALSE; // [IN] 1=Switch off cam while cam data are changed
myFM352_ctrlData.outputCam[1].posForceDirection := FALSE; // [IN] 1=Effective direction is positive
(plus)
myFM352_ctrlData.outputCam[1].negForceDirection := FALSE; // [IN] 1=Effective direction is negative
(minus)
myFM352_ctrlData.outputCam[1].beginOutputCam := 0; // [IN] New cam start
myFM352_ctrlData.outputCam[1].endOutputCam := 0; // [IN] New cam end / cam time
myFM352_ctrlData.outputCam[1].actuationTime := 0; // [IN] New lead time

// ************************
// Struct_FM352_paraData
// ************************
// myFM352_paraData : Struct_FM352_paraData;
myFM352_paraData.enableProcessAlarm := FALSE; // [IN_OUT] 1=Enable process interrupt: cam on / off
myFM352_paraData.minEdgeDistance := 0; // [IN_OUT] Minimum edge distance for edge acquisition
myFM352_paraData.unitDimension := 1; // [IN_OUT] System of units
myFM352_paraData.axisType := 0; // [IN_OUT] 0=Linear axis, 1=rotary axis
myFM352_paraData.endRotAxis := 100000; // [IN_OUT] End of rotary axis
myFM352_paraData.encType := 1; // [IN_OUT] Encoder type, frame length
myFM352_paraData.lenPerRevolution := 80000; // [IN_OUT] Displacement per encoder revolution
myFM352_paraData.incPerRevolution := 500; // [IN_OUT] Increments per encoder revolution
myFM352_paraData.cntOfRevolutions := 1024; // [IN_OUT] Number of encoder revolutions
myFM352_paraData.baudRate := 0; // [IN_OUT] Baud rate
myFM352_paraData.refPoint := 0; // [IN_OUT] Reference point coordinate
myFM352_paraData.absEncOffset := 0; // [IN_OUT] Absolute encoder adjustment
myFM352_paraData.refPointTrigMode := 0; // [IN_OUT] Type of retrigger reference point
myFM352_paraData.cntrDirection := FALSE; // [IN_OUT] Counting direction: 0=normal, 1=inverse
myFM352_paraData.openCircuit := TRUE; // [IN_OUT] 1=Monitor wirebreak
myFM352_paraData.transmissionError := TRUE; // [IN_OUT] 1=Monitor frame error
myFM352_paraData.missingPulse := TRUE; // [IN_OUT] 1=Monitor error pulses
myFM352_paraData.swLimitStart := -100000000; // [IN_OUT] Start of software limit switch
myFM352_paraData.swLimitEnd := 100000000; // [IN_OUT] End   of software limit switch
myFM352_paraData.numOfOutputCamsToSet := 0; // [IN_OUT] Cam quantity: 0,1,2,3 = max. 16,32,64,128 cams
myFM352_paraData.hysteresis := 0; // [IN_OUT] Hysteresis
```

```
myFM352_paraData.simSpeed := 0; // [IN_OUT] Simulation speed
myFM352_paraData.ctrlTrackOutputs := 16#0000; // [IN_OUT] Control of track outputs: 0=cam, 1=CPU;
Bitno=trackno
myFM352_paraData.enableInput3 := FALSE; // [IN_OUT] 1=Enable input I3
myFM352_paraData.track0CntrOutputCam := FALSE; // [IN_OUT] 1=Enable special track 0
myFM352_paraData.track1CntrOutputCam := FALSE; // [IN_OUT] 1=Enable special track 1
myFM352_paraData.track2CntrOutputCam := FALSE; // [IN_OUT] 1=Enable special track 2
myFM352_paraData.track0CntrLimit := 2; // [IN_OUT] Upper counter value of the counter cam track 0
myFM352_paraData.track1CntrLimit := 2; // [IN_OUT] Upper counter value of the counter cam track 1

// *************************
// Struct_FM352_outCamPara
// *************************
// first outputcam
myFM352_paraData.outputCam[1].valid := FALSE; // [IN_OUT] 1=Cam valid
myFM352_paraData.outputCam[1].posForceDirection := TRUE; // [IN_OUT] 1=Effective direction positive
(plus)
myFM352_paraData.outputCam[1].negForceDirection := TRUE; // [IN_OUT] 1=Effective direction negative
(minus)
myFM352_paraData.outputCam[1].outputCamType := FALSE; // [IN_OUT] 0=Displacement cam, 1= time cam
myFM352_paraData.outputCam[1].switchOnAlarm := FALSE; // [IN_OUT] 1=Process interrupt active on
switching on
myFM352_paraData.outputCam[1].switchOffAlarm := FALSE; // [IN_OUT] 1=Process interrupt active on
switching off
myFM352_paraData.outputCam[1].trackNumber := 16#00; // [IN_OUT] Track number
myFM352_paraData.outputCam[1].beginOutputCam := -100000000; // [IN_OUT] Cam start
myFM352_paraData.outputCam[1].endOutputCam := 100000000; // [IN_OUT] Cam end / switch-on time
myFM352_paraData.outputCam[1].actuationTime := 0; // [IN_OUT] Lead time
```

2.4.4.2 _FM352_diagnostic

The _FM352_diagnostic function block can be used to read out the complete diagnostic data from the FM 352.

Hardware/software requirements:

Hardware version: 6ES7 352-1AH01-0AE0 after product version E08

Software version: none

The online functions of the parameter assignment tool in STEP 7 HW Config can only be used for diagnostic purposes (read-only access to the module). Write access (control function) has no effect. The parameters set by the program can be read out using the parameter assignment tool.

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myFM352_diagnostic : _FM352_diagnostic; // instance of FB (_FM352_diagnostic)
myFM352_diagnostic(
enable := myEnable, // [IN] enable (BOOL)
periIn := myPeriIn, // [IN] I/O variable for access to I/O inputs from the FM 352 (ARRAY[0..15] OF
BYTE)
diagData := myFM352_diagData // [IN_OUT] Data structure for diagnostic data (Struct_FM352_diagData)
);

// ************************
// Struct_FM352_diagData
// ************************
// myFM352_diagData : Struct_FM352_diagData;
myFM352_diagData.moduleAddress := 256; // [IN] Module address
myFM352_diagData.diagInformation := FALSE; // [IN] 1=Read dignostic buffer unconditionaly
```

2.4.4.3 _FM352_initialize

The _FM352_initialize function block can be used to initialize the channel data after startup of the FM 352 module.

Hardware/software requirements:

Hardware version: 6ES7 352-1AH01-0AE0 after product version E08

Software version: none

The online functions of the parameter assignment tool in STEP 7 HW Config can only be used for diagnostic purposes (read-only access to the module). Write access (control function) has no effect. The parameters set by the program can be read out using the parameter assignment tool.

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myFM352_initialize : _FM352_initialize; // instance of FB (_FM352_initialize)
myFM352_initialize(
execute := myExecute, // [IN] execute (BOOL)
ctrlData := myFM352_ctrlData // [IN_OUT] Data structure whith channel-specific data
(Struct_FM352_ctrlData)
);

// ************************
// Struct_FM352_ctrlData
// ************************
// myFM352_ctrlData : Struct_FM352_ctrlData;
myFM352_ctrlData.moduleAddress := 256; // [IN] Module address
myFM352_ctrlData.FmType := FALSE; // [IN] Type of FM: 0=FM352, 1=FM452 or FM352 >= V5.0
myFM352_ctrlData.enableSimPositive := FALSE; // [IN] 1=Simulation in positive direction (plus)
myFM352_ctrlData.enableSimNegative := FALSE; // [IN] 1=Simulation in negative direction (minus)
myFM352_ctrlData.enableOutputCam := FALSE; // [IN] 1=Enable cam processing
myFM352_ctrlData.enableTrack0Counter := FALSE; // [IN] 1=Enable counter function of cam track 0
myFM352_ctrlData.enableTrack1Counter := FALSE; // [IN] 1=Enable counter function of cam track 1
myFM352_ctrlData.enableTrack := 16#00; // [IN] Enable cam tracks (Bit0=track0)
myFM352_ctrlData.floatActValue := FALSE; // [IN] 1=Set actual value on-the-fly executed
myFM352_ctrlData.enableEdgeDetection := FALSE; // [IN] 1=Edge acquisition on
myFM352_ctrlData.enableSimulation := FALSE; // [IN] 1=Simulation on
myFM352_ctrlData.enableLenMeasuring := FALSE; // [IN] 1=Length measurement on
myFM352_ctrlData.execRetrigRefPoint := FALSE; // [IN] 1=Retrigger reference point on
myFM352_ctrlData.switchOffSwLimit := FALSE; // [IN] 1=Software limit switch off
myFM352_ctrlData.execWrMachineData := FALSE; // [IN] 1=Write machine data
myFM352_ctrlData.execWrActivateMData := FALSE; // [IN] 1=Enable machine date
myFM352_ctrlData.execWrActValueRevoke := FALSE; // [IN] 1=Undo set actual value
myFM352_ctrlData.execWrOutputCamData1 := FALSE; // [IN] 1=Write cam data 1 (cams 0..15)
myFM352_ctrlData.execWrOutputCamData2 := FALSE; // [IN] 1=Write cam data 2 (cams 16..31)
myFM352_ctrlData.execWrOutputCamData3 := FALSE; // [IN] 1=Write cam data 3 (cams 32..47)
myFM352_ctrlData.execWrOutputCamData4 := FALSE; // [IN] 1=Write cam data 4 (cams 48..63)
myFM352_ctrlData.execWrOutputCamData5 := FALSE; // [IN] 1=Write cam data 5 (cams 64..79)
myFM352_ctrlData.execWrOutputCamData6 := FALSE; // [IN] 1=Write cam data 6 (cams 80..95)
myFM352_ctrlData.execWrOutputCamData7 := FALSE; // [IN] 1=Write cam data 7 (cams 96..111)
myFM352_ctrlData.execWrOutputCamData8 := FALSE; // [IN] 1=Write cam data 8 (cams 112..127)
myFM352_ctrlData.execWrSetRefPoint := FALSE; // [IN] 1=Set reference point
myFM352_ctrlData.execWrActValue := FALSE; // [IN] 1=Set actual value
myFM352_ctrlData.execWrActValSetOnTheFly := FALSE; // [IN] 1=set actual value on-the-fly
myFM352_ctrlData.execWrZeroOffset := FALSE; // [IN] 1=Set zero offset
myFM352_ctrlData.execWrOutputCamEdge1 := FALSE; // [IN] 1=Change 1 cam (cam edges)
myFM352_ctrlData.execWrOutputCamEdge16 := FALSE; // [IN] 1=Change 16 cams (high speed cam changing)
myFM352_ctrlData.execRdMachineData := FALSE; // [IN] 1=Read machine data
myFM352_ctrlData.execRdOutputCamData1 := FALSE; // [IN] 1=Read cam data 1 (cams 0..15)
myFM352_ctrlData.execRdOutputCamData2 := FALSE; // [IN] 1=Read cam data 2 (cams 16..31)
myFM352_ctrlData.execRdOutputCamData3 := FALSE; // [IN] 1=Read cam data 3 (cams 32..47)
myFM352_ctrlData.execRdOutputCamData4 := FALSE; // [IN] 1=Read cam data 4 (cams 48..63)
myFM352_ctrlData.execRdOutputCamData5 := FALSE; // [IN] 1=Read cam data 5 (cams 64..79)
myFM352_ctrlData.execRdOutputCamData6 := FALSE; // [IN] 1=Read cam data 6 (cams 80..95)
myFM352_ctrlData.execRdOutputCamData7 := FALSE; // [IN] 1=Read cam data 7 (cams 96..111)
myFM352_ctrlData.execRdOutputCamData8 := FALSE; // [IN] 1=Read cam data 8 (cams 112..127)
myFM352_ctrlData.execRdMeasValue := FALSE; // [IN] 1=Read measurement values
myFM352_ctrlData.execRdCntrValueTrack := FALSE; // [IN] 1=Read count track values
myFM352_ctrlData.execRdActPosition := FALSE; // [IN] 1=Read position and track data
myFM352_ctrlData.execRdEncValue := FALSE; // [IN] 1=Read encoder value / counter reading
myFM352_ctrlData.execRdOutputCamData := FALSE; // [IN] 1=Read cam and track signals
```

```
myFM352_ctrlData.zeroOffset := 0; // [IN] Value for zero offset
myFM352_ctrlData.actValue := 0; // [IN] Coordinate for actual value
myFM352_ctrlData.actValueSetOnTheFly := 0; // [IN] Coordinate for actual value on-the-fly
myFM352_ctrlData.refPoint := 0; // [IN] Coordinate for reference point
myFM352_ctrlData.outputCamNumber := 0; // [IN] Change cam edges: Cam number
myFM352_ctrlData.beginOutputCam := 0; // [IN] Cam start
myFM352_ctrlData.endOutputCam := 0; // [IN] Cam end

// *************************
// Struct_FM352_outCamCtrl
// *************************
// first outputcam
myFM352_ctrlData.outputCam[1].number := 16#00; // [IN] Number of the cam to be changed
myFM352_ctrlData.outputCam[1].setForceDirection := FALSE; // [IN] 1=Change effective direction of the
cam
myFM352_ctrlData.outputCam[1].setBegin := FALSE; // [IN] 1=Change cam start to the value begin
myFM352_ctrlData.outputCam[1].setEnd := FALSE; // [IN] 1=Change cam end / cam time to the value end
myFM352_ctrlData.outputCam[1].setActuationTime := FALSE; // [IN] 1=Change lead time to the value of
actuationTime
myFM352_ctrlData.outputCam[1].deactivate := FALSE; // [IN] 1=Switch off cam while cam data are changed
myFM352_ctrlData.outputCam[1].posForceDirection := FALSE; // [IN] 1=Effective direction is positive
(plus)
myFM352_ctrlData.outputCam[1].negForceDirection := FALSE; // [IN] 1=Effective direction is negative
(minus)
myFM352_ctrlData.outputCam[1].beginOutputCam := 0; // [IN] New cam start
myFM352_ctrlData.outputCam[1].endOutputCam := 0; // [IN] New cam end / cam time
myFM352_ctrlData.outputCam[1].actuationTime := 0; // [IN] New lead time
```

2.4.5 Communication modules - CP340

2.4.5.1 _CP340_byteToPrintData

Entry of a number of data type BYTE in data structure Struct_CP340_printData.

Hardware/software requirements:

Hardware version: 6ES7 340-1AH01-0AE0 after product version E07, 6ES7 340-1BH00-0AE0 after product version E07, 6ES7 340-1CH00-0AE0 after product version E07

Software version: none

Hardware version: 6ES7340-1AH02-0AE0 after product version E01, 6ES7340-1BH02-0AE0 after product version E01, 6ES7340-1CH02-0AE0 after product version E01

Software version: as of V1.0.2

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
myCP340_byteToPrintData : _CP340_byteToPrintData; // instance of FB (_CP340_byteToPrintData)
myCP340_byteToPrintData(
execute := myExecute, // [IN] Edge-triggered request activation (BOOL)
data := myData, // [IN] Variable/value to be entered in the data structure (BYTE)
numVariable := myNumVariable, // [IN] Number of the variable in whitch the entry is to be made 1 <=
NumberVariable <=4 (INT)
entryFormatString := myEntryFormatString, // [IN] Method of entry in the format string
(enumFormstring2)
entryAtByteNumber := myEntryAtByteNumber, // [IN] Specifies the byte at which the entry is to begin in
the substructure of the format string (USINT)
printData := myCP340_printData // [IN_OUT] Data structure for the printer data (Struct_CP340_printData)
);

// ************************
// Struct_CP340_printData
// ************************
// myCP340_printData : Struct_CP340_printData;
myCP340_printData.variable := myVariable; // [IN] Variable to be printed (ARRAY [0..3] of
Struct_CP340_datRecord   )
myCP340_printData.format := myFormat; // [IN] Format string (ARRAY [0..150] OF BYTE )

// ************************
// Struct_CP340_datRecord
// ************************
// myCP340_datRecord : Struct_CP340_datRecord;
myCP340_datRecord.datalength := 0; // [IN] Quantity of data (UDINT )
myCP340_datRecord.data := myData; // [IN] Print data (ARRAY [0..31] OF BYTE )
```

2.4.5.2 _CP340_dintToPrintData

Entry of a number of data type DINT in data structure Struct_CP340_printData.

Hardware/software requirements:

Hardware version: 6ES7 340-1AH01-0AE0 after product version E07, 6ES7 340-1BH00-0AE0 after product version E07, 6ES7 340-1CH00-0AE0 after product version E07

Software version: none

Hardware version: 6ES7340-1AH02-0AE0 after product version E01, 6ES7340-1BH02-0AE0 after product version E01, 6ES7340-1CH02-0AE0 after product version E01

Software version: as of V1.0.2

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myCP340_dintToPrintData : _CP340_dintToPrintData; // instance of FB (_CP340_dintToPrintData)
myCP340_dintToPrintData(
execute := myExecute, // [IN] Edge-triggered request activation (BOOL)
data := myData, // [IN] Variable/value to be entered in the data structure (DINT)
numVariable := myNumVariable, // [IN] Number of the variable in whitch the entry is to be made 1 <=
NumberVariable <=4 (INT)
entryFormatString := myEntryFormatString, // [IN] Method of entry in the format string
(enumFormstring3)
entryAtByteNumber := myEntryAtByteNumber, // [IN] Specifies the byte at which the entry is to begin in
the substructure of the format string (USINT)
printData := myCP340_printData // [IN_OUT] Data structure for the printer data (Struct_CP340_printData)
);

// ************************
// Struct_CP340_printData
// ************************
// myCP340_printData : Struct_CP340_printData;
myCP340_printData.variable := myVariable; // [IN] Variable to be printed (ARRAY [0..3] of
Struct_CP340_datRecord   )
myCP340_printData.format := myFormat; // [IN] Format string (ARRAY [0..150] OF BYTE )

// ************************
// Struct_CP340_datRecord
// ************************
// myCP340_datRecord : Struct_CP340_datRecord;
myCP340_datRecord.datalength := 0; // [IN] Quantity of data (UDINT )
myCP340_datRecord.data := myData; // [IN] Print data (ARRAY [0..31] OF BYTE )
```

2.4.5.3 _CP340_dwordToPrintData

Entry of a number of data type 'DWORD' in data structure 'Struct_CP340_printData'.

Hardware/software requirements:

Hardware version: 6ES7 340-1AH01-0AE0 after product version E07, 6ES7 340-1BH00-0AE0 after product version E07, 6ES7 340-1CH00-0AE0 after product version E07

Software version: none

Hardware version: 6ES7340-1AH02-0AE0 after product version E01, 6ES7340-1BH02-0AE0 after product version E01, 6ES7340-1CH02-0AE0 after product version E01

Software version: as of V1.0.2

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myCP340_dwordToPrintData : _CP340_dwordToPrintData; // instance of FB (_CP340_dwordToPrintData)
myCP340_dwordToPrintData(
execute := myExecute, // [IN] Edge-triggered request activation (BOOL)
data := myData, // [IN] Variable/value to be entered in the data structure (DWORD)
numVariable := myNumVariable, // [IN] Number of the variable in whitch the entry is to be made 1 <=
NumberVariable <=4 (INT)
entryFormatString := myEntryFormatString, // [IN] Method of entry in the format string
(enumFormstring2)
entryAtByteNumber := myEntryAtByteNumber, // [IN] Specifies the byte at which the entry is to begin in
the substructure of the format string (USINT)
printData := myCP340_printData // [IN_OUT] Data structure for the printer data (Struct_CP340_printData)
);

// ************************
// Struct_CP340_printData
// ************************
// myCP340_printData : Struct_CP340_printData;
myCP340_printData.variable := myVariable; // [IN] Variable to be printed (ARRAY [0..3] of
Struct_CP340_datRecord   )
myCP340_printData.format := myFormat; // [IN] Format string (ARRAY [0..150] OF BYTE )

// ************************
// Struct_CP340_datRecord
// ************************
// myCP340_datRecord : Struct_CP340_datRecord;
myCP340_datRecord.datalength := 0; // [IN] Quantity of data (UDINT )
myCP340_datRecord.data := myData; // [IN] Print data (ARRAY [0..31] OF BYTE )
```

2.4.5.4 _CP340_getV24Signals

The _CP340_getV24Signals function block reads the RS 232C accompanying signals from the CP 340 and makes them available to the user in the block parameters. The functionality of the _CP340_getV24Signals function block can only be used when an ASCII driver is assigned.

Hardware/software requirements:

Hardware version: 6ES7 340-1AH01-0AE0 after product version E07, 6ES7 340-1BH00-0AE0 after product version E07, 6ES7 340-1CH00-0AE0 after product version E07

Software version: none

Hardware version: 6ES7340-1AH02-0AE0 after product version E01, 6ES7340-1BH02-0AE0 after product version E01, 6ES7340-1CH02-0AE0 after product version E01

Software version: as of V1.0.2

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myCP340_getV24Signals : _CP340_getV24Signals; // instance of FB (_CP340_getV24Signals)
myCP340_getV24Signals(
enable := myEnable, // [IN] enable (BOOL)
periIn := myPeriIn // [IN] I/O inputs of the CP transferred to the FB (ARRAY [0..15] OF BYTE)
);
```

2.4.5.5 _CP340_intToPrintData

Entry of a number of data type 'INT' in data structure 'Struct_CP340_printData'.

Hardware/software requirements:

Hardware version: 6ES7 340-1AH01-0AE0 after product version E07, 6ES7 340-1BH00-0AE0 after product version E07, 6ES7 340-1CH00-0AE0 after product version E07

Software version: none

Hardware version: 6ES7340-1AH02-0AE0 after product version E01, 6ES7340-1BH02-0AE0 after product version E01, 6ES7340-1CH02-0AE0 after product version E01

Software version: as of V1.0.2

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myCP340_intToPrintData : _CP340_intToPrintData; // instance of FB (_CP340_intToPrintData)
myCP340_intToPrintData(
execute := myExecute, // [IN] Edge-triggered request activation (BOOL)
data := myData, // [IN] Variable/value to be entered in the data structure (INT)
numVariable := myNumVariable, // [IN] Number of the variable in whitch the entry is to be made 1 <=
NumberVariable <=4 (INT)
entryFormatString := myEntryFormatString, // [IN] Method of entry in the format string
(enumFormstring3)
entryAtByteNumber := myEntryAtByteNumber, // [IN] Specifies the byte at which the entry is to begin in
the substructure of the format string (USINT)
printData := myCP340_printData // [IN_OUT] Data structure for the printer data (Struct_CP340_printData)
);

// *************************
// Struct_CP340_printData
// *************************
// myCP340_printData : Struct_CP340_printData;
myCP340_printData.variable := myVariable; // [IN] Variable to be printed (ARRAY [0..3] of
Struct_CP340_datRecord   )
myCP340_printData.format := myFormat; // [IN] Format string (ARRAY [0..150] OF BYTE )

// *************************
// Struct_CP340_datRecord
// *************************
// myCP340_datRecord : Struct_CP340_datRecord;
myCP340_datRecord.datalength := 0; // [IN] Quantity of data (UDINT )
myCP340_datRecord.data := myData; // [IN] Print data (ARRAY [0..31] OF BYTE )
```

2.4.5.6 _CP340_printer

The CP340_printer function block can be used to send data of type 'Struct_CP340_printData' from the printer memory area to a serial printer. For example, the CP340_printer function block transfers a process message to the CP340. The CP340 prints out the process message on the connected printer.

Hardware/software requirements:

Hardware version: 6ES7 340-1AH01-0AE0 after product version E07, 6ES7 340-1BH00-0AE0 after product version E07, 6ES7 340-1CH00-0AE0 after product version E07

Software version: none

Hardware version: 6ES7340-1AH02-0AE0 after product version E01, 6ES7340-1BH02-0AE0 after product version E01, 6ES7340-1CH02-0AE0 after product version E01

Software version: as of V1.0.2

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myCP340_printer : _CP340_printer; // instance of FB (_CP340_printer)
myCP340_printer(
execute := myExecute, // [IN] Initiates request on positive edge (BOOL)
reset := myReset, // [IN] Aborts request (BOOL)
moduleAddress := myModuleAddress, // [IN] Module address of the CP for data set transfer (from HW
Config) (DINT)
periIn := myPeriIn, // [IN] I/O inputs of the CP transferred to the FB (ARRAY [0..15] OF BYTE)
printData := myCP340_printData, // [IN_OUT] Send data array (data to be printed)
(Struct_CP340_printData)
periOut := myPeriOut // [IN_OUT] Prepared FB data for the I/O outputs of the CP (ARRAY [0..15] OF BYTE)
);

// ************************
// Struct_CP340_printData
// ************************
// myCP340_printData : Struct_CP340_printData;
myCP340_printData.format[0] := 16#00; // [IN] Format string (ARRAY [0..150] OF BYTE )

// ************************
// Struct_CP340_datRecord
// ************************
//
myCP340_printData.variable[0].datalength := 4; // [IN] Quantity of data (UDINT )
myCP340_printData.variable[0].data[0] := 16#00; // [IN] Print data (ARRAY [0..31] OF BYTE)
```

2.4.5.7 _CP340_printMsgText

Selection of message texts stored in the CP340.

Hardware/software requirements:

Hardware version: 6ES7 340-1AH01-0AE0 after product version E07, 6ES7 340-1BH00-0AE0 after product version E07, 6ES7 340-1CH00-0AE0 after product version E07

Software version: none

Hardware version: 6ES7340-1AH02-0AE0 after product version E01, 6ES7340-1BH02-0AE0 after product version E01, 6ES7340-1CH02-0AE0 after product version E01

Software version: as of V1.0.2

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myCP340_printMsgText : _CP340_printMsgText; // instance of FB (_CP340_printMsgText)
myCP340_printMsgText(
execute := myExecute, // [IN] Edge-triggered request activation (BOOL)
numMsgText := myNumMsgText, // [IN] Variable/value to be entered in the data structure (USINT)
numVariable := myNumVariable, // [IN] Number of the variable in whitch the entry is to be made 1 <=
NumberVariable <=4 (INT)
entryFormatString := myEntryFormatString, // [IN] Method of entry in the format string
(enumFormstring3)
entryAtByteNumber := myEntryAtByteNumber, // [IN] Specifies the byte at which the entry is to begin in
the substructure of the format string (USINT)
printData := myCP340_printData // [IN_OUT] Data structure for the printer data (Struct_CP340_printData)
);

// ************************
// Struct_CP340_printData
// ************************
// myCP340_printData : Struct_CP340_printData;
myCP340_printData.variable := myVariable; // [IN] Variable to be printed (ARRAY [0..3] of
Struct_CP340_datRecord   )
myCP340_printData.format := myFormat; // [IN] Format string (ARRAY [0..150] OF BYTE )

// ************************
// Struct_CP340_datRecord
// ************************
// myCP340_datRecord : Struct_CP340_datRecord;
myCP340_datRecord.datalength := 0; // [IN] Quantity of data (UDINT )
myCP340_datRecord.data := myData; // [IN] Print data (ARRAY [0..31] OF BYTE )
```

2.4.5.8 _CP340_realToPrintData

Entry of a number of data type REAL in data structure Struct_CP340_printData.

Hardware/software requirements:

Hardware version: 6ES7 340-1AH01-0AE0 after product version E07, 6ES7 340-1BH00-0AE0 after product version E07, 6ES7 340-1CH00-0AE0 after product version E07

Software version: none

Hardware version: 6ES7340-1AH02-0AE0 after product version E01, 6ES7340-1BH02-0AE0 after product version E01, 6ES7340-1CH02-0AE0 after product version E01

Software version: as of V1.0.2

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myCP340_realToPrintData : _CP340_realToPrintData; // instance of FB (_CP340_realToPrintData)
myCP340_realToPrintData(
execute := myExecute, // [IN] Edge-triggered request activation (BOOL)
data := myData, // [IN] Variable/value to be entered in the data structure (REAL)
numVariable := myNumVariable, // [IN] Number of the variable in whitch the entry is to be made 1 <=
NumberVariable <=4 (INT)
entryFormatString := myEntryFormatString, // [IN] Method of entry in the format string
(enumFormstring1)
entryAtByteNumber := myEntryAtByteNumber, // [IN] Specifies the byte at which the entry is to begin in
the substructure of the format string (USINT)
printData := myCP340_printData // [IN_OUT] Data structure for the printer data (Struct_CP340_printData)
);

// ************************
// Struct_CP340_printData
// ************************
// myCP340_printData : Struct_CP340_printData;
myCP340_printData.variable := myVariable; // [IN] Variable to be printed (ARRAY [0..3] of
Struct_CP340_datRecord   )
myCP340_printData.format := myFormat; // [IN] Format string (ARRAY [0..150] OF BYTE )

// ************************
// Struct_CP340_datRecord
// ************************
// myCP340_datRecord : Struct_CP340_datRecord;
myCP340_datRecord.datalength := 0; // [IN] Quantity of data (UDINT )
myCP340_datRecord.data := myData; // [IN] Print data (ARRAY [0..31] OF BYTE )
```

2.4.5.9 _CP340_receive

The _CP340_receive function block can be used to receive data from a communication partner in the receive data array. 1024 bytes are available. For the transfer operation, you can use the 3964 (R) protocol or ASCII driver.

Hardware/software requirements:

Hardware version: 6ES7 340-1AH01-0AE0 after product version E07, 6ES7 340-1BH00-0AE0 after product version E07, 6ES7 340-1CH00-0AE0 after product version E07

Software version: none

Hardware version: 6ES7340-1AH02-0AE0 after product version E01, 6ES7340-1BH02-0AE0 after product version E01, 6ES7340-1CH02-0AE0 after product version E01

Software version: as of V1.0.2

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myCP340_receive : _CP340_receive; // instance of FB (_CP340_receive)
myCP340_receive(
enable := myEnable, // [IN] Receive enable (BOOL)
reset := myReset, // [IN] Aborts request (BOOL)
moduleAddress := myModuleAddress, // [IN] Module address of the CP for data set transfer(from HW
Config) (DINT)
dataOffset := myDataOffset, // [IN] Offset of the first element to be received (UDINT)
periIn := myPeriIn, // [IN] I/O inputs of the CP transferred to the FB (ARRAY [0..15] OF BYTE)
data := myData, // [IN_OUT] Receive data array (ARRAY [0..1023] OF BYTE)
periOut := myPeriOut // [IN_OUT] Prepared FB data for the I/O outputs of the CP (ARRAY [0..15] OF BYTE)
);
```

2.4.5.10 _CP340_send

The _CP340_send function block can be used to send data from the send data array to a communication partner. 1024 bytes are available. For the transfer operation, you can use the 3964 (R) protocol or ASCII driver.

Hardware/software requirements:

Hardware version: 6ES7 340-1AH01-0AE0 after product version E07, 6ES7 340-1BH00-0AE0 after product version E07, 6ES7 340-1CH00-0AE0 after product version E07

Software version: none

Hardware version: 6ES7340-1AH02-0AE0 after product version E01, 6ES7340-1BH02-0AE0 after product version E01, 6ES7340-1CH02-0AE0 after product version E01

Software version: as of V1.0.2

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myCP340_send : _CP340_send; // instance of FB (_CP340_send)
myCP340_send(
execute := myExecute, // [IN] Initiates request on positive edge (BOOL)
reset := myReset, // [IN] Aborts request (BOOL)
moduleAddress := myModuleAddress, // [IN] Module address of the CP for data set transfer (from HW
Config) (DINT)
dataOffset := myDataOffset, // [IN] Offset of the first element to be sent (UDINT)
dataLength := myDataLength, // [IN] Number of elements to be sent (UDINT)
periIn := myPeriIn, // [IN] I/O inputs of the CP transferred to the FB (ARRAY [0..15] OF BYTE)
data := myData, // [IN_OUT] Send data array (ARRAY [0..1023] OF BYTE)
periOut := myPeriOut // [IN_OUT] Prepared FB data for the I/O outputs of the CP (ARRAY [0..15] OF BYTE)
);
```

2.4.5.11 _CP340_setV24Signals

The _CP340_setV24Signals function block can be used to set or reset RS 232C accompanying signals. The functionality of the _CP340_setV24Signals function block can only be used when an ASCII driver have been parameterized.

Hardware/software requirements:

Hardware version: 6ES7 340-1AH01-0AE0 after product version E07, 6ES7 340-1BH00-0AE0 after product version E07, 6ES7 340-1CH00-0AE0 after product version E07

Software version: none

Hardware version: 6ES7340-1AH02-0AE0 after product version E01, 6ES7340-1BH02-0AE0 after product version E01, 6ES7340-1CH02-0AE0 after product version E01

Software version: as of V1.0.2

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myCP340_setV24Signals : _CP340_setV24Signals; // instance of FB (_CP340_setV24Signals)
myCP340_setV24Signals(
enable := myEnable, // [IN] enable (BOOL)
sigDtr := mySigDtr, // [IN] Signal 'Data terminal ready' (BOOL)
sigRts := mySigRts, // [IN] Signal 'Request to send' (BOOL)
periOut := myPeriOut // [IN_OUT] Prepared FB data for the I/O outputs of the CP (ARRAY [0..15] OF BYTE)
);
```

2.4.5.12 _CP340_wordToPrintData

Entry of a number of data type 'WORD' in data structure 'Struct_CP340_printData'.

Hardware/software requirements:

Hardware version: 6ES7 340-1AH01-0AE0 after product version E07, 6ES7 340-1BH00-0AE0 after product version E07, 6ES7 340-1CH00-0AE0 after product version E07

Software version: none

Hardware version: 6ES7340-1AH02-0AE0 after product version E01, 6ES7340-1BH02-0AE0 after product version E01, 6ES7340-1CH02-0AE0 after product version E01

Software version: as of V1.0.2

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myCP340_wordToPrintData : _CP340_wordToPrintData; // instance of FB (_CP340_wordToPrintData)
myCP340_wordToPrintData(
execute := myExecute, // [IN] Edge-triggered request activation (BOOL)
data := myData, // [IN] Variable/value to be entered in the data structure (WORD)
numVariable := myNumVariable, // [IN] Number of the variable in whitch the entry is to be made 1 <=
NumberVariable <=4 (INT)
entryFormatString := myEntryFormatString, // [IN] Method of entry in the format string
(enumFormstring2)
entryAtByteNumber := myEntryAtByteNumber, // [IN] Specifies the byte at which the entry is to begin in
the substructure of the format string (USINT)
printData := myCP340_printData // [IN_OUT] Data structure for the printer data (Struct_CP340_printData)
);

// ************************
// Struct_CP340_printData
// ************************
// myCP340_printData : Struct_CP340_printData;
myCP340_printData.variable := myVariable; // [IN] Variable to be printed (ARRAY [0..3] of
Struct_CP340_datRecord   )
myCP340_printData.format := myFormat; // [IN] Format string (ARRAY [0..150] OF BYTE )

// ************************
// Struct_CP340_datRecord
// ************************
// myCP340_datRecord : Struct_CP340_datRecord;
myCP340_datRecord.datalength := 0; // [IN] Quantity of data (UDINT )
myCP340_datRecord.data := myData; // [IN] Print data (ARRAY [0..31] OF BYTE )
```

2.4.6 Communication modules - CP341

2.4.6.1 _CP341_byteToPrintData

Entry of a number of data type 'BYTE' in data structure 'Struct_CP341_printData'.

Hardware/software requirements:

Hardware version: 6ES7 341-1AH02-0AE0 after product version E01

Software version: as of V2.0.0

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
myCP341_byteToPrintData : _CP341_byteToPrintData; // instance of FB (_CP341_byteToPrintData)
myCP341_byteToPrintData(
execute := myExecute, // [IN] Edge-triggered request activation (BOOL)
data := myData, // [IN] Variable/value to be entered in the data structure (BYTE)
numVariable := myNumVariable, // [IN] Number of the variable in whitch the entry is to be made 1 <=
NumberVariable <=4 (INT)
entryFormatString := myEntryFormatString, // [IN] Method of entry in the format string
(enumFormstring2)
entryAtByteNumber := myEntryAtByteNumber, // [IN] Specifies the byte at which the entry is to begin in
the substructure of the format string (USINT)
printData := myCP341_printData // [IN_OUT] Data structure for the printer data (Struct_CP341_printData)
);

// ************************
// Struct_CP341_printData
// ************************
// myCP341_printData : Struct_CP341_printData;
myCP341_printData.variable := myVariable; // [IN] Variable to be printed (ARRAY [0..3] of
Struct_CP341_datRecord   )
myCP341_printData.format := myFormat; // [IN] Format string (ARRAY [0..150] OF BYTE )

// ************************
// Struct_CP341_datRecord
// ************************
// myCP341_datRecord : Struct_CP341_datRecord;
myCP341_datRecord.datalength := 0; // [IN] Quantity of data (UDINT )
myCP341_datRecord.data := myData; // [IN] Print data (ARRAY [0..31] OF BYTE )
```

2.4.6.2 _CP341_dintToPrintData

Entry of a number of data type DINT in data structure Struct_CP341_printData.

Hardware/software requirements:

Hardware version: 6ES7 341-1AH02-0AE0 after product version E01

Software version: as of V2.0.0

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "SIMOTION Utilities & Applications" CD-ROM. The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myCP341_dintToPrintData : _CP341_dintToPrintData; // instance of FB (_CP341_dintToPrintData)
myCP341_dintToPrintData(
execute := myExecute, // [IN] Edge-triggered request activation (BOOL)
data := myData, // [IN] Variable/value to be entered in the data structure (DINT)
numVariable := myNumVariable, // [IN] Number of the variable in whitch the entry is to be made 1 <=
NumberVariable <=4 (INT)
entryFormatString := myEntryFormatString, // [IN] Method of entry in the format string
(enumFormstring3)
entryAtByteNumber := myEntryAtByteNumber, // [IN] Specifies the byte at which the entry is to begin in
the substructure of the format string (USINT)
printData := myCP341_printData // [IN_OUT] Data structure for the printer data (Struct_CP341_printData)
);

// *************************
// Struct_CP341_printData
// *************************
// myCP341_printData : Struct_CP341_printData;
myCP341_printData.variable := myVariable; // [IN] Variable to be printed (ARRAY [0..3] of
Struct_CP341_datRecord   )
myCP341_printData.format := myFormat; // [IN] Format string (ARRAY [0..150] OF BYTE )

// *************************
// Struct_CP341_datRecord
// *************************
// myCP341_datRecord : Struct_CP341_datRecord;
myCP341_datRecord.datalength := 0; // [IN] Quantity of data (UDINT )
myCP341_datRecord.data := myData; // [IN] Print data (ARRAY [0..31] OF BYTE )
```

2.4.6.3 _CP341_dwordToPrintData

Entry of a number of data type 'DWORD' in data structure 'Struct_CP341_printData'.

Hardware/software requirements:

Hardware version: 6ES7 341-1AH02-0AE0 after product version E01

Software version: as of V2.0.0

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myCP341_dwordToPrintData : _CP341_dwordToPrintData; // instance of FB (_CP341_dwordToPrintData)
myCP341_dwordToPrintData(
execute := myExecute, // [IN] Edge-triggered request activation (BOOL)
data := myData, // [IN] Variable/value to be entered in the data structure (DWORD)
numVariable := myNumVariable, // [IN] Number of the variable in whitch the entry is to be made 1 <=
NumberVariable <=4 (INT)
entryFormatString := myEntryFormatString, // [IN] Method of entry in the format string
(enumFormstring2)
entryAtByteNumber := myEntryAtByteNumber, // [IN] Specifies the byte at which the entry is to begin in
the substructure of the format string (USINT)
printData := myCP341_printData // [IN_OUT] Data structure for the printer data (Struct_CP341_printData)
);

// *************************
// Struct_CP341_printData
// *************************
// myCP341_printData : Struct_CP341_printData;
myCP341_printData.variable := myVariable; // [IN] Variable to be printed (ARRAY [0..3] of
Struct_CP341_datRecord   )
myCP341_printData.format := myFormat; // [IN] Format string (ARRAY [0..150] OF BYTE )

// *************************
// Struct_CP341_datRecord
// *************************
// myCP341_datRecord : Struct_CP341_datRecord;
myCP341_datRecord.datalength := 0; // [IN] Quantity of data (UDINT )
myCP341_datRecord.data := myData; // [IN] Print data (ARRAY [0..31] OF BYTE )
```

2.4.6.4 _CP341_getV24Signals

The _CP341_getV24Signals function block reads the RS 232C accompanying signals from the CP341 and makes them available on the output parameters. The functionality of the _CP341_getV24Signals function block can only be used when ASCII driver is assigned.

Hardware/software requirements:

Hardware version: 6ES7 341-1AH01-0AE0 after product version, E01 6ES7 341-1BH01-0AE0 after product version, E01 6ES7 341-1CH01-0AE0 after product version E01

Software version: none

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myCP341_getV24Signals : _CP341_getV24Signals; // instance of FB (_CP341_getV24Signals)
myCP341_getV24Signals(
enable := myEnable, // [IN] Receive enable (BOOL)
periIn := myPeriIn // [IN] I/O inputs of the CP transferred to the FB (ARRAY [0..15] OF BYTE)
);
```

2.4.6.5 _CP341_intToPrintData

Entry of a number of data type 'INT' in data structure 'Struct_CP341_printData'.

Hardware/software requirements:

Hardware version: 6ES7 341-1AH02-0AE0 after product version E01

Software version: as of V2.0.0

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myCP341_intToPrintData : _CP341_intToPrintData; // instance of FB (_CP341_intToPrintData)
myCP341_intToPrintData(
execute := myExecute, // [IN] Edge-triggered request activation (BOOL)
data := myData, // [IN] Variable/value to be entered in the data structure (INT)
numVariable := myNumVariable, // [IN] Number of the variable in whitch the entry is to be made 1 <=
NumberVariable <=4 (INT)
entryFormatString := myEntryFormatString, // [IN] Method of entry in the format string
(enumFormstring3)
entryAtByteNumber := myEntryAtByteNumber, // [IN] Specifies the byte at which the entry is to begin in
the substructure of the format string (USINT)
printData := myCP341_printData // [IN_OUT] Data structure for the printer data (Struct_CP341_printData)
);

// *************************
// Struct_CP341_printData
// *************************
// myCP341_printData : Struct_CP341_printData;
myCP341_printData.variable := myVariable; // [IN] Variable to be printed (ARRAY [0..3] of
Struct_CP341_datRecord   )
myCP341_printData.format := myFormat; // [IN] Format string (ARRAY [0..150] OF BYTE )

// *************************
// Struct_CP341_datRecord
// *************************
// myCP341_datRecord : Struct_CP341_datRecord;
myCP341_datRecord.datalength := 0; // [IN] Quantity of data (UDINT )
myCP341_datRecord.data := myData; // [IN] Print data (ARRAY [0..31] OF BYTE )
```

2.4.6.6 _CP341_printer

The CP341_printer function block can be used to send data of type 'Struct_CP341_printData' from the printer memory area to a serial printer. For example, the CP341_printer function block transfers a process message to the CP341. The CP341 prints out the process message on the connected printer.

Hardware/software requirements:

Hardware version: 6ES7 341-1AH02-0AE0 after product version E01

Software version: as of V2.0.0

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myCP341_printer : _CP341_printer; // instance of FB (_CP341_printer)
myCP341_printer(
execute := myExecute, // [IN] Initiates request on positive edge (BOOL)
reset := myReset, // [IN] Aborts request (BOOL)
moduleAddress := myModuleAddress, // [IN] Module address of the CP for data set transfer (from HW
Config) (DINT)
periIn := myPeriIn, // [IN] I/O inputs of the CP transferred to the FB (ARRAY [0..15] OF BYTE)
printData := myCP341_printData, // [IN_OUT] Send data array (data to be printed)
(Struct_CP341_printData)
periOut := myPeriOut // [IN_OUT] Prepared FB data for the I/O outputs of the CP (ARRAY [0..15] OF BYTE)
);

// ************************
// Struct_CP341_printData
// ************************
// myCP341_printData : Struct_CP341_printData;
myCP341_printData.format[0] := 16#00; // [IN] Format string (ARRAY [0..150] OF BYTE )

// ************************
// Struct_CP341_datRecord
// ************************
//
myCP341_printData.variable[0].datalength := 4; // [IN] Quantity of data (UDINT )
myCP341_printData.variable[0].data[0] := 16#00; // [IN] Print data (ARRAY [0..31] OF BYTE)
```

2.4.6.7 _CP341_printMsgText

Selection of message texts stored in the CP341.

Hardware/software requirements:

Hardware version: 6ES7 341-1AH02-0AE0 after product version E01

Software version: as of V2.0.0

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myCP341_printMsgText : _CP341_printMsgText; // instance of FB (_CP341_printMsgText)
myCP341_printMsgText(
execute := myExecute, // [IN] Edge-triggered request activation (BOOL)
numMsgText := myNumMsgText, // [IN] Variable/value to be entered in the data structure (USINT)
numVariable := myNumVariable, // [IN] Number of the variable in whitch the entry is to be made 1 <=
NumberVariable <=4 (INT)
entryFormatString := myEntryFormatString, // [IN] Method of entry in the format string
(enumFormstring3)
entryAtByteNumber := myEntryAtByteNumber, // [IN] Specifies the byte at which the entry is to begin in
the substructure of the format string (USINT)
printData := myCP341_printData // [IN_OUT] Data structure for the printer data (Struct_CP341_printData)
);

// *************************
// Struct_CP341_printData
// *************************
// myCP341_printData : Struct_CP341_printData;
myCP341_printData.variable := myVariable; // [IN] Variable to be printed (ARRAY [0..3] of
Struct_CP341_datRecord   )
myCP341_printData.format := myFormat; // [IN] Format string (ARRAY [0..150] OF BYTE )

// *************************
// Struct_CP341_datRecord
// *************************
// myCP341_datRecord : Struct_CP341_datRecord;
myCP341_datRecord.datalength := 0; // [IN] Quantity of data (UDINT )
myCP341_datRecord.data := myData; // [IN] Print data (ARRAY [0..31] OF BYTE )
```

2.4.6.8 _CP341_realToPrintData

Entry of a number of data type 'REAL' in data structure 'Struct_CP341_printData'.

Hardware/software requirements:

Hardware version: 6ES7 341-1AH02-0AE0 after product version E01

Software version: as of V2.0.0

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myCP341_realToPrintData : _CP341_realToPrintData; // instance of FB (_CP341_realToPrintData)
myCP341_realToPrintData(
execute := myExecute, // [IN] Edge-triggered request activation (BOOL)
data := myData, // [IN] Variable/value to be entered in the data structure (REAL)
numVariable := myNumVariable, // [IN] Number of the variable in whitch the entry is to be made 1 <=
NumberVariable <=4 (INT)
entryFormatString := myEntryFormatString, // [IN] Method of entry in the format string
(enumFormstring1)
entryAtByteNumber := myEntryAtByteNumber, // [IN] Specifies the byte at which the entry is to begin in
the substructure of the format string (USINT)
printData := myCP341_printData // [IN_OUT] Data structure for the printer data (Struct_CP341_printData)
);

// *************************
// Struct_CP341_printData
// *************************
// myCP341_printData : Struct_CP341_printData;
myCP341_printData.variable := myVariable; // [IN] Variable to be printed (ARRAY [0..3] of
Struct_CP341_datRecord   )
myCP341_printData.format := myFormat; // [IN] Format string (ARRAY [0..150] OF BYTE )

// *************************
// Struct_CP341_datRecord
// *************************
// myCP341_datRecord : Struct_CP341_datRecord;
myCP341_datRecord.datalength := 0; // [IN] Quantity of data (UDINT )
myCP341_datRecord.data := myData; // [IN] Print data (ARRAY [0..31] OF BYTE )
```

2.4.6.9 _CP341_receive

The _CP341_receive function block enables you to receive data from a communication partner in the receive data array (3964 (R) protocol, ASCII driver, RK 512) or to make data available for the communication partner (RK 512). The respective data array depends on the type of protocol used. If you are using the 3964 (R) protocol or ASCII driver, the receive data array is 'dataCI3964'. When an RK 512 computer interface is used, the data array is 'dataCI512'. In both cases, 4096 bytes are available. From the 'dataCI512' data array, data is also made available for the communication partner.

Hardware/software requirements:

Hardware version: 6ES7 341-1AH01-0AE0 after product version E01 6ES7 341-1AH02-0AE0 after product version E01 6ES7 341-1BH01-0AE0 after product version E01 6ES7 341-1CH01-0AE0 after product version E01

Software version: none

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myCP341_receive : _CP341_receive; // instance of FB (_CP341_receive)
myCP341_receive(
enable := myEnable, // [IN] Receive enable (BOOL)
reset := myReset, // [IN] Aborts request (BOOL)
moduleAddress := myModuleAddress, // [IN] Module address of the CP for data set transfer (from HW
Config) (DINT)
dataOffset := myDataOffset, // [IN] First element to be received (UDINT)
periIn := myPeriIn, // [IN] I/O inputs of the CP transferred to the FB (ARRAY [0..15] OF BYTE)
dataCI3964 := myDataCI3964, // [IN_OUT] Receive data array (ARRAY [0..4095] OF BYTE)
dataCI512 := myCP341_CI512CpData, // [IN_OUT] Data area for RK 512 two-dimensional array(6) (ARRAY
[0..6] of Struct_CP341_CI512CpData)
comFlags := myComFlags, // [IN_OUT] Communication flag area for RK 512 (ARRAY [0..31] of BYTE)
periOut := myPeriOut // [IN_OUT] Prepared FB data for the I/O outputs of the CP (ARRAY [0..15] OF BYTE)
);

// ***************************
// Struct_CP341_CI512CpData
// ***************************
// myCP341_CI512CpData : ARRAY [0..6] of Struct_CP341_CI512CpData;
// example for one element
myCP341_CI512CpData[0].data[0] := 16#01; // [IN_OUT] Data array for RK 512 computer interfacing (send
and fetch request) (ARRAY [0..4095] OF BYTE )
```

2.4.6.10 _CP341_send

The CP341_send function block can be used to send data from the send data array to a communication partner (3964 (R) protocol, RK 512, ASCII driver) or to retrieve data from a communication partner and store it there (RK 512). 4096 bytes are available.

Hardware/software requirements:

Hardware version: 6ES7 341-1AH01-0AE0 after product version E01 6ES7 341-1AH02-0AE0 after product version E01 6ES7 341-1BH01-0AE0 after product version E01 6ES7 341-1CH01-0AE0 after product version E01

Software version: none

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myCP341_send : _CP341_send; // instance of FB (_CP341_send)
myCP341_send(
mode := myMode, // [IN] SEND_CP: Sends data FETCH_CP: Fetches data from the communication partner
(EnumCP341SndFetch)
execute := myExecute, // [IN] Initiates request on positive edge (BOOL)
reset := myReset, // [IN] Aborts request (BOOL)
moduleAddress := myModuleAddress, // [IN] Module address of the CP for data set transfer (from HW
Config) (DINT)
dataOffset := myDataOffset, // [IN] First element to be sent (UDINT)
dataLength := myDataLength, // [IN] Number of elements to be sent (UDINT)
remoteCpuId := myRemoteCpuId, // [IN] Number of the remote communication partner (USINT)
remoteDataType := myRemoteDataType, // [IN] Area type in the remote communication partner (USINT)
remoteMemIndex := myRemoteMemIndex, // [IN] Memory index in the remote communication partner (1
<=remoteMemoryIndex<= 6) (UINT)
remoteDataOffset := myRemoteDataOffset, // [IN] First element in the remote communication partner
(UDINT)
remoteComFlagByte := myRemoteComFlagByte, // [IN] Communication flag index in the local communication
partner (USINT)
remoteComFlagBit := myRemoteComFlagBit, // [IN] Communication flag bit no. in the local communication
partner (USINT)
periIn := myPeriIn, // [IN] I/O inputs of the CP transferred to the FB (ARRAY [0..15] OF BYTE)
data := myData, // [IN_OUT] Send data array / Receive data for parameter  operatingMode=FETCH_CP
(ARRAY [0..4095] OF BYTE)
periOut := myPeriOut // [IN_OUT] Prepared FB data for the I/O outputs of the CP (ARRAY [0..15] OF BYTE)
);
```

2.4.6.11 _CP341_setV24Signals

The _CP341_setV24Signals function block can be used to set or reset RS 232C accompanying signals. The functionality of the _CP341_setV24Signals function block can only be used when an ASCII driver have been parameterized.

Hardware/software requirements:

Hardware version: 6ES7 341-1AH01-0AE0 after product version, E01 6ES7 341-1BH01-0AE0 after product version, E01 6ES7 341-1CH01-0AE0 after product version E01

Software version: none

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myCP341_setV24Signals : _CP341_setV24Signals; // instance of FB (_CP341_setV24Signals)
myCP341_setV24Signals(
enable := myEnable, // [IN] Receive enable (BOOL)
sigDtr := mySigDtr, // [IN] Signal 'Data terminal ready' (BOOL)
sigRts := mySigRts, // [IN] Signal 'Request to send' (BOOL)
periOut := myPeriOut // [IN_OUT] Prepared FB data for the I/O outputs of the CP (ARRAY [0..15] OF BYTE)
);
```

2.4.6.12 _CP341_wordToPrintData

Entry of a number of data type 'WORD' in data structure 'Struct_CP341_printData'.

Hardware/software requirements:

Hardware version: 6ES7 341-1AH02-0AE0 after product version E01

Software version: as of V2.0.0

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myCP341_wordToPrintData : _CP341_wordToPrintData; // instance of FB (_CP341_wordToPrintData)
myCP341_wordToPrintData(
execute := myExecute, // [IN] Edge-triggered request activation (BOOL)
data := myData, // [IN] Variable/value to be entered in the data structure (WORD)
numVariable := myNumVariable, // [IN] Number of the variable in whitch the entry is to be made 1 <=
NumberVariable <=4 (INT)
entryFormatString := myEntryFormatString, // [IN] Method of entry in the format string
(enumFormstring2)
entryAtByteNumber := myEntryAtByteNumber, // [IN] Specifies the byte at which the entry is to begin in
the substructure of the format string (USINT)
printData := myCP341_printData // [IN_OUT] Data structure for the printer data (Struct_CP341_printData)
);

// *************************
// Struct_CP341_printData
// *************************
// myCP341_printData : Struct_CP341_printData;
myCP341_printData.variable := myVariable; // [IN] Variable to be printed (ARRAY [0..3] of
Struct_CP341_datRecord   )
myCP341_printData.format := myFormat; // [IN] Format string (ARRAY [0..150] OF BYTE )

// *************************
// Struct_CP341_datRecord
// *************************
// myCP341_datRecord : Struct_CP341_datRecord;
myCP341_datRecord.datalength := 0; // [IN] Quantity of data (UDINT )
myCP341_datRecord.data := myData; // [IN] Print data (ARRAY [0..31] OF BYTE )
```

2.4.7 Communication modules - ET200S-SI

2.4.7.1 _ET200S_SI04_flowRts

The ET200S_SI04_flowRts function block can be used to modify additional parameters on the module if the module for "RTS/CTS" data flow control has been parameterized. Depending on which data interface you have chosen in the hardware configuration, you use either the _ET200S_SI04_flowRts for the 4-byte data interface, _ET200S_SI08_flowRts for the 8-byte data interface, or _ET200S_SI32_flowRts for the 32-byte data interface of the serial interface module.

Hardware/software requirements:

4-byte, 8-byte data interface:

Hardware version: 6ES7 138-4DF00-0AB0 after product version E03

Software version: none

4-byte, 8-byte, 32-byte data interface:

Hardware version: 6ES7 138-4DF01-0AB0 after product version E01

Software version: V1.0.4

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myET200S_SI04_flowRts : _ET200S_SI04_flowRts; // instance of FB (_ET200S_SI04_flowRts)
myET200S_SI04_flowRts(
execute := myExecute, // [IN] Initiates request on positive edge (BOOL)
reset := myReset, // [IN] Aborts request (BOOL)
waitForCts := myWaitForCts, // [IN] Delay time for CTS=ON, value range:20 ms<=waitForXON<=655350 ms
(DINT)
periIn := myPeriIn, // [IN] Transfers I/O inputs of the serial interface module to FB (ARRAY [0..3] OF
BYTE)
periOut := myPeriOut // [IN_OUT] Prepared data of the FB for the I/O outputs of the serial interface
module (ARRAY [0..3] OF BYTE)
);
```

2.4.7.2 _ET200S_SI04_flowV24

The _ET200S_SI04_flowV24 function block can be used to modify additional parameters on the module if the module "Automatic use of V24 signals" data flow control has been parameterized. Depending on which data interface you have chosen in the hardware configuration, you use either the _ET200S_SI04_flowV24 for the 4-byte data interface, _ET200S_SI08_flowV24 for the 8-byte data interface, or _ET200S_SI32_flowV24 for the 32-byte data interface of the serial interface module.

Hardware/software requirements:

4-byte, 8-byte data interface:

Hardware version: 6ES7 138-4DF00-0AB0 after product version E03

Software version: none

4-byte, 8-byte, 32-byte data interface:

Hardware version: 6ES7 138-4DF01-0AB0 after product version E01

Software version: V1.0.4

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myET200S_SI04_flowV24 : _ET200S_SI04_flowV24; // instance of FB (_ET200S_SI04_flowV24)
myET200S_SI04_flowV24(
execute := myExecute, // [IN] Initiates request on positive edge (BOOL)
reset := myReset, // [IN] Aborts request (BOOL)
timeRtsOff := myTimeRtsOff, // [IN] Time that must elapse after the transfer before RTS is disabled,
value range:0<=timeRTSOff<=655350 ms (DINT)
maxWaitTime := myMaxWaitTime, // [IN] Wait time for CTS=ON by partner after RTS=ON has been set, value
range:0<=maxWaitTime<=655350 ms (DINT)
periIn := myPeriIn, // [IN] Transfers I/O inputs of the serial interface module to FB (ARRAY [0..3] OF
BYTE)
periOut := myPeriOut // [IN_OUT] Prepared data of the FB for the I/O outputs of the serial interface
module (ARRAY [0..3] OF BYTE)
);
```

2.4.7.3 _ET200S_SI04_flowXon

The ET200S_SI04_flowXon function block can be used to modify parameters on the module independently of the parameter assignment interface if "XON/XOFF" data flow control has been assigned for the module. Depending on which data interface you have chosen in the hardware configuration, you use either the _ET200S_SI04_flowXon for the 4-byte data interface, _ET200S_SI08_flowXon for the 8-byte data interface, or _ET200S_SI32_flowXon for the 32-byte data interface of the serial interface module.

Hardware/software requirements:

4-byte, 8-byte data interface:

Hardware version: 6ES7 138-4DF00-0AB0 after product version E03

Software version: none

4-byte, 8-byte, 32-byte data interface:

Hardware version: 6ES7 138-4DF01-0AB0 after product version E01

Software version: V1.0.4

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myET200S_SI04_flowXon : _ET200S_SI04_flowXon; // instance of FB (_ET200S_SI04_flowXon)
myET200S_SI04_flowXon(
execute := myExecute, // [IN] Initiates request on positive edge (BOOL)
reset := myReset, // [IN] Aborts request (BOOL)
sigXon := mySigXon, // [IN] Code for XON characters (BYTE)
sigXoff := mySigXoff, // [IN] Code for XOFF characters (BYTE)
waitForXon := myWaitForXon, // [IN] Delay time for XON after XOFF, value range:20
ms<=waitForXON<=655350 ms (DINT)
periIn := myPeriIn, // [IN] Transfers I/O inputs of the serial interface module to FB (ARRAY [0..3] OF
BYTE)
periOut := myPeriOut // [IN_OUT] Prepared data of the FB for the I/O outputs of the serial interface
module (ARRAY [0..3] OF BYTE)
);
```

2.4.7.4 _ET200S_SI04_getV24Sig

When ASCII driver is assigned, the _ET200S_SI04_getV24Sig function block reads the RS 232C accompanying signals and provides them to the user as output parameters. Depending on which data interface you have chosen in the hardware configuration, you use either the _ET200S_SI04_getV24Sig for the 4-byte data interface, _ET200S_SI08_getV24Sig for the 8-byte data interface, or _ET200S_SI32_getV24Sig for the 32-byte data interface of the serial interface module.

Hardware/software requirements:

4-byte, 8-byte data interface:

Hardware version: 6ES7 138-4DF00-0AB0 after product version E03

Software version: none

4-byte, 8-byte, 32-byte data interface:

Hardware version: 6ES7 138-4DF01-0AB0 after product version E01

Software version: V1.0.4

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myET200S_SI04_getV24Sig : _ET200S_SI04_getV24Sig; // instance of FB (_ET200S_SI04_getV24Sig)
myET200S_SI04_getV24Sig(
execute := myExecute, // [IN] Initiates request on positive edge (BOOL)
reset := myReset, // [IN] Aborts request (BOOL)
periIn := myPeriIn, // [IN] Transfers I/O inputs of the serial interface module to FB (ARRAY [0..3] OF
BYTE)
periOut := myPeriOut // [IN_OUT] Prepared data of the FB for the I/O outputs of the serial interface
module (ARRAY [0..3] OF BYTE)
);
```

2.4.7.5 _ET200S_SI04_receive

The _ET200S_SI04_receive function block can be used to receive data from a communication partner in the receive data array. 1024 bytes are available. Depending on which data interface you have chosen in the hardware configuration, you use either the _ET200S_SI04_receive for the 4-byte data interface, _ET200S_SI08_receive for the 8-byte data interface, or _ET200S_SI32_receive for the 32-byte data interface of the serial interface module. For the transfer operation, you can use the 3964(R) protocol or ASCII driver.

Hardware/software requirements:

4-byte, 8-byte data interface:

Hardware version: 6ES7 138-4DF00-0AB0 after product version E03

Software version: none

4-byte, 8-byte, 32-byte data interface:

Hardware version: 6ES7 138-4DF01-0AB0 after product version E01

Software version: V1.0.4

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myET200S_SI04_receive : _ET200S_SI04_receive; // instance of FB (_ET200S_SI04_receive)
myET200S_SI04_receive(
enable := myEnable, // [IN] Receive enable (BOOL)
reset := myReset, // [IN] Aborts request (BOOL)
dataOffset := myDataOffset, // [IN] Offset of the first element to be received (INT)
periIn := myPeriIn, // [IN] Transfers I/O inputs of the serial interface module to FB (ARRAY [0..3] OF
BYTE)
data := myData, // [IN_OUT] Receive data array (ARRAY [0..1023] OF BYTE)
periOut := myPeriOut // [IN_OUT] Prepared data of the FB for the I/O outputs of the serial interface
module (ARRAY [0..3] OF BYTE)
);
```

2.4.7.6 _ET200S_SI04_send

The _ET200S_SI04_send function block can be used to send data from the 'data' send data array to a communication partner. 1024 bytes are available. Depending on which data interface you have chosen in the hardware configuration, you use either the _ET200S_SI04_send for the 4-byte data interface, _ET200S_SI08_send for the 8-byte data interface, or _ET200S_SI32_send for the 32-byte data interface of the serial interface module. For the transfer operation, you can use the 3964(R) protocol or ASCII driver.

Hardware/software requirements:

4-byte, 8-byte data interface:

Hardware version: 6ES7 138-4DF00-0AB0 after product version E03

Software version: none

4-byte, 8-byte, 32-byte data interface:

Hardware version: 6ES7 138-4DF01-0AB0 after product version E01

Software version: V1.0.4

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myET200S_SI04_send : _ET200S_SI04_send; // instance of FB (_ET200S_SI04_send)
myET200S_SI04_send(
execute := myExecute, // [IN] Initiates request on positive edge (BOOL)
reset := myReset, // [IN] Aborts request (BOOL)
dataOffset := myDataOffset, // [IN] Offset of the first element to be sent (INT)
datalength := myDatalength, // [IN] Number of elements to be sent 0 = dataLength = 224 (INT)
periIn := myPeriIn, // [IN] Transfers I/O inputs of the serial interface module to FB (ARRAY [0..3] OF
BYTE)
data := myData, // [IN_OUT] Send data array (ARRAY [0..1023] OF BYTE)
periOut := myPeriOut // [IN_OUT] Prepared data of the FB for the I/O outputs of the serial interface
module (ARRAY [0..3] OF BYTE)
);
```

2.4.7.7 _ET200S_SI04_setV24Sig

The _ET200S_SI04_setV24Sig function block can be used to set or reset RS 232C accompanying signals if ASCII driver is assigned.. Depending on which data interface you have chosen in the hardware configuration, you use either the _ET200S_SI04_setV24Sig for the 4-byte data interface, _ET200S_SI08_setV24Sig for the 8-byte data interface, or _ET200S_SI32_setV24Sig for the 32-byte data interface of the serial interface module.

Hardware/software requirements:

4-byte, 8-byte data interface:

Hardware version: 6ES7 138-4DF00-0AB0 after product version E03

Software version: none

4-byte, 8-byte, 32-byte data interface:

Hardware version: 6ES7 138-4DF01-0AB0 after product version E01

Software version: V1.0.4

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myET200S_SI04_setV24Sig : _ET200S_SI04_setV24Sig; // instance of FB (_ET200S_SI04_setV24Sig)
myET200S_SI04_setV24Sig(
execute := myExecute, // [IN] Initiates request on positive edge (BOOL)
reset := myReset, // [IN] Aborts request (BOOL)
sigRts := mySigRts, // [IN] Signal 'Request to send' (BOOL)
sigDtr := mySigDtr, // [IN] Signal 'Data terminal ready' (BOOL)
periIn := myPeriIn, // [IN] Transfers I/O inputs of the serial interface module to FB (ARRAY [0..3] OF
BYTE)
periOut := myPeriOut // [IN_OUT] Prepared data of the FB for the I/O outputs of the serial interface
module (ARRAY [0..3] OF BYTE)
);
```

2.4.7.8 _ET200S_SI08_flowRts

The ET200S_SI08_flowRts function block can be used to modify additional parameters on the module if the module for "RTS/CTS" data flow control has been parameterized. Depending on which data interface you have chosen in the hardware configuration, you use either the _ET200S_SI04_flowRts for the 4-byte data interface, _ET200S_SI08_flowRts for the 8-byte data interface, or _ET200S_SI32_flowRts for the 32-byte data interface of the serial interface module.

Hardware/software requirements:

4-byte, 8-byte data interface:

Hardware version: 6ES7 138-4DF00-0AB0 after product version E03

Software version: none

4-byte, 8-byte, 32-byte data interface:

Hardware version: 6ES7 138-4DF01-0AB0 after product version E01

Software version: V1.0.4

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myET200S_SI08_flowRts : _ET200S_SI08_flowRts; // instance of FB (_ET200S_SI08_flowRts)
myET200S_SI08_flowRts(
execute := myExecute, // [IN] Initiates request on positive edge (BOOL)
reset := myReset, // [IN] Aborts request (BOOL)
waitForCts := myWaitForCts, // [IN] Delay time for CTS=ON, value range:20 ms<=waitForXON<=655350 ms
(DINT)
periIn := myPeriIn, // [IN] Transfers I/O inputs of the serial interface module to FB (ARRAY [0..7] OF
BYTE)
periOut := myPeriOut // [IN_OUT] Prepared data of the FB for the I/O outputs of the serial interface
module (ARRAY [0..7] OF BYTE)
);
```

2.4.7.9 _ET200S_SI08_flowV24

The _ET200S_SI08_flowV24 function block can be used to modify additional parameters on the module if the module "Automatic use of V24 signals" data flow control has been parameterized. Depending on which data interface you have chosen in the hardware configuration, you use either the _ET200S_SI04_flowV24 for the 4-byte data interface, _ET200S_SI08_flowV24 for the 8-byte data interface, or _ET200S_SI32_flowV24 for the 32-byte data interface of the serial interface module.

Hardware/software requirements:

4-byte, 8-byte data interface:

Hardware version: 6ES7 138-4DF00-0AB0 after product version E03

Software version: none

4-byte, 8-byte, 32-byte data interface:

Hardware version: 6ES7 138-4DF01-0AB0 after product version E01

Software version: V1.0.4

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "SIMOTION Utilities & Applications" CD-ROM. The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myET200S_SI08_flowV24 : _ET200S_SI08_flowV24; // instance of FB (_ET200S_SI08_flowV24)
myET200S_SI08_flowV24(
execute := myExecute, // [IN] Initiates request on positive edge (BOOL)
reset := myReset, // [IN] Aborts request (BOOL)
timeRtsOff := myTimeRtsOff, // [IN] Time that must elapse after the transfer before RTS is disabled,
value range:0<=timeRTSOff<=655350 ms (DINT)
maxWaitTime := myMaxWaitTime, // [IN] Wait time for CTS=ON by partner after RTS=ON has been set, value
range:0<=maxWaitTime<=655350 ms (DINT)
periIn := myPeriIn, // [IN] Transfers I/O inputs of the serial interface module to FB (ARRAY [0..7] OF
BYTE)
periOut := myPeriOut // [IN_OUT] Prepared data of the FB for the I/O outputs of the serial interface
module (ARRAY [0..7] OF BYTE)
);
```

2.4.7.10 _ET200S_SI08_flowXon

The _ET200S_SI08_flowXon function block can be used to modify parameters on the module independently of the parameter assignment interface if "XON/XOFF" data flow control has been assigned for the module. Depending on which data interface you have chosen in the hardware configuration, you use either the _ET200S_SI04_flowXon for the 4-byte data interface, _ET200S_SI08_flowXon for the 8-byte data interface, or _ET200S_SI32_flowXon for the 32-byte data interface of the serial interface module.

Hardware/software requirements:

4-byte, 8-byte data interface:

Hardware version: 6ES7 138-4DF00-0AB0 after product version E03

Software version: none

4-byte, 8-byte, 32-byte data interface:

Hardware version: 6ES7 138-4DF01-0AB0 after product version E01

Software version: V1.0.4

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myET200S_SI08_flowXon : _ET200S_SI08_flowXon; // instance of FB (_ET200S_SI08_flowXon)
myET200S_SI08_flowXon(
execute := myExecute, // [IN] Initiates request on positive edge (BOOL)
reset := myReset, // [IN] Aborts request (BOOL)
sigXon := mySigXon, // [IN] Code for XON characters (BYTE)
sigXoff := mySigXoff, // [IN] Code for XOFF characters (BYTE)
waitForXon := myWaitForXon, // [IN] Delay time for XON after XOFF, value range:20
ms<=waitForXON<=655350 ms (DINT)
periIn := myPeriIn, // [IN] Transfers I/O inputs of the serial interface module to FB (ARRAY [0..7] OF
BYTE)
periOut := myPeriOut // [IN_OUT] Prepared data of the FB for the I/O outputs of the serial interface
module (ARRAY [0..7] OF BYTE)
);
```

2.4.7.11 _ET200S_SI08_getV24Sig

When ASCII driver is assigned, the _ET200S_SI08_getV24Sig function block reads the RS 232C accompanying signals and provides them to the user as output parameters. Depending on which data interface you have chosen in the hardware configuration, you use either the _ET200S_SI04_getV24Sig for the 4-byte data interface, _ET200S_SI08_getV24Sig for the 8-byte data interface, or _ET200S_SI32_getV24Sig for the 32-byte data interface of the serial interface module.

Hardware/software requirements:

4-byte, 8-byte data interface:

Hardware version: 6ES7 138-4DF00-0AB0 after product version E03

Software version: none

4-byte, 8-byte, 32-byte data interface:

Hardware version: 6ES7 138-4DF01-0AB0 after product version E01

Software version: V1.0.4

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myET200S_SI08_getV24Sig : _ET200S_SI08_getV24Sig; // instance of FB (_ET200S_SI08_getV24Sig)
myET200S_SI08_getV24Sig(
execute := myExecute, // [IN] Initiates request on positive edge (BOOL)
reset := myReset, // [IN] Aborts request (BOOL)
periIn := myPeriIn, // [IN] Transfers I/O inputs of the serial interface module to FB (ARRAY [0..7] OF
BYTE)
periOut := myPeriOut // [IN_OUT] Prepared data of the FB for the I/O outputs of the serial interface
module (ARRAY [0..7] OF BYTE)
);
```

2.4.7.12 _ET200S_SI08_receive

The _ET200S_SI08_receive function block can be used to receive data from a communication partner in the 'data' receive data array. 1024 bytes are available. Depending on which data interface you have chosen in the hardware configuration, you use either the _ET200S_SI04_receive for the 4-byte data interface, _ET200S_SI08_receive for the 8-byte data interface, or _ET200S_SI32_receive for the 32-byte data interface of the serial interface module. For the transfer operation, you can use the 3964(R) protocol or ASCII driver.

Hardware/software requirements:

4-byte, 8-byte data interface:

Hardware version: 6ES7 138-4DF00-0AB0 after product version E03

Software version: none

4-byte, 8-byte, 32-byte data interface:

Hardware version: 6ES7 138-4DF01-0AB0 after product version E01

Software version: V1.0.4

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myET200S_SI08_receive : _ET200S_SI08_receive; // instance of FB (_ET200S_SI08_receive)
myET200S_SI08_receive(
enable := myEnable, // [IN] Receive enable (BOOL)
reset := myReset, // [IN] Aborts request (BOOL)
dataOffset := myDataOffset, // [IN] Offset of the first element to be received (INT)
periIn := myPeriIn, // [IN] Transfers I/O inputs of the serial interface module to FB (ARRAY [0..7] OF
BYTE)
data := myData, // [IN_OUT] Receive data array (ARRAY [0..1023] OF BYTE)
periOut := myPeriOut // [IN_OUT] Prepared data of the FB for the I/O outputs of the serial interface
module (ARRAY [0..7] OF BYTE)
);
```

2.4.7.13 _ET200S_SI08_send

The _ET200S_SI08_send function block can be used to send data from the send data array to a communication partner. 1024 bytes are available. Depending on which data interface you have chosen in the hardware configuration, you use either the _ET200S_SI04_send for the 4-byte data interface, _ET200S_SI08_send for the 8-byte data interface, or _ET200S_SI32_send for the 32-byte data interface of the serial interface module. For the transfer operation, you can use the 3964(R) protocol or ASCII driver.

Hardware/software requirements:

4-byte, 8-byte data interface:

Hardware version: 6ES7 138-4DF00-0AB0 after product version E03

Software version: none

4-byte, 8-byte, 32-byte data interface:

Hardware version: 6ES7 138-4DF01-0AB0 after product version E01

Software version: V1.0.4

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "SIMOTION Utilities & Applications" CD-ROM. The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myET200S_SI08_send : _ET200S_SI08_send; // instance of FB (_ET200S_SI08_send)
myET200S_SI08_send(
execute := myExecute, // [IN] Initiates request on positive edge (BOOL)
reset := myReset, // [IN] Aborts request (BOOL)
dataOffset := myDataOffset, // [IN] Offset of the first element to be sent (INT)
datalength := myDatalength, // [IN] Number of elements to be sent 0 = dataLength = 224 (INT)
periIn := myPeriIn, // [IN] Transfers I/O inputs of the serial interface module to FB (ARRAY [0..7] OF
BYTE)
data := myData, // [IN_OUT] Send data array (ARRAY [0..1023] OF BYTE)
periOut := myPeriOut // [IN_OUT] Prepared data of the FB for the I/O outputs of the serial interface
module (ARRAY [0..7] OF BYTE)
);
```

2.4.7.14 _ET200S_SI08_setV24Sig

The _ET200S_SI08_setV24Sig function block can be used to set or reset RS 232C accompanying signals if ASCII driver is assigned.. Depending on which data interface you have chosen in the hardware configuration, you use either the _ET200S_SI04_setV24Sig for the 4-byte data interface, _ET200S_SI08_setV24Sig for the 8-byte data interface, or _ET200S_SI32_setV24Sig for the 32-byte data interface of the serial interface module.

Hardware/software requirements:

4-byte, 8-byte data interface: Hardware version: 6ES7 138-4DF00-0AB0 as of product version E03. Software version: None. 4-byte, 8-byte, 32-byte data interface: Hardware version: 6ES7 138-4DF01-0AB0 as of product version E01. Software version: V1.0.4

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myET200S_SI08_setV24Sig : _ET200S_SI08_setV24Sig; // instance of FB (_ET200S_SI08_setV24Sig)
myET200S_SI08_setV24Sig(
execute := myExecute, // [IN] Initiates request on positive edge (BOOL)
reset := myReset, // [IN] Aborts request (BOOL)
sigRts := mySigRts, // [IN] Signal 'Request to send' (BOOL)
sigDtr := mySigDtr, // [IN] Signal 'Data terminal ready' (BOOL)
periIn := myPeriIn, // [IN] Transfers I/O inputs of the serial interface module to FB (ARRAY [0..7] OF
BYTE)
periOut := myPeriOut // [IN_OUT] Prepared data of the FB for the I/O outputs of the serial interface
module (ARRAY [0..7] OF BYTE)
);
```

2.4.7.15 _ET200S_SI32_flowRts

The ET200S_SI32_flowRts function block can be used to modify additional parameters on the module if the module for "RTS/CTS" data flow control has been parameterized. Depending on which data interface you have chosen in the hardware configuration, you use either the _ET200S_SI04_flowRts for the 4-byte data interface, _ET200S_SI08_flowRts for the 8-byte data interface, or _ET200S_SI32_flowRts for the 32-byte data interface of the serial interface module.

Hardware/software requirements:

4-byte, 8-byte data interface:

Hardware version: 6ES7 138-4DF00-0AB0 after product version E03

Software version: none

4-byte, 8-byte, 32-byte data interface:

Hardware version: 6ES7 138-4DF01-0AB0 after product version E01

Software version: V1.0.4

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myET200S_SI32_flowRts : _ET200S_SI32_flowRts; // instance of FB (_ET200S_SI32_flowRts)
myET200S_SI32_flowRts(
execute := myExecute, // [IN] Initiates request on positive edge (BOOL)
reset := myReset, // [IN] Aborts request (BOOL)
waitForCts := myWaitForCts, // [IN] Delay time for CTS=ON, value range:20 ms<=waitForXON<=655350 ms
(DINT)
periIn := myPeriIn, // [IN] Transfers I/O inputs of the serial interface module to FB (ARRAY [0..31]
OF BYTE)
periOut := myPeriOut // [IN_OUT] Prepared data of the FB for the I/O outputs of the serial interface
module (ARRAY [0..31] OF BYTE)
);
```

2.4.7.16 _ET200S_SI32_flowV24

The _ET200S_SI32_flowV24 function block can be used to modify additional parameters on the module if the module "Automatic use of V24 signals" data flow control has been parameterized. Depending on which data interface you have chosen in the hardware configuration, you use either the _ET200S_SI04_flowV24 for the 4-byte data interface, _ET200S_SI08_flowV24 for the 8-byte data interface, or _ET200S_SI32_flowV24 for the 32-byte data interface of the serial interface module.

Hardware/software requirements:

4-byte, 8-byte data interface:

Hardware version: 6ES7 138-4DF00-0AB0 after product version E03

Software version: none

4-byte, 8-byte, 32-byte data interface:

Hardware version: 6ES7 138-4DF01-0AB0 after product version E01

Software version: V1.0.4

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myET200S_SI32_flowV24 : _ET200S_SI32_flowV24; // instance of FB (_ET200S_SI32_flowV24)
myET200S_SI32_flowV24(
execute := myExecute, // [IN] Initiates request on positive edge (BOOL)
reset := myReset, // [IN] Aborts request (BOOL)
timeRtsOff := myTimeRtsOff, // [IN] Time that must elapse after the transfer before RTS is disabled,
value range:0<=timeRTSOff<=655350 ms (DINT)
maxWaitTime := myMaxWaitTime, // [IN] Wait time for CTS=ON by partner after RTS=ON has been set, value
range:0<=maxWaitTime<=655350 ms (DINT)
periIn := myPeriIn, // [IN] Transfers I/O inputs of the serial interface module to FB (ARRAY [0..31]
OF BYTE)
periOut := myPeriOut // [IN_OUT] Prepared data of the FB for the I/O outputs of the serial interface
module (ARRAY [0..31] OF BYTE)
);
```

2.4.7.17 _ET200S_SI32_flowXon

The ET200S_SI32_flowXon function block can be used to modify parameters on the module independently of the parameter assignment interface if "XON/XOFF" data flow control has been assigned for the module. Depending on which data interface you have chosen in the hardware configuration, you use either the _ET200S_SI04_flowXon for the 4-byte data interface, _ET200S_SI08_flowXon for the 8-byte data interface, or _ET200S_SI32_flowXon for the 32-byte data interface of the serial interface module.

Hardware/software requirements:

4-byte, 8-byte data interface:

Hardware version: 6ES7 138-4DF00-0AB0 after product version E03

Software version: none

4-byte, 8-byte, 32-byte data interface:

Hardware version: 6ES7 138-4DF01-0AB0 after product version E01

Software version: V1.0.4

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myET200S_SI32_flowXon : _ET200S_SI32_flowXon; // instance of FB (_ET200S_SI32_flowXon)
myET200S_SI32_flowXon(
execute := myExecute, // [IN] Initiates request on positive edge (BOOL)
reset := myReset, // [IN] Aborts request (BOOL)
sigXon := mySigXon, // [IN] Code for XON characters (BYTE)
sigXoff := mySigXoff, // [IN] Code for XOFF characters (BYTE)
waitForXon := myWaitForXon, // [IN] Delay time for XON after XOFF, value range:20
ms<=waitForXON<=655350 ms (DINT)
periIn := myPeriIn, // [IN] Transfers I/O inputs of the serial interface module to FB (ARRAY [0..31]
OF BYTE)
periOut := myPeriOut // [IN_OUT] Prepared data of the FB for the I/O outputs of the serial interface
module (ARRAY [0..31] OF BYTE)
);
```

2.4.7.18 _ET200S_SI32_getV24Sig

When ASCII driver is assigned, the _ET200S_SI32_getV24Sig function block reads the RS 232C accompanying signals and provides them to the user as output parameters. Depending on which data interface you have chosen in the hardware configuration, you use either the _ET200S_SI04_getV24Sig for the 4-byte data interface, _ET200S_SI08_getV24Sig for the 8-byte data interface, or _ET200S_SI32_getV24Sig for the 32-byte data interface of the serial interface module.

Hardware/software requirements:

4-byte, 8-byte data interface: Hardware version: 6ES7 138-4DF00-0AB0 as of product version E03. Software version: None. 4-byte, 8-byte, 32-byte data interface: Hardware version: 6ES7 138-4DF01-0AB0 as of product version E01. Software version: V1.0.4

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myET200S_SI32_getV24Sig : _ET200S_SI32_getV24Sig; // instance of FB (_ET200S_SI32_getV24Sig)
myET200S_SI32_getV24Sig(
execute := myExecute, // [IN] Initiates request on positive edge (BOOL)
reset := myReset, // [IN] Aborts request (BOOL)
periIn := myPeriIn, // [IN] Transfers I/O inputs of the serial interface module to FB (ARRAY [0..31]
OF BYTE)
periOut := myPeriOut // [IN_OUT] Prepared data of the FB for the I/O outputs of the serial interface
module (ARRAY [0..31] OF BYTE)
);
```

2.4.7.19 _ET200S_SI32_receive

The _ET200S_SI32_receive function block can be used to receive data from a communication partner in the 'data' receive data array. 1024 bytes are available. Depending on which data interface you have chosen in the hardware configuration, you use either the _ET200S_SI04_receive for the 4-byte data interface, _ET200S_SI08_receive for the 8-byte data interface, or _ET200S_SI32_receive for the 32-byte data interface of the serial interface module. For the transfer operation, you can use the 3964(R) protocol or ASCII driver.

Hardware/software requirements:

4-byte, 8-byte data interface: Hardware version: 6ES7 138-4DF00-0AB0 as of product version E03. Software version: None. 4-byte, 8-byte, 32-byte data interface: Hardware version: 6ES7 138-4DF01-0AB0 as of product version E01. Software version: V1.0.4

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myET200S_SI32_receive : _ET200S_SI32_receive; // instance of FB (_ET200S_SI32_receive)
myET200S_SI32_receive(
enable := myEnable, // [IN] Receive enable (BOOL)
reset := myReset, // [IN] Aborts request (BOOL)
dataOffset := myDataOffset, // [IN] Offset of the first element to be received (INT)
periIn := myPeriIn, // [IN] Transfers I/O inputs of the serial interface module to FB (ARRAY [0..31]
OF BYTE)
data := myData, // [IN_OUT] Receive data array (ARRAY [0..1023] OF BYTE)
periOut := myPeriOut // [IN_OUT] Prepared data of the FB for the I/O outputs of the serial interface
module (ARRAY [0..31] OF BYTE)
);
```

2.4.7.20 _ET200S_SI32_send

The _ET200S_SI32_send function block can be used to send data from the 'data' send data array to a communication partner. 1024 bytes are available. Depending on which data interface you have chosen in the hardware configuration, you use either the _ET200S_SI04_send for the 4-byte data interface, _ET200S_SI08_send for the 8-byte data interface, or _ET200S_SI32_send for the 32-byte data interface of the serial interface module. For the transfer operation, you can use the 3964(R) protocol or ASCII driver.

Hardware/software requirements:

4-byte, 8-byte data interface: Hardware version: 6ES7 138-4DF00-0AB0 as of product version E03. Software version: None. 4-byte, 8-byte, 32-byte data interface: Hardware version: 6ES7 138-4DF01-0AB0 as of product version E01. Software version: V1.0.4

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myET200S_SI32_send : _ET200S_SI32_send; // instance of FB (_ET200S_SI32_send)
myET200S_SI32_send(
execute := myExecute, // [IN] Initiates request on positive edge (BOOL)
reset := myReset, // [IN] Aborts request (BOOL)
dataOffset := myDataOffset, // [IN] Offset of the first element to be sent (INT)
datalength := myDatalength, // [IN] Number of elements to be sent 0 <= dataLength <= 224 (INT)
periIn := myPeriIn, // [IN] Transfers I/O inputs of the serial interface module to FB (ARRAY [0..31]
OF BYTE)
data := myData, // [IN_OUT] Send data array (ARRAY [0..1023] OF BYTE)
periOut := myPeriOut // [IN_OUT] Prepared data of the FB for the I/O outputs of the serial interface
module (ARRAY [0..31] OF BYTE)
);
```

2.4.7.21 _ET200S_SI32_setV24Sig

The _ET200S_SI32_setV24Sig function block can be used to set or reset RS 232C accompanying signals if ASCII driver is assigned.. Depending on which data interface you have chosen in the hardware configuration, you use either the _ET200S_SI04_setV24Sig for the 4-byte data interface, _ET200S_SI08_setV24Sig for the 8-byte data interface, or _ET200S_SI32_setV24Sig for the 32-byte data interface of the serial interface module.

Hardware/software requirements:

4-byte, 8-byte data interface: Hardware version: 6ES7 138-4DF00-0AB0 as of product version E03. Software version: None. 4-byte, 8-byte, 32-byte data interface: Hardware version: 6ES7 138-4DF01-0AB0 as of product version E01. Software version: V1.0.4

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myET200S_SI32_setV24Sig : _ET200S_SI32_setV24Sig; // instance of FB (_ET200S_SI32_setV24Sig)
myET200S_SI32_setV24Sig(
execute := myExecute, // [IN] Initiates request on positive edge (BOOL)
reset := myReset, // [IN] Aborts request (BOOL)
sigRts := mySigRts, // [IN] Signal 'Request to send' (BOOL)
sigDtr := mySigDtr, // [IN] Signal 'Data terminal ready' (BOOL)
periIn := myPeriIn, // [IN] Transfers I/O inputs of the serial interface module to FB (ARRAY [0..31]
OF BYTE)
periOut := myPeriOut // [IN_OUT] Prepared data of the FB for the I/O outputs of the serial interface
module (ARRAY [0..31] OF BYTE)
);
```

2.4.8 Ident systems

2.4.8.1 _PIB_001KB

The function blocks act as the communication interface between a standard profile RFID system (e.g. ASM 456) and the user program.

The _PIB_001KB function block supports the following functions:

- configuration - command execution - reading and writing data - diagnostics

Hardware/software requirements:

All PNO standard profile ASM456 RFID devices: Hardware version: As of 6GT2002-0ED00. Software version: as of V3.0.0

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myPIB_001KB : _PIB_001KB; // instance of FB (_PIB_001KB)
myPIB_001KB(
EXECUTE := myEXECUTE, // [IN]  rising edge: start of command            ( BOOL )
ID := myID, // [IN]  base address of peripheral device        (DWORD )
INDEX := myINDEX, // [IN]  identification of channel data           (  INT )
OFFSET := myOFFSET, // [IN]  offset of channel to base address        (  INT )
RPTCMD := myRPTCMD, // [IN]  selects command repetition               ( BOOL )
SRESET := mySRESET, // [IN]  soft-reset                               ( BOOL )
INIT := myINIT, // [IN]  initializes ident unit                    ( BOOL )
UOUT0 := myUOUT0, // [IN]  user output 0 of the ident unit          ( BOOL )
UOUT1 := myUOUT1, // [IN]  user output 1 of the ident unit          ( BOOL )
UOUT2 := myUOUT2, // [IN]  user output 2 of the ident unit          ( BOOL )
UOUT3 := myUOUT3, // [IN]  user output 3 of the ident unit          ( BOOL )
RDGATE := myRDGATE, // [IN]  activates reading gate                  ( BOOL )
CMDDIM := myCMDDIM, // [IN]  number of commands in buffer CMDREF      (  INT )
CMDSEL := myCMDSEL, // [IN]  commandselector                          (  INT )
TXBUFLEN := myTXBUFLEN, // [IN]  used length of transmitbuffer            ( DINT )
TXSTART := myTXSTART, // [IN]  index of 1st data item in transmitbuffer ( DINT )
RXBUFLEN := myRXBUFLEN, // [IN]  used length of receivebuffer             ( DINT )
RXSTART := myRXSTART, // [IN]  index of 1st data item in receivebuffer  ( DINT )
CMDREF := myCMDREF, // [IN_OUT] commandbuffer  (ARRAY [1..10] OF _PIB_COMMAND)
TXREF := myTXREF, // [IN_OUT] transmitbuffer       (ARRAY [1..1024] OF BYTE)
RXREF := myRXREF // [IN_OUT] receivebuffer        (ARRAY [1..1024] OF BYTE)
);

// ***************
// _PIB_COMMAND
// ***************
// myCMDREF : ARRAY [1..10] OF _PIB_COMMAND;  //   commandbuffer
// index 10 is for example only
myCMDREF[10].CMD := 16#00; // command-ID                                                    (BYTE)
myCMDREF[10].Config := 16#00; // configuration byte                                          (BYTE)
myCMDREF[10].OffsetBuffer := 0; // relative offset within RXBUF                              (INT)
myCMDREF[10].UID[1] := 16#00; // identifies a single transponder           (ARRAY [1..8] OF BYTE)
myCMDREF[10].FileName[1] := 16#00; // specifies the file to be accessed         (ARRAY [1..8] OF BYTE)
myCMDREF[10].Offset := 0; // specifies the relative offset within the specified file   (DINT)
myCMDREF[10].Length := 0; // specifies the number of bytes to be transmitted            (INT)
myCMDREF[10].StartAddress := 0; // specifies a physical address within the tag memory       (DINT)
myCMDREF[10].Attributes := 16#00; // encoded file attributes                               (BYTE)
myCMDREF[10].NextMode := 16#00; // specifies how tags are handled                         (BYTE)
myCMDREF[10].Timeout := 0; // maximum time for barcode to be read                         (INT)
myCMDREF[10].ObjectNumber := 0; // used to concatenate an object with its barcode information (INT)
myCMDREF[10].FileType := 16#0000; // groups files according to user specific criteria          (WORD)
```

2.4.8.2 _PIB_016KB

The function blocks act as the communication interface between a standard profile RFID system (e.g. ASM 456) and the user program.

The _PIB_016KB function block supports the following functions:

- configuration - command execution - reading and writing data - diagnostics

Hardware/software requirements:

All PNO standard profile ASM456 RFID devices: Hardware version: As of 6GT2002-0ED00. Software version: as of V3.0.0

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myPIB_016KB : _PIB_016KB; // instance of FB (_PIB_016KB)
myPIB_016KB(
EXECUTE := myEXECUTE, // [IN]  rising edge: start of command            ( BOOL )
ID := myID, // [IN]  base address of peripheral device        (DWORD )
INDEX := myINDEX, // [IN]  identification of channel data           (  INT )
OFFSET := myOFFSET, // [IN]  offset of channel to base address        (  INT )
RPTCMD := myRPTCMD, // [IN]  selects command repetition               ( BOOL )
SRESET := mySRESET, // [IN]  soft-reset                               ( BOOL )
INIT := myINIT, // [IN]  initializes ident unit                    ( BOOL )
UOUT0 := myUOUT0, // [IN]  user output 0 of the ident unit          ( BOOL )
UOUT1 := myUOUT1, // [IN]  user output 1 of the ident unit          ( BOOL )
UOUT2 := myUOUT2, // [IN]  user output 2 of the ident unit          ( BOOL )
UOUT3 := myUOUT3, // [IN]  user output 3 of the ident unit          ( BOOL )
RDGATE := myRDGATE, // [IN]  activates reading gate                  ( BOOL )
CMDDIM := myCMDDIM, // [IN]  number of commands in buffer CMDREF      (  INT )
CMDSEL := myCMDSEL, // [IN]  commandselector                          (  INT )
TXBUFLEN := myTXBUFLEN, // [IN]  used length of transmitbuffer            ( DINT )
TXSTART := myTXSTART, // [IN]  index of 1st data item in transmitbuffer ( DINT )
RXBUFLEN := myRXBUFLEN, // [IN]  used length of receivebuffer             ( DINT )
RXSTART := myRXSTART, // [IN]  index of 1st data item in receivebuffer  ( DINT )
CMDREF := myCMDREF, // [IN_OUT] commandbuffer  (ARRAY [1..10] OF _PIB_COMMAND)
TXREF := myTXREF, // [IN_OUT] transmitbuffer       (ARRAY [1..1024] OF BYTE)
RXREF := myRXREF // [IN_OUT] receivebuffer        (ARRAY [1..1024] OF BYTE)
);

// ***************
// _PIB_COMMAND
// ***************
// myCMDREF : ARRAY [1..10] OF _PIB_COMMAND;  //   commandbuffer
// index 10 is for example only
myCMDREF[10].CMD := 16#00; // command-ID                                                    (BYTE)
myCMDREF[10].Config := 16#00; // configuration byte                                          (BYTE)
myCMDREF[10].OffsetBuffer := 0; // relative offset within RXBUF                              (INT)
myCMDREF[10].UID[1] := 16#00; // identifies a single transponder          (ARRAY [1..8] OF BYTE)
myCMDREF[10].FileName[1] := 16#00; // specifies the file to be accessed         (ARRAY [1..8] OF BYTE)
myCMDREF[10].Offset := 0; // specifies the relative offset within the specified file   (DINT)
myCMDREF[10].Length := 0; // specifies the number of bytes to be transmitted            (INT)
myCMDREF[10].StartAddress := 0; // specifies a physical address within the tag memory       (DINT)
myCMDREF[10].Attributes := 16#00; // encoded file attributes                               (BYTE)
myCMDREF[10].NextMode := 16#00; // specifies how tags are handled                         (BYTE)
myCMDREF[10].Timeout := 0; // maximum time for barcode to be read                         (INT)
myCMDREF[10].ObjectNumber := 0; // used to concatenate an object with its barcode information (INT)
myCMDREF[10].FileType := 16#0000; // groups files according to user specific criteria          (WORD)
```

2.4.8.3 _PIB_032KB

The function blocks act as the communication interface between a standard profile RFID system (e.g. ASM 456) and the user program.

The _PIB_032KB function block supports the following functions:

- configuration - command execution - reading and writing data - diagnostics

Hardware/software requirements:

All PNO standard profile ASM456 RFID devices: Hardware version: As of 6GT2002-0ED00. Software version: as of V3.0.0

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myPIB_032KB : _PIB_032KB; // instance of FB (_PIB_032KB)
myPIB_032KB(
EXECUTE := myEXECUTE, // [IN]  rising edge: start of command            ( BOOL )
ID := myID, // [IN]  base address of peripheral device        (DWORD )
INDEX := myINDEX, // [IN]  identification of channel data           (  INT )
OFFSET := myOFFSET, // [IN]  offset of channel to base address        (  INT )
RPTCMD := myRPTCMD, // [IN]  selects command repetition               ( BOOL )
SRESET := mySRESET, // [IN]  soft-reset                               ( BOOL )
INIT := myINIT, // [IN]  initializes ident unit                    ( BOOL )
UOUT0 := myUOUT0, // [IN]  user output 0 of the ident unit          ( BOOL )
UOUT1 := myUOUT1, // [IN]  user output 1 of the ident unit          ( BOOL )
UOUT2 := myUOUT2, // [IN]  user output 2 of the ident unit          ( BOOL )
UOUT3 := myUOUT3, // [IN]  user output 3 of the ident unit          ( BOOL )
RDGATE := myRDGATE, // [IN]  activates reading gate                   ( BOOL )
CMDDIM := myCMDDIM, // [IN]  number of commands in buffer CMDREF      (  INT )
CMDSEL := myCMDSEL, // [IN]  commandselector                          (  INT )
TXBUFLEN := myTXBUFLEN, // [IN]  used length of transmitbuffer            ( DINT )
TXSTART := myTXSTART, // [IN]  index of 1st data item in transmitbuffer ( DINT )
RXBUFLEN := myRXBUFLEN, // [IN]  used length of receivebuffer             ( DINT )
RXSTART := myRXSTART, // [IN]  index of 1st data item in receivebuffer  ( DINT )
CMDREF := myCMDREF, // [IN_OUT] commandbuffer  (ARRAY [1..10] OF _PIB_COMMAND)
TXREF := myTXREF, // [IN_OUT] transmitbuffer       (ARRAY [1..1024] OF BYTE)
RXREF := myRXREF // [IN_OUT] receivebuffer        (ARRAY [1..1024] OF BYTE)
);

// ***************
// _PIB_COMMAND
// ***************
// myCMDREF : ARRAY [1..10] OF _PIB_COMMAND;  //   commandbuffer
// index 10 is for example only
myCMDREF[10].CMD := 16#00; // command-ID                                                    (BYTE)
myCMDREF[10].Config := 16#00; // configuration byte                                          (BYTE)
myCMDREF[10].OffsetBuffer := 0; // relative offset within RXBUF                              (INT)
myCMDREF[10].UID[1] := 16#00; // identifies a single transponder           (ARRAY [1..8] OF BYTE)
myCMDREF[10].FileName[1] := 16#00; // specifies the file to be accessed         (ARRAY [1..8] OF BYTE)
myCMDREF[10].Offset := 0; // specifies the relative offset within the specified file   (DINT)
myCMDREF[10].Length := 0; // specifies the number of bytes to be transmitted            (INT)
myCMDREF[10].StartAddress := 0; // specifies a physical address within the tag memory       (DINT)
myCMDREF[10].Attributes := 16#00; // encoded file attributes                               (BYTE)
myCMDREF[10].NextMode := 16#00; // specifies how tags are handled                         (BYTE)
myCMDREF[10].Timeout := 0; // maximum time for barcode to be read                         (INT)
myCMDREF[10].ObjectNumber := 0; // used to concatenate an object with its barcode information (INT)
myCMDREF[10].FileType := 16#0000; // groups files according to user specific criteria          (WORD)
```

2.4.9 Weighing technology - SIWAREX FTA

2.4.9.1 _FTA_control

The _FTA_control function block is used to exchange data with the SIWAREX FTA weighing module. The _FTA_control function block executes commands on the SIWAREX FTA, acknowledges technology messages upon request, and visualizes the weighing status on the output parameters.

Hardware/software requirements:

SIWAREX FTA hardware version: 1.0.0 and higher. Software version: 2.1.8 and higher

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "Documentation, Utilities & Applications" SCOUT DVD. The "Documentation, Utilities & Applications" SCOUT DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myFTA_control : _FTA_control; // instance of FB (_FTA_control)
myFTA_control(
periIn := myPeriIn, // [IN]     Positive edge: the command entered in cmdNumber is executed.
(ARRAY[0..15] of BYTE)
moduleAddress := myModuleAddress, // [IN]     Provides the command number, The command start is
initiated with the execute parameter. (DINT)
execute := myExecute, // [IN]     TRUE = error message is acknowledged (BOOL)
cmdNumber := myCmdNumber, // [IN]     Transfers I/O inputs to _FTA_control FB (UINT)
ackErrorFTA := myAckErrorFTA, // [IN]     Module address from HW Config (BOOL)
simValue := mySimValue, // [IN]     Value for weight simulation when simulation is enabled; selection
is made in DR7 (REAL)
setAO := mySetAO, // [IN]     Value specification for analog output of weighing module with control
priority of higher-level controller; selection is made in DR7 (REAL)
forceDO := myForceDO, // [IN]     Specifies digital outputs of weighing module with forced control;
selection is made in DR7 (BYTE)
transitions := myTransitions, // [IN]     Disables and enables weighing steps (BYTE)
scaleData := myFTA_scaleData, // [IN_OUT] Data structure for the parameters of the SIWAREX FTA
weighing module (Struct_FTA_scaleData)
periOut := myPeriOut // [IN_OUT] Prepared FB data for I/O outputs of SIWAREX FTA weighing module
(ARRAY[0..15] of BYTE)
);

// *****************
// Struct_FTA_DR3
// *****************
// myFTA_DR3 : Struct_FTA_DR3;
myFTA_DR3.calibDigits0 := 1398101; // [IN]      Calibration digits for the zero point
myFTA_DR3.calibDigits1 := 15379113; // [IN]      Calibration digits for calibration weight 1
myFTA_DR3.calibDigits2 := 0; // [IN]       Calibration digits for calibration weight 2
myFTA_DR3.calibDigits3 := 0; // [IN]       Calibration digits for calibration weight 3
myFTA_DR3.calibDigits4 := 0; // [IN]       Calibration digits for calibration weight 4
myFTA_DR3.calibWeight1 := 100.0; // [IN]       Calibration weight 1
myFTA_DR3.calibWeight2 := 0.0; // [IN]       Calibration weight 2
myFTA_DR3.calibWeight3 := 0.0; // [IN]       Calibration weight 3
myFTA_DR3.calibWeight4 := 0.0; // [IN]       Calibration weight 4
myFTA_DR3.sigRange := 16#02; // [IN]       Characteristic value range,1 = characteristic value to 1
mV/V, 2 = characteristic value to 2 mV/V, 4 = characteristic value to 4 mV/V
myFTA_DR3.filtSequence := 16#00; // [IN]       Signal filter sequence, The 8 bits in this byte have
the following meaning: Bit 0 = 0: averaging filter before digital filter Bit 0 = 1: digital filter
before averaging filter Bits 1-7: not set
myFTA_DR3.filtType := 16#00; // [IN]       Low-pass filter type 0 = critically damped 1 = Bessel
filter 2 = Butterworth filter Other definitions not permitted.
myFTA_DR3.filtCutOffFreq := 16#04; // [IN]       Filter cut-off frequency (Hz) Specifies the response
time of the scale to a measured-value change 0 = no filter 1 = 20 Hz 2 = 10 Hz 3 = 5 Hz 4 = 2 Hz 5 = 1
Hz 6 = 0.5 Hz 7 = 0.2 Hz 8 = 0.1 Hz 9 = 0.05 Hz Other definitions not permitted.
myFTA_DR3.filtDepth := 10; // [IN]       Depth of averaging filterm Number of values for averaging [0
to 250] x 2.5 ms 0 = averaging filter deactivated
myFTA_DR3.scaleId := 'SIWAREXxx'; // [IN] [IN] Scale name
myFTA_DR3.numRanges := 16#01; // [IN]      Number of scale ranges
myFTA_DR3.scaleType := FALSE; // [IN]      Scale type FALSE = multi-range scale TRUE = multi-
resolution scale
myFTA_DR3.setZeroPwrOn := FALSE; // [IN]      TRUE = zero setting activated upon weighing module
startup (power ON)
myFTA_DR3.setZeroTare := FALSE; // [IN]      TRUE = zero setting activated upon weighing module
startup (power ON) and tare <> 0
myFTA_DR3.setZeroAutomatic := FALSE; // [IN]      TRUE = automatic zero correction switched on
```

```
myFTA_DR3.minWeightRange1 := 1.0; // [IN]      Minimum weight for weighing range 1
myFTA_DR3.maxWeightRange1 := 100.0; // [IN]      Maximum weight for weighing range 1
myFTA_DR3.incRange1 := 0.02; // [IN]      Numerical increment for weighing range 1 Numerical increment
for weighing range 1 (1*10 k, 2*10 k, 5*10 k, -4 = k = 1)
myFTA_DR3.minWeightRange2 := 0.0; // [IN]      Minimum weight for weighing range 2
myFTA_DR3.maxWeightRange2 := 0.0; // [IN]      Maximum weight for weighing range 2
myFTA_DR3.incRange2 := 0.0; // [IN]      Numerical increment for weighing range 2 Numerical increment
for weighing range 2 (1*10 k, 2*10 k, 5*10 k, -4 = k = 1)
myFTA_DR3.minWeightRange3 := 0.0; // [IN]      Minimum weight for weighing range 3
myFTA_DR3.maxWeightRange3 := 0.0; // [IN]      Maximum weight for weighing range 3
myFTA_DR3.incRange3 := 0.0; // [IN]      Numerical increment for weighing range 3 Numerical increment
for weighing range 3 (1*10 k, 2*10 k, 5*10 k, -4 = k = 1)
myFTA_DR3.standStillTime1 := T#1s; // [IN]      Standstill time If the fluctuation range of the weight
during standstill time is smaller than the fluctuation range defined in the
standStillWeight1parameter, the scale standstill signal is set to TRUE.
myFTA_DR3.standStillWeight1 := 0.02; // [IN]      Fluctuation range of the weight for detection/
generation of the scale standstill signal
myFTA_DR3.timeOutStandStill1 := T#2s; // [IN]      Maximum wait time for scale standstill If a scale
standstill is not detected during this time, a technology alarm is issued.
myFTA_DR3.maxPosWeightPwrOn := 16#0A; // [IN]      Maximum positive weight in [% of WBmax] for zero
setting after Power On
myFTA_DR3.minNegWeightPwrOn := 16#0A; // [IN]      Maximum negative weight in [% of WBmax] for zero
setting after Power On
myFTA_DR3.maxPosWeightZero := 16#01; // [IN]      Maximum positive weight in [% of WBmax] for zero
setting
myFTA_DR3.minNegWeightZero := 16#03; // [IN]      Maximum negative weight in [% of WBmax] for zero
setting
myFTA_DR3.maxTare := 16#64; // [IN]      Maximum tare capacity in [% of WBmax] with respect to the
maximum weighing range
myFTA_DR3.setLCType := 16#00; // [IN]      Selection of load cell type
myFTA_DR3.timeOutDigLC := 80; // [IN]      Monitoring time in [ms] for digital load cell If a weight
message frame is not received withing the monitoring timeout, the SIWAREX FTA weighing module
generates an operating error.
myFTA_DR3.setRestriction := '----'; // [IN]      The entry 'OIML' activates restrictions/regulations
for scales that require official calibration
myFTA_DR3.weightUnit := 'kg--'; // [IN]      Unit of mass (for example, t, kg, g)
myFTA_DR3.standStillWeight2 := 0.02; // [IN]      Standstill time If the fluctuation range of the
weight during standstill time is smaller than the fluctuation range defined in the
standStillWeight2parameter, the scale standstill signal is set to TRUE.
myFTA_DR3.standStillTime2 := T#1s; // [IN]      Fluctuation range of the weight for detection/
generation of the scale standstill signal
myFTA_DR3.timeOutStandStill2 := T#2s; // [IN]      Maximum wait time for scale standstill If a scale
standstill is not detected during this time, a technology alarm is issued.
myFTA_DR3.standStillWeight3 := 0.02; // [IN]      Fluctuation range of the weight for detection/
generation of the scale standstill signal
myFTA_DR3.standStillTime3 := T#1s; // [IN]      Standstill time If the fluctuation range of the weight
during standstill time is smaller than the fluctuation range defined in the
standStillWeight3parameter, the scale standstill signal is set to TRUE.
myFTA_DR3.timeOutStandStill3 := T#2s; // [IN]      Maximum wait time for scale standstill If a scale
standstill is not detected during this time, a technology alarm is issued.
myFTA_DR3.minDosingValTot := 100; // [IN]      Smallest set weight for weight totalizing
myFTA_DR3.incTotalizing := 0.1; // [IN]      Scale interval for weight totalizing
myFTA_DR3.xxxReserve0 := 0.0; // [IN]      Reserve (use not permitted)
myFTA_DR3.xxxReserve1 := 16#00; // [IN]      Reserve (use not permitted)
myFTA_DR3.xxxReserve2 := 16#00; // [IN]      Reserve (use not permitted)

// *****************
```

```
// Struct_FTA_DR4
// *****************
// myFTA_DR4 : Struct_FTA_DR4;
myFTA_DR4.scaleMode := 16#03; // [IN] Scale mode, selection of weighing program 0 = NAWI: filling
procedure 1 = NAWI: emptying procedure 2 = AWI: single/continuous operation filling 3 = AWI: catchw.
filling 4 = AWI: catchw. emptying 5 = AWI: check 6 = AWI: totalizing Other definitions are not
permitted.
myFTA_DR4.xxxReserve0 := 16#00; // [IN] Reserve (use not permitted)
myFTA_DR4.xxxReserve1 := 16#0000; // [IN] Reserve (use not permitted)
myFTA_DR4.timeOutLogOutput := T#2s; // [IN] Monitoring time in [ms] for output of log data to a
connected printer.
myFTA_DR4.selectLogOutput := 16#01; // [IN] Selection of device for log output Bit 0 = 0: log output
to RS232 interface Bit 0 = 1: log output to micro memory card (MMC)
myFTA_DR4.xxxReserve2 := 16#00; // [IN] Reserve (use not permitted)
myFTA_DR4.baseWeightLimit1 := FALSE; // [IN] Base weight for limit value 1 TRUE = basis for limit
value 1 is gross weight of scale FALSE = basis for limit value 1 is net weight of scale
myFTA_DR4.baseWeightLimit2 := FALSE; // [IN] Base weight for limit value 2 TRUE = basis for limit
value 2 is gross weight of scale FALSE = basis for limit value 2 is net weight of scale
myFTA_DR4.xxxReserve3 := FALSE; // [IN] Reserve (use not permitted)
myFTA_DR4.baseEmptyRange := FALSE; // [IN] Base weight for empty range monitoring FALSE = basis for
empty range is gross weight of scale TRUE = basis for empty range is net weight of scale
myFTA_DR4.xxxReserve4 := 16#00; // [IN] Reserve (use not permitted)
myFTA_DR4.emptyRange := 1.0; // [IN] Empty range
myFTA_DR4.limit1On := 1.0; // [IN] Start-up weight limit value 1
myFTA_DR4.limit1Off := 1.1; // [IN] Shut-down weight limit value 1
myFTA_DR4.limit2On := 50.0; // [IN] Start-up weight limit value 2
myFTA_DR4.limit2Off := 49.0; // [IN] Shut-down weight limit value 2
myFTA_DR4.limit3On := 99.0; // [IN] Start-up weight limit value 3
myFTA_DR4.limit3Off := 98.0; // [IN] Shut-down weight limit value 3
myFTA_DR4.minFlowLimit1 := 0.0; // [IN] Limit value 1 for minimum flow rate 0.0 = no limit value in
effect >0.0 = minimum flow rate in [1/s]
myFTA_DR4.minFlowLimit2 := 0.0; // [IN] Limit value 2 for minimum flow rate 0.0 = no limit value in
effect >0.0 = minimum flow rate in [1/s]
myFTA_DR4.filtDepthMinFlow := 16#05; // [IN] Filter depth of mean value filter for flow rate
calculation [0 to 255] * 10 ms
myFTA_DR4.xxxReserve5 := 16#00; // [IN] Reserve (use not permitted)

// *****************
// Struct_FTA_DR7
// *****************
// myFTA_DR7 : Struct_FTA_DR7;
myFTA_DR7.xxxReserve0 := 16#00; // [IN] Reserve (use not permitted)
myFTA_DR7.srcWeightSim := 16#00; // [IN] Weight simulation source 0 = weight simulation is inactive 1
= weight simulation value from SIMOTION (defined externally via DR 16) 2 = weight simulation value
from RS232
myFTA_DR7.deciDigitsProcVal := 16#03; // [IN] Number of decimal places/decade used for rounding the
decimal places of process values (REAL weight values) 0 = rounding to 0 decimal places 1 = rounding to
1 decimal place 2 = rounding to 2 decimal places 3 = rounding to 3 decimal places 4 = rounding to 4
decimal places 5 = rounding to 5 decimal places 6 = rounding to 6 decimal plac
myFTA_DR7.xxxReserve1 := 16#00; // [IN] Reserve (use not permitted)
myFTA_DR7.enableForceDO := FALSE; // [IN] TRUE = force control possible in service mode
myFTA_DR7.indexProcessValue1 := 16#02; // [IN] Selection of process value 1 from list at output
parameter val1Process of function block _FTA_control for fast output to the SIMOTION device (see
appendix "Selection List for Process Values")
myFTA_DR7.indexProcessValue2 := 16#1E; // [IN] Selection of process value 2 from list at output
parameter val2Process of function block _FTA_control for fast output to the SIMOTION device (see
appendix "Selection List for Process Values")
```

```
myFTA_DR7.xxxReserve2 := 16#00; // [IN] Reserve (use not permitted)
myFTA_DR7.defProcessAlarm0 := 16#00; // [IN] Definition of process alarm 0 Value 0 = process alarm is
not triggered Value range 1 - 0x0FF: number of technology error Value range 0x100 - 0x13F: 100h+ 0x00
to 0x1F bit no. incoming NAWI status bit 100h+ 0x20 to 0x3F bit no. incoming AWI status bit Value
range 0x200 - 0x23F: 0x200+ 0x00 to 0x1Fh bit no. outgoing NAWI status bit 0x2
myFTA_DR7.defProcessAlarm1 := 16#00; // [IN] Definition of process alarm 1 (see defProcessAlarm0)
myFTA_DR7.defProcessAlarm2 := 16#00; // [IN] Definition of process alarm 2 (see defProcessAlarm0)
myFTA_DR7.defProcessAlarm3 := 16#00; // [IN] Definition of process alarm 3 (see defProcessAlarm0)
myFTA_DR7.defProcessAlarm4 := 16#00; // [IN] Definition of process alarm 4 (see defProcessAlarm0)
myFTA_DR7.defProcessAlarm5 := 16#00; // [IN] Definition of process alarm 5 (see defProcessAlarm0)
myFTA_DR7.defProcessAlarm6 := 16#00; // [IN] Definition of process alarm 6 (see defProcessAlarm0)
myFTA_DR7.defProcessAlarm7 := 16#00; // [IN] Definition of process alarm 7 (see defProcessAlarm0)
myFTA_DR7.timeOutLifeBit := T#0ms; // [IN] Sign-of-life timeout in higher-level control 0 = sign-of-
life timeout is switched off 1 to n = timeout duration [ms]
myFTA_DR7.weightAOZero := 0.0; // [IN] Weight for zero point (0 or 4 mA)
myFTA_DR7.weightAOEnd := 0.0; // [IN] Weight for end value (20 mA)
myFTA_DR7.weightAOonOD := 0.0; // [IN] Substitute value for analog output for CPU-STOP (OS, output
disable)
myFTA_DR7.srcAO := 16#00; // [IN] Source for the analog output 0 = SIMOTION control signals 1 =
external default value via DR17 2 = gross 3 = net 4 = coarse/fine default values
myFTA_DR7.rangeAO := FALSE; // [IN] Current range for the analog output TRUE = 0 to 20 mA FALSE = 4 to
20 mA
myFTA_DR7.printerBdRate := 16#03; // [IN] RS232 - printer baud rate 0 = 1200 bits/s 1 = 2400 bits/s 2
= 4800 bits/s 3 = 9600 bits/s
myFTA_DR7.setXonXoff := TRUE; // [IN] Transfer control for the RS232 interfaces TRUE = XON/XOFF
transfer control ON FALSE = XON/XOFF transfer control OFF
myFTA_DR7.setRtsCts := FALSE; // [IN] Transfer control for the RS232 interfaces TRUE = CTS/RTS
transfer control ON FALSE = CTS/RTS transfer control OFF
myFTA_DR7.setRS485Prot := 16#00; // [IN] R485 log selection 0 = no device 1 = SIEBERT display S11 2 =
reserve 3 = SIEBERT display S102
myFTA_DR7.digitsRemDisplay := 16#00; // [IN] Decimal place for remote display (0 to 4)
myFTA_DR7.rs485BdRate := 16#03; // [IN] RS485 baud rate 0 = 1200 bits/s 1 = 2400 bits/s 2 = 4800
bits/s 3 = 9600 bits/s 4 = 19200 bits/s 5 = 38400 bits/s
myFTA_DR7.rs485Parity := FALSE; // [IN] RS485 bit parity TRUE = odd FALSE = even
myFTA_DR7.Rs485NumDataBits := TRUE; // [IN] RS485- data bits TRUE = 8 data bits FALSE = 7 data bits
myFTA_DR7.rs485NumStopBits := FALSE; // [IN] RS485- stop bits TRUE = 1 stop bit FALSE = 2 stop bits
myFTA_DR7.defDO1 := 16#FF; // [IN] Definition of digital output 1, Value range 0x00 - 0x3F, 0x00 to
0x1F bit no. NAWI status bit, 0x20 to 0x3F bit no. AWI status bit, 0xFD start of cyclic measured value
transfer, digital load cell, 0xFE stop of cyclic measured value transfer, digital load cell, 0xFF
output always inactive, Value range 0x40 to 0xFC is not permitted.
myFTA_DR7.defDO2 := 16#FF; // [IN] Definition of digital output 2 (see defDO1)
myFTA_DR7.defDO3 := 16#FF; // [IN] Definition of digital output 3 (see defDO1)
myFTA_DR7.defDO4 := 16#FF; // [IN] Definition of digital output 4 (see defDO1)
myFTA_DR7.defDO5 := 16#FF; // [IN] Definition of digital output 5 (see defDO1)
myFTA_DR7.defDO6 := 16#FF; // [IN] Definition of digital output 6 (see defDO1)
myFTA_DR7.defDO7 := 16#FF; // [IN] Definition of digital output 7 (see defDO1)
myFTA_DR7.defDO8 := 16#FF; // [IN] Definition of digital output 8 (see defDO1)
myFTA_DR7.lowActiveDO1 := FALSE; // [IN] Level definition of digital output 1 TRUE = output low active
FALSE = output high active
myFTA_DR7.lowActiveDO2 := FALSE; // [IN] Level definition of digital output 2 TRUE = output low active
FALSE = output high active
myFTA_DR7.lowActiveDO3 := FALSE; // [IN] Level definition of digital output 3 TRUE = output low active
FALSE = output high active
myFTA_DR7.lowActiveDO4 := FALSE; // [IN] Level definition of digital output 4 TRUE = output low active
FALSE = output high active
myFTA_DR7.lowActiveDO5 := FALSE; // [IN] Level definition of digital output 5 TRUE = output low active
FALSE = output high active
```

```
myFTA_DR7.lowActiveDO6 := FALSE; // [IN] Level definition of digital output 6 TRUE = output low active
FALSE = output high active
myFTA_DR7.lowActiveDO7 := FALSE; // [IN] Level definition of digital output 7 TRUE = output low active
FALSE = output high active
myFTA_DR7.lowActiveDO8 := FALSE; // [IN] Level definition of digital output 8 TRUE = output low active
FALSE = output high active
myFTA_DR7.DO1onOD := FALSE; // [IN] Substitute value for digital output 1 for CPU-STOP (OD) or fault
myFTA_DR7.DO2onOD := FALSE; // [IN] Substitute value for digital output 2 for CPU-STOP (OD) or fault
myFTA_DR7.DO3onOD := FALSE; // [IN] Substitute value for digital output 3 for CPU-STOP (OD) or fault
myFTA_DR7.DO4onOD := FALSE; // [IN] Substitute value for digital output 4 for CPU-STOP (OD) or fault
myFTA_DR7.DO5onOD := FALSE; // [IN] Substitute value for digital output 5 for CPU-STOP (OD) or fault
myFTA_DR7.DO6onOD := FALSE; // [IN] Substitute value for digital output 6 for CPU-STOP (OD) or fault
myFTA_DR7.DO7onOD := FALSE; // [IN] Substitute value for digital output 7 for CPU-STOP (OD) or fault
myFTA_DR7.DO8onOD := FALSE; // [IN] Substitute value for digital output 8 for CPU-STOP (OD) or fault
myFTA_DR7.enableDOonError := FALSE; // [IN] Activate substitute value output for operational fault
TRUE = substitute value for operational fault enabled FALSE = substitute value for operational fault
disabled
myFTA_DR7.xxxReserve3 := 16#00; // [IN] Reserve (use not permitted)
myFTA_DR7.defDI1 := 16#00; // [IN] Definition of digital input 1, 0 = no command, 1 to 254 = command
code, 255 = step enabling (see DR23, step control weighing parameters)
myFTA_DR7.defDI2 := 16#00; // [IN] Definition of digital input 2 (see defDI1)
myFTA_DR7.defDI3 := 16#00; // [IN] Definition of digital input 3 (see defDI1)
myFTA_DR7.defDI4 := 16#00; // [IN] Definition of digital input 4 (see defDI1)
myFTA_DR7.defDI5 := 16#00; // [IN] Definition of digital input 5 (see defDI1)
myFTA_DR7.defDI6 := 16#00; // [IN] Definition of digital input 6 (see defDI1)
myFTA_DR7.defDI7 := 16#00; // [IN] Definition of digital input 7 (see defDI1)
myFTA_DR7.lowActiveDI1 := FALSE; // [IN] Level definition of digital input 1 TRUE = input low active
FALSE = input high active
myFTA_DR7.lowActiveDI2 := FALSE; // [IN] Level definition of digital input 2 (see lowActiveDI1)
myFTA_DR7.lowActiveDI3 := FALSE; // [IN] Level definition of digital input 3 (see lowActiveDI1)
myFTA_DR7.lowActiveDI4 := FALSE; // [IN] Level definition of digital input 4 (see lowActiveDI1)
myFTA_DR7.lowActiveDI5 := FALSE; // [IN] Level definition of digital input 5 (see lowActiveDI1)
myFTA_DR7.lowActiveDI6 := FALSE; // [IN] Level definition of digital input 6 (see lowActiveDI1)
myFTA_DR7.lowActiveDI7 := FALSE; // [IN] Level definition of digital input 7 (see lowActiveDI1)
myFTA_DR7.measTimeForCntr := T#999ms; // [IN] Measuring time for pulse input/counter input
myFTA_DR7.xxxReserve4 := 16#00; // [IN] Reserve (use not permitted)
myFTA_DR7.modeLogOverflow := TRUE; // [IN] Behavior during log overflow (MMC card full) TRUE =
overwrite oldest entries when MMC memory is full FALSE = stop logging with MMC memory is full
myFTA_DR7.modeTraceOverflow := TRUE; // [IN] Behavior during trace overflow (MMC card full) TRUE =
overwrite oldest trace data when MMC card is full FALSE = trace data cannot be overwritten
myFTA_DR7.traceDataToMMC := FALSE; // [IN] Trace data memory location TRUE = store trace data on MMC
FALSE = store trace data in RAM
myFTA_DR7.traceSizeMMC := 16#32; // [IN] Memory segment on MMC for trace function in [%] Up to 100%;
however, the sum of the trace function and logs cannot exceed 100%
myFTA_DR7.logSizeMMC := 16#32; // [IN] Memory segment on MMC for logs in [%] Up to 100%; however, the
sum of the trace function and logs cannot exceed 100%
myFTA_DR7.traceCycle := 16#01; // [IN] Recording cycle for trace function in [ms] 1 to n x 10 ms

// *****************
// Struct_FTA_DR8
// *****************
// myFTA_DR8 : Struct_FTA_DR8;
myFTA_DR8.dateAndTime := DT#1992-01-01-0:0:0.0; // [IN] Date and time for the SIWAREX FTA weighing
module (write)

// *****************
// Struct_FTA_DR9
```

```
// *****************
// myFTA_DR9 : Struct_FTA_DR9;
myFTA_DR9.crcCheckSumFw := 16#00; // [IN]               Firmware checksum
myFTA_DR9.lenFw := 16#00; // [IN]               Firmware length in bytes
myFTA_DR9.moduleInfo := ''; // [IN]               Siemens AG module serial number
myFTA_DR9.moduleName := ''; // [IN]               Module name
myFTA_DR9.application[1] := ''; // [IN]          Application identification
myFTA_DR9.application[2] := ''; // [IN]          Application identification
myFTA_DR9.application[3] := ''; // [IN]          Application identification
myFTA_DR9.application[4] := ''; // [IN]          Application identification
myFTA_DR9.application[5] := ''; // [IN]          Application identification
myFTA_DR9.application[6] := ''; // [IN]          Application identification
myFTA_DR9.application[7] := ''; // [IN]          Application identification
myFTA_DR9.application[8] := ''; // [IN]          Application identification
myFTA_DR9.fileName := 'NAWI'; // [IN]               File name
myFTA_DR9.typeVersion := 16#00; // [IN]               Version type B = Laboratory status P = Pilot R
= Release S = Special status V = Version K = Revision status
myFTA_DR9.fctVersion := 0; // [IN]               Function status Major function changes or
calibration-related changes (0 to 99)
myFTA_DR9.dataStructVersion := 0; // [IN]               Data record structure version, Identifies
changes in the data record structure (0 to 99)
myFTA_DR9.corrVersion := 0; // [IN]               Revision status, Small changes or error corrections
(0 to 99)
myFTA_DR9.dateCreation := ''; // [IN]               Creation date
myFTA_DR9.timeCreation := ''; // [IN]               Creation time
myFTA_DR9.bootVersion := 16#00; // [IN]               Bootloader version
myFTA_DR9.scaleType := ''; // [IN]               AWI - Automatic Weighing Instrument, NAWI - Non
Automatic Weighing Instrument
myFTA_DR9.xxxReserve1 := 16#00; // [IN]               Reserve (use not permitted)

// *****************
// Struct_FTA_DR15
// *****************
// myFTA_DR15 : Struct_FTA_DR15;
myFTA_DR15.tareSetValue := 0.0; // [IN] External tare weight definition

// *****************
// Struct_FTA_DR16
// *****************
// myFTA_DR16 : Struct_FTA_DR16;
myFTA_DR16.weightSimValue := 0.0; // [IN] Default value for weight simulation, This value is used in
place of the gross weight, for example, during testing.

// *****************
// Struct_FTA_DR17
// *****************
// myFTA_DR17 : Struct_FTA_DR17;
myFTA_DR17.setAO := 0.0; // [IN] External definition for the analog output

// *****************
// Struct_FTA_DR18
// *****************
// myFTA_DR18 : Struct_FTA_DR18;
myFTA_DR18.setValRemDisplay := 0.0; // [IN]  External definition for the remote display

// *****************
// Struct_FTA_DR20
```

```
// ******************
// myFTA_DR20 : Struct_FTA_DR20;
myFTA_DR20.dosingSetpoint := 50.0; // [IN] Setpoint weight for the weighing operation

// ******************
// Struct_FTA_DR21
// ******************
// myFTA_DR21 : Struct_FTA_DR21;
myFTA_DR21.loadingSetpoint := 1000.0; // [IN] Load quantity/total setpoint for loading mode

// ******************
// Struct_FTA_DR22
// ******************
// myFTA_DR22 : Struct_FTA_DR22;
myFTA_DR22.maxDosingTime := T#0ms; // [IN] Maximum dosing time, 0 = deactivated
myFTA_DR22.inFlightWeight := 1.0; // [IN] In-flight weight, Amount that is still trailing (in-flight)
after the fine signal has been switched off
myFTA_DR22.fineWeight := 20; // [IN] Fine weight,Amount to be dosed during the fine signal
myFTA_DR22.switchOffCorr := 0.0; // [IN] Switch-off compensation value, Additional positive/negative
offset of the fine-signal switch-off point
myFTA_DR22.timePreDosing := T#0ms; // [IN] Pre-dosing time,0 = deactivated,> 0 = pre-dosing time
myFTA_DR22.upperTolValTO1 := 0.2; // [IN] Upper tolerance limit 1 (value of the permissible positive
deviation from the setpoint weight)
myFTA_DR22.lowerTolValTU1 := 0.2; // [IN] Lower tolerance limit 1 (value of the permissible negative
deviation from the setpoint weight)
myFTA_DR22.upperTolValTO2 := 0.5; // [IN] Value of upper tolerance limit 2, upperTolValTO2 must be
larger than upperTolValTO1
myFTA_DR22.lowerTolValTU2 := 0.5; // [IN] Value of lower tolerance limit 2, lowerTolValTU2 must be
larger than lowerTolValTU1

// ******************
// Struct_FTA_DR23
// ******************
// myFTA_DR23 : Struct_FTA_DR23;
myFTA_DR23.selectText := 16#01; // [IN] Text selection for automatic logging, 0 = no automatic logging
after weighing, 1 = automatic logging with text 1, 2 = automatic logging with text 2, 3 = automatic
logging with text 3, 4 = automatic logging with text 4
myFTA_DR23.xxxReserve1 := 16#00; // [IN] Reserve (use not permitted)
myFTA_DR23.xxxReserve2 := 16#00; // [IN] Reserve (use not permitted)
myFTA_DR23.maxSetpointDosing := 90.0; // [IN] Maximum setpoint weight for single dosing
myFTA_DR23.disableTimeCoarse := T#500ms; // [IN] Disable time coarse, 0 = deactivated, After the
coarse signal is switched on, weight evaluation is suspended for the specified time.
myFTA_DR23.disableTimeFine := T#500ms; // [IN] Disable time fine, 0 = deactivated, After the coarse
signal is switched off, weight evaluation is suspended for the specified time.
myFTA_DR23.disableTimeCompare := T#0ms; // [IN] Disable time for setpoint-actual comparison, After the
disable time is initiated by command, the current weight monitoring is suspended for the specified
time during the weighing procedure.
myFTA_DR23.valAOCoarse := 16#3C; // [IN] Default value in [%] for analog output when coarse signal is
active
myFTA_DR23.valAOFine := 16#14; // [IN] Default value in [%] for analog output when fine signal is
active
myFTA_DR23.filtTypeDosing := 16#00; // [IN] Filter type for dosage control, 0 = critically damped, 1 =
Bessel filter, 2 = Butterworth filter, Other definitions not permitted.
myFTA_DR23.filtCutOffFreq := 16#04; // [IN] Dosing filter cut-off frequency, 0: no filter, 1: fg = 20
Hz, 2: fg = 10 Hz, 3: fg = 5 Hz, 4: fg = 2 Hz, 5: fg = 1 Hz, 6: fg = 0.5 Hz, 7: fg = 0.2 Hz, 8: fg =
0.1 Hz, 9: fg = 0.05 Hz, Other definitions not permitted.
myFTA_DR23.modeZeroTare := 16#02; // [IN] Tare/zeroing mode, 0 = do not tare or zero when staring
```

```
scale, 1 = zero, 2 = tare, 3 = tare with mean value, 4 = tare with external tare input, Other
definitions not permitted.
myFTA_DR23.cycleZeroTare := 16#00; // [IN] Tare/zeroing cycle, 0 = each weighing procedure is zeroed
or tared, 1 = one fill is not zeroed or tared, 2 to 99 = 2 to 99 fills are not zeroed or tared, Other
definitions not permitted.
myFTA_DR23.xxxReserve3 := 16#00; // [IN] Reserve (use not permitted)
myFTA_DR23.minTareValue := 0.0; // [IN] Minimum tare weight, Taring or external tare input is
performed only if gross > minimum tare weight, 0 = minimum tare weight is not monitored
myFTA_DR23.maxTareValue := 0.0; // [IN] Maximum tare value, Taring or external tare input is performed
only if gross < maximum tare weight, 0 = maximum tare weight is not monitored
myFTA_DR23.timeAutoZeroing := T#5m; // [IN] Cycle time for zeroing, = 0: no time-controlled zeroing,
<> 0: time between two zeroing operations, Note: For weighing mode AWI and country code "OIML",
zeroing/taring is performed after 15 min., at the latest.
myFTA_DR23.waitDI1InStepX := 16#00; // [IN] Step control via digital input 1, Instead of using
definable command codes for the digital inputs, step, enabling for weighing control can also be
controlled via the inputs. The requirement for this is that DR7 (interface parameter) must contain
0xFF.,
// [IN] 0 = weighing waits at step 0 if DI1 is active, 1 = weighing waits at step 1 if DI1 is active,
2 = weighing waits at step 2 if DI1 is active...7 = weighing waits at step 7 if DI1 is active, Other
definitions not permitted.
myFTA_DR23.waitDI2InStepX := 16#00; // [IN] Step control via digital input 2 (see waitDI1InStepX)
myFTA_DR23.waitDI3InStepX := 16#00; // [IN] Step control via digital input 3 (see waitDI1InStepX)
myFTA_DR23.waitDI4InStepX := 16#00; // [IN] Step control via digital input 4 (see waitDI1InStepX)
myFTA_DR23.waitDI5InStepX := 16#00; // [IN] Step control via digital input 5 (see waitDI1InStepX)
myFTA_DR23.waitDI6InStepX := 16#00; // [IN] Step control via digital input 6 (see waitDI1InStepX)
myFTA_DR23.waitDI7InStepX := 16#00; // [IN] Step control via digital input 7 (see waitDI1InStepX)
myFTA_DR23.xxxReserve4 := 16#00; // [IN] Reserve (use not permitted)
myFTA_DR23.timeOutOneStep := T#0s; // [IN] Timeout for step control = 0: no timeout, > 0: timeout time
is defined, If a transition to the next step does not occur within the defined time, the "transition
timeout" technology error is issued.
myFTA_DR23.stopAfterStep1 := FALSE; // [IN] Check stop definition, TRUE = weighing goes to check stop
after step 1, FALSE = no check stop
myFTA_DR23.stopAfterStep2 := FALSE; // [IN] Check stop definition, TRUE = weighing goes to check stop
after step 2, FALSE = no check stop
myFTA_DR23.stopAfterStep3 := FALSE; // [IN] Check stop definition, TRUE = weighing goes to check stop
after step 3, FALSE = no check stop
myFTA_DR23.stopAfterStep4 := FALSE; // [IN] Check stop definition, TRUE = weighing goes to check stop
after step 4, FALSE = no check stop
myFTA_DR23.stopAfterStep5 := FALSE; // [IN] Check stop definition, TRUE = weighing goes to check stop
after step 5, FALSE = no check stop
myFTA_DR23.stopAfterStep6 := FALSE; // [IN] Check stop definition, TRUE = weighing goes to check stop
after step 6, FALSE = no check stop
myFTA_DR23.stopAfterStep7 := FALSE; // [IN] Check stop definition, TRUE = weighing goes to check stop
after step 7, FALSE = no check stop
myFTA_DR23.xxxReserve5 := 16#00; // [IN] Reserve (use not permitted)
myFTA_DR23.autoPostDosing := FALSE; // [IN] Automatic post-dosing, FALSE = no automatic post-dosing,
TRUE = automatic post-dosing for deviation from tolerance limit
myFTA_DR23.modePostDosing := FALSE; // [IN] Post-dosing type, FALSE = post-dosing with continuous fine
signal, TRUE = post-dosing in pulse inching mode
myFTA_DR23.stopTO1Limit := FALSE; // [IN] Default: stop when upper tolerance limit 1 (TO1) violated,
FALSE = weighing is not stopped due to tolerance error, TRUE = weighing is stopped due to tolerance
error (weight above TO1)
myFTA_DR23.stopTO2Limit := FALSE; // [IN] Stop when upper tolerance limit 2 (TO2) violated, FALSE =
weighing is not stopped due to tolerance error, TRUE = weighing is stopped due to tolerance error
(weight above TO2)
myFTA_DR23.stopTU1Limit := FALSE; // [IN] Stop when lower tolerance limit 1 (TU1) violated, FALSE =
weighing is not stopped due to tolerance error, TRUE = weighing is stopped due to tolerance error
```

```
(weight below TU1)
myFTA_DR23.stopTU2Limit := FALSE; // [IN] Stop when lower tolerance limit 2 (TU2) violated, FALSE =
weighing is not stopped due to tolerance error, TRUE = weighing is stopped due to tolerance error
(weight below TU2)
myFTA_DR23.continueTolStop := FALSE; // [IN] Continue after stop due to tolerance error, FALSE = cycle
cannot be continued if tolerance error exists, TRUE = cycle can be continued despite existing
tolerance error
myFTA_DR23.numNoTolCheck := 16#00; // [IN] Check for tolerance deviations, 0 = all weighing operations
are checked for tolerance deviations, 1 = one weighing operation is not checked for tolerance
deviation, 2 to 99 = 2 to 99 weighing operations are not checked for tolerance deviations, Other
definitions not permitted.
myFTA_DR23.timePulseInching := T#1s; // [IN] Pulse duration of fine signal
myFTA_DR23.ctrlErrReaction := 16#00; // [IN] Controller response to weighing error Bit 0 = 0: reset
controller in response to technology error (weighing error), Bit 0 = 1: limit controller to maximum
control action, Bits 1 to 7 not used
myFTA_DR23.typeController := 16#00; // [IN] Selection of controller type, 0 = no control for switching
off coarse/fine signal, 1 = proportional controller without fine signal time controller, 2 =
proportional controller with fine signal time controller, 3 = fine signal time controller without
proportional controller, Other definitions not permitted.
myFTA_DR23.factorController := 16#1E; // [IN] Control factor for proportional controller [0 to 100%]
myFTA_DR23.xxxReserve6 := 16#00; // [IN] Reserve (use not permitted)
myFTA_DR23.limitController := 1.0; // [IN] Maximum one-time control action, Limitation of maximum one-
time control action of proportional controller
myFTA_DR23.optiPlusCtrl := 0.0; // [IN] Optimum plus controller
myFTA_DR23.optiMinusCtrl := 0.0; // [IN] Optimum minus controller
myFTA_DR23.setFineTime := T#3s; // [IN] Fine time setpoint
myFTA_DR23.factorFineTime := 16#14; // [IN] Control factor for fine-time controller [0 to 100 %]
myFTA_DR23.xxxReserve7 := 16#00; // [IN] Reserve (use not permitted)
myFTA_DR23.xxxReserve8 := 16#00; // [IN] Reserve (use not permitted)
myFTA_DR23.timeOverlap := T#0s; // [IN] Overlap time, The overlap time (timeOverlap) must be less than
the emptying time (timeEmptying). The next weighing operation can be advanced by the overlap time and
can begin during the emptying operation, after a predefined time has elapsed.
myFTA_DR23.timeEmptying := T#0s; // [IN] Emptying time = 0: emptying depends on the emptying range, >
0: emptying occurs after a predefined time
myFTA_DR23.timeOutEmptying := T#0s; // [IN] Maximum emptying time, = 0: timeout is switched off, > 0:
a technology error is issued if the empty range has not been reached by the emptying timeout.
myFTA_DR23.modeLoading := 16#00; // [IN] Loading with coarse, 0 = all weighing operations during
loading mode are controlled with coarse and fine signals,
myFTA_DR23.xxxReserve9 := 16#00; // [IN] 1 = only coarse feed is used for weighing; however, the last
5 weighing operations are performed with coarse and fine signals

// ******************
// Struct_FTA_DR26
// ******************
// myFTA_DR26 : Struct_FTA_DR26;
myFTA_DR26.presetTareActive := FALSE; // [IN] TRUE = tare memory is assigned and activated by an
external default value
myFTA_DR26.xxxReserve1 := 16#00; // [IN] Reserve (use not permitted)
myFTA_DR26.xxxReserve2 := 16#00; // [IN] Reserve (use not permitted)
myFTA_DR26.internState := 16#00; // [IN] Internal status, Bit 0 = 0: SIMOTION operation activated, Bit
0 = 1: stand-alone mode activated, Bit 1 = 0: recording of digital load cell stopped, Bit 1 = 1:
recording of digital load cell activated, Bits 2 to 7 not used (use not permitted)
myFTA_DR26.actTareWeight := 0.0; // [IN] Actual tare weight (process value)
myFTA_DR26.actAverTareWeight := 0.0; // [IN] Actual tare average (when taring with average value)
myFTA_DR26.pwrOnZeroValue := 0.0; // [IN] Zero value Value is set during power-up if "power-on zero
value" is activated.
myFTA_DR26.zeroValue := 0.0; // [IN] Zero value Value is set during zeroing
```

```
myFTA_DR26.zeroCorrValue := 0.0; // [IN] Zero compensation value Value is controlled by automatic zero
correction
myFTA_DR26.impedRefValue := 0; // [IN] Impedance reference value
myFTA_DR26.actImpedValue := 0; // [IN] Actual impedance value from last measurement
myFTA_DR26.lastMaxWeight := 0.0; // [IN] Last maximum weight value
myFTA_DR26.numOpMinutes := 0; // [IN] Operating minutes counter
myFTA_DR26.maxTemperature := 0; // [IN] Current maximum internal temperature measured since the last
loading of default values [0.1°C] If a temperature value is not available: -100.0°C.
myFTA_DR26.xxxReserve3 := 16#00; // [IN] Reserve (use not permitted)
myFTA_DR26.xxxReserve4 := 16#00; // [IN] Reserve (use not permitted)
myFTA_DR26.crc := 16#0000; // [IN] Checksum

// ******************
// Struct_FTA_DR30
// ******************
// myFTA_DR30 : Struct_FTA_DR30;

// ******************
// Struct_FTA_DR31
// ******************
// myFTA_DR31 : Struct_FTA_DR31;

// ******************
// Struct_FTA_DR32
// ******************
// myFTA_DR32 : Struct_FTA_DR32;

// ******************
// Struct_FTA_DR34
// ******************
// myFTA_DR34 : Struct_FTA_DR34;

// ******************
// Struct_FTA_DR35
// ******************
// myFTA_DR35 : Struct_FTA_DR35;

// ******************
// Struct_FTA_DR39
// ******************
// myFTA_DR39 : Struct_FTA_DR39;

// ******************
// Struct_FTA_DR44
// ******************
// myFTA_DR44 : Struct_FTA_DR44;

// ******************
// Struct_FTA_DR45
// ******************
// myFTA_DR45 : Struct_FTA_DR45;
myFTA_DR45.varText1 := ''; // [IN] Variable text (string 1)
myFTA_DR45.varText2 := ''; // [IN] Variable text (string 2)
myFTA_DR45.varText3 := ''; // [IN] Variable text (string 3)
myFTA_DR45.varText4 := ''; // [IN] Variable text (string 4)

// ******************
```

```
// Struct_FTA_DR46
// ******************
// myFTA_DR46 : Struct_FTA_DR46;
myFTA_DR46.selectLogId := 0; // [IN] ID number for read-out, With this setting, the log is read out
with its ID number when DR47 is read out. If request last data record (reqLastLog = 1) is activated,
the ID number is ignored.
myFTA_DR46.reqLastLog := 16#00; // [IN] Request for last data record,  If this parameter setting = 1,
the last log is output via data record DR47.
myFTA_DR46.xxxReserve1 := 16#00; // [IN] xxxReserve1 BYTE 16#00

// ******************
// Struct_FTA_DR47
// ******************
// myFTA_DR47 : Struct_FTA_DR47;

// ********************
// Struct_FTA_DR123
// ********************
// myFTA_DR123 : Struct_FTA_DR123;

// ************************
// Struct_FTA_scaleData
// ************************
// myFTA_scaleData : Struct_FTA_scaleData;
myFTA_scaleData.DR3 := myDR3; // [IN_OUT] (Struct_FTA_DR3)
myFTA_scaleData.DR4 := myDR4; // [IN_OUT] (Struct_FTA_DR4)
myFTA_scaleData.DR7 := myDR7; // [IN_OUT] (Struct_FTA_DR7)
myFTA_scaleData.DR8 := myDR8; // [IN_OUT] (Struct_FTA_DR8)
myFTA_scaleData.DR9 := myDR9; // [IN_OUT] (Struct_FTA_DR9)
myFTA_scaleData.DR15 := myDR15; // [IN_OUT] (Struct_FTA_DR15)
myFTA_scaleData.DR16 := myDR16; // [IN_OUT] (Struct_FTA_DR16)
myFTA_scaleData.DR17 := myDR17; // [IN_OUT] (Struct_FTA_DR17)
myFTA_scaleData.DR18 := myDR18; // [IN_OUT] (Struct_FTA_DR18)
myFTA_scaleData.DR20 := myDR20; // [IN_OUT] (Struct_FTA_DR20)
myFTA_scaleData.DR21 := myDR21; // [IN_OUT] (Struct_FTA_DR21)
myFTA_scaleData.DR22 := myDR22; // [IN_OUT] (Struct_FTA_DR22)
myFTA_scaleData.DR23 := myDR23; // [IN_OUT] (Struct_FTA_DR23)
myFTA_scaleData.DR26 := myDR26; // [IN_OUT] (Struct_FTA_DR26)
myFTA_scaleData.DR30 := myDR30; // [IN_OUT] (Struct_FTA_DR30)
myFTA_scaleData.DR31 := myDR31; // [IN_OUT] (Struct_FTA_DR31)
myFTA_scaleData.DR32 := myDR32; // [IN_OUT] (Struct_FTA_DR32)
myFTA_scaleData.DR34 := myDR34; // [IN_OUT] (Struct_FTA_DR34)
myFTA_scaleData.DR35 := myDR35; // [IN_OUT] (Struct_FTA_DR35)
myFTA_scaleData.DR44 := myDR44; // [IN_OUT] (Struct_FTA_DR44)
myFTA_scaleData.DR45 := myDR45; // [IN_OUT] (Struct_FTA_DR45)
myFTA_scaleData.DR46 := myDR46; // [IN_OUT] (Struct_FTA_DR46)
myFTA_scaleData.DR47 := myDR47; // [IN_OUT] (Struct_FTA_DR47)
myFTA_scaleData.DR123 := myDR123; // [IN_OUT] (Struct_FTA_DR123)
```

2.5 Additional system functions

2.5.1 Operating state

2.5.1.1 _changeOperationMode

This function causes the operating mode to change to STOP or STOPU. There is no return value.

Additional references

Additional information is available at:

- SIMOTION SCOUT Configuration Manual
  Section Controlling the operating mode with SIMOTION SCOUT
- or in the SIMOTION online help.

Parameters:

OperationMode

Direction:	Input parameter
Data type:	EnumOperationMode
Parameter index:	1

EnumOperationMode

STOPU (0)	STOPU
STOP (134)	STOP

Specifies the state change that occurs with STOPU or STOP. With STOPU, a transition to STOPU takes place. With STOP, a transition to STOP takes place.

2.5.1.2 _restart

This function restarts the SIMOTION D410-2 device. To make the SIMOTION device assume 'RUN' operating mode after the restart, you must configure _startUpData.operationMode = "RUN".

Additional references

Further information is available at:

- SIMOTION D410-2 Commissioning and Hardware Installation Manual
- or in the online help

Return value: DINT

Description of the return values:

16#00000000	Function performed without errors.
16#FFFF8091	Functionality is not supported for this device. The _restart system function only supports synchronous command execution.

Note

In the case of a return of the system function and the executed functionality (return value 16#00000000), the user program is to ensure that there are no more axis motions or other possible undesired influences on the machine caused by the user program.

2.5.2 File access

2.5.2.1 _directoryPathDelete

The function block deletes a directory path and all subdirectories, including directory contents.

Parameters:

EXECUTE

Direction:	Input parameter
Data type:	BOOL
Parameter index:	1

Starting a new request with a rising edge

STORAGETYPE

Direction:	Input parameter
Data type:	EnumStorageType
Parameter index:	2

EnumStorageType

RAMDISK (0)	Temporary data storage; deleted on power off, not non-volatile
CARD (1)	Permanent data storage; non-volatile

Specifies the storage medium

LOCATION

Direction:	Input parameter
Data type:	EnumUserLocation
Parameter index:	3

EnumUserLocation

USERFILES (0)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERFILES'
USERLOG (1)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERLOG'
USERDATASETS (2)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\UPP\UNITDS'; the contents, including subdirectories, are deleted with 'Delete user data on card' if the 'Unit data sets' checkmark is activated
USERWEB (3)	Corresponds to the following path: '\USER \SIMOTION\HMI\FILES'
USERTEMP (4)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\USERTEMP'; the contents, including subdirectories, are deleted with 'Delete user data on card'

Specifies the storage location

DIRECTORYPATHNAME

Direction:	Input parameter
Data type:	STRING
Parameter index:	4

Path to be deleted under LOCATION

DONE

Direction:	Output parameter
Data type:	BOOL
Parameter index:	5

The block was ended.

BUSY

Direction:	Output parameter
Data type:	BOOL
Parameter index:	6

The block is being processed.

ERROR

Direction:	Output parameter
Data type:	BOOL
Parameter index:	7

The block was executed with errors.

ERRORID

Direction:	Output parameter
Data type:	DWORD
Parameter index:	8

Error specification

16#00000000	Error-free - job performed successfully
16#FFFF8003	Command buffer could not be created.
16#FFFF8009	Internal error.
16#FFFF800E	Invalid file name.
16#FFFF800F	No memory space available.
16#FFFF8010	Access was denied.

Note

About error value 16#FFFF8010

Check whether one of the following cases may apply: a) The file is open. b) The file is write-protected, i.e. the file is open as read-only.

2.5.2.2 _fileClose

The function block closes the specified file.

Any changes that might have been made to the data are saved on the medium. The system resources are also released.

Additional references

Further information is available at:

- SIMOTION Runtime Basic Functions Function Manual
  Section Reading and writing files from a user program (as of V4.4)
  Section Managing files from a user program (as of V4.4)
- or in the online help

Parameters:

EXECUTE

Direction:	Input parameter
Data type:	BOOL
Parameter index:	1

Starts a new job with a rising edge.

HANDLE

Direction:	Input/output parameter
Data type:	_filehandle
Parameter index:	2

EnumYesNo

NO (91)	No
YES (173)	Yes

EnumStorageType

RAMDISK (0)	Temporary data storage; deleted on power off, not non-volatile
CARD (1)	Permanent data storage; non-volatile

EnumUserLocation

USERFILES (0)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERFILES'
USERLOG (1)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERLOG'
USERDATASETS (2)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\UPP\UNITDS'; the contents, including subdirectories, are deleted with 'Delete user data on card' if the 'Unit data sets' checkmark is activated
USERWEB (3)	Corresponds to the following path: '\USER \SIMOTION\HMI\FILES'
USERTEMP (4)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\USERTEMP'; the contents, including subdirectories, are deleted with 'Delete user data on card'

_filehandle

Structure	Name	Data type	Unit
DONE		BOOL	-
BUSY		BOOL	-
ERROR		BOOL	-
ERRORID		DWORD	-
NUMBEROFOPENFILES		UDINT	-
AVAILABLESTORAGE		UDINT	-
ISFILEOPEN		EnumYesNo	-
STORAGETYPE		EnumStorageType	-

LOCATION		EnumUserLocation	-
FILENAME		STRING	-
ACTUALFILEPOSITION		DINT	-
_INST_DATA_		ARRAY [0..0] OF DINT	-

Transfer of the _filehandle function block

DONE

Direction:	Output parameter
Data type:	BOOL
Parameter index:	14

The block was ended.

BUSY

Direction:	Output parameter
Data type:	BOOL
Parameter index:	15

The block is being processed.

ERROR

Direction:	Output parameter
Data type:	BOOL
Parameter index:	16

The block was executed with errors.

ERRORID

Direction:	Output parameter
Data type:	DWORD
Parameter index:	17

Error specification

16#00000000	Error-free - job performed successfully
16#FFFF8002	Command cannot be executed.
16#FFFF8003	Command buffer could not be created.
16#FFFF8009	Internal error.
16#FFFF800D	Invalid handle.
16#FFFF8010	Access was denied.

Note

About error value 16#FFFF8010

Check whether one of the following cases may apply: a) The file is open. b) The file is write-protected, i.e. the file is open as read-only.

2.5.2.3 _fileCopy

The function block creates a copy of a file, e.g. a file can be copied from RAMDISK to CARD. The storage medium for the source and target files is specified with the SOURCESTORAGETYPE and DESTINATIONSTORAGETYPE parameters. The file can only be copied if it is closed (_fileClose might have to be called first).

Additional references

Further information is available at:

- SIMOTION Runtime Basic Functions Function Manual
  Section Managing files from a user program (as of V4.4)
- or in the online help

Parameters:

EXECUTE

Direction:	Input parameter
Data type:	BOOL
Parameter index:	1

Starting a new request with a rising edge

SOURCESTORAGETYPE

Direction:	Input parameter
Data type:	EnumStorageType
Parameter index:	2

EnumStorageType

RAMDISK (0)	Temporary data storage; deleted on power off, not non-volatile
CARD (1)	Permanent data storage; non-volatile

Specifies the storage medium for the source file

SOURCELOCATION

Direction:	Input parameter
Data type:	EnumUserLocation
Parameter index:	3

EnumUserLocation

USERFILES (0)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERFILES'
USERLOG (1)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERLOG'
USERDATASETS (2)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\UPP\UNITDS'; the contents, including subdirectories, are deleted with 'Delete user data on card' if the 'Unit data sets' checkmark is activated
USERWEB (3)	Corresponds to the following path: '\USER \SIMOTION\HMI\FILES'
USERTEMP (4)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\USERTEMP'; the contents, including subdirectories, are deleted with 'Delete user data on card'

Specifies the storage location for the source file

SOURCEFILENAME

Direction:	Input parameter
Data type:	STRING
Parameter index:	4

Name and path of the source file under LOCATION

DESTINATIONSTORAGETYPE

Direction:	Input parameter
Data type:	EnumStorageType
Parameter index:	5

EnumStorageType

RAMDISK (0)	Temporary data storage; deleted on power off, not non-volatile
CARD (1)	Permanent data storage; non-volatile

Specifies the storage medium for the destination file

DESTINATIONLOCATION

Direction:	Input parameter
Data type:	EnumUserLocation
Parameter index:	6

EnumUserLocation

USERFILES (0)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERFILES'
USERLOG (1)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERLOG'
USERDATASETS (2)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\UPP\UNITDS'; the contents, including subdirectories, are deleted with 'Delete user data on card' if the 'Unit data sets' checkmark is activated
USERWEB (3)	Corresponds to the following path: '\USER \SIMOTION\HMI\FILES'
USERTEMP (4)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\USERTEMP'; the contents, including subdirectories, are deleted with 'Delete user data on card'

Specifies the storage location for the destination file

DESTINATIONFILENAME

Direction:	Input parameter
Data type:	STRING
Parameter index:	7

Name and path of the target file under LOCATION

DONE

Direction:	Output parameter
Data type:	BOOL
Parameter index:	8

The block was ended.

BUSY

Direction:	Output parameter
Data type:	BOOL
Parameter index:	9

The block is being processed.

ERROR

Direction:	Output parameter
Data type:	BOOL
Parameter index:	10

The block was executed with errors.

ERRORID

Direction:	Output parameter
Data type:	DWORD
Parameter index:	11

Error specification

16#00000000	Error-free - job performed successfully
16#FFFF8003	Command buffer could not be created.
16#FFFF8008	File is open.
16#FFFF8009	Internal error.
16#FFFF800A	File not found.
16#FFFF800B	File already exists.
16#FFFF800E	Invalid file name.
16#FFFF800F	No memory space available.
16#FFFF8010	Access was denied.

Note

About error value 16#FFFF8010

Check whether one of the following cases may apply: a) The file is open. b) The file is write-protected, i.e. the file is open as read-only.

2.5.2.4 _fileDelete

The function block deletes a file. The file can only be deleted if it is closed (_fileClose might have to be called first).

Additional references

Further information is available at:

- SIMOTION Runtime Basic Functions Function Manual
  Section Managing files from a user program (as of V4.4)
- or in the online help

Parameters:

EXECUTE

Direction:	Input parameter
Data type:	BOOL
Parameter index:	1

Starting a new request with a rising edge

STORAGETYPE

Direction:	Input parameter
Data type:	EnumStorageType
Parameter index:	2

EnumStorageType

RAMDISK (0)	Temporary data storage; deleted on power off, not non-volatile
CARD (1)	Permanent data storage; non-volatile

Specifies the storage medium

LOCATION

Direction:	Input parameter
Data type:	EnumUserLocation
Parameter index:	3

EnumUserLocation

USERFILES (0)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERFILES'
USERLOG (1)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERLOG'
USERDATASETS (2)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\UPP\UNITDS'; the contents, including subdirectories, are deleted with 'Delete user data on card' if the 'Unit data sets' checkmark is activated
USERWEB (3)	Corresponds to the following path: '\USER \SIMOTION\HMI\FILES'
USERTEMP (4)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\USERTEMP'; the contents, including subdirectories, are deleted with 'Delete user data on card'

Specifies the storage location

FILENAME

Direction:	Input parameter
Data type:	STRING
Parameter index:	4

Name and path of the file under LOCATION

DONE

Direction:	Output parameter
Data type:	BOOL
Parameter index:	5

The block was ended.

BUSY

Direction:	Output parameter
Data type:	BOOL
Parameter index:	6

The block is being processed.

ERROR

Direction:	Output parameter
Data type:	BOOL
Parameter index:	7

The block was executed with errors.

ERRORID

Direction:	Output parameter
Data type:	DWORD
Parameter index:	8

Error specification

16#00000000	Error-free - job performed successfully
16#FFFF8003	Command buffer could not be created.
16#FFFF8008	File is open.
16#FFFF8009	Internal error.
16#FFFF800A	File not found.
16#FFFF800E	Invalid file name.
16#FFFF8010	Access was denied.

Note

About error value 16#FFFF8010

Check whether one of the following cases may apply: a) The file is open. b) The file is write-protected, i.e. the file is open as read-only.

2.5.2.5 _filehandle

Status values can be ascertained by calling the instance of _filehandle separately. The _filehandle function block always indentified precisely one file. Therefore, just one file name (STORAGETYPE, LOCATION, and FILENAME) and one file position (ACTUALFILEPOSITION) will exist for a _filehandle if the file is open. The ISFILEOPEN parameter indicates whether the file is open via the current instance of _filehandle.

The NUMBEROFOPENFILES parameter shows all files that have been opened by the user. The available memory on the storage medium is returned in the AVAILABLESTORAGE output parameter, which is specified via the STORAGETYPE parameter. If the corresponding file for a _filehandle is closed, the AVAILABLESTORAGE output parameter outputs the available memory on the memory card. The information from the NUMBEROFOPENFILES and AVAILABLESTORAGE parameters are to be considered independently of _filehandle. The NUMBEROFOPENFILES parameter shows all files that have been opened by the user. This information is independent of _filehandle.

The resource for accessing the file is linked to the _filehandle function block. The declaration of the _fileHandle variable determines the service life of the function block and thus also the enable of the resource in relation to the file system (see also variable model). If the instance of the _filehandle FB is declared as a static or global variable, the user must ensure that the file is closed at an appropriate time (_flieClose), otherwise the system resources remain occupied and retain their validity when the mode changes from STOP->RUN. If the instance of the _filehandle FB is declared as a temporary variable, the system resources are released automatically when the validity range of the FB is left. If the user changes to STOP state during an active job, the job is finished in the background and then the system resources are released. This means that if there is a fast change of the mode from RUN->STOP->RUN, the system resources may still be occupied in the background, as functions, such as writing, are still active and will not be available to the user until a later time.

Note

Generally valid note on using local variables

We strongly recommend that for waiting loops, e.g. on completion of a function block, you call the _waitTime system function between the individual cycles to keep system utilization as low as possible and to free up computing time for other tasks.

The user creates an instance of _filehandle in his program and transfers this instance to the function blocks for opening, reading, writing, and closing as an IN/OUT parameter as soon as the call comes. The instance of _filehandle is, as it were, initialized with the corresponding data and linked to the file by means of a call to _fileOpen.

Additional references

Further information is available at:

- SIMOTION Runtime Basic Functions Function Manual
  Section Reading and writing files from a user program (as of V4.4)
- or in the online help

Parameters:

EXECUTE

Direction:	Input parameter
Data type:	BOOL
Parameter index:	1

Starting a new request with a rising edge

DONE

Direction:	Output parameter
Data type:	BOOL
Parameter index:	2

The block was ended.

BUSY

Direction:	Output parameter
Data type:	BOOL
Parameter index:	3

The block is being processed.

ERROR

Direction:	Output parameter
Data type:	BOOL
Parameter index:	4

The block was executed with errors.

ERRORID

Direction:	Output parameter
Data type:	DWORD
Parameter index:	5

Error specification

16#00000000	Error-free - job performed successfully
16#FFFF8003	Command buffer could not be created.
16#FFFF8009	Internal error.
16#FFFF8010	Access was denied.

Note

About error value 16#FFFF8010

Check whether one of the following cases may apply: a) The file is open. b) The file is write-protected, i.e. the file is open as read-only.

NUMBEROFOPENFILES

Direction:	Output parameter
Data type:	UDINT
Parameter index:	6
System default:	0

Number of open files

AVAILABLESTORAGE

Direction:	Output parameter
Data type:	UDINT
Parameter index:	7
System default:	0

Available memory space on the STORAGETYPE storage medium if the file is open, otherwise available memory space on the memory card.

ISFILEOPEN

Direction:	Output parameter
Data type:	EnumYesNo
Parameter index:	8
System default:	NO

EnumYesNo

NO (91)	No
YES (173)	Yes

YES - The file is open via this instance. NO - No file is open via this instance

STORAGETYPE

Direction:	Output parameter
Data type:	EnumStorageType
Parameter index:	9

EnumStorageType

RAMDISK (0)	Temporary data storage; deleted on power off, not non-volatile
CARD (1)	Permanent data storage; non-volatile

Storage medium if the file is open

LOCATION

Direction:	Output parameter
Data type:	EnumUserLocation
Parameter index:	10

EnumUserLocation

USERFILES (0)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERFILES'
USERLOG (1)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERLOG'
USERDATASETS (2)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\UPP\UNITDS'; the contents, including subdirectories, are deleted with 'Delete user data on card' if the 'Unit data sets' checkmark is activated
USERWEB (3)	Corresponds to the following path: '\USER \SIMOTION\HMI\FILES'
USERTEMP (4)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\USERTEMP'; the contents, including subdirectories, are deleted with 'Delete user data on card'

Storage location for the file if the file is open

FILENAME

Direction:	Output parameter
Data type:	STRING
Parameter index:	11

Name and path for the file under LOCATION if the file is open

ACTUALFILEPOSITION

Direction:	Output parameter
Data type:	DINT
Parameter index:	12
System default:	-1

Current position within the file if the file is not open is -1

2.5.2.6 _fileOpen

The function block opens the specified text file on the specified storage medium. The storage medium is specified at the STORAGETYPE parameter. The storage location is selected via the LOCATION parameter. In addition to the file name, the user can also specify a user-defined path at the FILENAME parameter; this is then set up under LOCATION.

The file can be opened in "read-only" or "read and write" mode; this is specified via the OPENFILE parameter. If the file does not exist when accessed in read-only mode, an error is output. When accessed in write mode, the file is recreated. The size of the file is only relevant when recreating a file.

If the path does not exist when a file is recreated, it is set up.

The number of files open at the same time must not exceed five; otherwise, an error is output.

If a file is opened as "read and write", this is an exclusive access right, i.e. if the file is already open as "read only", the file can no longer be opened as "read and write".

This also applies in reverse if the file is already open as "read and write", it can no longer be opened as "read only".

However, several instances of a file can be open simultaneously as "read only".

The available memory space on the storage medium is returned in the AVAILABLESTORAGE output parameter.

Additional references

Further information is available at:

- SIMOTION Runtime Basic Functions Function Manual
  Section Reading and writing files from a user program (as of V4.4)
  Section Managing files from a user program (as of V4.4)
- or in the online help

Parameters:

EXECUTE

Direction:	Input parameter
Data type:	BOOL
Parameter index:	1

Starting a new request with a rising edge

STORAGETYPE

Direction:	Input parameter
Data type:	EnumStorageType
Parameter index:	2

EnumStorageType

RAMDISK (0)	Temporary data storage; deleted on power off, not non-volatile
CARD (1)	Permanent data storage; non-volatile

Specifies the storage medium

LOCATION

Direction:	Input parameter
Data type:	EnumUserLocation
Parameter index:	3

EnumUserLocation

USERFILES (0)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERFILES'
USERLOG (1)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERLOG'
USERDATASETS (2)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\UPP\UNITDS'; the contents, including subdirectories, are deleted with 'Delete user data on card' if the 'Unit data sets' checkmark is activated
USERWEB (3)	Corresponds to the following path: '\USER \SIMOTION\HMI\FILES'
USERTEMP (4)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\USERTEMP'; the contents, including subdirectories, are deleted with 'Delete user data on card'

Specifies the storage location

FILENAME

Direction:	Input parameter
Data type:	STRING
Parameter index:	4

Name and path of the file under LOCATION

OPENFILE

Direction:	Input parameter
Data type:	EnumOpenFile
Parameter index:	5

EnumOpenFile

READONLY (0)	open in read-only mode
READWRITE (1)	open in read and write mode

Access mode to the file

FILESIZE

Direction:	Input parameter
Data type:	UDINT
Parameter index:	6

Size of the file to be created in bytes only significant if the file is being recreated

HANDLE

Direction:	Input/output parameter
Data type:	_filehandle
Parameter index:	7

EnumYesNo

NO (91)	No
YES (173)	Yes

EnumStorageType

RAMDISK (0)	Temporary data storage; deleted on power off, not non-volatile
CARD (1)	Permanent data storage; non-volatile

EnumUserLocation

USERFILES (0)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERFILES'
USERLOG (1)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERLOG'
USERDATASETS (2)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\UPP\UNITDS'; the contents, including subdirectories, are deleted with 'Delete user data on card' if the 'Unit data sets' checkmark is activated
USERWEB (3)	Corresponds to the following path: '\USER \SIMOTION\HMI\FILES'
USERTEMP (4)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\USERTEMP'; the contents, including subdirectories, are deleted with 'Delete user data on card'

_filehandle

Structure	Name	Data type	Unit
DONE		BOOL	-
BUSY		BOOL	-
ERROR		BOOL	-
ERRORID		DWORD	-
NUMBEROFOPENFILES		UDINT	-
AVAILABLESTORAGE		UDINT	-
ISFILEOPEN		EnumYesNo	-
STORAGETYPE		EnumStorageType	-

LOCATION		EnumUserLocation	-
FILENAME		STRING	-
ACTUALFILEPOSITION		DINT	-
_INST_DATA_		ARRAY [0..0] OF DINT	-

Transfer of the _filehandle function block

DONE

Direction:	Output parameter
Data type:	BOOL
Parameter index:	19

The block was ended.

BUSY

Direction:	Output parameter
Data type:	BOOL
Parameter index:	20

The block is being processed.

ERROR

Direction:	Output parameter
Data type:	BOOL
Parameter index:	21

The block was executed with errors.

ERRORID

Direction:	Output parameter
Data type:	DWORD
Parameter index:	22

Error specification

16#00000000	Error-free - job performed successfully
16#FFFF8003	Command buffer could not be created.
16#FFFF8009	Internal error.
16#FFFF800A	File not found.
16#FFFF800C	Maximum number of open files exceeded.
16#FFFF800D	Invalid handle.
16#FFFF800E	Invalid file name.
16#FFFF800F	No memory space available.
16#FFFF8010	Access was denied.
16#FFFF8014	Handle already assigned. Another file is opened via the handle.

Note

About error value 16#FFFF8010

Check whether one of the following cases may apply: a) The file is open. b) The file is write-protected, i.e. the file is open as read-only.

AVAILABLESTORAGE

Direction:	Output parameter
Data type:	UDINT
Parameter index:	23

Available memory space on the STORAGETYPE storage medium.

2.5.2.7 _fileRead

The function block reads data from a file in "binary" format and saves the data read in the BUFFER parameter.

Additional references

Further information is available at:

- SIMOTION Runtime Basic Functions Function Manual
  Section Reading and writing files from a user program (as of V4.4)
- or in the online help

Parameters:

EXECUTE

Direction:	Input parameter
Data type:	BOOL
Parameter index:	1

Starting a new request with a rising edge

DATALENGTH

Direction:	Input parameter
Data type:	UDINT
Parameter index:	2

Data length to be read in bytes

BUFFERPOSITION

Direction:	Input parameter
Data type:	UDINT
Parameter index:	3

Position relative to BUFFER, the data read is written to the buffer starting from the position

HANDLE

Direction:	Input/output parameter
Data type:	_filehandle
Parameter index:	4

EnumYesNo

NO (91)	No
YES (173)	Yes

EnumStorageType

RAMDISK (0)	Temporary data storage; deleted on power off, not non-volatile
CARD (1)	Permanent data storage; non-volatile

EnumUserLocation

USERFILES (0)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERFILES'
USERLOG (1)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERLOG'
USERDATASETS (2)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\UPP\UNITDS'; the contents, including subdirectories, are deleted with 'Delete user data on card' if the 'Unit data sets' checkmark is activated
USERWEB (3)	Corresponds to the following path: '\USER \SIMOTION\HMI\FILES'
USERTEMP (4)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\USERTEMP'; the contents, including subdirectories, are deleted with 'Delete user data on card'

_filehandle

Structure	Name	Data type	Unit
DONE		BOOL	-
BUSY		BOOL	-
ERROR		BOOL	-
ERRORID		DWORD	-
NUMBEROFOPENFILES		UDINT	-
AVAILABLESTORAGE		UDINT	-
ISFILEOPEN		EnumYesNo	-
STORAGETYPE		EnumStorageType	-

LOCATION		EnumUserLocation	-
FILENAME		STRING	-
ACTUALFILEPOSITION		DINT	-
_INST_DATA_		ARRAY [0..0] OF DINT	-

Transfer of the _filehandle function block

BUFFER

Direction:	Input/output parameter
Data type:	ARRAY [..] OF BYTE
Parameter index:	16

Data read

DONE

Direction:	Output parameter
Data type:	BOOL
Parameter index:	17

The block was ended.

BUSY

Direction:	Output parameter
Data type:	BOOL
Parameter index:	18

The block is being processed.

ERROR

Direction:	Output parameter
Data type:	BOOL
Parameter index:	19

The block was executed with errors.

ERRORID

Direction:	Output parameter
Data type:	DWORD
Parameter index:	20

Error specification

16#00000000	Error-free - job performed successfully
16#FFFF8003	Command buffer could not be created.
16#FFFF8005	Error when writing.
16#FFFF8009	Internal error.

16#FFFF800D	Invalid handle.
16#FFFF8010	Access was denied.
16#FFFF8011	Data not read in full.

Note

About error value 16#FFFF8010

Check whether one of the following cases may apply: a) The file is open. b) The file is write-protected, i.e. the file is open as read-only.

BYTESREAD

Direction:	Output parameter
Data type:	UDINT
Parameter index:	21

Number of bytes read.

2.5.2.8 _fileReadLn

The function block reads a line from the text file in text format as far as CR/LF and writes the data read to the LINE parameter.

Additional references

Further information is available at:

- SIMOTION Runtime Basic Functions Function Manual
  Section Reading and writing files from a user program (as of V4.4)
- or in the online help

Parameters:

EXECUTE

Direction:	Input parameter
Data type:	BOOL
Parameter index:	1

Starting a new request with a rising edge

HANDLE

Direction:	Input/output parameter
Data type:	_filehandle
Parameter index:	2

EnumYesNo

NO (91)	No
YES (173)	Yes

EnumStorageType

RAMDISK (0)	Temporary data storage; deleted on power off, not non-volatile
CARD (1)	Permanent data storage; non-volatile

EnumUserLocation

USERFILES (0)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERFILES'
USERLOG (1)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERLOG'
USERDATASETS (2)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\UPP\UNITDS'; the contents, including subdirectories, are deleted with 'Delete user data on card' if the 'Unit data sets' checkmark is activated
USERWEB (3)	Corresponds to the following path: '\USER \SIMOTION\HMI\FILES'
USERTEMP (4)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\USERTEMP'; the contents, including subdirectories, are deleted with 'Delete user data on card'

_filehandle

Structure	Name	Data type	Unit
DONE		BOOL	-
BUSY		BOOL	-
ERROR		BOOL	-
ERRORID		DWORD	-
NUMBEROFOPENFILES		UDINT	-
AVAILABLESTORAGE		UDINT	-
ISFILEOPEN		EnumYesNo	-
STORAGETYPE		EnumStorageType	-

LOCATION		EnumUserLocation	-
FILENAME		STRING	-
ACTUALFILEPOSITION		DINT	-
_INST_DATA_		ARRAY [0..0] OF DINT	-

Transfer of the _filehandle function block

DONE

Direction:	Output parameter
Data type:	BOOL
Parameter index:	14

The block was ended.

BUSY

Direction:	Output parameter
Data type:	BOOL
Parameter index:	15

The block is being processed.

ERROR

Direction:	Output parameter
Data type:	BOOL
Parameter index:	16

The block was executed with errors.

ERRORID

Direction:	Output parameter
Data type:	DWORD
Parameter index:	17

Error specification

16#00000000	Error-free - job performed successfully
16#FFFF8003	Command buffer could not be created.
16#FFFF8005	Error when writing.
16#FFFF8009	Internal error.
16#FFFF800D	Invalid handle.
16#FFFF8010	Access was denied.
16#FFFF8011	Data not read in full.
16#FFFF8013	CR/LF not found during reading.

Note

About error value 16#FFFF8010

Check whether one of the following cases may apply: a) The file is open. b) The file is write-protected, i.e. the file is open as read-only.

LINE

Direction:	Output parameter
Data type:	STRING
Parameter index:	18

Buffer for read data.

2.5.2.9 _fileRename

The function block renames a file inside a directory. The file can only be renamed if it is closed (_fileClose might have to be called first).

Additional references

Further information is available at:

- SIMOTION Runtime Basic Functions Function Manual
  Section Managing files from a user program (as of V4.4)
- or in the online help

Parameters:

EXECUTE

Direction:	Input parameter
Data type:	BOOL
Parameter index:	1

Starting a new request with a rising edge

STORAGETYPE

Direction:	Input parameter
Data type:	EnumStorageType
Parameter index:	2

EnumStorageType

RAMDISK (0)	Temporary data storage; deleted on power off, not non-volatile
CARD (1)	Permanent data storage; non-volatile

Specifies the storage medium

LOCATION

Direction:	Input parameter
Data type:	EnumUserLocation
Parameter index:	3

EnumUserLocation

USERFILES (0)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERFILES'
USERLOG (1)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERLOG'
USERDATASETS (2)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\UPP\UNITDS'; the contents, including subdirectories, are deleted with 'Delete user data on card' if the 'Unit data sets' checkmark is activated
USERWEB (3)	Corresponds to the following path: '\USER \SIMOTION\HMI\FILES'
USERTEMP (4)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\USERTEMP'; the contents, including subdirectories, are deleted with 'Delete user data on card'

Specifies the storage location

FILENAME

Direction:	Input parameter
Data type:	STRING
Parameter index:	4

Name and path of the file under LOCATION

NEWFILENAME

Direction:	Input parameter
Data type:	STRING
Parameter index:	5

New name of file without path

DONE

Direction:	Output parameter
Data type:	BOOL
Parameter index:	6

The block was ended.

BUSY

Direction:	Output parameter
Data type:	BOOL
Parameter index:	7

The block is being processed.

ERROR

Direction:	Output parameter
Data type:	BOOL
Parameter index:	8

The block was executed with errors.

ERRORID

Direction:	Output parameter
Data type:	DWORD
Parameter index:	9

Error specification

16#00000000	Error-free - job performed successfully
16#FFFF8003	Command buffer could not be created.
16#FFFF8008	File is open.
16#FFFF8009	Internal error.
16#FFFF800A	File not found.
16#FFFF800B	File already exists.
16#FFFF800E	Invalid file name.
16#FFFF800F	No memory space available.
16#FFFF8010	Access was denied.

Note

About error value 16#FFFF8010

Check whether one of the following cases may apply: a) The file is open. b) The file is write-protected, i.e. the file is open as read-only.

2.5.2.10 _fileSetPosition

The function block defines the position within a text file that is already open starting from which data is to be read or written. The FILEPOSITIONOFFSET parameter is indicated in bytes and refers to the position specified with the FILEPOSITION parameter. The FILEPOSITION parameter can be the start or end of the file or the current position.

It is possible to explicitly determine a position in the file which is required for an application. For example, there is a code number at the end of a file, and only this code number is to be read out. It is then not necessary to read the complete file, but rather to go to the desired position in the file and then call the read command.

The function block (FB) does not have to be called every time before performing a read or write operation. After opening, the position is set to the start of the file automatically and moves accordingly during reading or writing based on the number of bytes read/written. If the end of a file is reached (through reading, for example), the _fileSetPosition function block can be used to reset the position to the start of the file without having to open the file again.

Additional references

Further information is available at:

- SIMOTION Runtime Basic Functions Function Manual
  Section Reading and writing files from a user program (as of V4.4)
- or in the online help

Parameters:

EXECUTE

Direction:	Input parameter
Data type:	BOOL
Parameter index:	1

Starting a new request with a rising edge

FILEPOSITION

Direction:	Input parameter
Data type:	EnumFilePosition
Parameter index:	2

EnumFilePosition

BEGINNING_OF_FILE (0)	Start of file
ACTUAL_FILEPOSITION (1)	Current position in file
END_OF_FILE (2)	End of file

Position in the file

FILEPOSITIONOFFSET

Direction:	Input parameter
Data type:	DINT
Parameter index:	3

Offset in bytes relative to the position in the file

HANDLE

Direction:	Input/output parameter
Data type:	_filehandle
Parameter index:	4

EnumYesNo

NO (91)	No
YES (173)	Yes

EnumStorageType

RAMDISK (0)	Temporary data storage; deleted on power off, not non-volatile
CARD (1)	Permanent data storage; non-volatile

EnumUserLocation

USERFILES (0)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERFILES'
USERLOG (1)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERLOG'
USERDATASETS (2)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\UPP\UNITDS'; the contents, including subdirectories, are deleted with 'Delete user data on card' if the 'Unit data sets' checkmark is activated
USERWEB (3)	Corresponds to the following path: '\USER \SIMOTION\HMI\FILES'
USERTEMP (4)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\USERTEMP'; the contents, including subdirectories, are deleted with 'Delete user data on card'

_filehandle

Structure	Name	Data type	Unit
DONE		BOOL	-
BUSY		BOOL	-
ERROR		BOOL	-
ERRORID		DWORD	-
NUMBEROFOPENFILES		UDINT	-
AVAILABLESTORAGE		UDINT	-
ISFILEOPEN		EnumYesNo	-
STORAGETYPE		EnumStorageType	-

LOCATION		EnumUserLocation	-
FILENAME		STRING	-
ACTUALFILEPOSITION		DINT	-
_INST_DATA_		ARRAY [0..0] OF DINT	-

Transfer of the _filehandle function block

DONE

Direction:	Output parameter
Data type:	BOOL
Parameter index:	16

The block was ended.

BUSY

Direction:	Output parameter
Data type:	BOOL
Parameter index:	17

The block is being processed.

ERROR

Direction:	Output parameter
Data type:	BOOL
Parameter index:	18

The block was executed with errors.

ERRORID

Direction:	Output parameter
Data type:	DWORD
Parameter index:	19

Error specification

16#00000000	Error-free - job performed successfully
16#FFFF8003	Command buffer could not be created.
16#FFFF8009	Internal error.
16#FFFF800D	Invalid handle.
16#FFFF8010	Access was denied.

Note

About error value 16#FFFF8010

Check whether one of the following cases may apply: a) The file is open. b) The file is write-protected, i.e. the file is open as read-only.

NEWPOSITION

Direction:	Output parameter
Data type:	DINT
Parameter index:	20

Offset in bytes of the new position in the file relative to the start of the file.

2.5.2.11 _fileWrite

The function block writes data to a file in "binary" format.

The data to be written is transferred in the BUFFER parameter. If the specified file size is reached during writing, an error is output and only the data up to the end of the file is written.

The function block must not be used in time-monitored or synchronous tasks!

Additional references

Further information is available at:

- SIMOTION Runtime Basic Functions Function Manual
  Section Reading and writing files from a user program (as of V4.4)
- or in the online help

Parameters:

EXECUTE

Direction:	Input parameter
Data type:	BOOL
Parameter index:	1

Starting a new request with a rising edge

DATALENGTH

Direction:	Input parameter
Data type:	UDINT
Parameter index:	2

Data length to be written in bytes

BUFFERPOSITION

Direction:	Input parameter
Data type:	UDINT
Parameter index:	3

Position relative to BUFFER, the data to be written to the buffer is written starting from the position

HANDLE

Direction:	Input/output parameter
Data type:	_filehandle
Parameter index:	4

EnumYesNo

NO (91)	No
YES (173)	Yes

EnumStorageType

RAMDISK (0)	Temporary data storage; deleted on power off, not non-volatile
CARD (1)	Permanent data storage; non-volatile

EnumUserLocation

USERFILES (0)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERFILES'
USERLOG (1)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERLOG'
USERDATASETS (2)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\UPP\UNITDS'; the contents, including subdirectories, are deleted with 'Delete user data on card' if the 'Unit data sets' checkmark is activated
USERWEB (3)	Corresponds to the following path: '\USER \SIMOTION\HMI\FILES'
USERTEMP (4)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\USERTEMP'; the contents, including subdirectories, are deleted with 'Delete user data on card'

_filehandle

Structure	Name	Data type	Unit
DONE		BOOL	-
BUSY		BOOL	-
ERROR		BOOL	-
ERRORID		DWORD	-
NUMBEROFOPENFILES		UDINT	-
AVAILABLESTORAGE		UDINT	-
ISFILEOPEN		EnumYesNo	-
STORAGETYPE		EnumStorageType	-

LOCATION		EnumUserLocation	-
FILENAME		STRING	-
ACTUALFILEPOSITION		DINT	-
_INST_DATA_		ARRAY [0..0] OF DINT	-

Transfer of the _filehandle function block

BUFFER

Direction:	Input/output parameter
Data type:	ARRAY [..] OF BYTE
Parameter index:	16

Data to be written

DONE

Direction:	Output parameter
Data type:	BOOL
Parameter index:	17

The block was ended.

BUSY

Direction:	Output parameter
Data type:	BOOL
Parameter index:	18

The block is being processed.

ERROR

Direction:	Output parameter
Data type:	BOOL
Parameter index:	19

The block was executed with errors.

ERRORID

Direction:	Output parameter
Data type:	DWORD
Parameter index:	20

Error specification

16#00000000	Error-free - job performed successfully
16#FFFF8003	Command buffer could not be created.
16#FFFF8006	Error while writing.
16#FFFF8009	Internal error.

16#FFFF800D	Invalid handle.
16#FFFF8010	Access was denied.
16#FFFF8011	Data not written in full.

Note

About error value 16#FFFF8010

Check whether one of the following cases may apply: a) The file is open. b) The file is write-protected, i.e. the file is open as read-only.

BYTESWRITTEN

Direction:	Output parameter
Data type:	UDINT
Parameter index:	21

Number of bytes written.

2.5.2.12 _fileWriteLn

The function block writes a line to a text file in "text" format and completes the line with CR/LF. The data to be written is transferred in the LINE parameter.

The size of the file does not change. If the specified data volume exceeds the current file size, the data is not written and an error is output.

Additional references

Further information is available at:

- SIMOTION Runtime Basic Functions Function Manual
  Section Reading and writing files from a user program (as of V4.4)
- or in the online help

Parameters:

EXECUTE

Direction:	Input parameter
Data type:	BOOL
Parameter index:	1

Starting a new request with a rising edge

LINE

Direction:	Input parameter
Data type:	STRING
Parameter index:	2

Data to be written

HANDLE

Direction:	Input/output parameter
Data type:	_filehandle
Parameter index:	3

EnumYesNo

NO (91)	No
YES (173)	Yes

EnumStorageType

RAMDISK (0)	Temporary data storage; deleted on power off, not non-volatile
CARD (1)	Permanent data storage; non-volatile

EnumUserLocation

USERFILES (0)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERFILES'
USERLOG (1)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERLOG'
USERDATASETS (2)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\UPP\UNITDS'; the contents, including subdirectories, are deleted with 'Delete user data on card' if the 'Unit data sets' checkmark is activated
USERWEB (3)	Corresponds to the following path: '\USER \SIMOTION\HMI\FILES'
USERTEMP (4)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\USERTEMP'; the contents, including subdirectories, are deleted with 'Delete user data on card'

_filehandle

Structure	Name	Data type	Unit
DONE		BOOL	-
BUSY		BOOL	-
ERROR		BOOL	-
ERRORID		DWORD	-
NUMBEROFOPENFILES		UDINT	-
AVAILABLESTORAGE		UDINT	-
ISFILEOPEN		EnumYesNo	-
STORAGETYPE		EnumStorageType	-

LOCATION		EnumUserLocation	-
FILENAME		STRING	-
ACTUALFILEPOSITION		DINT	-
_INST_DATA_		ARRAY [0..0] OF DINT	-

Transfer of the _filehandle function block

DONE

Direction:	Output parameter
Data type:	BOOL
Parameter index:	15

The block was ended.

BUSY

Direction:	Output parameter
Data type:	BOOL
Parameter index:	16

The block is being processed.

ERROR

Direction:	Output parameter
Data type:	BOOL
Parameter index:	17

The block was executed with errors.

ERRORID

Direction:	Output parameter
Data type:	DWORD
Parameter index:	18

Error specification

16#00000000	Error-free - job performed successfully
16#FFFF8003	Command buffer could not be created.
16#FFFF8006	Error while writing.
16#FFFF8009	Internal error.
16#FFFF800D	Invalid handle.
16#FFFF8010	Access was denied.
16#FFFF8011	Data not written in full.
16#FFFF8012	Data not written. There are not enough bytes available in the file to write the line incl. CR/LF.

Note

About error value 16#FFFF8010

Check whether one of the following cases may apply: a) The file is open. b) The file is write-protected, i.e. the file is open as read-only.

WRITTENLINE

Direction:	Output parameter
Data type:	STRING
Parameter index:	19

Buffer for written data.

2.5.2.13 _getStateOfFile

The function block supplies status information about a file. The file attributes are set implicitly when the file is created. Creating write-protected files or setting write protection retrospectively is excluded.

Parameters:

EXECUTE

Direction:	Input parameter
Data type:	BOOL
Parameter index:	1

Starting a new request with a rising edge

STORAGETYPE

Direction:	Input parameter
Data type:	EnumStorageType
Parameter index:	2

EnumStorageType

RAMDISK (0)	Temporary data storage; deleted on power off, not non-volatile
CARD (1)	Permanent data storage; non-volatile

Specifies the storage medium

LOCATION

Direction:	Input parameter
Data type:	EnumUserLocation
Parameter index:	3

EnumUserLocation

USERFILES (0)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERFILES'
USERLOG (1)	Corresponds to the following path: '\USER \SIMOTION\HMI\USERLOG'
USERDATASETS (2)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\UPP\UNITDS'; the contents, including subdirectories, are deleted with 'Delete user data on card' if the 'Unit data sets' checkmark is activated
USERWEB (3)	Corresponds to the following path: '\USER \SIMOTION\HMI\FILES'
USERTEMP (4)	Corresponds to the following path: '\USER \SIMOTION\USER_DIR\USERTEMP'; the contents, including subdirectories, are deleted with 'Delete user data on card'

Specifies the storage location

FILENAME

Direction:	Input parameter
Data type:	STRING
Parameter index:	4

Name and path of the file under LOCATION

DONE

Direction:	Output parameter
Data type:	BOOL
Parameter index:	5

The block was ended.

BUSY

Direction:	Output parameter
Data type:	BOOL
Parameter index:	6

The block is being processed.

ERROR

Direction:	Output parameter
Data type:	BOOL
Parameter index:	7

The block was executed with errors.

ERRORID

Direction:	Output parameter
Data type:	DWORD
Parameter index:	8

Error specification

16#00000000	Error-free - job performed successfully
16#FFFF8003	Command buffer could not be created.
16#FFFF8009	Internal error.
16#FFFF800A	File not found.
16#FFFF800E	Invalid file name.
16#FFFF800F	No memory space available.
16#FFFF8010	Access was denied.

DATEOFFILE

Direction:	Output parameter
Data type:	DATE
Parameter index:	9

Date of last change made to the file.

TIMEOFFILE

Direction:	Output parameter
Data type:	TIME
Parameter index:	10

Time of last change made to the file.

ACTUALFILESIZE

Direction:	Output parameter
Data type:	UDINT
Parameter index:	11

Current size of the file in bytes.

ACTUALFILEPOSITION

Direction:	Output parameter
Data type:	DINT
Parameter index:	12

Current position in the file (if the specified FILENAME refers to a file) if the file or a directory is closed -1

ISREADONLY

Direction:	Output parameter
Data type:	EnumYesNo
Parameter index:	13

EnumYesNo

NO (91)	No
YES (173)	Yes

YES - The file is write-protected and cannot be changed

ISDIRECTORY

Direction:	Output parameter
Data type:	EnumYesNo
Parameter index:	14

EnumYesNo

NO (91)	No
YES (173)	Yes

YES - the FILENAME specified represents a directory

FILEEXISTING

Direction:	Output parameter
Data type:	EnumYesNo
Parameter index:	15

EnumYesNo

NO (91)	No
YES (173)	Yes

YES - The file exists

2.5.3 Data storage

2.5.3.1 _resetUnitData

This system function resets the data of a source to its initial values.

Return value: StructRetUnitDataSetCommand

EnumDeviceUnitDataSetCommand

DONE (0)	Command successful
ACTIVE (1)	Command is being processed
INTERNEL_ERROR (2)	Internal error
COMMAND_FAILED (3)	Command failed
NO_COMMAND_BUFFER_AVAILABLE (4)	No free command buffer
COMMAND_NOT_FOUND (5)	Command not found
DATASET_ID_NOT_VALID (6)	Invalid data set ID
READ_ERROR (7)	Error occurred while reading the data set
NO_STORAGE_AVAILABLE (8)	No memory available for writing the data set
ACCESS_DENIED (9)	Missing access right
DATASET_ALREADY_EXISTS (10)	Data set already exists
DATASET_NOT_FOUND (11)	Data set not found
UNIT_NOT_FOUND (12)	Data set unit not found
VERSION_MISSMATCH (13)	Incorrect data set for the current project
DATA_INCOMPLETE (14)	Incomplete loading/import of the data
SYMBOL_INFORMATION_NOT_AVAILABLE (15)	No symbol information available
DATA_MISMATCH (16)	Data selection not contained in the data set
DATA_INCOMPATIBLE (17)	Data could not be converted

StructRetUnitDataSetCommand
Return parameters for unit data set commands

Structure	Name	Data type	Unit
functionResult	Return value of function	EnumDeviceUnitDataSetCommand	-
handle	Handle for following call	UDINT	-

Description of the return values

Return value "functionResult" has the following meaning:

DONE means all selected data segments were successfully loaded. ACTIVE means the command is still being executed. INTERNAL_ERROR indicates that an internal error has occurred. COMMAND_FAILED means the command has not been executed. NO_COMMAND_BUFFER_AVAILABLE means there is no free command buffer available. COMMAND_NOT_FOUND means the command has not been found. DATA_SET_ID_NOT_VALID means there is an invalid data set ID. READ_ERROR means an error occurred while a data set was being read. NO_STORAGE_AVAILABLE means there is no free memory available for writing the data set. ACCESS_DENIED means that no access rights have been granted. DATASET_ALREADY_EXISTS means the data set already exists. DATASET_NOT_FOUND means the required data set has not been found. UNIT_NOT_FOUND means the data set unit has not been found. VERSION_MISMATCH means the required data set does not match the current project. DATA_INCOMPLETE means the data set has not been completely imported. SYMBOL_INFORMATION_NOT_AVAILABLE means no symbol information is available. DATA_MISMATCH means the data set does not contain a selected data segment, with the result that no data set can be loaded. DATA_INCOMPATIBLE means the data set contains additional data segments or that selected data segments are missing from the data set.

Result + Handle for the subsequent call of 'getStateOfUnitDataSetCommand' (continuation mode IMMEDIATELY).

Parameters:

unitName

Direction:	Input parameter
Data type:	STRING
Parameter index:	1

Name of source whose data are to be re-initialized.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	2

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Continuation mode:

IMMEDIATELY -> Start command only, scan result later.

WHEN_COMMAND_DONE -> Wait for end of command.

dataScope

Direction:	Input parameter
Data type:	EnumDeviceDataScope
Parameter index:	3

EnumDeviceDataScope

_INTERFACE (0)	Global data from INTERFACE
_IMPLEMENTATION (1)	Global data from IMPLEMENTATION
_INTERFACE_AND_IMPLEMENTATION (2)	Global data from INTERFACE and IMPLEMENTATION

Selection of data to be initialized:

_INTERFACE -> Global data from INTERFACE

_IMPLEMENTATION -> Global data from IMPLEMENTATION

_INTERFACE_AND_IMPLEMENTATION -> Global data from INTERFACE and IMPLEMENTATION

kindOfData

Direction:	Input parameter
Data type:	EnumDeviceKindOfData
Parameter index:	4

EnumDeviceKindOfData

NO_RETAIN_GLOBAL (0)	Global non-RETAIN data
_RETAIN (1)	Global RETAIN data
ALL_GLOBAL (2)	Global RETAIN and non-RETAIN data

Selection of data to be initialized:

NO_RETAIN_GLOBAL -> Global non-RETAIN data

_RETAIN -> Global RETAIN data

ALL_GLOBAL -> Global RETAIN and non-RETAIN data

2.5.3.2 _saveConfigData

This function saves modified TO configuration data in the RAM disk ("Copy current to RAM") and then copies the RAM disk to the ROM ("Copy RAM to ROM").

Note

Note the following to avoid errors:

- Do not use this system function cyclically
- Prevent Power Off while the system function is active
- Write access to the memory card is limited (see manuals)

Return value: EnumSaveState

EnumSaveState

SAVE_ABORTED (-1)	Saving aborted
SAVE_FINISHED (0)	Saving completed

With SAVE_ABORTED, the system function was not executed or was aborted. With SAVE_FINISHED, the system function was executed successfully.

2.5.3.3 _saveDiagnosticFiles

The function saves the diagnostics data in a ZIP file with a name specified as [name].zip in directory / User/Simotion/HMI/Files/[path] on the memory card.

Note

Calling the function:

This function can only be called of from motion tasks or the background task.

Parameters:

EXECUTE

Direction:	Input parameter
Data type:	BOOL
Parameter index:	1

Starts a new job with a rising edge.

PATH

Direction:	Input parameter
Data type:	STRING
Parameter index:	2

Path of the file under /USER/SIMOTION/HMI/Files/...

NAME

Direction:	Input parameter
Data type:	STRING
Parameter index:	3

Name of the ZIP file; if required, the extension .zip is attached.

OVERWRITE (optional)

Direction:	Input parameter
Data type:	BOOL
Parameter index:	4
System default:	FALSE

If a ZIP file of the same name already exists, it will be overwritten with the new data.

DONE

Direction:	Output parameter
Data type:	BOOL
Parameter index:	5
System default:	FALSE

The job has been completed.

BUSY

Direction:	Output parameter
Data type:	BOOL
Parameter index:	6
System default:	FALSE

The job is being processed.

ERROR

Direction:	Output parameter
Data type:	BOOL
Parameter index:	7
System default:	FALSE

Job completed with errors.

ERRORID

Direction:	Output parameter
Data type:	DWORD
Parameter index:	8
System default:	0

Description of the error messages:

16#00000000	Error-free - job performed successfully
16#FFFF8009	Internal error.
16#FFFF800B	ZIP file is already available.
16#FFFF800D	Path or file name invalid.
16#FFFF800E	Too little memory space available.

2.5.3.4 _savePersistentMemoryData

The '_savePersistentMemoryData' system function is used to save the contents of the SIMOTION non-volatile data to the memory card. This backup prevents the retain variables and the absolute encoder position from being lost if a component is replaced.

The contents are saved to the "PMEMORY.XML" file in the "USER/SIMOTION" directory.

The file contents are copied back to the SRAM when the controller ramps up if any of the following conditions have occurred:

- SRAM content lost due to a discharged battery. In this case, the entire contents of the SRAM are restored from the file.
- After an overall reset. The portions of the SRAM affected by the overall reset are restored from the file.

If consistency is required for certain applicative data, the user must ensure that this data is not changed while the backup is running. If the 'PMEMORY.XML' file saved on the MemoryCard should be deleted, this can be dones with the help of an external card reader and the Windows Explorer.

On the system side, this system function ensures that a consistent overall image of the SIMOTION non-volatile data is always available the next time the unit is powered on, even if there is a power failure during backup. To this end, before a new backup file is created, any existing backup file is renamed as 'PMEMORY.BAK'. If the save operation to the new backup file fails (e.g. because the capacity of the CF card is insufficient), this backup copy of the backup file is used the next time an attempt is made to restore the contents of the SIMOTION non-volatile data. If the new file is successfully created, the backup copy of the backup file is deleted.

The origin of the current SRAM content can be determined from system variable "persistentDataPowerMonitoring.persistentDataState".

Because the function is not called synchronously, it can only be used in a Motion Task.

Note

The non-volatile SINAMICS data (NVRAM data) is backed up by setting the CU parameter p7775 to 1. (valid as of SINAMICS V4.5)

Note

In earlier versions of SIMOTION, the 'PMEMORY.DAT' file was generated by calling 'savePersistentMemoryData'.

In Version 3.2 and higher, the generated file is called 'PMEMORY.XML'. This file has a fundamentally different structure. Any existing 'PMEMORY.DAT' files are ignored starting in V3.2.

NOTICE

Preventing data loss

If you do not save the data to the CF card, it will be lost if a spare part must be used (defective module). If an absolute encoder overflow occurs after _savePersistentMemoryData, the actual position value will no longer be correct after the non-volatile data have been restored. In this case, homing (absolute encoder adjustment) must be repeated.

NOTICE
Save retain data Retain data of a module can be saved by means of _savePersistentMemoryData. After the upgrade of a module with subsequent loading of the project, the saved retain data is not restored.

Causes

Under certain circumstances (e.g. if system-defined structures that have been extended in a new SIMOTION version are used within a VAR_GLOBAL RETAIN block), retentive variables are automatically initialized after an upgrade. Retain data of the technology objects (absolute encoders) is not restored if the system variable _configurationManagement.preserveToRetainData is set to 'NO'.

Note

As of SIMOTION version V4.1.5.6, the default value of the system variable _configurationManagement.preserveToRetainData is YES.

Remedy

Use the SCOUT functions "Save variables" and "Restore variables" to save the retain data and restore it after the upgrade. Set the system variable _configurationManagement.preserveToRetainData to YES.

Additional references

Additional information is available at:

- SIMOTION Basic Functions Function Manual
  Section Memory access
- or in the SIMOTION online help.

Return value: EnumSaveState

EnumSaveState

SAVE_ABORTED (-1)	Saving aborted
SAVE_FINISHED (0)	Saving completed

With SAVE_ABORTED, the system function was not executed or was aborted. With SAVE_FINISHED, the system function was executed successfully.

In addition to the _savePersistentMemoryData function, the non-volatile data can also be saved via the switch position as of V4.1 SP2 of the SIMOTION D4xx (for further information see SIMOTION D410 and SIMOTION D4x5 Commissioning Manuals).

2.5.3.5 _saveTraceAndDiagnosticFiles

The function saves the available non-volatile data of the TO and device trace and the diagnostics data in a ZIP file with a name specified as [NAME].zip in directory /USER/SIMOTION/HMI/Files/[PATH].

The traces are not automatically restarted after storing.

Note

Calling the function:

This function can only be called of from motion tasks or the background task.

Parameters:

EXECUTE

Direction:	Input parameter
Data type:	BOOL
Parameter index:	1

Starts a new job with a rising edge.

PATH

Direction:	Input parameter
Data type:	STRING
Parameter index:	2

Path of the file under /USER/SIMOTION/HMI/Files/...

NAME

Direction:	Input parameter
Data type:	STRING
Parameter index:	3

Name of the ZIP file; if required, the extension .zip is attached.

OVERWRITE (optional)

Direction:	Input parameter
Data type:	BOOL
Parameter index:	4
System default:	FALSE

If a ZIP file of the same name already exists, it will be overwritten with the new data.

DONE

Direction:	Output parameter
Data type:	BOOL
Parameter index:	5
System default:	FALSE

The job has been completed.

BUSY

Direction:	Output parameter
Data type:	BOOL
Parameter index:	6
System default:	FALSE

The job is being processed.

ERROR

Direction:	Output parameter
Data type:	BOOL
Parameter index:	7
System default:	FALSE

Job completed with errors.

ERRORID

Direction:	Output parameter
Data type:	DWORD
Parameter index:	8
System default:	0

Description of the error messages:

16#00000000	Error-free - job performed successfully
16#FFFF8009	Internal error.
16#FFFF800B	ZIP file is already available.
16#FFFF800D	Path or file name invalid.
16#FFFF800E	Too little memory space available.

2.5.4 Modular machines

2.5.4.1 _activateConfiguration

This system function activates the configurations specified in the parameters.

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
  Section Activating a configuration
  Section Activating a kernel
- or in the SIMOTION online help.

Return value: StructRetConfiguration

EnumDeviceDataActivationState

NOT_ACTIVATED (0)	Configuration not activated
ACTIVATED (1)	The configuration is effective
NO_REQUEST_TO_ACTIVATE (2)	Desired configuration is not designated as the configuration to be activated
NO_VALID_CONFIGURATION_ID (3)	The configuration identification is outside of the permissible limits
CONFIGURATION_ID_NOT_FOUND (4)	Configuration identification not available on memory card
NOT_ENOUGH_RAM (5)	Insufficient RAM memory to take the configuration
RAM_DISK_FULL (6)	RAM disk is full
MEMORY_CARD_FULL (7)	Memory card is full
TIME_OUT (8)	Monitoring time has expired
INTERNAL_ERROR (9)	Internal error
ACCESS_DENIED (10)	Access denied, e.g. if access protection active

StructRetConfiguration
Structure for the return of the system function results _activateConfiguration _setFallBackConfiturationActive _setFallBackConfigurationInactive

Structure	Name	Data type	Unit
result	Display of the system function	UDINT	-
stateActivationState	Activation state	StructDeviceDataActivationState	-

StructDeviceDataActivationState
Activation state of the configurations

Structure	Name	Data type	Unit
projectData	Current status for project data activation	EnumDeviceDataActivationState	-
technologicalPackage	Current status for technology package activation	EnumDeviceDataActivationState	-
driveData	Current status for drive data activation	EnumDeviceDataActivationState	-
kernel	Current status for kernel activation	EnumDeviceDataActivationState	-

Description of the return values:

The required actions are asynchronous. The system function signals the status of the execution in the return code.

16#00000000	The configuration to be activated is already active. The system function only returns to the user program when no further internal actions are necessary. In other words, the configuration is not saved and there is no implicit reset of SIMOTION.
16#FFFF8091	Functionality is already requested (if functionality is running, another system function call).
16#FFFF8093	No internal memory available for executing the system function. The system function cannot be executed.
16#FFFF8094	Ram disk is full, the system function cannot be executed.
16#FFFF8095	The memory card is full. The configuration data cannot be saved.
16#FFFF8098	Functionality terminated with timeout.
16#FFFF8099	Internal error, function terminated.
16#FFFF80A1	The identification of the configuration to be activated lies outside the valid range.
16#FFFF80A3	The configuration to be activated is not on the memory card.
16#FFFF80A6	Exclusive use of SIMOTION cannot be disabled.
16#FFFF80A7	Archive cannot be extracted.
16#FFFF80B6	Access denied, e.g. if access protection active
16#FFFF80C1	Resources for system functions are limited. This display signals that the function cannot be executed due to a lack of resources. Resources are released automatically by SIMOTION. The user program must continue calling the function until this display no longer occurs.

16#FFFF80C3	SIMOTION is used exclusively, e.g. for the following user actions: - Download of project via SCOUT - Test operation of SIMOTION by means of SCOUT - Master control via machine control panel

It is not possible to change the configuration (project data) during the exclusive use of SIMOTION.

Parameters:

projectData (optional)

Direction:	Input parameter
Data type:	EnumYesNo
Parameter index:	1
System default:	NO

EnumYesNo

NO (91)	No
YES (173)	Yes

The project data is requested and activated. (HW Config, IO Config, user program, icons, etc.).

projectDataId (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	2
System default:	4294967295

ID for project data configuration.

technologicalPackage (optional)

Direction:	Input parameter
Data type:	EnumYesNo
Parameter index:	3
System default:	NO

EnumYesNo

NO (91)	No
YES (173)	Yes

The technology packages (TO) are requested and activated (Cam, Cam_ext, TControl).

technologicalPackageId (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	4
System default:	4294967295

Identification of the technology packages (TP) (Cam, Cam_ext, TControl).

driveData (optional)

Direction:	Input parameter
Data type:	EnumYesNo
Parameter index:	5
System default:	NO

EnumYesNo

NO (91)	No
YES (173)	Yes

The drive data is requested and activated (not V3.2).

driveDataId (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	6
System default:	4294967295

Identification of drive data configuration (not V3.2).

kernel (optional)

Direction:	Input parameter
Data type:	EnumYesNo
Parameter index:	7
System default:	NO

EnumYesNo

NO (91)	No
YES (173)	Yes

The kernel is requested and activated (not V3.2).

kernelId (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	8
System default:	4294967295

Identification of kernel (not V3.2).

timeOut (optional)

Direction:	Input parameter
Data type:	TIME
Parameter index:	9
System default:	T#5M

Monitoring time for change from Stop to Run mode once a configuration is activated after a SIMOTION restart. Five minutes are set as system default.

storageType (optional)

Direction:	Input parameter
Data type:	EnumDeviceStorageType
Parameter index:	10
System default:	PERMANENT_STORAGE

EnumDeviceStorageType

TEMPORARY_STORAGE (0)	Temporary data storage
PERMANENT_STORAGE (1)	Permanent data storage
USER_STORAGE (2)	Data storage in user path

Selection of the storage location In the case of TEMPORARY_STORAGE, there is temporary data storage on the RAM disk. In the case of a network failure, the data will be deleted. In the case of PERMANENT_STORAGE there will be permanent data storage on a memory card (MMC). The data is preserved in the case of a network failure. In the case of USER_Storage where the path is stated, the data storage is permitted when the default settings have been made.

2.5.4.2 _activateDpSlave

This system function establishes cyclical communication for the entire DP station (slave) or the PROFINET station (device), even if the station has multiple logical addresses (diagnostic addresses).

This system function cannot be called when the _deactivateDpSlave system function is active with the same 'logicalAddressOfDpStation'.

This system function applies to both PROFIBUS and PROFINET.

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- SIMOTION Communication System Manual
- or in the online help

Return value: DINT

The return value reports the job status.

Description of the return values:

16#00000000	Abort job. No jobs are now being executed for the logical address.
16#00000001	PROFIBUS slave/IO device activated - job completed with positive result. Job okay.
16#00000005	PROFIBUS slave/IO device is activated. The system function has not waited for a 'station recovery' alarm. Job completed with positive result. Job okay. A PROFIBUS slave/IO device can also be activated if it is switched off or is not located on the bus. If the PROFIBUS slave/IO device is switched back on or joins the bus, it continues to be treated as activated.
16#00007000	Functionality not active. Initial call required with REQUEST_TRUE
16#00007001	Functionality set up, only for asynchronous execution.
16#00007002	Functionality active, only for asynchronous execution.
16#00007003	Abort job. The job has already been accepted, but its execution could not continue as feedback is expected from the system.
16#FFFF8090	Specified logical base address invalid.
16#FFFF8091	No configuration present for the specified DP station address or device number.
16#FFFF8092	"Activate" requested while "Activate" or "Deactivate" is being executed.
16#FFFF8093	Not a valid DP station
16#FFFF8098	Functionality terminated after timeout.
16#FFFF8099	Internal error, function terminated.
16#FFFF80C1	"Activate" requested while "Deactivate" is being executed.
16#FFFF80C3	Temporary bottleneck: No additional function resource currently available.

Parameters:

logicalAddressOfDpStation

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

The logical diagnostic address of the DP station should be specified for PROFIBUS. The logical diagnostic address of the IO device/I device should be specified for PROFINET.

reqActDeactGetStateMode

Direction:	Input parameter
Data type:	EnumReqActDeactGetStateMode
Parameter index:	2

EnumReqActDeactGetStateMode

REQUEST_TRUE (0)	Start functionality
REQUEST_FALSE (1)	Query status with REQUEST_TRUE
REQUEST_ABORT (2)	Abort of the function execution

REQUEST_TRUE starts the functionality. REQUEST_FALSE scans the status of the functionality started with REQUEST_TRUE. REQUEST_ABORT aborts the function execution.

dpAlarmMode

Direction:	Input parameter
Data type:	EnumDeviceDpAlarmMode
Parameter index:	3

EnumDeviceDpAlarmMode

SET_DP_ALARM (0)	Set up user interface
DO_NOT_SET_DP_ALARM (1)	Suppress user interface call

Specifies the alarm mode. SET_DP_ALARM generates the standard DP alarms (e.g. station failure) that are triggered by the function call. DO_NOT_SET_DP_ALARM suppresses the standard DP alarms that are triggered by the function call.

Note

The alarm is only suppressed if it occurs within the time set in parameter 'timeToWaitForStationAlarm'.

Alarms that occur later are not suppressed.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	4

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and the function cannot be completed.

With WHEN_COMMAND_DONE, the transition takes place when the command is finished. With ABORT_CURRENT_COMMAND and the same 'commandId' as in the previous function call with IMMEDIATELY, the active function is aborted.

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	5
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of the logical diagnostic address of the distributed station or IO device or I device. For INPUT, the logical diagnostic address of the distributed station or IO device or I device is in the input range. For OUTPUT, the logical diagnostic address of the distributed station or IO device or I device is in the output range. Diagnostic addresses are always of type INPUT.

timeToWaitForStationAlarm (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	6
System default:	60

The system function waits for the specified time (in seconds) for the station recovery alarm to arrive. After this time, the internal processing is completed. The system function signals "16#00000005"; i.e. the station is "activated".

Behavior when the 'timeToWaitForStationAlarm' input parameter (wait time) is set to the following values:

- T#0s: The system function does not wait for a station recovery alarm.
- T#60s (default): Behavior as in version < V4.4; the system waits for the station recovery alarm for up to 60 seconds.
- > T#0s: The system function waits for the station recovery alarm for up to the specified wait time.
- The version V4.3 value is applied as the system default.

2.5.4.3 _activateDpSlaveAddress

This system function activates all DP slave addresses set beforehand.

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Return value: DINT

Description of the return values:

16#FFFF8090	Invalid address Outside of valid range / invalid DP segment adapter
16#FFFF8099	Internal error, function terminated.
16#FFFF80C1	Resources for executing the functionality are not available.
16#FFFFFF51	This function is supported in the simulation.

Parameters:

logicalAddressCommunicationAdapter

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

Specifies the logical diagnostic address of the interface.

2.5.4.4 _activateNameOfStation

The function activates all names previously set for the interfaces with the system function _setNameOfStation. Internally, SIMOTION changes to STOP and then triggers a restart.

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Return value: DINT

The return value returns the job status and name.

Description of the return values:

16#FFFF8090	The specified logical diagnostic address is invalid.
16#FFFF8091	No valid logical diagnostic address for a PROFINET interface.
16#FFFF8092	PROFINET interface is configured, but not available. Can only occur if the PROFINET interface is inserted in SIMOTION via a module.
16#FFFF8095	The SIMOTION restart necessary for activation of NameOfStation did not take place.
16#FFFF8096	A device name was not set with _setNameOfStation before this system function was called.
16#FFFF8097	The job was not changed in the STOP operating mode.
16#FFFF8099	Internal error. Function terminated.
16#FFFF80C1	There are currently no resources available to execute this system function. The user program can continue to call this system function until this return value no longer appears.
16#FFFF80CF	A system function call for the relevant PROFINET interface is already in progress. The user program can call this function until this return value no longer appears.

Parameters:

logicalAddressPnInterface

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

Specifies logical base address of PNIO interface (diagnostic address).

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	2
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

The iold parameter indicates whether the logical base address of the Profinet interface is an input address or an output address.

2.5.4.5 _activateTo

This system function enables activation of technology objects (TO). The functions of a technology object (e.g. _enableAxis on the TO axis) cannot be used until the object is ready for operation.

Note

The functions _activateTo, _deactivateTo and _getStateOfTo belong to a function pool. Maximum 50 active calls from this pool can be processed simultaneously.

The return value '16#FFFF80C3' is signaled when the maximum number is exceeded. Each running instance requires additional computation time in the technology object environment (in the cyclic levels); therefore it is more efficient to first query the respective state and then to set the required desired state by means of _activateTo and _deactivateTo.

Additional references

More detailed information on this system function can be found in

- the Function Manual titled "Motion Control Basic Functions for Modular Machines".

Return value: DINT

Description of the return values:

The required actions are asynchronous. The system function signals the status of the execution in the return code.

16#00000000	The technology object is activated.
16#00007000	The system function is in 'Idle' status.
16#00007001	The system function is started (only reported when execution is asynchronous).
16#00007002	The system function is running (only reported when execution is asynchronous, a new call in the control program is required until the functionality ends).
16#FFFF8090	Technology object not available.
16#FFFF8092	'"Activate TO" system function requested while "Deactivate TO" system function is running.
16#FFFF8098	Functionality terminated with timeout.
16#FFFF8099	Internal error, function terminated.
16#FFFF80C1	'Activate TO' is already started due to a new call of the system function with call parameter 'requestMode:=REQUEST_TRUE'.
16#FFFF80C3	Temporary bottleneck (internal resources for executing the function not currently available). The maximum number of 50 calls active simultaneously has been exceeded. Wait until an active call is completed.

Parameters:

TO_Instance

Direction:	Input parameter
Data type:	AnyObject
Parameter index:	1

Reference to the technology object to be activated.

reqActDeactGetStateMode

Direction:	Input parameter
Data type:	EnumReqActDeactGetStateMode
Parameter index:	2

EnumReqActDeactGetStateMode

REQUEST_TRUE (0)	Start functionality
REQUEST_FALSE (1)	Query status with REQUEST_TRUE
REQUEST_ABORT (2)	Abort of the function execution

REQUEST_TRUE starts the functionality. REQUEST_FALSE scans the status of the functionality started with REQUEST_TRUE. REQUEST_ABORT aborts the function execution.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	3

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	5

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and the function cannot be completed.

With WHEN_COMMAND_DONE, the transition takes place when the command has been executed. With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

2.5.4.6 _assignNameOfStationToDevice

The function block assigns the PROFINET device name 'nameOfStation' to an IO device with the MAC address 'MAC'.

Before setting the name, the function implicitly resets the communication parameters of the IO device in order to set the IO device into a defined initial state for the communication if it was previously in operation in another plant, for example. Whereby, a permanent IP address is deleted. NameOfStation is always set permanently. A temporary setting is not supported in contrast to the system function _setPnNameOfStation.

Additional references

Further information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Parameters:

EXECUTE

Direction:	Input parameter
Data type:	BOOL
Parameter index:	1

Starts a new job with a rising edge.

Note

If the input 'EXECUTE' is set back to 'FALSE' after starting a job, the output signals ERROR, ERRORID, DONE and LEN are pending for exactly one cycle.

ioId

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	2

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Specifies the I/O direction.

logicalAddressPnInterface

Direction:	Input parameter
Data type:	DINT
Parameter index:	3

PROFINET interface of the SIMOTION module. The interface must be configured for the IO controller operating mode.

MAC

Direction:	Input parameter
Data type:	STRING
Parameter index:	4

MAC address (as string) of the device whose name is to be set.

NameOfStation

Direction:	Input parameter
Data type:	STRING
Parameter index:	5

Name of the device to be set.

DONE

Direction:	Output parameter
Data type:	BOOL
Parameter index:	6
System default:	FALSE

Message that the block was completed correctly.

BUSY

Direction:	Output parameter
Data type:	BOOL
Parameter index:	7
System default:	FALSE

Message that the block is being processed.

ERROR

Direction:	Output parameter
Data type:	BOOL
Parameter index:	8
System default:	FALSE

Message that the block was executed with errors.

ERRORID

Direction:	Output parameter
Data type:	DWORD
Parameter index:	9
System default:	0

Description of the error messages:

In an error situation, the documentation for the associated module should be consulted for the specific error analysis.

16#00000000	Error-free - job performed successfully
16#FFFF8009	Internal error
16#FFFF800A	The specified logical diagnostic address is invalid.
16#FFFF800B	PROFINET interface is configured, but not available.
16#FFFF800C	PROFINET interface is not configured as IO controller.
16#FFFF800D	The STRING specified in the 'MAC' parameter does not correspond to the format specification for the MAC address.
16#FFFF800E	The STRING specified in the 'nameOfStation'' parameter is too long.
16#FFFF800F	The contents of the STRING specified in the 'NameOfStation' parameter do not comply with the conventions that apply for a 'NameOfStation'.
16#FFFF8010	A system function call for the relevant PROFINET interface is already in progress. The user program can call this function until this return value no longer appears.
16#FFFF8011	Job was rejected by the IO device.
16#FFFF8012	There are currently no resources available to execute this system function. The user program can call this function until this return value no longer appears.

2.5.4.7 _deactivateDpSlave

This system function aborts cyclical communication for the entire DP station (slave) or the PROFINET station (device), even if the station has multiple logical addresses (diagnostic addresses).

This system function cannot be called when the _activateDpSlave system function is active with the same 'logicalAddressOfDpStation'. This system function applies to both PROFIBUS and PROFINET.

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- SIMOTION Communication System Manual
- or in the online help

Return value: DINT

The return value reports the job status.

Description of the return values:

16#00000000	Abort job. No jobs are now being executed for the logical address.
16#00000002	PROFIBUS slave/IO device deactivated. Job completed with positive result. Job okay.
16#00000006	PROFIBUS slave/IO device is deactivated. The system function has not waited for a 'station failure' alarm. Job completed with positive result. Job okay. A PROFIBUS slave/IO device can also be deactivated if it is switched off or is not located on the bus. If the PROFIBUS slave/IO device is switched back on or joins the bus, it continues to be treated as deactivated.
16#00007000	Functionality not active.
16#00007001	Functionality initiated, only for asynchronous execution.
16#00007002	Functionality active, only for asynchronous execution.
16#00007003	Abort job. The job has already been accepted, but its execution could not continue as feedback is expected from the system.
16#FFFF8090	Specified logical base address invalid.
16#FFFF8091	No configuration present for the specified DP station address or device number.
16#FFFF8092	"Deactivate" requested while "Activate" or "Deactivate" is being executed.
16#FFFF8093	Not a valid DP station
16#FFFF8098	Functionality of system function has been terminated with timeout.
16#FFFF8099	Internal error, function terminated.
16#FFFF80C1	"Activate" requested while "Deactivate" is being executed.
16#FFFF80C3	Temporary bottleneck: No additional function resource currently available.

Parameters:

logicalAddressOfDpStation

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

The logical diagnostic address of the DP station should be specified for PROFIBUS. The logical diagnostic address of the IO device/I device should be specified for PROFINET.

reqActDeactGetStateMode

Direction:	Input parameter
Data type:	EnumReqActDeactGetStateMode
Parameter index:	2

EnumReqActDeactGetStateMode

REQUEST_TRUE (0)	Start functionality
REQUEST_FALSE (1)	Query status with REQUEST_TRUE
REQUEST_ABORT (2)	Abort of the function execution

REQUEST_TRUE starts the functionality. REQUEST_FALSE scans the status of the functionality started with REQUEST_TRUE. REQUEST_ABORT aborts the function execution.

dpAlarmMode

Direction:	Input parameter
Data type:	EnumDeviceDpAlarmMode
Parameter index:	3

EnumDeviceDpAlarmMode

SET_DP_ALARM (0)	Set up user interface
DO_NOT_SET_DP_ALARM (1)	Suppress user interface call

Specifies the alarm mode. SET_DP_ALARM generates the standard DP alarms (e.g. station failure) that are triggered by the function call. DO_NOT_SET_DP_ALARM suppresses the standard DP alarms that are triggered by the function call.

Note

The alarm is only suppressed if it occurs within the time set in parameter 'timeToWaitForStationAlarm'.

Alarms that occur later are not suppressed.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	4

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and the function cannot be completed.

With WHEN_COMMAND_DONE, the transition takes place when the command is finished. With ABORT_CURRENT_COMMAND and the same 'commandId' as in the previous function call with IMMEDIATELY, the active function is aborted.

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	5
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of the logical diagnostic address of the distributed station, IO device, or I device. For INPUT, the logical diagnostic address of the distributed station, IO device, or I device is in the input range. For OUTPUT, the logical diagnostic address of the distributed station, IO device, or I device is in the output range. Diagnostic addresses can be of type INPUT or OUTPUT.

timeToWaitForStationAlarm (optional)

Direction:	Input parameter
Data type:	UDINT
Parameter index:	6
System default:	0

The system function waits for this time (in seconds) for the station failure alarm to arrive. After this time, the internal processing is completed. The system function reads the activation state and signals "16#00000006"; i.e. the station is "deactivated".
Behavior when the timeToWaitForStationAlarm input parameter (wait time) is set to the following values:
0: The system function does not wait for a station restoration alarm.
60: (default): Behavior as in version < V4.4; the system waits for the station failure alarm for up to 60 seconds.
> 0: The system function waits for the station failure alarm for up to the specified wait time in seconds.
The Version V4.3 value applies by default.

2.5.4.8 _deactivateTo

This system function deactivates technology objects (TO).

You must also ensure that all technology objects that are dependent on the deactivated technology object are also deactivated.

For example:

A slave axis always requires a synchronous object. Therefore both technology objects must be deactivated.

The deactivation sequence should always start with the technology objects that are not dependent on other technology objects.

Note

The functions _activateTo, _deactivateTo and _getStateOfTo belong to a function pool. Maximum 50 active calls from this pool can be processed simultaneously.
The return value '16#FFFF80C3' is signaled when the maximum number is exceeded. Each running instance requires additional computation time in the technology object environment (in the cyclic levels); therefore it is more efficient to first query the respective state and then to set the required desired state by means of _activateTo and _deactivateTo.

Additional references

More detailed information on this system function can be found in

- the Function Manual titled "Motion Control Basic Functions for Modular Machines".

Return value: DINT

Description of the return values:

The required actions are asynchronous. The system function signals the status of the execution in the return code.

16#00000000	The technology object (TO) is deactivated.
16#00007000	The system function is in 'Idle' status (inactive).
16#00007001	The system function is started (only reported in asynchronous execution).
16#00007002	The system function is running (only reported when execution is asynchronous, a new call in the control program is required until the functionality ends).
16#FFFF8090	Invalid technology object reference.
16#FFFF8092	"Deactivate TO" requested while "Activate TO" is running.
16#FFFF8098	Functionality terminated with timeout.
16#FFFF8099	Internal error, function terminated.
16#FFFF80C1	'Deactivate TO' is already started. due to a new call of the system function with call parameter 'requestMode:=REQUEST_TRUE'.
16#FFFF80C3	Temporary bottleneck (internal resources for executing the function not currently available). The maximum number of 50 calls active simultaneously has been exceeded. Wait until an active call is completed.

Parameters:

TO_Instance

Direction:	Input parameter
Data type:	AnyObject
Parameter index:	1

Reference to the technology object to be deactivated.

reqActDeactGetStateMode

Direction:	Input parameter
Data type:	EnumReqActDeactGetStateMode
Parameter index:	2

EnumReqActDeactGetStateMode

REQUEST_TRUE (0)	Start functionality
REQUEST_FALSE (1)	Query status with REQUEST_TRUE
REQUEST_ABORT (2)	Abort of the function execution

REQUEST_TRUE starts the functionality. REQUEST_FALSE scans the status of the functionality started with REQUEST_TRUE. REQUEST_ABORT aborts the function execution.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	3

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	5

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and the function cannot be completed.

. With WHEN_COMMAND_DONE, the transition takes place when the command has been executed. With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

2.5.4.9 _getActiveDpSlaveAddress

This system function reads the active communication address of the specified interface.

Additional references

More detailed information on this system function can be found in

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the SIMOTION online help

Return value: StructRetDpSlaveAddress

StructRetDpSlaveAddress
Display structure of the system function _getActiveDpSlaveAddress

Structure	Name	Data type	Unit
functionResult	Result of the function execution	DINT	-
dpSlaveAddress	DP slave address	SINT	-

Description of the return values:

Return value "functionResult" has the following meaning:

16#00000000	Function terminated successfully.
16#FFFF8090	Invalid address, outside of valid range/invalid DP segment adapter.
16#FFFF8091	Invalid DP slave address.
16#FFFF80C1	Resources for implementing the functionality are occupied.

Parameters:

logicalAddressCommunicationAdapter

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

Specifies the logical diagnostic address of the interface.

2.5.4.10 _getActiveNameOfStation

Note

Applies from SIMOTION version V4.4

Use the '_getPnNameOfStation' system function instead of '_getActiveNameOfStation' when you create a new SIMOTION project. This system function supplies the effective 'NameOfStation' name of the PROFINET interface. This originates from the configuration or was set with the DCP service using 'TcpIp' or by means of the _setNameOfStation system function.

The system function reads the currently active NameOfStation. This originates from the configuration or was set with the DCP service using 'TcpIp' or by means of the _setNameOfStation system function.

A maximum of 239 characters is returned. According to the standard, 240 characters are permitted, but this is not currently supported by SIMOTION.

If NameOfStation is not available, a string with a length of 0 is returned. In pnPortNumber, 16#FFFFFFFF is always returned. In pnSlotNumber, 16#FFFFFFFF is always returned.

Additional references

More detailed information on this system function can be found in

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Return value: StructRetDeviceNameOfStation

StructRetDeviceNameOfStation
Return values for _getPnInterfacePortNeighbour and _getActiveNameOfStation

Structure	Name	Data type	Unit
functionResult	Result of the execution of the system function	DINT	-
nameOfStation	nameOfStation of own device (_getActivNameOfStation) or neighboring device (_getPnInterfacePortNeighbour)	STRING	-
pnPortNumber	pnPortNumber of the neighboring device, with which this is connected to the local device.	UDINT	-
pnSlotNumber	pnSlotNumber of the neighboring device, with which this is connected to the local device.	UDINT	-

Description of the return values:

functionResult

The return value returns the job status and name.

16#00000000	System function completed successfully. NameOfStation is contained in the return value.
16#00007000	The system function is in inactive status. The functionality associated with the system function is not active. An initial call with REQUEST_TRUE is required.
16#00007001	The system function is started with REQUEST_TRUE. A new call in the user program until completion of the system function is required with REQUEST_FALSE. This return value is reported only for asynchronous execution.
16#00007002	The system function is still running. A new call in the user program until completion of the system function is required with REQUEST_FALSE. This return value is reported only for asynchronous execution.
16#00007003	System function aborted successfully. If an abort is not possible, the system function returns 16#FFFF8094.
16#FFFF8090	The specified logical diagnostic address is invalid.
16#FFFF8091	No valid logical diagnostic address for a PROFINET interface or a port of a PROFINET interface.
16#FFFF8092	PROFINET interface is configured, but not available. Can only occur if the PROFINET interface is inserted in SIMOTION via a module.
16#FFFF8094	An abort is no longer possible in the current system function status.
16#FFFF8099	Internal error. Function terminated.
16#FFFF809A	The PROFINET interface does not support this service. Function terminated.
16#FFFF80A0	The "Obtain device names in another way" option is deselected in HW Config. Function terminated.
16#FFFF80C1	There are currently no resources available to perform this system function. The user program can continue to call this system function until this return value no longer appears.
16#FFFF80CF	A system function call for the relevant PROFINET interface is already in progress. The user program can continue to call this system function until this return value no longer appears.

nameOfStation

Device name of relevant PROFINET interface. A maximum of 239 characters is returned. According to the standard, 240 characters are permitted, but this is not currently supported by SIMOTION.

If NameOfStation is not available, a string with a length of 0 is returned.

pnPortNumber

16#FFFFFFFF is always returned.

pnSlotNumber

16#FFFFFFFF is always returned.

Parameters:

logicalAddressPnInterface

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

Specifies the logical diagnostic address of the PROFINET interface, or the logical diagnostic addresses of the associated ports on the PROFINET interface, from which the active device name is to be read.

requestMode

Direction:	Input parameter
Data type:	EnumReqSysFunctMode
Parameter index:	2

EnumReqSysFunctMode

REQUEST_TRUE (0)	Start functionality
REQUEST_FALSE (1)	Status query of functionality started with REQUEST_TRUE
REQUEST_ABORT (2)	Abort of the function execution

REQUEST_TRUE starts the functionality. REQUEST_FALSE scans the status of the functionality started with REQUEST_TRUE. REQUEST_ABORT aborts the function execution.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	3

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

This parameter specifies a 'CommandId' of type 'commandIdType' that is unique from all others in the system. The user can make the system generate a 'syncCommandId' by calling _getSyncCommandId. The same 'syncCommandId' is used for subsequent calls.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	5

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and the function cannot be completed.

With WHEN_COMMAND_DONE, the transition takes place when the command is finished. With ABORT_CURRENT_COMMAND and the same 'commandId' as in the previous function call with IMMEDIATELY, the active function is aborted.

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	6
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

The ioId parameter indicates whether logicalAddressPnInterface of the Profinet interface is an input address or an output address.

2.5.4.11 _getConfigurationData

This system function supplies data to the user program for activation of configurations.

Additional references

More detailed information on this system function can be found in

- the Function Manual titled "Motion Control Basic Functions for Modular Machines".

Return value: StructRetGetConfigurationData

EnumDeviceConfigurationActivationState

NOT_ACTIVATED (0)	Not started yet
CONFIGURATION_NOT_FOUND (101)	Configuration not found
CONFIGURATION_DELETED (102)	Configuration deleted
CONFIGURATION_ERROR (103)	Configuration error
WAITING_FOR_RESTART (104)	Waiting for SIMOTION restart
STOP (105)	STOP operating mode reached
TIME_OUT (106)	Abort after timeout
RUN (107)	New configuration active
ACTIVATE_FALL_BACK_CONFIGURATION (108)	Emergency configuration will be activated

EnumDeviceDataActivationState

NOT_ACTIVATED (0)	Configuration not activated
ACTIVATED (1)	The configuration is effective
NO_REQUEST_TO_ACTIVATE (2)	Desired configuration is not designated as the configuration to be activated
NO_VALID_CONFIGURATION_ID (3)	The configuration identification is outside of the permissible limits
CONFIGURATION_ID_NOT_FOUND (4)	Configuration identification not available on memory card
NOT_ENOUGH_RAM (5)	Insufficient RAM memory to take the configuration
RAM_DISK_FULL (6)	RAM disk is full
MEMORY_CARD_FULL (7)	Memory card is full
TIME_OUT (8)	Monitoring time has expired
INTERNAL_ERROR (9)	Internal error
ACCESS_DENIED (10)	Access denied, e.g. if access protection active

EnumYesNo

NO (91)	No
YES (173)	Yes

StructRetGetConfigurationData
Return parameter for _getConfigurationData

Structure	Name	Data type	Unit
functionResult	Total job return value	UDINT	-
configuration	Information for the activation of the configurations	StructDeviceConfigurationData	-

StructDeviceConfigurationData
Data for the configurations to be activated

Structure	Name	Data type	Unit
activationState	Current state of the activation	EnumDeviceConfigurationActivationState	-
projectDataState	Current state of the project data activation	EnumDeviceDataActivationState	-
projectDataRequest	Status of the activation request	EnumYesNo	-
projectDataId	Identification of the project data	UDINT	-
technologicalPackageState	Current state of the technological package activation	EnumDeviceDataActivationState	-
technologicalPackageRequest	Status of the activation request	EnumYesNo	-
technologicalPackageId	Identification of technological package	UDINT	-
driveDataState	Current state of the activation	EnumDeviceDataActivationState	-
driveDataRequest	Status of the activation request	EnumYesNo	-
driveDataId	Identification of the drive data	UDINT	-
kernelState	Current state of the kernel version activation	EnumDeviceDataActivationState	-
kernelRequest	Status of the activation request	EnumYesNo	-
kernelId	Identification of the kernel version	UDINT	-
timeOut	Timeout	TIME	-

With ACTIVATED, the selected configuration is active. With NO_REQUEST_TO_ACTIVATE, the desired configuration is not identified as the configuration to be activated. When _activateConfiguration is called, the 'projectData', 'technologicalPackage', and/or firmware parameters are set to NO. With NOT_FOUND, the configuration is not found. Check to determine whether the memory card contains the configuration.

Description of the return values:

Return value "functionResult" has the following meaning:

16#00000000	The system function is finished - result is in 'commandIdState'.
16#00007000	The system function is in 'Idle' status.
16#00007001	The system function is started (only reported when execution is asynchronous).
16#00007002	The system function is running (only reported when execution is asynchronous, a new call in the control program is required until the functionality ends).
16#FFFF8094	The functionality has already been started with the same 'commandId'. The new request is rejected.
16#FFFF8098	Functionality terminated after timeout. Function not executed, all resources are released.
16#FFFF8099	Internal error, function terminated. Function not executed, all resources are released.

Parameters:

configurationInfoId

Direction:	Input parameter
Data type:	EnumConfigurationInfoId
Parameter index:	1

EnumConfigurationInfoId

ACTUAL_ACTIVATED (0)	Status of the currently effective configuration
PREVIOUS_ACTIVATED (1)	Status of the previously effective configuration
CONFIGURATION_VERSION (2)	Status of the currently effective version

Specifies the status of the configuration. Specifies the data to be read: current, previous, or emergency configuration.

ACTUAL_ACTIVATED displays the status of the currently active configuration. PREVIOUS_ACTIVATED displays the status of the previous configuration, e.g. in order to investigate the reason for an error. FALL_BACK enables the set emergency configuration to be read.

requestMode

Direction:	Input parameter
Data type:	EnumReqSysFunctMode
Parameter index:	2

EnumReqSysFunctMode

REQUEST_TRUE (0)	Start functionality
REQUEST_FALSE (1)	Status query of functionality started with REQUEST_TRUE
REQUEST_ABORT (2)	Abort of the function execution

REQUEST_TRUE starts the functionality. REQUEST_FALSE scans the status of the functionality started with REQUEST_TRUE. REQUEST_ABORT aborts the function execution.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	3

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	5

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and the function cannot be completed.

With WHEN_COMMAND_DONE, the transition takes place when the command has been executed. With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

2.5.4.12 _getIPConfig

This function determines the IP configuration of a selected Ethernet interface.

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Return value: StructRetIPConfig

StructRetIPConfig
Return parameter for _getIPConfig

Structure	Name	Data type	Unit
functionResult	Result	UDINT	-
ipAddress	Current IP address of the Ethernet interface	ARRAY [0..5] OF USINT	-
subnetMask	Current subnet screen of the Ethernet interface	ARRAY [0..5] OF USINT	-
gatewayAddress	Address of the default router in the network	ARRAY [0..5] OF USINT	-

Result, IP address, subnet screen form, and address of the default router

Description of the return values:

Return value "functionResult" has the following meaning:

16#00000000 - IP_CONF_RLST_READY The function has been completed successfully. The IP configuration is available without error.

16#FFFF8090 - IP_CONF_WRONG_PARAM The function has been called with incorrect parameters.

16#FFFF8099 - IP_CONF_INTERNAL_ERROR An internal error has occurred while executing the function.

16#FFFF80C1 - IP_CONF_NO_RESSOURCES No resources are available to supply the IP configuration.

16#FFFF80C3 - IP_CONF_NOT_AVAILABLE The function has been called for a module that does not have an Ethernet interface.

Parameters:

ethernetInterface

Direction:	Input parameter
Data type:	EnumInterfaceID
Parameter index:	1

EnumInterfaceID

IE_01 (0)	Ethernet interface 1: All platforms
IE_02 (1)	Ethernet interface 2: All platforms

Ethernet interface for which the configuration is to be read.

2.5.4.13 _getObjectName

The system function returns the configured name of a technology object (TO) as a string. If the TO name is available as a string, this can be processed directly in the user program. When the function is called in the TechnologicalFaultTask, TSI#toInst can be transferred directly to the ToInst input parameter from the TaskStartInfo of the TechnologicalFaultTask.

Return value: STRING

The function supplies a string as a return value.

Parameters:

ToInst

Direction:	Input parameter
Data type:	AnyObject
Parameter index:	1

Reference to the technology object whose name is to be returned.

2.5.4.14 _getPnInterfacePortNeighbour

Note

Applies from SIMOTION version V4.4

Use the '_getPnPortNeighbour' system function instead of '_getPnInterfacePortNeighbour' when you create a new SIMOTION project. This system function supplies the name of the PROFINET device connected to the port of the PROFINET interface.

This system function reads information from the PN device connected to the port.

If no identifiable neighbor device, or no device at all, is connected to the relevant port, an empty string will be returned in 'nameOfStation' and 16#FFFFFFFF in 'pnPortNumber'.

If a project-wide unique device name has not been assigned to the connected PN device, the hardware address (MAC address) is returned. The MAC address is output in the notation 11:22:33:44:55:66:77.

If an identifiable neighbor device has been found, a string with a length other than 0 will be returned in 'nameOfStation' and a value 0 < x < 255 will be returned in 'pnPortNumber'.

For this system function, only exactly one function call per port is permitted simultaneously.

Additional references

More detailed information on this system function can be found in

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the SIMOTION online help

Return value: StructRetDeviceNameOfStation

StructRetDeviceNameOfStation
Return values for _getPnInterfacePortNeighbour and _getActiveNameOfStation

Structure	Name	Data type	Unit
functionResult	Result of the execution of the system function	DINT	-
nameOfStation	nameOfStation of own device (_getActivNameOfStation) or neighboring device (_getPnInterfacePortNeighbour)	STRING	-
pnPortNumber	pnPortNumber of the neighboring device, with which this is connected to the local device.	UDINT	-
pnSlotNumber	pnSlotNumber of the neighboring device, with which this is connected to the local device.	UDINT	-

The return value returns the job status and name.

Description of the return values:

functionResult

16#00000000	System function completed successfully. The device name and port number of the relevant neighbor device will be returned if it can be identified.
16#00007000	The system function is in inactive status. The functionality of the system function is not active. An initial call with REQUEST_TRUE is required.
16#00007001	The system function is started with REQUEST_TRUE. A new call in the user program until completion of the system function is required with REQUEST_FALSE. This return value is reported only for asynchronous execution.
16#00007002	The system function is still running. A new call in the user program until completion of the system function is required with REQUEST_FALSE. This return value is reported only for asynchronous execution.

16#00007003	System function aborted successfully. If an abort is not possible, the system function returns the value 16#FFFF8094.
16#FFFF8090	The specified logical diagnostic address is invalid.
16#FFFF8091	No valid logical diagnostic address for a PROFINET interface port.
16#FFFF8092	PROFINET interface is configured, but not available. Can only occur if the PROFINET interface is inserted in SIMOTION via a module.
16#FFFF8093	The STRING returned under 'nameOfStation' is too long. Only 235 characters are permitted. According to the standard, 240 characters are permitted, but this is not currently supported by SIMOTION.
16#FFFF8094	An abort is no longer possible in the current system function status.
16#FFFF8099	Internal error. Function terminated.
16#FFFF809A	The PROFINET interface does not support this service. Function terminated.
16#FFFF809D	Several neighbors are connected to the port addressed by 'logicalAddressOfPnPort'. Function terminated.
16#FFFF80C1	There are currently no resources available to execute this system function. The user program can call this function until this return value no longer appears.
16#FFFF80CF	A system function call for the relevant PROFINET interface port is already in progress. The user program can call this function until this return value no longer appears.

nameOfStation

Device name of neighbor device

pnPortNumber

Port number of neighbor device used connect to the local device. If no identifiable neighbor device, or no device at all, is connected to the relevant port,

- an empty string will be returned in 'nameOfStation'
- and 16#FFFFFFFF in 'pnPortNumber'.

If an identifiable neighbor device has been found,

- a string with a length other than 0 will be returned in 'nameOfStation' and
- a value of 0 < x < 255 will be returned in 'pnPortNumber'.

pnSlotNumber

Slot number of neighbor device containing the neighbor port. Only returned if the neighbor device can be identified and contains several slots with ports. If a slot number is not available, 16#FFFFFFFF is returned. For this system function, only exactly one function call per port is permitted simultaneously.

Parameters:

logicalAddressOfPnInterfacePort

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

Specifies the logical diagnostic address of the port used to determine the device name, port number, and slot number of its neighbor.

requestMode

Direction:	Input parameter
Data type:	EnumReqSysFunctMode
Parameter index:	2

EnumReqSysFunctMode

REQUEST_TRUE (0)	Start functionality
REQUEST_FALSE (1)	Status query of functionality started with REQUEST_TRUE
REQUEST_ABORT (2)	Abort of the function execution

REQUEST_TRUE starts the functionality. REQUEST_FALSE scans the status of the functionality started with REQUEST_TRUE. REQUEST_ABORT aborts the function execution.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	3

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND. The same CommandId is used for subsequent calls.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	5

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and the function cannot be completed.

With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	6
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

The ioId parameter indicates whether logicalAddressOfPnInterfacePort of the Profinet interface is an input address or an output address.

2.5.4.15 _getPnIpConfig

This function determines the IP configuration of a selected Ethernet interface.

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Return value: StructRetPnIpConfig

StructRetPnIpConfig
Return parameter for _getPnIpConfig

Structure	Name	Data type	Unit
functionResult	Result	UDINT	-
ipAddress	Current IP address of the Ethernet interface	ARRAY [0..3] OF USINT	-
subnetMask	Current subnet screen of the Ethernet interface	ARRAY [0..3] OF USINT	-
gatewayAddress	Address of the default router in the network	ARRAY [0..3] OF USINT	-

Result, IP address, subnet screen form, and address of the default router

Description of the return values:

functionResult

has the following meaning:

16#00000000	The active IP configuration was read successfully.
16#00007001	First call, the trigger to read the IP configuration OK (only when command is issued asynchronously). Must be repeated in the next program cycle.
16#00007002	Intermediate call, the reading of the IP configuration is still active. Repeat in the next program cycle where necessary until the IP configuration has been read successfully or canceled (only when the command is issued asynchronously).
16#00007003	Reading of IP configuration aborted (only when command is issued asynchronously).
16#FFFF8090	The specified logical diagnostic address is invalid.
16#FFFF8091	No valid logical diagnostic address for a PROFINET interface or a port of a PROFINET interface.
16#FFFF8092	PROFINET interface is configured, but not available. Can only occur if the PROFINET interface is inserted in SIMOTION as a module.
16#FFFF8094	An abort is no longer possible in the current system function status.
16#FFFF8099	An error occurred while the function was being processed.
16#FFFF809A	Setting the device name is not supported by the PROFINET interface.

16#FFFF80A0	In HW Config, the options "Obtain device names in another way" or "Obtain IP address in another way" are deselected for the respective communication interface. Select both of these options for the corresponding communication interface in HW Config. Function terminated.
16#FFFF80FF	There is no configuration information on a PROFINET interface at the logical diagnostic address. To remedy this, please load a project with a configuration for the PROFINET interface into the device.
16#FFFF80C1	There are currently no resources available to execute this system function. The user program can call this function until this return value no longer appears.

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Specifies whether the diagnostic address in the logicalAddressPnInterface parameter is an input or output address.

logicalAddressPnInterface

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

You must specify a valid diagnostic address of a PROFINET interface/port of a PROFINET interface, from which the IP configuration is to be read. Both Ethernet interfaces with PROFINET basic services (PN-IE) and PN-IO interfaces of a SIMOTION device can be addressed via this address.

nextCommand (optional)

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	3
System default:	IMMEDIATELY

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). IMMEDIATELY means: The functionality is executed asynchronously. With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	4

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND. The same CommandId is used for subsequent calls.

2.5.4.16 _getPnNameOfStation

Note

Applies from SIMOTION version V4.4

Use the '_getPnNameOfStation' system function instead of '_getActiveNameOfStation' when you create a new SIMOTION project. This system function supplies the effective 'NameOfStation' name of the PROFINET interface. This originates from the configuration or was set with the DCP service using 'TcpIp' or by means of the _setNameOfStation system function.

If NameOfStation is not available, a string with a length of 0 is returned. In 'pnPortNumber', 16#FFFFFFFF is then returned. In 'pnSlotNumber', 16#FFFFFFFF is then returned.

Additional references

More detailed information on this system function can be found in

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Return value: StructRetDeviceNameOfStation

StructRetDeviceNameOfStation
Return values for _getPnInterfacePortNeighbour and _getActiveNameOfStation

Structure	Name	Data type	Unit
functionResult	Result of the execution of the system function	DINT	-
nameOfStation	nameOfStation of own device (_getActivNameOfStation) or neighboring device (_getPnInterfacePortNeighbour)	STRING	-
pnPortNumber	pnPortNumber of the neighboring device, with which this is connected to the local device.	UDINT	-
pnSlotNumber	pnSlotNumber of the neighboring device, with which this is connected to the local device.	UDINT	-

Description of the return values:

functionResult

The return value returns the job status and name.

16#00000000	System function completed successfully. The device name was read successfully and is available in the return value.
16#00007001	Initial call, the trigger to read the device name is still active. Repetition necessary in the next program cycle until reading has been completed successfully or aborted (only when command is issued asynchronously).
16#00007002	Intermediate call, the reading of the device name is still active. Repetition necessary in the next program cycle until the reading of the device name has been completed successfully or aborted (only when command is issued asynchronously).
16#00007003	Reading the device name aborted. If an abort is not possible, the system function returns 16#FFFF8094.
16#FFFF8090	The specified logical diagnostic address is invalid.
16#FFFF8091	No valid logical diagnostic address for an interface or port of a PROFINET interface.
16#FFFF8092	PROFINET interface is configured, but not available. Can only occur if the PROFINET interface is inserted in SIMOTION via a module.
16#FFFF8094	An abort is no longer possible in the current system function status.
16#FFFF8099	An error occurred while the function was being processed.
16#FFFF809A	Readout of device name is not supported by the PROFINET interface.
16#FFFF80FF	There is no configuration information on a PROFINET interface at the logical diagnostic address. To remedy this, please load a project with a configuration for the PROFINET interface into the device.
16#FFFF80C1	There are currently no resources available to perform this system function. The user program can continue to call this system function until this return value no longer appears.
16#FFFF80C2	A system function call for the relevant PROFINET interface is already in progress. The user program can call this function until this return value no longer appears.

nameOfStation

Device name of relevant PROFINET interface.

If NameOfStation is not available, a string with a length of 0 is returned.

pnPortNumber

16#FFFFFFFF is always returned.

pnSlotNumber

16#FFFFFFFF is always returned.

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

The iold parameter indicates whether logicalAddressPnInterface of the Profinet interface is an input address or an output address.

logicalAddressPnInterface

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical diagnostic address of the PROFINET interface, or the logical diagnostic addresses of the associated ports on the PROFINET interface, from which the active device name is to be read.

nextCommand (optional)

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	3
System default:	IMMEDIATELY

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. With IMMEDIATELY, an initial call starts the asynchronous processing of the system function. The system function is to be called until it reports that it has been completed (OK or error indications).

With WHEN_COMMAND_DONE, the transition takes place when the command is finished. With ABORT_CURRENT_COMMAND and the same 'commandId' as in the previous function call with IMMEDIATELY, the active function is aborted.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	4

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

This parameter specifies a 'CommandId' of type 'commandIdType' that is unique from all others in the system. The user can make the system generate a 'syncCommandId' by calling _getSyncCommandId. The same 'syncCommandId' is used for subsequent calls.

2.5.4.17 _getPnPortNeighbour

Note

Applies from SIMOTION version V4.4

Use the '_getPnPortNeighbour' system function instead of '_getPnInterfacePortNeighbour' when you create a new SIMOTION project. This system function supplies the name of the PROFINET device connected to the port of the PROFINET interface.

If no identifiable neighbor device, or no device at all, is connected to the relevant port, an empty string will be returned in 'nameOfStation' and the value 16#FFFFFFFF in 'pnPortNumber' and 'pnSlotNumber'.

If an identifiable neighbor device has been found, a string with a length other than 0 will be returned in 'nameOfStation' and a value 0 < x < 255 will be returned in 'pnPortNumber'.

For this system function, only exactly one function call per port is permitted simultaneously.

Additional references

More detailed information on this system function can be found in

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the SIMOTION online help

Return value: StructRetDeviceNameOfStation

StructRetDeviceNameOfStation
Return values for _getPnInterfacePortNeighbour and _getActiveNameOfStation

Structure	Name	Data type	Unit
functionResult	Result of the execution of the system function	DINT	-
nameOfStation	nameOfStation of own device (_getActivNameOfStation) or neighboring device (_getPnInterfacePortNeighbour)	STRING	-
pnPortNumber	pnPortNumber of the neighboring device, with which this is connected to the local device.	UDINT	-
pnSlotNumber	pnSlotNumber of the neighboring device, with which this is connected to the local device.	UDINT	-

The return value returns the job status and name.

Description of the return values:

functionResult

16#00000000	System function completed successfully. The device name and port number of the relevant neighbor device will be returned if it can be identified.
16#00007001	Initial call, the trigger to read out the device name is still active. Repetition necessary in the next program cycle (only when command is issued asynchronously).
16#00007002	Intermediate call, the readout of the neighboring device is still running. A new call in the user program is required until completion of the system function. This return value is reported only for asynchronous execution.
16#00007003	System function aborted successfully. If an abort is not possible, the system function returns the value 16#FFFF8094.
16#FFFF8090	The specified logical diagnostic address is invalid.
16#FFFF8091	No valid logical diagnostic address for a PROFINET interface port.
16#FFFF8092	PROFINET interface is configured, but not available. Can only occur if the PROFINET interface is inserted in SIMOTION via a module.
16#FFFF8093	The STRING returned under 'nameOfStation' is too long. According to the standard, only 240 characters are permitted.
16#FFFF8094	An abort is no longer possible in the current system function status.
16#FFFF8098	The device name returned under 'pnNameOfStation' is invalid, e.g. illegal characters.
16#FFFF8099	Internal error. Function terminated.
16#FFFF809A	The PROFINET interface does not support this service. Function terminated.
16#FFFF809B	The system function cannot be aborted. Function terminated.
16#FFFF809C	There is an unidentifiable neighboring device which is returning invalid data for the slot number, port number, and/or the device name. Function terminated.
16#FFFF809D	Several neighbors are connected to the port addressed by 'logicalAddressOfPnPort'. Function terminated.
16#FFFF80A2	No port neighbor was found on the port addressed by 'logicalAddressOfPnPort'. Function terminated.
16#FFFF80A4	A port neighbor was found on the port addressed by 'logicalAddressOfPnPort', which has an empty or invalid name. Function terminated.
16#FFFF80FF	There is no configuration information on a PROFINET interface at the logical diagnostic address. To remedy this, load a project with a configuration for the PROFINET interface into the device.
16#FFFF80C1	There are currently no resources available to execute this system function. The user program can call this function until this return value no longer appears.
16#FFFF80C2	An instance of the system function with identical call parameters is already running. The user program can call this function until this return value no longer appears.

nameOfStation

Device name of neighbor device

pnPortNumber

Port number of neighbor device used connect to the local device. If no identifiable neighbor device, or no device at all, is connected to the relevant port,

- an empty string will be returned in 'nameOfStation'
- and 16#FFFFFFFF in 'pnPortNumber'.

If an identifiable neighbor device has been found,

- a string with a length other than 0 will be returned in 'nameOfStation' and
- a value of 0 < x < 255 will be returned in 'pnPortNumber'.

pnSlotNumber

Slot number of neighbor device containing the neighbor port. Only returned if the neighbor device can be identified and contains several slots with ports. If a slot number is not available, 16#FFFFFFFF is returned. For this system function, only exactly one function call per port is permitted simultaneously.

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

The 'ioId' parameter indicates whether 'logicalAddressOfPnInterfacePort' of the PROFINET interface is an input address or an output address.

logicalAddressOfPnPort

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical diagnostic address of the port used to determine the device name, port number, and slot number of its neighbor.

nextCommand (optional)

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	3
System default:	IMMEDIATELY

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. With IMMEDIATELY, an initial call starts the asynchronous processing of the system function. The system function is to be called until it reports that it has been completed (OK or error indications).

With WHEN_COMMAND_DONE, the transition takes place when the command is finished. With ABORT_CURRENT_COMMAND and the same 'commandId' as in the previous function call with IMMEDIATELY, the active function is aborted.

With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	4

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND. The same CommandId is used for subsequent calls.

2.5.4.18 _getStateOfAllDpSlaves

This function returns a list of status data for all existing DP slaves. This function only provides information for DP slaves with PROFIBUS DP.

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Return value: StructRetGetStateOfAllDpSlaves

EnumStateOfDpSlave

INACTIVE (0)	Slave/IO device deactivated by program
IN_OPERATION (1)	Input/output data exchange with slave/IO device
DATA_EXCHANGE_INACTIVE (2)	No input/output data exchange with slave/IO device
NOT_PRESENT (3)	Slave/IO device is not available (no supply voltage, no connection to bus, etc.)

StructRetGetStateOfAllDpSlaves
Return parameter for _getStateOfAllDPSlaves

Structure	Name	Data type	Unit
functionResult	Total job return value	DINT	-
ActualStateOfAllDpSlaves	Status data of the available DP slaves	ARRAY [0..127] OF EnumStateOfDpSlave	-

Description of the return values:

16#00000000	Okay - system function completed successfully
16#00007000	Functionality not active
16#00007001	Functionality initiated
16#00007002	Functionality active
16#FFFF8090	No valid logical address for DP master interface
16#FFFF8091	Not a valid DP segment
16#FFFF8092	A new call of the function with these parameters is not permitted
16#FFFF8093	Not a valid DP station
16#FFFF8098	Functionality terminated with timeout
16#FFFF8099	Internal error, function terminated
16#FFFF80C1	Function already started, new call with parameter 'ReqGetStateOfAllDPSlavesStateMode=TRUE'
16#FFFF80C3	Temporary resource bottleneck
16#FFFF80CF	CommandId not unique
16#FFFF809E	Abort of function not allowed
16#FFFF809D	Abort successful

Describes the state of the requested DP slave:

INACTIVE means the slave is deactivated by the program. IN_OPERATION means input/output data exchange takes place via the slave.

DATA_EXCHANGE_INACTIVE means the station is disturbed (the head-end of the station or one of its modules is disturbed).

Possible causes:

- Fault in station head-end
  - No data exchange with station head-end or modules in the station is possible.
- Fault on a module in the station
  - Data exchange with station head-end or modules in the station is possible. One or more diagnostics of station head-end or modules in the station are pending. The user program can receive additional information about the causes by reading the diagnostics of the station head-end or the individual modules.

NOT_PRESENT means the slave is not available (no supply voltage, no connection to bus, etc.).

Parameters:

logicalDiagnosticAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

Logical diagnostic address of DP master interface

ReqGetStateOfAllDpSlavesStateMode

Direction:	Input parameter
Data type:	EnumReqActDeactGetStateMode
Parameter index:	2

EnumReqActDeactGetStateMode

REQUEST_TRUE (0)	Start functionality
REQUEST_FALSE (1)	Query status with REQUEST_TRUE
REQUEST_ABORT (2)	Abort of the function execution

REQUEST_TRUE starts the functionality. REQUEST_FALSE scans the status of the functionality started with REQUEST_TRUE. REQUEST_ABORT aborts the function execution.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	3

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and the function cannot be completed.

With WHEN_COMMAND_DONE, the transition takes place when the command has been executed. With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	4

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status.

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	6
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of the diagnostic address of the distributed station. For INPUT, the diagnostic address of the distributed station is in the input range. For OUTPUT, the diagnostic address of the distributed station is in the output range. Diagnostic addresses are always of type INPUT.

2.5.4.19 _getStateOfAllDpStations

The system function determines the status of the communication to cyclical communications partners (device - PROFINET, slave - PROFIBUS, sender or recipient controller-controller internode communication - PROFINET).

The system function _getStateOfAllDpStations supplies the overall status of the stations configured for the intended interface (PROFIBUS, PROFINET IO) where this information is available. This status is formed from the status of the individual modules of the station.

If the system function _getStateOfAllDpStations returns the overall status DISTURBED, _getStateOfIO can be used to query the status of the slots/sub-slots or _getStateOfSingleDpSlave for diagnosing the communication interface.

The user program calls the system function the first time with 'RequestMode = REQUEST_TRUE'. This call returns a maximum of 10 results. In the 'functionResult', 0x00000001 signals that other status options are available.

The user program then calls the system function with the identifier 'RequestMode = REQUEST_FALSE'. The call is repeated until all status values have been read out. This is signaled with 'functionResult' 0x00000000.

Note

Where further function calls are made with 'RequestMode = REQUEST_FALSE', the same CommandID must be used as when calling with 'REQUEST_TRUE'. Unless the same CommandId is used, the function will continue to be restarted and only the first 10 status values will be displayed.

If the process of reading out the status values is to be aborted, the system function must be called with 'RequestMode = REQUEST_ABORT'.

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- Communication System Manual
- or in the online help

Return value: StructRetDeviceGetStateOfAllDpStations

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

EnumStateOfDpStation

NO_VALID_STATE (0)	Default value or a valid state could not be determined
INACTIVE (1)	Slave/IO device deactivated by program
IN_OPERATION (2)	Input/output data exchange with slave/IO device
DISTURBED (3)	DP slave / IO device is faulty (diagnostics pending, modules faulty ...)
NOT_PRESENT (4)	The diagnostics, input/output address of one or all modules in a station cannot be accessed by the controller. (Possible causes: Missing power supply of the module, missing connection to the bus, station failure, for example, caused by faulty or removed module in an ET200.)

StructRetDeviceGetStateOfAllDpStations
Return value for _getStateOfAllDpStations

Structure	Name	Data type	Unit
functionResult	Result	DINT	-
NumberResultsDpStations	Number of states that are signaled in actualStateOfDpStations	UDINT	-
StructStateOfDpStations	Status data of the available DP slaves / IO devices	ARRAY [0..9] OF StructStateOfDpStations	-

StructStateOfDpStations

Return parameters of the individual devices

Structure	Name	Data type	Unit
ioIdType	IO type	EnumIoIdType	-
logicalDiagnosticAddressDpStations	Diagnostics address of the PROFIBUS slave or IO device	DINT	-
actualStateOfDpStation	Status of the relevant PROFIBUS slave or IO device	EnumStateOfDpStation	-

Description of the return values: Return value 'functionResult'

16#00000000	The system function has been executed. All information on the configured PROFIBUS DP slaves or PROFINET IO devices is reported.
16#00000001	Ok - even more status values for PROFINET IO devices or PROFIBUS DP slaves are available. The system function must be called again with 'REQUEST_FALSE' if the other status values are to be read out.
16#00007000	Functionality is not active, only for asynchronous execution.
16#00007001	Functionality initiated, only possible for asynchronous execution.
16#00007002	Functionality active, only possible for asynchronous execution.
16#00007003	System function aborted successfully. If an abort is not possible, the system function returns the value 16#FFFF8094.
16#FFFF8090	The specified logical diagnostic address is invalid. Address is outside of value range for the logical diagnostic addresses.
16#FFFF8091	Specified logical diagnostic address is not an address for a PROFIBUS DP / PROFINET segment.
16#FFFF8092	The relevant PROFIBUS DP / PROFINET segment does not contain any configured PROFIBUS DP stations or PROFINET IO devices.
16#FFFF8093	The status values readout is not supported by the relevant PROFIBUS DP / PROFINET segment. The status values can only be read out by a PROFIBUS DP master interface or a PROFINET IO controller of a PROFINET interface.
16#FFFF8094	An abort is no longer possible in the current system function status.
16#FFFF8097	No status can be provided for the logical diagnostic address.
16#FFFF8099	Internal error, function terminated.
16#FFFF809E	Job aborted. Attempt to abort a non-active function.
16#FFFF80C1	There are currently no resources available to execute this system function. The user program can continue to call this system function until this return value no longer appears.
16#FFFF80C2	A re-querying of the PROFIBUS DP / PROFINET segment with REQUEST_TRUE was rejected because a function call for querying the status values of the relevant PROFINET / PROFIBUS DP segment is still in progress with an identical 'CommandId'. The user program can call this function until this return value no longer appears.

Return value 'NumberResultsDpStations'

'NumberResultsDpStations' supplies the number of result elements in the 'StructStateOfDpStations' array. If 'NumberResultsDpStations' is less than 10, the system function enters the value NO_VALID_STATE the logical diagnostic address 16#00007FFE (invalid diagnostic address) and in 'actualStateOfDpStation' for those array elements that are larger than 'NumberResultsDpStations'.

Return value 'actualStateOfDpStation'

Status of the relevant PROFIBUS DP slave or PROFINET IO device. NO_VALID_STATE system default or no valid state could be determined. INACTIVE means the slave or the IO device is deactivated by the program. IN_OPERATION means that input/output data exchange with slave or IO device is in progress. DISTURBED means the station is disturbed (head-end of station or a module in the station).

Possible causes:

- Fault in station head-end,
  - i.e. no data exchange with station head-end or modules in the station is possible.
- Fault on a module in the station.
- Data exchange with station head-end or modules in the station is possible.
  - One or more diagnostics of station head-end or modules in the station are pending.
- The user program can receive additional information about the causes by reading the diagnostics of the station head-end or the individual modules.

NOT_PRESENT means the diagnostics, input/output address of one or all modules in a station cannot be accessed by the controller (possible causes: Missing power supply of the module, missing connection to the bus, station failure, for example, caused by faulty or removed module in an ET200). Data exchange with the station or the modules in the station is not possible (system-internal functionality, user program, engineering system).

Parameters:

logicalDiagnosticAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

The logical diagnostic address of a PROFIBUS DP master interface or an IO controller's PROFINET interface should be entered under the 'logicalDiagnosticAddress' parameter.

Unless all elements are filled out in the 'actualStateOfDpStation' ARRAY, then the return value 16#00007FFE (invalid diagnostic address) is output. This means the content of 'actualStateOfDpStation' is irrelevant.

requestMode

Direction:	Input parameter
Data type:	EnumReqSysFunctMode
Parameter index:	2

EnumReqSysFunctMode

REQUEST_TRUE (0)	Start functionality
REQUEST_FALSE (1)	Status query of functionality started with REQUEST_TRUE
REQUEST_ABORT (2)	Abort of the function execution

REQUEST_TRUE starts the functionality. REQUEST_FALSE queries the status with the started functionality REQUEST_TRUE. REQUEST_ABORT aborts the function execution.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	3

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and cannot be completed. If the system function reports a display, the internal processing will be finished (other than for nextCommand, ABORT_CURRENT_COMMAND) and a restart of the system function possible with REQUEST_TRUE.

With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	4

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

This parameter specifies a 'commandId' of type 'commandIdType' that is unique from all others in the system for the purpose of tracking the command status. A 'commandId' is valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND. The system can generate the parameter 'syncCommandId' by calling the system function _getSyncCommandId. The same 'syncCommandId' is to be used for subsequent calls.

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	6
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of the diagnostic address of the communication interface (PROFIBUS DP/ PROFINET). For INPUT, the diagnostic address of the communication interface (PROFIBUS DP/ PROFINET) is in the input range. For OUTPUT, the diagnostic address of the communication interface (PROFIBUS DP/PROFINET) is in the output range. Diagnostic addresses are always of type INPUT.

2.5.4.20 _getStateOfDpSlave

This function supplies the information whether the PROFIBUS DP slave or the PROFINET IO device is activated or deactivated.

Note that for evaluating the _getStateOfDpSlave system function result, the 'functionResult' and 'commandIdState' return values are related to each other and are evaluated together.

If the 'functionResult' return parameter has the value 0, a single call of the system function completed successfully. The current status of the slave/device can be derived from the 'commandIdState' return parameter and represents the device status associated with the logical address when the system function was executed.

For the asynchronous call of the _getStateOfDpSlave function, this relationship means that a single call was successful but a subsequent call to determine the status is required.

Example

1st-nth Asynchronous start:

functionResult = 0, commandIdState = FUNCTION_IS_ACTIVE(28674) Meaning: Although the function was successfully called asynchronously, the device status could not yet be determined.

N+1 asynchronous call:

functionResult = 0, commandIdState = ACTIVE(1) / INACTIVE(2) / IN_ACTIVATION(3) / IN_DEACTIVATION(4) Meaning: The function was successfully called asynchronously, the device status could be determined. No further subsequent call is required.

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- SIMOTION Communication System Manual
- or in the online help

Return value: StructRetDeviceGetStateOfDpSlave

EnumDeviceStateOfDpSlave

NO_RESSOURCES_AVAILABLE (-32573)	No further resources available (16#FFFF80c3)
ACTIVE (1)	DP station activated (16#00000001)
INACTIVE (2)	DP station deactivated (16#00000002)
IN_ACTIVATION (3)	Activating DP station (16#00000003)
IN_DEACTIVATION (4)	Deactivating DP station (16#00000004)
ACTIVATED_NO_ALARM (5)	Internal value - not relevant for user (16#00000005)
DEACTIVATED_NO_ALARM (6)	Internal value - not relevant for user (16#00000006)
FUNCTION_IS_WAITING_FOR_ACTIVATION (28672)	Function is waiting for activation to complete (16#00007000)
FUNCTION_IS_ACTIVATED (28673)	Function is activated (16#00007001)
FUNCTION_IS_ACTIVE (28674)	Function is active (16#00007002)

StructRetDeviceGetStateOfDpSlave

Return parameters for device commands

Structure	Name	Data type	Unit
functionResult	Return value of function	DINT	-
commandIdState	State of job	EnumDeviceStateOfDpSlave	-

Description of the return values:

16#00000000	Abort job. No jobs are now being executed for the logical address.
16#00007003	Abort job. The job has already been accepted, but its execution could not continue as feedback is expected from the system.
The return value reports the job status.	
16#FFFF8090	Specified logical base address invalid.
16#FFFF8091	No valid DP adapter segment, job aborted: Function not executed, all resources are released.
16#FFFF8093	Not a valid DP station.
16#FFFF8098	Functionality terminated after timeout.
16#FFFF8099	Internal error: Function terminated.

Parameters:

logicalAddressOfDpStation

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

Specifying the logical diagnostic address of the DP station or IO device or I device.

reqActDeactGetStateMode

Direction:	Input parameter
Data type:	EnumReqActDeactGetStateMode
Parameter index:	2

EnumReqActDeactGetStateMode

REQUEST_TRUE (0)	Start functionality
REQUEST_FALSE (1)	Query status with REQUEST_TRUE
REQUEST_ABORT (2)	Abort of the function execution

REQUEST_TRUE starts the functionality. REQUEST_FALSE scans the status of the functionality started with REQUEST_TRUE. REQUEST_ABORT aborts the function execution.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	3

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and the function cannot be completed.

With WHEN_COMMAND_DONE, the transition takes place when the command is finished. With ABORT_CURRENT_COMMAND and the same 'commandId' as in the previous function call with IMMEDIATELY, the active function is aborted.

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	4
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of the logical diagnostic address of the distributed station or IO device or I device. For INPUT, the logical diagnostic address of the distributed station, IO device, or I device is in the input range. For OUTPUT, the logical diagnostic address of the distributed station, IO device, or I device is in the output range.

Note

Diagnostic addresses are always of type INPUT.

2.5.4.21 _getStateOfIO

This system function supplies the status of the device, the module or the submodule. This is addressed with the "Logical address", "Input/output identification" and "Scope" parameters.

The user program calls the system function the first time with 'RequestMode = REQUEST_TRUE'. This call returns a maximum of 10 results. In the 'functionResult', 0x00000001 signals that other status values are available.

The user program then calls the system function with the identifier 'RequestMode = REQUEST_FALSE'. The call is repeated until all status values have been read out. This is signaled with 'functionResult' 0x00000000.

Note

Where further function calls are made with 'RequestMode = REQUEST_FALSE', the same CommandID must be used as when calling with 'REQUEST_TRUE'. Unless the same CommandId is used, the function will continue to be restarted and only the first 10 status values will be displayed.

If the process of reading out the status values is to be aborted, the system function must be called with 'RequestMode = REQUEST_ABORT'.

Example:

Configuration of the I device in HW Config

Transfer address: 300

Diagnostics address of the interface: 16378

Via the diagnostics address of the PN interface:

Example

```
myRetStructRetDeviceGetStateOfIO_1 :=_getStateOfIO(
 logicalAddress := 16378
 ,ioId := INPUT
 ,scope := OBJECT
 ,requestMode := REQUEST_TRUE
 ,nextCommand := when_command_done
 ,commandId := _getcommandid());
```

Via the transfer address of the I device:

Example

```
myRetStructRetDeviceGetStateOfIO_1 :=_getStateOfIO(
 logicalAddress := 300
 ,ioId := INPUT
 ,scope := OBJECT
 ,requestMode := REQUEST_TRUE
 ,nextCommand := when_command_done
 ,commandId := _getcommandid());
```

Return value: StructRetDeviceGetStateOfIO

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

EnumStateOfIO

NO_VALID_STATE (0)	Default value or a valid status could not be determined
INACTIVE (1)	PROFIBUS DP slave / PROFINET IO device deactivated by program
IN_OPERATION (2)	Data exchange performed with PROFIBUS DP slave, PROFINET IO device, module or submodule
DISTURBED (3)	PROFIBUS DP slave / PROFINET IO device, module or submodule is faulty (diagnostics pending, modules faulty ...)
NOT_PRESENT (4)	PROFIBUS DP slave / PROFINET IO device, module or submodule is not present (missing supply voltage, missing connection to the bus ...)

EnumStateOfMaintenanceOfIO

MAINTENANCE_NO_INFO (0)	No information
MAINTENANCE_REQUIRED (1)	Maintenance needed soon
MAINTENANCE_DEMANDED (2)	Maintenance required

StructRetDeviceGetStateOfIO
Return value for _getStateOfIO

Structure	Name	Data type	Unit
functionResult	Result	DINT	-
NumberOfResults	Number of states that are signaled in actualStateOfDpStations	UDINT	-
stateOfIO	Status for the PROFIBUS DP slave, PROFINET IO device, module or submodule	ARRAY [0..9] OF StructStateOfIO	-

StructStateOfIO
Return value of a unit

Structure	Name	Data type	Unit
logicalAddress	Logical address of the PROFIBUS DP slave, PROFINET IO device, module or submodule	DINT	-
ioIdType	IO type	EnumIoIdType	-
actualStateOfDpStation	Status for the PROFIBUS DP slave, PROFINET IO device, module or submodule	EnumStateOfIO	-
stateOfMaintenace	Status for maintenance of the PROFIBUS DP slave, PROFINET IO device, module or submodule objects	EnumStateOfMaintenanceOfIO	-

Description of the return values:

Return value 'functionResult'

16#00000000	The system function has been executed. All information about the configured modules / submodules is reported.
16#00000001	OK - other status values for objects are present. The system function must be called again with 'REQUEST_FALSE' if the other status values are to be read out.
16#00007000	Functionality is not active, only for asynchronous execution.
16#00007001	Functionality initiated, only possible for asynchronous execution.
16#00007002	Functionality active, only possible for asynchronous execution.
16#00007003	System function aborted successfully. If an abort is not possible, the system function returns the value 16#FFFF8094.
16#FFFF8090	The specified logical diagnostic address is invalid. Address is outside of value range for the logical diagnostic addresses.
16#FFFF8091	Specified logical diagnostic address is not an address for a PROFIBUS DP / PROFINET segment.

16#FFFF8092	Depending on the logical address and the input/output assignment, the relevant PROFIBUS DP / PROFINET segment does not contain any configured PROFIBUS DP stations or PROFINET IO devices or input or output addresses.
16#FFFF8093	The status values readout is not supported by the relevant PROFIBUS DP / PROFINET segment. The status values can only be read out by a PROFIBUS DP master interface or a PROFINET IO controller of a PROFINET interface.
16#FFFF8094	An abort is no longer possible in the current system function status.
16#FFFF8097	No status can be provided for the logical diagnostic address.
16#FFFF8092	The relevant PROFIBUS DP / PROFINET segment does not contain any configured PROFIBUS stations or IO devices.
16#FFFF8093	The status values readout is not supported by the relevant PROFIBUS DP / PROFINET segment. The status values can only be read out by a PROFIBUS DP master interface or a PROFINET IO controller of a PROFINET interface.
16#FFFF8094	An abort is no longer possible in the current system function status.
16#FFFF8097	No status can be provided for the logical diagnostic address.
16#FFFF8099	Internal error, function terminated.
16#FFFF809E	Job aborted. Attempt to abort a non-active function.
16#FFFF80C1	There are currently no resources available to execute this system function. The user program can continue to call this system function until this return value no longer appears.
16#FFFF80C2	A re-querying of the PROFIBUS DP / PROFINET segment with REQUEST_TRUE was rejected because a function call for querying the status values of the relevant PROFINET / PROFIBUS DP segment is still in progress with an identical 'CommandId'. The user program can call this function until this return value no longer appears.

Return value 'stateOfIO'

Logical address of the associated PROFIBUS DP slave, IO device, module or submodule.

Input/output identification for the logical address.

Status.

NO_VALID_STATE means no default value or valid state could be determined.

INACTIVE means the slave or IO device is deactivated by the program.

IN_OPERATION means that input/output data exchange with slave or IO device is in progress.

DISTURBED means the station is disturbed (head-end of station or a module in the station).

Possible causes:

- Fault on station head-end, so no data exchange with station head-end or modules in the station.
- Fault on a module in the station. Data exchange with station head-end or modules in the station is possible. One or more diagnostics of station head-end or modules in the station are pending.
- The user program can receive additional information about the causes by reading the diagnostics of the station head-end or the individual modules.

NOT_PRESENT means the diagnostics, input/output address of one or all modules in a station cannot be accessed by SIMOTION (possible causes: Missing supply voltage, missing connection to the bus, station has failed, e.g. station does not report, module has an internal error or has been removed, for example for an ET200S, etc.)

Parameters:

logicalAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

The logical address of a PROFIBUS master interface, or of a PROFINET interface of an IO controller, a module or a submodule should be entered under the 'logicalAddress' parameter.

Unless all elements are filled out in the 'actualStateOfDpStation' ARRAY, then the return value 16#00007FFE (invalid diagnostic address) is output. This means the content of 'actualStateOfDpStation' is irrelevant.

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	2
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of the logical address With INPUT, the logical address is in the input range. With OUTPUT, the logical address is in the output range.

scope (optional)

Direction:	Input parameter
Data type:	EnumScopeOfIO
Parameter index:	3
System default:	OBJECT

EnumScopeOfIO

OBJECT (0)	Status of all elements of the object referenced by its logical address.
CONNECTED (1)	Status of all elements connected to the addressed object.

Scope of the logical address

For OBJECT, the status of all submodules, which are to be referenced to an object via the logical address, is returned. The following are designated in a representative role as the object: PROFINET IO interface, PROFINET IO device, PROFIBUS DP interface or PROFIBUS DP slave (DP station).

For PROFINET IO / PROFINET IO device, the status of all submodules addressed via PROFINET IO (device) is reported, for example, the status of the submodules or the status of all ports. For PROFIBUS DP interface or PROFIBUS DP slave, the status of the submodules is also returned.

For CONNECTED, the logical address is that of a distributed station. For CONNECTED, the status of all modules of the distributed station is output.

For PROFINET IO interface, the status of all PROFINET IO devices is returned. For PROFINET IO device, the status of all modules is returned. For PROFIBUS DP interface, the status of all PROFIBUS DP slaves is output. For PROFIBUS DP slave, the status of all modules is output.

requestMode

Direction:	Input parameter
Data type:	EnumReqSysFunctMode
Parameter index:	4

EnumReqSysFunctMode

REQUEST_TRUE (0)	Start functionality
REQUEST_FALSE (1)	Status query of functionality started with REQUEST_TRUE
REQUEST_ABORT (2)	Abort of the function execution

REQUEST_TRUE starts the functionality. REQUEST_FALSE queries the status with the started functionality REQUEST_TRUE. REQUEST_ABORT aborts the function execution.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	5

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and the function cannot be completed.

With WHEN_COMMAND_DONE, the transition takes place when the command has been executed (synchronous call). With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	6

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

This parameter specifies a 'commandId' of type 'commandIdType' that is unique from all others in the system for the purpose of tracking the command status. A 'commandId' is valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND. The system can generate the parameter 'syncCommandId' by calling the system function _getSyncCommandId. The same 'syncCommandId' is to be used for subsequent calls.

2.5.4.22 _getStateOfSingleDpSlave

Note

Do not use the _getStateOfSingleDpSlave function anymore as it is obsolete. Use the _getStateOfIO function instead.

The function determines the status of the communication to a cyclical communications partner (device - PROFINET, slave - PROFIBUS, sender or recipient controller-controller internode communication - PROFINET).

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- Communication System Manual
- or in the online help

Return value: StructRetGetStateOfSingleDpSlave

EnumStateOfDpSlave

INACTIVE (0)	Slave/IO device deactivated by program
IN_OPERATION (1)	Input/output data exchange with slave/IO device
DATA_EXCHANGE_INACTIVE (2)	No input/output data exchange with slave/IO device
NOT_PRESENT (3)	Slave/IO device is not available (no supply voltage, no connection to bus, etc.)

StructRetGetStateOfSingleDpSlave
Return parameter for _getStateOfSingleDPSlave

Structure	Name	Data type	Unit
functionResult	Total job return value	DINT	-
ActualStateOfSingleDpSlave	Status data of the individual DP slave	EnumStateOfDpSlave	-

Describes the state of the requested DP slave:

INACTIVE means the slave or IO device is deactivated by the program. IN_OPERATION means that input/output data exchange with slave or IO device is in progress.

DATA_EXCHANGE_INACTIVE means the station is disturbed (the head-end of the station or one of its modules is disturbed).

Possible causes:

- Fault in station head-end
  - No data exchange with station head-end or modules in the station is possible.
- Fault on a module in the station
  - Data exchange with station head-end or modules in the station is possible. One or more diagnostics of station head-end or modules in the station are pending. The user program can receive additional information about the causes by reading the diagnostics of the station head-end or the individual modules.

NOT_PRESENT means the slave or IO device is not available (no supply voltage, no connection to bus, etc.).

Description of the return values:

16#00000000	System function completed successfully.
16#00007000	Functionality is not active, only for asynchronous execution.
16#00007001	Functionality initiated, only for asynchronous execution.
16#00007002	Functionality active, only for asynchronous execution.
16#FFFF8090	No valid PROFIBUS or PROFINET segment.
16#FFFF8091	No configuration present for the specified DP station address or device number.
16#FFFF8092	A new call of the function with these parameters is not permitted.
16#FFFF8093	No valid logical diagnostic address of a DP station or an IO device.
16#FFFF8098	The automated status of the system function terminated with timeout.
16#FFFF8099	Internal error, terminate function.
16#FFFF80C1	Function already started, new call with parameter 'ReqGetStateOfSingleDPSlaveStateMode=TRUE'.
16#FFFF80C3	Temporary resource bottleneck.
16#FFFF80CF	CommandId not unique.
16#FFFF809E	Abort of function not allowed.
16#FFFF809D	Abort successful.

Parameters:

logicalDiagnosticAddress

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

Diagnostic address of the communications partner or the receive connection of the internode communication.

Note

Several connections are possible at the receiver side of the controller-controller internode communication. Each connection is represented as communications partner.

ReqGetStateOfSingleDPSlaveStateMode

Direction:	Input parameter
Data type:	EnumReqActDeactGetStateMode
Parameter index:	2

EnumReqActDeactGetStateMode

REQUEST_TRUE (0)	Start functionality
REQUEST_FALSE (1)	Query status with REQUEST_TRUE
REQUEST_ABORT (2)	Abort of the function execution

REQUEST_TRUE starts the functionality. REQUEST_FALSE scans the status of the functionality started with REQUEST_TRUE. REQUEST_ABORT aborts the function execution.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	3

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and the function cannot be completed.

With WHEN_COMMAND_DONE, the transition takes place when the command has been executed. With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	4

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status.

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	6
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of the logical diagnostic address of the distributed station. For INPUT, the logical diagnostic address of the distributed station is in the input range. For OUTPUT, the logical diagnostic address of the distributed station is in the output range. Diagnostic addresses can be of type INPUT or OUTPUT.

2.5.4.23 _getStateOfTo

This function supplies information on whether the technology object (TO) is activated or deactivated. It can be executed both synchronously and asynchronously.

Note

The functions _activateTo, _deactivateTo and _getstateOfTo belong to a function pool. Maximum 50 active calls from this pool can be processed simultaneously. The return value '16#FFFF80C3' is signaled when the maximum number is exceeded. Each running instance requires additional computation time in the technology object environment (in the cyclic levels); therefore it is more efficient to first query the respective state and then to set the required desired state by means of _activateTo and _deactivateTo.

Additional references

More detailed information on this system function can be found in

- the Function Manual entitled Motion Control Basic Functions for Modular Machines
- or in the SIMOTION online help.

Return value: StructRetGetStateOfTo

EnumStateOfTo

ACTIVE (1)	16#00000001 TO activated
INACTIVE (2)	16#00000002 TO deactivated
IN_ACTIVATION (3)	16#00000003 TO will be activated
IN_DEACTIVATION (4)	16#00000004 TO will be deactivated

EnumToSetStateOfTo

ACTIVATED (1)	16#00000001 TO activated
DEACTIVATED (2)	16#00000002 TO deactivated

StructRetGetStateOfTo
Return parameters for device commands

Structure	Name	Data type	Unit
functionResult	Return value of function	DINT	-
commandIdState	Current state of the technology object	EnumStateOfTo	-
setState	Desired state of the TO: ACTIVATED / DEACTIVATED	EnumToSetStateOfTo	-

Description of the return values

Return value "functionResult" has the following meaning:

16#00000000	The system function is finished. The result is in 'commandIdState'.

16#00007000	The system function is in 'Idle' status.
16#00007001	The system function is started (only reported when execution is asynchronous).
16#00007002	The system function is running (only reported when execution is asynchronous, a new call in the control program is required until the functionality ends).
16#FFFF8090	Technology object not available. Function not executed, all resources are released.
16#FFFF8091	No valid device, job aborted: Function not executed, all resources are released.
16#FFFF8098	Functionality terminated after timeout, Function not executed, all resources are released.
16#FFFF8099	Internal error, function terminated. Function not executed, all resources are released.
16#FFFF80C3	Temporary bottleneck (internal resources for executing the function not currently available). The maximum number of 50 calls active simultaneously has been exceeded. Wait until an active call is completed.

Parameters:

TO_Instance

Direction:	Input parameter
Data type:	AnyObject
Parameter index:	1

Reference to the technology object.

reqActDeactGetStateMode

Direction:	Input parameter
Data type:	EnumReqActDeactGetStateMode
Parameter index:	2

EnumReqActDeactGetStateMode

REQUEST_TRUE (0)	Start functionality
REQUEST_FALSE (1)	Query status with REQUEST_TRUE
REQUEST_ABORT (2)	Abort of the function execution

REQUEST_TRUE starts the functionality. REQUEST_FALSE scans the status of the functionality started with REQUEST_TRUE. REQUEST_ABORT aborts the function execution.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	3

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	5

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and the function cannot be completed.

With WHEN_COMMAND_DONE, the transition takes place when the command is finished. With ABORT_CURRENT_COMMAND and the same 'commandId' as in the previous function call with IMMEDIATELY, the active function is aborted.

2.5.4.24 _setDpSlaveAddress

This system function sets a new DP address (node number) of a DP interface on a SIMOTION device.

Additional references

Further information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Return value: DINT

Description of the return values:

16#00000000	The system function is successfully completed
16#FFFF8090	Invalid address Outside of valid range / invalid DP segment adapter
16#FFFF8091	Invalid DP slave address.
16#FFFF8092	DP adapter is not configured as a slave.
16#FFFF8093	DP adapter is not configured passively.
16#FFFF8094	Requested slave address not stored.
16#FFFF8099	Internal error, function terminated.
16#FFFF80C1	Resources for implementing the functionality are occupied.
16#FFFFFF51	This function is supported in the simulation.

Parameters:

logicalAddressCommunicationAdapter

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

Specifies the logical diagnostic address of the interface.

dpSlaveAddress

Direction:	Input parameter
Data type:	SINT
Parameter index:	2

Communication address.

2.5.4.25 _setIPConfig

Note

Applies from SIMOTION version V4.4

Use the '_setPnIpConfig' system function instead of '_setIpConfig' when you create a new SIMOTION project.

This function sets the IP configuration of a selected Ethernet interface. The parameters are not permanently saved. The configured IP address data are not backed up with RamtoRom.

Additional references

Further information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Return value: UDINT

Description of the return values:

16#00000000	IP configuration was set successfully
16#FFFF80C1	Resource bottleneck
16#FFFF8099	Internal error.
16#FFFF8090	Incorrect return value output in the transfer parameter 'ethernetInterface', subnet mask or gateway.

Parameters:

ethernetInterface

Direction:	Input parameter
Data type:	EnumInterfaceID
Parameter index:	1

EnumInterfaceID

IE_01 (0)	Ethernet interface 1: All platforms
IE_02 (1)	Ethernet interface 2: All platforms

Selected Ethernet interface

ipAddress

Direction:	Input parameter
Data type:	ARRAY [0..5] OF USINT
Parameter index:	2

Ethernet interface IP address to be transferred

subnetMask

Direction:	Input parameter
Data type:	ARRAY [0..5] OF USINT
Parameter index:	3

Ethernet interface subnet screen form to be transferred

gatewayAddress

Direction:	Input parameter
Data type:	ARRAY [0..5] OF USINT
Parameter index:	4

Ethernet interface gateway address to be transferred

2.5.4.26 _setModeSelfAdaptingConfiguration

This system function sets the mode to ON or OFF.

Additional references

Further information is available at:

- the Function Manual entitled Motion Control Basic Functions for Modular Machines
- or in the online help

Return value: DINT

This system function sets the mode if storage for this mode is available in SIMOTION.

Description of the return values:

16#00000000 OK - the system function is finished - the mode has been saved.

16#00000001 Storage not available for this information.

16#FFFF8099 Internal error, function terminated.

16#FFFF80C1 The resources to execute this functionality are missing.

16#FFFF80C3 Access to the mode storage location disabled.

Parameters:

mode

Direction:	Input parameter
Data type:	EnumModeSelfAdaptingConfiguration
Parameter index:	1

EnumModeSelfAdaptingConfiguration

OFF (0)	Switched off
ON (1)	Switched on

Value "ON" or "OFF" to be set.

2.5.4.27 _setNameOfStation

Note

Applies from SIMOTION version V4.4

Use the '_setPnNameOfStation' system function instead of '_setNameOfStation' when you create a new SIMOTION project.

This function saves the transferred NameOfStation. The stored 'NameOfStation' only then takes effect using _activateNameOfStation.

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Return value: DINT

The return value returns the job status and name.

Description of the return values:

16#0000000	System function completed successfully. 'NameOfStation' is successfully set during the next activation using _activateNameOfStation.
16#FFFF8090	The specified logical diagnostic address is invalid.
16#FFFF8091	No valid logical diagnostic address for a PROFINET interface or a port of a PROFINET interface.
16#FFFF8092	PROFINET interface is configured, but not available. Can only occur if the PROFINET interface is inserted in SIMOTION via a module.
16#FFFF8093	The STRING specified in the 'nameOfStation" parameter is too long. Only 239 characters are permitted. According to the standard, 240 characters are permitted, but this is not currently supported by SIMOTION.
16#FFFF8094	An abort is no longer possible in the current system function status.
16#FFFF8098	The contents of the STRING specified under the parameter 'nameOfStation' do not respect the usual 'NameOfStation' conventions.
16#FFFF8099	Internal error. Function terminated.
16#FFFF809A	Setting the 'NameOfStation' is not supported by the PROFINET interface.
16#FFFF80A0	In HW Config, the options "Obtain device names in another way" or "Obtain IP address in another way" are deselected for the respective communication interface. Select both of these options for the corresponding communication interface in HW Config. Function terminated.
16#FFFF80C1	There are currently no resources available to execute this system function. The user program can call this function until this return value no longer appears.
16#FFFF80CF	A system function call for the relevant PROFINET interface is already in progress. The user program can call this function until this return value no longer appears.

Parameters:

logicalAddressPnInterface

Direction:	Input parameter
Data type:	DINT
Parameter index:	1

Specifies the logical diagnostic address of the PROFINET interface, or the logical diagnostic addresses of the associated ports on the PROFINET interface, from which the active device name is to be read.

The ports' logical diagnostic addresses clearly identify the PROFINET interface. The requested active 'NameOfStation' is supplied.

NameOfStation

Direction:	Input parameter
Data type:	STRING
Parameter index:	2

Specification of the station name. This name is active in the specified Profinet interface (x120) x127, (x130, x150), x1400. If the name to be set exceeds a length of 63 characters, it must be divided into 63-character sections separated by the "." character.

requestMode

Direction:	Input parameter
Data type:	EnumReqSysFunctMode
Parameter index:	3

EnumReqSysFunctMode

REQUEST_TRUE (0)	Start functionality
REQUEST_FALSE (1)	Status query of functionality started with REQUEST_TRUE
REQUEST_ABORT (2)	Abort of the function execution

REQUEST_TRUE starts the functionality. REQUEST_FALSE scans the status of the functionality started with REQUEST_TRUE. REQUEST_ABORT aborts the function execution.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	4

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

This parameter specifies a 'CommandId' of type 'commandIdType' that is unique from all others in the system. The user can make the system generate a 'syncCommandId' by calling _getSyncCommandId. The same syncCommandId is used for subsequent calls.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	6

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and the function cannot be completed.

With WHEN_COMMAND_DONE, the transition takes place when the command is finished. With ABORT_CURRENT_COMMAND and the same 'commandId' as in the previous function call with IMMEDIATELY, the active function is aborted.

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	7
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

The ioId parameter indicates whether logicalAddressPnInterface of the Profinet interface is an input address or an output address.

2.5.4.28 _setPnIpConfig

Note

Applies from SIMOTION version V4.4

Use the '_setPnIpConfig' system function instead of '_setIpConfig' when you create a new SIMOTION project.

This function sets the IP configuration of a selected PROFINET interface.

Additional references

Further information is available at:

- SIMOTION Communication System Manual
- or in the online help

Return value: UDINT

Return value UDINT:

16#00000000	IP configuration has been set successfully and is active.
16#00007001	First call, the trigger to set the IP configuration OK (only when command is issued asynchronously). Must be repeated in the next program cycle.
16#00007002	Intermediate call, the setting of the IP configuration is still active. Repeat in the next program cycle where necessary until the IP configuration has been set successfully or canceled (only when the command is issued asynchronously).
16#00007003	Setting IP configuration aborted (only when command is issued asynchronously).
16#FFFF8090	No valid logical diagnostic address for a PROFINET interface or a port of a PROFINET interface.
16#FFFF8091	No valid logical diagnostic address for a PROFINET interface or a port of a PROFINET interface.
16#FFFF8092	PROFINET interface is configured, but not available. Can only occur if the PROFINET interface is inserted into the SIMOTION device via a module.
16#FFFF8094	An abort is not possible in the current system function status.
16#FFFF8099	An error occurred while the function was being processed.
16#FFFF80A5	An attempt has been made to set an IP subnet which is already functioning at another PROFINET interface on the device. The subnet in which a PROFINET interface is located is defined in its IP configuration. The rule is that each PROFINET interface on a device must be located in its own IP subnet.
16#FFFF80A7	The IP configuration could not be set because there are illegal values in the 'ipAddress', 'subnetMask', and/or 'gatewayAddress' parameters.
16#FFFF80C1	There are currently no resources available to execute this system function. The user program can call this function until this return value no longer appears.

16#FFFF80C2	An attempt is made to change the configuration of the PROFINET interface while a job for changing the PROFINET interface is running.
16#FFFF80FF	There is no configuration information on a PROFINET interface at the logical address. To remedy this, load a project with a configuration for the PROFINET interface into the device.

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

States the diagnostic address. Selects whether the diagnostic address in the 'logicalAddressPnInterface' parameter is an input or output address.

logicalAddressPnInterface

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

States the diagnostic address of interfaces / ports. You must specify a valid diagnostics address of interfaces/ports of PROFINET interfaces, from which the IP configuration is to be set. This can be used to address both Ethernet interfaces with basic PROFINET data (PN-IE) and PN-IO interfaces of a SIMOTION device.

ipAddress

Direction:	Input parameter
Data type:	ARRAY [0..3] OF USINT
Parameter index:	3

States the IP address of the Ethernet interface to be transferred.

subnetMask

Direction:	Input parameter
Data type:	ARRAY [0..3] OF USINT
Parameter index:	4

States the subnet screen of the Ethernet interface to be transferred.

gatewayAddress

Direction:	Input parameter
Data type:	ARRAY [0..3] OF USINT
Parameter index:	5

States the gateway address of the Ethernet interface to be transferred.

storeIpConfigPermanent (optional)

Direction:	Input parameter
Data type:	EnumYesNo
Parameter index:	6
System default:	NO

EnumYesNo

NO (91)	No
YES (173)	Yes

States the validity of the IP configuration set. If the input parameter is set to NO, the set IP configuration remains active until the next start-up (e.g. download or network on/off). If the input parameter is set to YES, the set IP configuration remains active even after the next start-up (e.g. download or network on/off).

Call options _setPnIpConfig

The following are possible call options of the system function _setPnIpConfig and their results after a successful execution:

_setPnIpConfig and 'obtain IP address in another way' not set.

After an error-free execution, the IP configuration stated on _setPnIpConfig is temporarily active. The IP configuration loaded via a project is then temporarily suspended. After a restart, the IP configuration from the project will be reactivated. The input parameter 'storeIpConfigPermanent' of Systemfunktion _setPnIpConfig has no effect.

_setPnIpconfig with 'storeIpConfigPermanent=YES' and 'Obtain IP address in another way' set.

After an error-free execution, the IP configuration stated on _setPnIpConfig is active. The IP configuration is written into the permanent files. After the next restart, the IP configuration set via _setPnIpConfig remains active.

_setPnIpConfig with 'storeIpConfigPermanent=NO' and 'Obtain IP address in another way' set.

After an error-free execution, the IP configuration stated on _setPnIpConfig is active. After the next restart, an invalid IP configuration (IP address = subnet screen = gateway address = 0.0.0.0) is active.

nextCommand (optional)

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	7
System default:	IMMEDIATELY

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Selection of synchronous or asynchronous call of the system function. Familiar functionality.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	8

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status. A 'commandId' is always valid for the duration of the job, and an asynchronous job can only be aborted with a 'commandId' by means of the transition condition ABORT_CURRENT_COMMAND. The same 'commandId' is used for subsequent calls.

2.5.4.29 _setPnNameOfStation

Note

Applies from SIMOTION version V4.4

Use the '_setPnNameOfStation' system function instead of '_setNameOfStation' when you create a new SIMOTION project.

This function saves the transferred NameOfStation. The system function is completed using 'OK'. The return value 'OK' means 'NameOfStation effective'. Subsequent activation using '_activateNameOfStation' is no longer required and is even acknowledged with an error in the system function return value. This name is active in the specified Profinet interface (x120) x127, (x130, x150), x1400. If the name to be set exceeds a length of 63 characters, it must be divided into 63-character sections separated by the "." character.

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Return value: DINT

The return value returns the job status and name.

Description of the return values:

16#0000000	The device name has been set and activated successfully.
16#00007001	Initial call, the trigger to set the device name is still active. Repetition necessary in the next program cycle (only when command is issued asynchronously).
16#00007002	Intermediate call, the setting of the device name is still active. Repetition necessary in the next program cycle until the setting of the device name has been completed successfully or aborted (only when command is issued asynchronously).
16#00007003	Setting the device name aborted. If an abort is not possible, the system function returns 16#FFFF8094.
16#FFFF8090	The specified logical diagnostic address is invalid.
16#FFFF8091	No valid logical diagnostic address for an interface or a port of a PROFINET interface.
16#FFFF8092	PROFINET interface is configured, but not available. This error only occurs if the PROFINET interface is inserted in SIMOTION via a module.
16#FFFF8093	The STRING specified in the 'nameOfStation'' parameter is too long.
16#FFFF8094	An abort is no longer possible in the current system function status.
16#FFFF8098	The contents of the STRING specified under the parameter 'nameOfStation' do not respect the usual 'NameOfStation' conventions.
16#FFFF8099	An error occurred while the function was being processed.
16#FFFF809A	Setting the device name is not supported by the PROFINET interface.

16#FFFF80A0	In HW Config, the options "Obtain device names in another way" or "Obtain IP address in another way" are deselected for the respective communication interface. Both these options must be selected for the corresponding communication interface in HW Config so that the device name can be set.
16#FFFF80FF	There is no configuration information on a PROFINET interface at the logical address. To remedy this, load a project with a configuration for the PROFINET interface into the device.
16#FFFF80C1	There are currently no resources available to execute this system function. The user program can call this function until this return value no longer appears.
16#FFFF80C2	A system function call for the relevant PROFINET interface is already in progress. The user program can call this function until this return value no longer appears.

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of the logical diagnostic address. Specifying this is optional. With INPUT, the logical diagnostic address is in the input range. With OUTPUT, the logical diagnostic address is in the output range.

logicalAddressPnInterface

Direction:	Input parameter
Data type:	DINT
Parameter index:	2

Specifies the logical diagnostic address of the PROFINET interface, or the logical diagnostic address of the associated ports on the PROFINET interface, from which the active device name is to be set.

NameOfStation

Direction:	Input parameter
Data type:	STRING
Parameter index:	3

Specification of the station name. According to the standard, 240 characters are permitted.

storeNameOfStationPermanent (optional)

Direction:	Input parameter
Data type:	EnumYesNo
Parameter index:	4
System default:	NO

EnumYesNo

NO (91)	No
YES (173)	Yes

With 'storeNameOfStationPermanent := NO', the device name is effective until the next change and/or restart; after a restart, the device name is deleted and returns an empty string. With 'storeNameOfStationPermanent := YES', the device name is retained even following a restart.

nextCommand (optional)

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	5
System default:	IMMEDIATELY

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. With IMMEDIATELY, an initial call starts the asynchronous processing of the system function. The system function is to be called until it reports that it has been completed (OK or error indications).

With WHEN_COMMAND_DONE, the transition takes place when the command is finished. With ABORT_CURRENT_COMMAND and the same 'commandId' as in the previous function call with IMMEDIATELY, the active function is aborted.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	6

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

This parameter specifies a 'CommandId' of type 'commandIdType' that is unique from all others in the system. The user can make the system generate a 'syncCommandId' by calling _getSyncCommandId. The same syncCommandId is used for subsequent calls.

2.5.4.30 _synchronizeDpInterfaces

This function synchronizes the DP master interface with the DP slave interface. This requires a corresponding interface configuration.

The function is complete when the interfaces are synchronous or if synchronization is not possible. This function can be called multiple times (using the same 'commandId') to scan the execution status.

This function can only be active once at any given time. Multiple starts (multiple calls using a different 'commandId') cause an error message.

If the operating mode is switched from RUN to STOP while the function is being executed, the function is aborted. After the transition from STOP to RUN, the function can be called again.

Additional references

Additional information is available at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Return value: DINT

Description of the return values:

Asynchronous call: (in addition to return values for synchronous call)	
16#00007000	DP synchronization started: Return value signals that the function has been started.
Synchronous call:	
16#00007010	DP interfaces synchronized: Return value signals that the interfaces are synchronized. The internal state machine is advanced to DP_INTERFACES_SYNCHRONIZED.

Note

With PROFIBUS, the function must be started when RUN is reached; with PROFINET, the call for PROFINET can be made during startup. With PROFINET, the call only functions when the system variable StateOfDpInterfaceSynchronization is at DP_CLOCK_DETECTED when the function is called.

16#00007080	DP synchronization active: Return value signals that the function is still being executed.
16#FFFF805x	DP synchronization not enabled: Return value signals that synchronization was ended unsuccessfully. The internal state machine is advanced to WAIT_FOR_DP_CLOCK. Details x: 0 This error is set if the _enableDpInterfaceSynchronizationMode function was not called with parameter MASTER_SLAVE_ALARMMESSAGES_1. 1 This error is set if the interfaces could not be synchronized. One possible cause is interference in the clock signal. 2 Temporary internal synchronization problem. The user state machine must be advanced.

16#FFFF8080	DP synchronization not enabled: It is not permissible to call the function in the current system state. The internal state machine is not advanced.
16#FFFF8081	DP_Synchronization already active: Return value signals that the function has already been called with another 'commandId'. The new call was aborted or rejected. The internal state machine is not advanced.
16#FFFF8082	Function cannot be called synchronously due to resource bottleneck: The return value signals that the function cannot be called synchronously at the moment for resource reasons (no semaphore available). The internal state machine is not advanced.
16#FFFF8083	Required alarm messages are not enabled: The function has been called, although the alarm messages required for sequential control have not been activated or the C230-2 has not been configured as an isochronous slave (see also _enableDpInterfaceSynchronizationMode function). The internal state machine is not advanced.
16#FFFF8090	DP synchronization not supported: Return value signals that the function is not supported by the platform. The internal state machine is not advanced.

Parameters:

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	1

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and the function cannot be completed.

With WHEN_COMMAND_DONE, the transition takes place when the command is finished. With ABORT_CURRENT_COMMAND and the same 'commandId' as in the previous function call with IMMEDIATELY, the active function is aborted.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	2

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status.

2.5.5 Controller

2.5.5.1 _CTRL_pid

The _CTRL_pid function block is used to control technical processes with continuous input and output variables on SIMOTION systems. Via parameter assignment, you can enable or disable subfunctions of the PID controller and thus adapt it to the controlled system in question.

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "SIMOTION Utilities & Applications" CD-ROM. The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myCTRL_pid : _CTRL_pid; // instance of FB (_CTRL_pid)
myCTRL_pid(
enable := myEnable, // [IN] enable (BOOL)
dataPid := myCTRL_dataPid // [IN_OUT] dataPid (Struct_CTRL_dataPid)
);

// **********************
// Struct_CTRL_dataPid
// **********************
// myCTRL_dataPid : Struct_CTRL_dataPid;
myCTRL_dataPid.reset := FALSE; // [IN] complete restart
myCTRL_dataPid.manualMode := TRUE; // [IN] manual value on
myCTRL_dataPid.actValueType := FALSE; // [IN] process variable peripherie on
myCTRL_dataPid.setPController := TRUE; // [IN] proportional action on
myCTRL_dataPid.setIController := TRUE; // [IN] integral action on
myCTRL_dataPid.holdIValue := FALSE; // [IN] integral action hold
myCTRL_dataPid.setIValue := FALSE; // [IN] initialization of the integral action
myCTRL_dataPid.setDController := FALSE; // [IN] derivative action on
myCTRL_dataPid.cycleTime := 1000; // [IN] sample time in ms
myCTRL_dataPid.setpointValue := 0.0; // [IN] internal setpoint
myCTRL_dataPid.numericActValue := 0.0; // [IN] process variable in
myCTRL_dataPid.binActValue := 0; // [IN] process variable peripherie
myCTRL_dataPid.manualValue := 0.0; // [IN] manual value
myCTRL_dataPid.propGain := 2.0; // [IN] proportional gain
myCTRL_dataPid.integTime := 20000; // [IN] integration time in ms
myCTRL_dataPid.derivativeTime := 10000; // [IN] derivative time in ms
myCTRL_dataPid.delayTime := 2000; // [IN] time lag of the derivative action in ms
myCTRL_dataPid.deadBand := 0.0; // [IN] dead band width
myCTRL_dataPid.upperLimit := 100.0; // [IN] manipulated value high limit
myCTRL_dataPid.lowerLimit := 0.0; // [IN] manipulated value low limit
myCTRL_dataPid.actValueFactor := 1.0; // [IN] process variable factor
myCTRL_dataPid.actValueOffset := 0.0; // [IN] process variable offset
myCTRL_dataPid.outValueFactor := 1.0; // [IN] manipulated value factor
myCTRL_dataPid.outValueOffset := 0.0; // [IN] manipulated value offset
myCTRL_dataPid.initialIValue := 0.0; // [IN] initialization value of the integral action
myCTRL_dataPid.disturbValue := 0.0; // [IN] disturbance variable
```

2.5.5.2 _CTRL_piStep

The _CTRL_piStep function block is used to control technical processes with binary manipulated variable output signals for integrating actuators on SIMOTION systems. Via parameter assignment, you can enable or disable subfunctions of the PI step controller and thus adapt it to the controlled system in question.

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "SIMOTION Utilities & Applications" CD-ROM. The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myCTRL_piStep : _CTRL_piStep; // instance of FB (_CTRL_piStep)
myCTRL_piStep(
enable := myEnable, // [IN] enable (BOOL)
dataPiStep := myCTRL_dataPiStep // [IN_OUT] dataPiStep (Struct_CTRL_dataPiStep)
);

// *************************
// Struct_CTRL_dataPiStep
// *************************
// myCTRL_dataPiStep : Struct_CTRL_dataPiStep;
myCTRL_dataPiStep.reset := FALSE; // [IN] complete restart
myCTRL_dataPiStep.upperLimit := FALSE; // [IN] high limit signal of repeated manipulated value
myCTRL_dataPiStep.lowerLimit := FALSE; // [IN] low limit signal of repeated manipulated value
myCTRL_dataPiStep.manualMode := TRUE; // [IN] manipulated signals value
myCTRL_dataPiStep.setOutHigh := FALSE; // [IN] manipulated value signals up
myCTRL_dataPiStep.setOutLow := FALSE; // [IN] manipulated signals down
myCTRL_dataPiStep.actValueType := FALSE; // [IN] process variable peripherie on
myCTRL_dataPiStep.cycleTime := 1000; // [IN] sample time in ms
myCTRL_dataPiStep.setPointValue := 0.0; // [IN] internal setpoint
myCTRL_dataPiStep.numericActValue := 0.0; // [IN] process variable in
myCTRL_dataPiStep.binActValue := 16#0000; // [IN] process variable periphery W#16#0000
myCTRL_dataPiStep.propGain := 2.0; // [IN] proportional gain
myCTRL_dataPiStep.integTime := 20000; // [IN] integration time in ms
myCTRL_dataPiStep.deadBand := 1.0; // [IN] dead band width
myCTRL_dataPiStep.actValueFactor := 1.0; // [IN] process variable factor
myCTRL_dataPiStep.actValueOffset := 0.0; // [IN] process variable offset
myCTRL_dataPiStep.minPulseTime := 3000; // [IN] minimum pulse time in ms
myCTRL_dataPiStep.minIdleTime := 3000; // [IN] minimum break time in ms
myCTRL_dataPiStep.actuatingTime := 30000; // [IN] motor manipulated value in ms
myCTRL_dataPiStep.disturbanceValue := 0.0; // [IN] disturbance variable
```

2.5.5.3 _CTRL_pwm

The _CTRL_pwm function block is used to set up a PID controller with pulse output for proportional actuators.

ST:

Note

You can import the following program segments into the ST program using drag-and-drop. Additions must be made for executability (see manual, e.g. creating variables, etc.). An executable application example can be found on the "SIMOTION Utilities & Applications" CD-ROM. The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

```
// myCTRL_pwm : _CTRL_pwm; // instance of FB (_CTRL_pwm)
myCTRL_pwm(
enable := myEnable, // [IN] enable (BOOL)
dataPwm := myCTRL_dataPwm // [IN_OUT] dataPwm (Struct_CTRL_dataPwm)
);

// **********************
// Struct_CTRL_dataPwm
// **********************
// myCTRL_dataPwm : Struct_CTRL_dataPwm;
myCTRL_dataPwm.reset := FALSE; // [IN] complete restart
myCTRL_dataPwm.inValue := 0.0; // [IN] input variable
myCTRL_dataPwm.periodTime := 1000; // [IN] period time in ms
myCTRL_dataPwm.minPulseIdleTime := 0; // [IN] minimum break time in ms
myCTRL_dataPwm.ratioFactor := 1.0; // [IN] ratio factor
myCTRL_dataPwm.set3StepControl := TRUE; // [IN] three step signal on
myCTRL_dataPwm.set2StepControl := FALSE; // [IN] two step signal for bipolar manipulated value on
myCTRL_dataPwm.setManualMode := FALSE; // [IN] manual mode on
myCTRL_dataPwm.setPosPulse := FALSE; // [IN] positiv pulse on
myCTRL_dataPwm.setNegPulse := FALSE; // [IN] negativ pulse on
myCTRL_dataPwm.cycleTime := 10; // [IN] sample Time in ms
```

2.5.6 System time

2.5.6.1 _getInternalTimeStamp

This function returns an internal timestamp which can be used for high-resolution runtime measurements. The value depends on the CPU and can only be interpreted by the _getTimeDifferenceOfInternalTimeStamps function for the purpose of calculating a period of time.

Return value: UDINT

The return value is the internal timestamp at the time of the function call.

2.5.6.2 _getPNSyncCounter

This function provides the current counter value of a counter synchronized via PROFINET, which is read from the PN module. The value of the numerator corresponds to the unit ns (nanoseconds). For modules with several PN modules (e.g. CBE and switch module), the logical address or diagnostics address is used to distinguish the PN modules.

Return value: StructPNSyncCounter

StructPNSyncCounter
Identification of the configurations to be activated

Structure	Name	Data type	Unit
PNSyncCounterLow	Low part of 64-bit counter	UDINT	-
PNSyncCounterHigh	High part of 64-bit counter	UDINT	-

The return parameter contains the current 64-bit counter value as a field with two 32-bit values (low-order component and high-order component).

Parameters:

ioId (optional)

Direction:	Input parameter
Data type:	EnumIoIdType
Parameter index:	1
System default:	INPUT

EnumIoIdType

INPUT (198)	Input address
OUTPUT (199)	Output address

Input/output assignment of logical address of the I/O module. With INPUT, the logical address is in the input range. With OUTPUT, the logical address is in the output range.

logAddress (optional)

Direction:	Input parameter
Data type:	DINT
Parameter index:	2
System default:	0

Specifies the logical address or the diagnostics address of the PN module. For addresses that cannot be assigned to PROFINET, the count value 0 is returned.

Note

The value of the numerator is only output with a valid logical address, when an IRT connection is configured and the SIMOTION CPU is synchronized with PROFINET.

2.5.6.3 _getTimeDifferenceOfInternalTimeStamps

This function calculates the time gap between two time stamps, usually determined via _getInternalTimeStamp, in microseconds. The time that elapses between two time stamps is different for each device. In the case of SIMOTION D410-2, the maximum length of a measuring interval should not exceed 14 seconds. Otherwise the result may be undefined.

Note

The sequence of the time stamp parameters must be taken into account when calling the function to determine the time difference. The numeric values of the time stamp are not significant for this.

Return value: UDINT

The return value is the runtime of t1 to t2 in microseconds.

Parameters:

t1

Direction:	Input parameter
Data type:	UDINT
Parameter index:	1

The first (i.e. older) time stamp for time measurement. It indicates the start of the interval.

t2

Direction:	Input parameter
Data type:	UDINT
Parameter index:	2

The second (i.e. more recent) time stamp for time measurement. It indicates the end of the interval.

2.5.7 Device functions

2.5.7.1 _changeOperationMode

This function causes the operating mode to change to STOP or STOPU. There is no return value.

Additional references

Additional information is available at:

- SIMOTION SCOUT Configuration Manual
  Section Controlling the operating mode with SIMOTION SCOUT
- or in the SIMOTION online help.

Parameters:

OperationMode

Direction:	Input parameter
Data type:	EnumOperationMode
Parameter index:	1

EnumOperationMode

STOPU (0)	STOPU
STOP (134)	STOP

Specifies the state change that occurs with STOPU or STOP. With STOPU, a transition to STOPU takes place. With STOP, a transition to STOP takes place.

2.5.7.2 _getDeviceId

This function reads out special HW identifiers from the CPU HW Info block.

Additional references

More detailed information on this system function can be found in

- Basic Functions function manual
- or in the online help

Return value: StructRetGetDeviceId

StructRetGetDeviceId
Return parameter for _getDeviceId

Structure	Name	Data type	Unit
functionResult	Result	UDINT	-
id	ID	STRING	-

Description of the return values:

functionResult

16#00000000	Readout of HW identifiers completed without errors.
16#FFFF80C3	Information is not available.
16#FFFF8090	Incorrect transfer parameter.
16#FFFF8099	Internal error.

Parameters:

idType

Direction:	Input parameter
Data type:	EnumDeviceIdType
Parameter index:	1

EnumDeviceIdType

SERIAL_NUMBER (0)	CPU serial number
HW_TYPE (1)	Module type
SPECIFIC_NUMBER (2)	Special OEM number
ORDER_ID (3)	Module order number

Specifications for reading out the hardware identifier.

2.5.7.3 _getMemoryCardId

This function reads out special HW identifiers from the MemoryCard HW Info block.

Additional references

Further information is available at:

- SIMOTION Basic Functions Function Manual
- or in the online help

Return value: StructRetGetMemoryCardId

StructRetGetMemoryCardId
Return parameter for getMemoryCardId

Structure	Name	Data type	Unit
functionResult	Result	UDINT	-
id	ID	STRING	-

Description of the return values:

functionResult

16#00000000	Readout of HW identifiers completed without errors.
16#FFFFFFFD	Internal error.
16#FFFFFFF8	Incorrect parameter.

Parameters:

idType

Direction:	Input parameter
Data type:	EnumMemoryCardIdType
Parameter index:	1

EnumMemoryCardIdType

MEMORY_CARD_SERIAL_NUMBER (0)	Memory card serial number

Specification for reading out the hardware identifier.

Note

A leading 'S' in the MemoryCard serial number will be ignored and not returned in the return value. For determining the license key in the Web License Manager, it does not play any role whether or not the serial number is specified with a leading 'S'.

2.5.7.4 _restart

This function restarts the SIMOTION D410-2 device. To make the SIMOTION device assume 'RUN' operating mode after the restart, you must configure _startUpData.operationMode = "RUN".

Additional references

Further information is available at:

- SIMOTION D410-2 Commissioning and Hardware Installation Manual
- or in the online help

Return value: DINT

Description of the return values:

16#00000000	Function performed without errors.
16#FFFF8091	Functionality is not supported for this device. The _restart system function only supports synchronous command execution.

Note

In the case of a return of the system function and the executed functionality (return value 16#00000000), the user program is to ensure that there are no more axis motions or other possible undesired influences on the machine caused by the user program.

2.5.7.5 _setDeviceErrorLED

This function serves to set the error sublicensing (LED flashing).

Additional references

Additional information is available at:

- SIMOTION Basic Functions Function Manual
- or in the SIMOTION online help

Return value: DINT

Description of the return values:

16#00000000	LED set without errors.
16#FFFFFFFD	Internal error.

2.6 Task system

2.6.1 _disableScheduler

This function enables commands within a task to be executed in succession without being interrupted by the task system sequence.

The function disables all user tasks (except IPOsynchronousTask, IPOsynchronousTask_2, and the task for TController) until '_enableScheduler' is called.

It does not, however, affect system tasks.

The programs assigned to the disabled tasks are no longer executed.

The watchdog timer for cyclic tasks is not suspended.

This system function also prevents a switch to 'SystemInterruptTasks' and 'UserInterruptTasks'.

Additional references

Additional information is available at:

- SIMOTION Basic Functions Function Manual
  Section Overview of task control commands
- or in the SIMOTION online help.

Return value: DINT

This function is always executable; the return value = 0.

2.6.2 _enableScheduler

This function cancels the effect of '_disableScheduler'.

The disabled tasks are included in the task succession again.

Additional references

Additional information is available at:

- SIMOTION Basic Functions Function Manual
  Section Overview of task control commands
- or in the SIMOTION online help.

Return value: DINT

This function is always executable; the return value = 0.

2.6.3 _getCurrentTaskName

The system function returns the configured name of the task currently in progress as a string.

If the task name is available as a string, this can be processed directly in the user program.

Additional references

Further information about the system function

- In the online help

Return value: STRING

The function supplies a string as a return value.

The function returns an "empty string" if no task name is determined.

2.6.4 _getStateOfNTPClockSynchronisation

This function supplies the task name of the task in progress.

Additional references

Further information on the system function is available at:

- SIMOTION D410-2 Commissioning Manual
- SIMOTION D4x5-2 Commissioning Manual
- or in the SIMOTION online help

Return value: StructRetDeviceGetStateOfNTPClockSynchronisation

EnumStateOfNTPClockSync

synchronisationNotConfigured (0)	Functionality is not configured in SDBs
synchronisationNotActive (1)	NTP functionality is not active
clockIsNotSynchronous (2)	NTP clock is not synchronized
clockIsSynchronous (3)	NTP clock is synchronized

StructRetDeviceGetStateOfNTPClockSynchronisation
Return values for _getStateOfNTPClockSynchronisation

Structure	Name	Data type	Unit
functionResult	Result of the execution	DINT	-
info	Info about the status of the NTP synchronization	EnumStateOfNTPClock Sync	-
usingNTPClockServer	IP address of the NTP server	ARRAY [0..3] OF USINT	-

2.6.5 Diagnostics

2.6.5.1 _taskTraceStart

This function starts the Task Trace recording.

Additional references

Additional information is available at:

- Function Manual: Task Trace
  Section Control of Task Tracer
- or in the SIMOTION online help.

Return value: UDINT

Description of the return values:

16#00000000	The Task Tracer has been started.
16#FFFF80E2	The write task to the memory card is still active. The Task Tracer has not been started.

2.6.5.2 _taskTraceStop

This function stops an active Task Tracer recording.

Additional references

Additional information is available at:

- Function Manual: Task Trace
  Section Control of Task Tracer
- or in the SIMOTION online help.

Return value: UDINT

Description of the return values:

16#00000000	The Task Tracer has been stopped.
16#FFFF8111	The Task Tracer was already in the STOP status.

2.6.5.3 _taskTraceUserEvent

This functions enters an event with a user-defined ID from 0 to 255 in the buffer of the SIMOTION Task Tracer.

Additional references

Additional information is available at:

- Function Manual: Task Trace
  Section Control of Task Tracer
- or in the SIMOTION online help.

Return value: UDINT

Description of the return values:

16#00000000	A user event has been entered.
16#FFFF8101	A user event outside the valid range of 0 to 255 has been entered.
16#FFFF8102	The Task Tracer is not active. No user event has been entered in the buffer.

Parameters:

userevent

Direction:	Input parameter
Data type:	UDINT
Parameter index:	1

The ID flag of the trace event.

2.6.5.4 _taskTraceWriteout

The function stops a running Task Tracer recording and saves the trace data to the memory card.

Additional references

Additional information is available at:

- Function Manual: Task Trace
  Section Control of Task Tracer
- or in the SIMOTION online help.

Return value: UDINT

Description of the return values:

16#00000000	The data has been successfully written to the memory card.
16#00007001	The write task for the data backup to the memory card has been started.
16#00007002	The write task for the data backup to the memory card is still active.
16#FFFF80E3	An error occurred while writing to the memory card. Check whether sufficient free space is available on the memory card.

2.7 Technology

2.7.1 Shared functions

2.7.1.1 BEGIN_SYNC

This function is used to tag functions for synchronous execution up to the END_SYNC call. Synchronous execution is started with _startSyncCommands.

Additional references

Additional information is available at:

- Basic Functions function manual
  Section Synchronous start
- or in the SIMOTION online help.

Parameters:

SyncId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	1

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

This parameter specifies a 'syncCommandId' of type 'commandId' that is unique from all others in the system. The user can make the system generate a 'syncCommandId' by calling _getSyncCommandId. The same syncCommandId is used in the _startSyncCommands command.

2.7.1.2 END_SYNC

This function ends a command sequence intended for synchronous execution. Synchronous execution is started with _startSyncCommands.

Additional references

Additional information is available at:

- Basic Functions function manual
  Section Synchronous start
- or in the SIMOTION online help.

2.7.1.3 _activateTo

This system function enables activation of technology objects (TO). The functions of a technology object (e.g. _enableAxis on the TO axis) cannot be used until the object is ready for operation.

Note

The functions _activateTo, _deactivateTo and _getStateOfTo belong to a function pool. Maximum 50 active calls from this pool can be processed simultaneously.

The return value '16#FFFF80C3' is signaled when the maximum number is exceeded. Each running instance requires additional computation time in the technology object environment (in the cyclic levels); therefore it is more efficient to first query the respective state and then to set the required desired state by means of _activateTo and _deactivateTo.

Additional references

More detailed information on this system function can be found in

- the Function Manual titled "Motion Control Basic Functions for Modular Machines".

Return value: DINT

Description of the return values:

The required actions are asynchronous. The system function signals the status of the execution in the return code.

16#00000000	The technology object is activated.
16#00007000	The system function is in 'Idle' status.
16#00007001	The system function is started (only reported when execution is asynchronous).
16#00007002	The system function is running (only reported when execution is asynchronous, a new call in the control program is required until the functionality ends).
16#FFFF8090	Technology object not available.
16#FFFF8092	'"Activate TO" system function requested while "Deactivate TO" system function is running.
16#FFFF8098	Functionality terminated with timeout.
16#FFFF8099	Internal error, function terminated.
16#FFFF80C1	'Activate TO' is already started due to a new call of the system function with call parameter 'requestMode:=REQUEST_TRUE'.
16#FFFF80C3	Temporary bottleneck (internal resources for executing the function not currently available). The maximum number of 50 calls active simultaneously has been exceeded. Wait until an active call is completed.

Parameters:

TO_Instance

Direction:	Input parameter
Data type:	AnyObject
Parameter index:	1

Reference to the technology object to be activated.

reqActDeactGetStateMode

Direction:	Input parameter
Data type:	EnumReqActDeactGetStateMode
Parameter index:	2

EnumReqActDeactGetStateMode

REQUEST_TRUE (0)	Start functionality
REQUEST_FALSE (1)	Query status with REQUEST_TRUE
REQUEST_ABORT (2)	Abort of the function execution

REQUEST_TRUE starts the functionality. REQUEST_FALSE scans the status of the functionality started with REQUEST_TRUE. REQUEST_ABORT aborts the function execution.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	3

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	5

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and the function cannot be completed.

With WHEN_COMMAND_DONE, the transition takes place when the command has been executed. With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

2.7.1.4 _deactivateTo

This system function deactivates technology objects (TO).

You must also ensure that all technology objects that are dependent on the deactivated technology object are also deactivated.

For example:

A slave axis always requires a synchronous object. Therefore both technology objects must be deactivated.

The deactivation sequence should always start with the technology objects that are not dependent on other technology objects.

Note

The functions _activateTo, _deactivateTo and _getStateOfTo belong to a function pool. Maximum 50 active calls from this pool can be processed simultaneously.

The return value '16#FFFF80C3' is signaled when the maximum number is exceeded. Each running instance requires additional computation time in the technology object environment (in the cyclic levels); therefore it is more efficient to first query the respective state and then to set the required desired state by means of _activateTo and _deactivateTo.

Additional references

More detailed information on this system function can be found in

- the Function Manual titled "Motion Control Basic Functions for Modular Machines".

Return value: DINT

Description of the return values:

The required actions are asynchronous. The system function signals the status of the execution in the return code.

16#00000000	The technology object (TO) is deactivated.
16#00007000	The system function is in 'Idle' status (inactive).
16#00007001	The system function is started (only reported in asynchronous execution).
16#00007002	The system function is running (only reported when execution is asynchronous, a new call in the control program is required until the functionality ends).
16#FFFF8090	Invalid technology object reference.
16#FFFF8092	"Deactivate TO" requested while "Activate TO" is running.
16#FFFF8098	Functionality terminated with timeout.
16#FFFF8099	Internal error, function terminated.
16#FFFF80C1	'Deactivate TO' is already started. due to a new call of the system function with call parameter 'requestMode:=REQUEST_TRUE'.
16#FFFF80C3	Temporary bottleneck (internal resources for executing the function not currently available). The maximum number of 50 calls active simultaneously has been exceeded. Wait until an active call is completed.

Parameters:

TO_Instance

Direction:	Input parameter
Data type:	AnyObject
Parameter index:	1

Reference to the technology object to be deactivated.

reqActDeactGetStateMode

Direction:	Input parameter
Data type:	EnumReqActDeactGetStateMode
Parameter index:	2

EnumReqActDeactGetStateMode

REQUEST_TRUE (0)	Start functionality
REQUEST_FALSE (1)	Query status with REQUEST_TRUE
REQUEST_ABORT (2)	Abort of the function execution

REQUEST_TRUE starts the functionality. REQUEST_FALSE scans the status of the functionality started with REQUEST_TRUE. REQUEST_ABORT aborts the function execution.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	3

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	5

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and the function cannot be completed.

. With WHEN_COMMAND_DONE, the transition takes place when the command has been executed. With ABORT_CURRENT_COMMAND, the active function is aborted immediately.

2.7.1.5 _getObjectName

The system function returns the configured name of a technology object (TO) as a string. If the TO name is available as a string, this can be processed directly in the user program. When the function is called in the TechnologicalFaultTask, TSI#toInst can be transferred directly to the ToInst input parameter from the TaskStartInfo of the TechnologicalFaultTask.

Return value: STRING

The function supplies a string as a return value.

Parameters:

ToInst

Direction:	Input parameter
Data type:	AnyObject
Parameter index:	1

Reference to the technology object whose name is to be returned.

2.7.1.6 _getStateOfTo

This function supplies information on whether the technology object (TO) is activated or deactivated. It can be executed both synchronously and asynchronously.

Note

The functions _activateTo, _deactivateTo and _getstateOfTo belong to a function pool. Maximum 50 active calls from this pool can be processed simultaneously. The return value '16#FFFF80C3' is signaled when the maximum number is exceeded. Each running instance requires additional computation time in the technology object environment (in the cyclic levels); therefore it is more efficient to first query the respective state and then to set the required desired state by means of _activateTo and _deactivateTo.

Additional references

More detailed information on this system function can be found in

- the Function Manual entitled Motion Control Basic Functions for Modular Machines
- or in the SIMOTION online help.

Return value: StructRetGetStateOfTo

EnumStateOfTo

ACTIVE (1)	16#00000001 TO activated
INACTIVE (2)	16#00000002 TO deactivated
IN_ACTIVATION (3)	16#00000003 TO will be activated
IN_DEACTIVATION (4)	16#00000004 TO will be deactivated

EnumToSetStateOfTo

ACTIVATED (1)	16#00000001 TO activated
DEACTIVATED (2)	16#00000002 TO deactivated

StructRetGetStateOfTo
Return parameters for device commands

Structure	Name	Data type	Unit
functionResult	Return value of function	DINT	-
commandIdState	Current state of the technology object	EnumStateOfTo	-
setState	Desired state of the TO: ACTIVATED / DEACTIVATED	EnumToSetStateOfTo	-

Description of the return values

Return value "functionResult" has the following meaning:

16#00000000	The system function is finished. The result is in 'commandIdState'.

16#00007000	The system function is in 'Idle' status.
16#00007001	The system function is started (only reported when execution is asynchronous).
16#00007002	The system function is running (only reported when execution is asynchronous, a new call in the control program is required until the functionality ends).
16#FFFF8090	Technology object not available. Function not executed, all resources are released.
16#FFFF8091	No valid device, job aborted: Function not executed, all resources are released.
16#FFFF8098	Functionality terminated after timeout, Function not executed, all resources are released.
16#FFFF8099	Internal error, function terminated. Function not executed, all resources are released.
16#FFFF80C3	Temporary bottleneck (internal resources for executing the function not currently available). The maximum number of 50 calls active simultaneously has been exceeded. Wait until an active call is completed.

Parameters:

TO_Instance

Direction:	Input parameter
Data type:	AnyObject
Parameter index:	1

Reference to the technology object.

reqActDeactGetStateMode

Direction:	Input parameter
Data type:	EnumReqActDeactGetStateMode
Parameter index:	2

EnumReqActDeactGetStateMode

REQUEST_TRUE (0)	Start functionality
REQUEST_FALSE (1)	Query status with REQUEST_TRUE
REQUEST_ABORT (2)	Abort of the function execution

REQUEST_TRUE starts the functionality. REQUEST_FALSE scans the status of the functionality started with REQUEST_TRUE. REQUEST_ABORT aborts the function execution.

commandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	3

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

Unique, system-wide 'commandId' to track command status.

nextCommand

Direction:	Input parameter
Data type:	EnumNextCommandMode
Parameter index:	5

EnumNextCommandMode

IMMEDIATELY (60)	Asynchronous
WHEN_COMMAND_DONE (160)	Synchronous
ABORT_CURRENT_COMMAND (260)	Abort

Specifies condition for transition to next command. IMMEDIATELY means an initial call should be executed with REQUEST_TRUE. This starts the system function's internal sequence. Repeated calling of the system function is required with REQUEST_FALSE until the function confirms it has been completed. Unless these subsequent calls are made, the system function will not continue and the function cannot be completed.

With WHEN_COMMAND_DONE, the transition takes place when the command is finished. With ABORT_CURRENT_COMMAND and the same 'commandId' as in the previous function call with IMMEDIATELY, the active function is aborted.

2.7.1.7 _resetTechnologicalErrors

This function acknowledges all active technological alarms requiring acknowledgment.

Additional references

Additional information is available at:

- Basic Functions function manual
  Section Acknowledgement using a user program
- or in the SIMOTION online help.

Return value: DINT

This function is always executable; the return value = 0.

2.7.1.8 _startSyncCommands

This function starts the synchronous commands. By calling this function, the user initiates the one-time synchronous start of all commands that were specified between BEGIN_SYNC and END_SYNC with the appropriate SyncId.

Additional references

Additional information is available at:

- Basic Functions function manual
  Section Synchronous start
- or in the SIMOTION online help.

Return value: DINT

The return code indicates whether or not a synchronous start has been executed. If synchronous start was not executed, the reason is specified.

Description of the return values:

0 OK

200 No technology package found

-200 Synchronous start not executable

Parameters:

syncCommandId

Direction:	Input parameter
Data type:	CommandIdType
Parameter index:	1

CommandIdType
System-wide unique ID

Structure	Name	Data type	Unit
SystemId_low	Low_part of ID	UDINT	-
SystemId_high	High_part of ID	UDINT	-

This parameter specifies a 'syncCommandId' of type 'commandId' that is unique from all others in the system. The user can make the system generate a 'syncCommandId' by calling _getSyncCommandId.

System variables – devices

3

3.1 systemClock

Basic cycle of the RT system

Effective:	read only
Data type:	DINT
Update:	For change

Description

The 'systemClock' variable shows the basic cycle clock of the RT system. The variable is shown in the µsec. unit.
The variable is a read-only variable.

3.2 servoControlClock

Position control cycle clock

Effective:	read only
Data type:	DINT
Update:	For change

Description

The 'servoControlClock' variable shows the position control cycle clock. The variable is shown in the µsec. unit.
The variable is a read-only variable.

3.3 servoControlClock_fast

Fast position control cycle clock

Effective:	read only
Data type:	DINT

Description

The 'servoControlClock_fast' variable shows the fast position control cycle clock. The variable is shown in the µsec. unit.
The variable is a read-only variable.

3.4 ipoClock

Interpolation cycle clock

Effective:	read only
Data type:	DINT
Update:	For change

Description

The 'ipoClock' variable shows the interpolator cycle clock. The variable is shown in the µsec. unit.
The variable is a read-only variable.

3.5 ipoClock_fast

Fast interpolation cycle clock

Effective:	read only
Data type:	DINT

Description

The 'ipoClock_fast' variable shows the fast interpolation cycle clock. The variable is shown in the µsec. unit.
The variable is a read-only variable.

3.6 servoTaskCycle

Position controller cycle number

Effective:	read only
Data type:	UDINT
Update:	For change

Description

The 'servoTaskCycle' variable shows the position controller cycle number within an IPO cycle clock. The value of the 'servoTaskCycle' variables corresponds to the number of times the Servo task has been called within the current IPO cycle clock.

Example

A cycle clock of 1 ms is set for the Servo task and a cycle clock of 3 ms is set for the IPO task.
The 'servoTaskCycle' variable can be either 1, 2, or 3 depending on whether the Servo task has been started once, twice, or three times so far.

The 'servoTaskCycle' query is particularly useful in the Servo task: If the variable has the value 1, this shows that a new IPO cycle clock has begun.

The variable is a read-only variable.

3.7 servoTaskCycle_fast

Fast position controller cycle number

Effective:	read only
Data type:	UDINT
Update:	For change

Description

The 'servoTaskCycle_fast' variable shows the fast position controller cycle number within a fast IPO cycle clock.
The value of the 'servoTaskCycle_fast' variables corresponds to the number of times the Servo task has been called within the current IPO cycle clock.

Example

A cycle clock of 1 ms is set for the Servo task and a cycle clock of 3 ms is set for the IPO task.
The 'servoTaskCycle_fast' variable can be either 1, 2, or 3 depending on whether the Servo task has been started once, twice, or three times so far.

The 'servoTaskCycle_fast' query is particularly useful in the Servo task: If the variable has the value 1, this shows that a new IPO cycle clock has begun.

The variable is a read-only variable.

3.8 traceControl

TIO functions control

Effective:	immediately
Data type:	ARRAY [0..1] OF StructTraceControl
Update:	For change

Additional references

Further information about trace is available at:

- See the SIMOTION MCC Programming and Operating Manual
- or in the SIMOTION online help.

Description

StructTraceControl
TIO functions control

Structure	Name	Data type	Unit
fctGenEnable	Enable function generator	EnumEnableDisable	-
fctGenStart	Start function generator	EnumFctGenStartStop	-
traceStart	-	EnumStartStop	-
traceTrigger	Trace trigger signal	EnumStartStop	-

EnumEnableDisable

DISABLE (42)	Enable reset
ENABLE (48)	Enable set

EnumFctGenStartStop

FCTGEN_START (1)	Start
FCTGEN_START_FG1 (2)	Synchronous start with function generator 1
FCTGEN_START_FG2 (3)	Synchronous start with function generator 2
FCTGEN_STOP (134)	Stop

EnumStartStop

START (1)	Start
STOP (134)	Stop

The function generator is controlled by means of the variables of 'trace Control'.
The parameters in the function generator are set during configuration.
'fctGenStart:=ENABLE' enables the function generator.
'fctGenStart:=DISABLE' disables or cancels the function generator.
'fctGenStart:=FCTGEN_START' starts the function generator.
'fctGenStart:=FCTGEN_STOP' stops or cancels the function generator.
'fctGenStart:=FCT_GEN_START_FG1' starts the function generator 2 synchronously to function generator 1.
Function generator 1 is started synchronously with function generator 2 with 'fctGenStart:= FCT_GEN_START_FG2'.
The 'traceStart' variable is no longer supported.
The stop-start edge of the 'traceTrigger' variable can be selected in the SIMOTION SCOUT trace tool as the trigger condition for the trace recording.
The variables can be read and written.

3.9 traceState

TIO functions status

Effective:	read only
Data type:	ARRAY [0..1] OF StructTraceState
Update:	For change

Additional references

Further information about trace is available at:

- See the SIMOTION MCC Programming and Operating Manual
- or in the SIMOTION online help.

Description

StructTraceState
TIO functions status

Structure	Name	Data type	Unit
fctGenState	Function generator status	EnumFctGenState	-
traceState	-	EnumTraceState	-

EnumFctGenState

FCTGEN_DISABLED (0)	The function generator is waiting for a parameterization
FCTGEN_PARAM_VALID (1)	The function generator is correctly parameterized and is now waiting for enable
FCTGEN_ENABLED (2)	The function generator is ready to start

FCTGEN_WAIT_FG1 (3)	The function generator is waiting for function generator 1 to start
FCTGEN_WAIT_FG2 (4)	The function generator is waiting for function generator 2 to start
FCTGEN_RUNNING (5)	The function generator is running
FCTGEN_STARTING (6)	The function generator travels the offset ramp (0 -> offset)
FCTGEN_STOPPING (7)	The function generator travels the offset ramp (offset -> 0)
FCTGEN_LIMIT_EXCEEDED (8)	The function generator was cancelled because the system variable limits were exceeded

EnumTraceState

TRACE_INACTIVE (0)	Trace inactive
TRACE_RUNNING (1)	The recording is running
TRACE_WAIT_FOR_TRIGGER (2)	Wait for trigger condition
TRACE_FINISHED (3)	The recording has completed
TRACE_MISMATCH (4)	The recording has failed
TRACE_PARAM_VALID (5)	Parameterization valid
TRACE_ABORTED (6)	Recording cancelled (because of performance bottleneck)
TRACE_NO_TIME (7)	Trace deactivated because of acute performance bottleneck at the moment

The 'traceState.fctGenState' variable shows the status of the function generator.
The 'traceState.traceState' variable is no longer supported.
The variables are read-only variables.

3.10 TCOFctGenOverride

Function generator override

Effective:	immediately
Data type:	ARRAY [0..1] OF StructTCOFctGenOverride
Update:	For change

Description

StructTCOFctGenOverride
Function generator override

Structure	Name	Data type	Unit
amplitude	Amplitude override	LREAL	%
offset	Offset override	LREAL	%

System settings TCOFctGenOverride[2]

Parameter name	min. value	Factory setting	max. value	Unit	Axis type
amplitude	0.0	100.0	1000.0	[%]	-
offset	0.0	100.0	1000.0	[%]	-

The amplitude of the function generator signal specified in the parameter assignment and the offset are weighted with the override. The override is specified as a percentage.

3.11 userData

User data

Effective:	immediately
Data type:	StructUserData
Update:	For change

Description

StructUserData
User data

Structure	Name	Data type	Unit
user1	User data 1	DINT	-
user2	User data 2	DINT	-
user3	User data 3	DINT	-
user4	User data 4	DINT	-
user5	User data 5	DINT	-

user6	User data 6	DINT	-
user7	User data 7	DINT	-
user8	User data 8	DINT	-

System settings userData

Parameter name	min. value	Factory setting	max. value	Unit	Axis type
user1	-2147483648	0	2147483647	-	-
user2	-2147483648	0	2147483647	-	-
user3	-2147483648	0	2147483647	-	-
user4	-2147483648	0	2147483647	-	-
user5	-2147483648	0	2147483647	-	-
user6	-2147483648	0	2147483647	-	-
user7	-2147483648	0	2147483647	-	-
user8	-2147483648	0	2147483647	-	-

The structural elements for 'userData' are system variables that are reserved for the user. For example, cyclical values can be entered in these elements and then recorded using the trace function.

3.12 taskRuntimeMonitoring

Measurement of the task run times

Effective:	immediately
Data type:	EnumEnableDisable
Update:	For change

Additional references

Further information about task run times can be found in the

- SIMOTION Basic Functions Function Manual
- SIMOTION SCOUT Configuration Manual
- or in the online help

Description

EnumEnableDisable

DISABLE (42)	Enable reset
ENABLE (48)	Enable set

Measurement of task runtimes can be activated using the 'taskRuntimeMonitoring' system variable.

3.13 taskRuntime

Task run times

Effective:	read only
Data type:	StructTaskRuntimeType
Update:	For change

Additional references

Additional information about task run times is available at:

- SIMOTION Basic Functions Function Manual
- SIMOTION SCOUT Configuration Manual, device diagnostics: Task manager
- or in the online help

Description

StructTaskRuntimeType
Tasks

Structure	Name	Data type	Unit
servo_fast	ServoFast task	StructTaskRuntimeValues	-
ipo_fast	IpoFast task	StructTaskRuntimeValues	-
servo	Servo task	StructTaskRuntimeValues	-
ipo	IPO task	StructTaskRuntimeValues	-
ipo_2	IPO task 2	StructTaskRuntimeValues	-
servoSynchronous_fast	Servo-synchronous task	StructTaskRuntimeValues	-
servoSynchronous	Servo-synchronous task	StructTaskRuntimeValues	-
ipoSynchronous	IPOsynchronous task	StructTaskRuntimeValues	-
ipoSynchronous_fast	IPOFastsynchronous task	StructTaskRuntimeValues	-
ipoSynchronous_2	IPOsynchronous task 2	StructTaskRuntimeValues	-
servoDcc	DCC task servo	StructTaskRuntimeValues	-
ipoDcc	DCC task IPO	StructTaskRuntimeValues	-
ipoDcc_2	DCC task IPO_2	StructTaskRuntimeValues	-

DccAux	DCC task free level 1	StructTaskRuntimeValues	-
DccAux_2	DCC task free level 2	StructTaskRuntimeValues	-
background	Background task	StructTaskRuntimeValues	-

StructTaskRuntimeValues
Task runtimes

Structure	Name	Data type	Unit
maximalValue	Maximum	UDINT	µs
meanValue	Average	UDINT	µs
actualValue	Current	UDINT	µs

The 'taskRuntime' system variable contains the task runtimes.

Note

System tasks system behavior

When a module is in the STOP operating state, the system tasks still use computing time.

3.14 effectiveTaskRuntime

Level run times

Effective:	read only
Data type:	StructEffectiveTaskRuntimeType
Update:	For change

Additional references

Further information about task run times, time and level overflows, and device diagnostics is available in:

- SIMOTION Basic Functions Function Manual
- Overview of Service and Diagnostics Options, Product Information
- or in the online help

Description

StructEffectiveTaskRuntimeType
Levels

Structure	Name	Data type	Unit
servo_fast	ServoFast level	StructTaskRuntimeValues	-
ipo_fast	IPOFast level	StructTaskRuntimeValues	-
servo	Servo level	StructTaskRuntimeValues	-
ipo	IPO level	StructTaskRuntimeValues	-
ipo_2	IPO level 2	StructTaskRuntimeValues	-

DccAux	DCC free level 1	StructTaskRuntimeValues	-
DccAux_2	DCC free level 2	StructTaskRuntimeValues	-
background	Background level task	StructTaskRuntimeValues	-

StructTaskRuntimeValues
Task runtimes

Structure	Name	Data type	Unit
maximalValue	Maximum	UDINT	µs
meanValue	Average	UDINT	µs
actualValue	Current	UDINT	µs

The 'effectiveTaskRuntime' system variable contains the level runtimes of the ServoFast, IpoFast, servo, Ipo, Ipo2, DccAux/2 and background levels.

'effectiveTaskRuntime' supplies the system variables with values from the system control only for 'TaskRuntimeMonitoring:=ENABLE'.

3.15 systemLoad

CPU loading caused by cyclical tasks

Effective:	read only
Data type:	StructSystemLoad
Update:	For change

Additional references

Further information about the execution system, execution levels, system tasks and system cycle clocks can be found in:

- SIMOTION Basic Functions Function Manual
- SIMOTION Communication System Manual
- SIMOTION SCOUT Configuration Manual
- Technology Objects Synchronous Operation, Cam Function Manual
- Output Cams and Measuring Inputs Function Manual
- Function Manual "TO Axis Electric/Hydraulic, External Encoder".
- or in the online help

Description

StructSystemLoad
Tasks

Structure	Name	Data type	Unit
servo_fast	ServoFast task	UDINT	-
servoSynchronous_fast	ServoFast synchronous task	UDINT	-
ipo_fast	IpoFast task	UDINT	-
ipoSynchronous_fast	IpoFast synchronous task	UDINT	-
servo	Servo task	UDINT	-
ipo	IPO task	UDINT	-
ipo_2	IPO task 2	UDINT	-
servoSynchronous	Servo-synchronous task	UDINT	-
ipoSynchronous	IPOsynchronous task	UDINT	-
ipoSynchronous_2	IPOsynchronous task 2	UDINT	-
ipoServo	Cyclical task	UDINT	-

The 'ipoServoLoad' system variable contains the CPU loading (as a percentage) by the servo, Ipo, IpoSync, Ipo2, and Ipo2Sync tasks.

3.16 numberOfSummarizedTaskOverflow

Number of level overflows that have occurred

Effective:	read only
Data type:	StructTaskOverflowType
Update:	For change

Additional references

Further information about time and level overflows is available at:

- SIMOTION Basic Functions Function Manual
- or in the online help

Description

StructTaskOverflowType
Tasks

Structure	Name	Data type	Unit
ipoTask	IPO task	UDINT	-
ipoTask_2	IPO task 2	UDINT	-
ipoSynchronousTask	IPOsynchronous task	UDINT	-
ipoSynchronousTask_2	IPOsynchronous task 2	UDINT	-
ipoDcc	DCC task IPO	UDINT	-
ipoDcc_2	DCC task IPO2	UDINT	-
DccAux	DCC task free level 1	UDINT	-
DccAux_2	DCC task free level 2	UDINT	-

The 'numberOfSummarizedTaskOverflow' system variable contains the level overflows that occurred since the last STOP to STOP_U transition as well as the time overflows of the synchronous user tasks that occurred.
The variable is a read-only variable.

3.17 ipoClock_2

Interpolation cycle clock 2

Effective:	read only
Data type:	DINT
Update:	For change

Description

The 'ipoClock_2' variable shows interpolator cycle clock 2. The variable is shown in the µsec. unit.
The variable is a read-only variable.

3.18 modeOfDpInterfaceSynchronization

Display of the DP synchronization mode

Effective:	read only
Data type:	EnumDpInterfaceSyncMode
Update:	For change

Additional references

Further information about the synchronization of a SIMOTION device without isochronous DP master interface can be found at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Description

EnumDpInterfaceSyncMode

NO_ALARMMESSAGES (0)	Alarm messages not enabled
MASTER_SLAVE_ALARMMESSAGES_1 (1)	Alarm messages for DP-master-slave synchronization activated
SLAVE_ALARMMESSAGES_1 (2)	Alarm messages for DP-slave synchronization activated
AUTOMATIC_INTERFACE_SYNCHRONIZATION (3)	Automatic DP-master-slave synchronization activated

System variable 'modeOfDpInterfacesSynchronization' shows which synchronization mechanisms (alarm messages) are activated or currently available for use. The system variable matches the parameter passed to system function _enableDpInterfaceSynchronizationMode if this system function has been executed correctly.

The NO_ALARMMESSAGES system default is set when the _enableDpInterfaceSynchronizationMode system function was not called.
The variable is a read-only variable.

With NO_ALARMESSAGES, no synchronization-specific alarm messages are enabled (default value)

For MASTER_SLAVE_ALARMMESSAGES_1, the following events for initiating the PeripherialFaultTask are enabled:

- _SC_DP_CLOCK_DETECTED
- _SC_DP_SYNCHRONIZATION_LOST

For SLAVE_ALARMMESSAGES_1, the following events for initiating the PeripherialFaultTask are enabled:

- _SC_DP_SLAVE_SYNCHRONIZED
- _SC_DP_SLAVE_NOT_SYNCHRONIZED
- AUTOMATIC_INTERFACE_SYNCHRONIZATION

The following events for initiating the PeripherialFaultTask are enabled:

_SC_DP_SYNCHRONIZATION_LOST.

If one DP interface is configured as an equidistant DP slave and the other DP interface is configured as an equidistant DP master, then synchronization is performed automatically with the incoming or existing master.

3.19 stateOfDpInterfaceSynchronization

Current state of the interface synchronization

Effective:	read only
Data type:	EnumDpInterfaceSyncState
Update:	For change

Additional references

Further information about the synchronization of a SIMOTION device with isochronous DP master interface can be found at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Description

EnumDpInterfaceSyncState

WAIT_FOR_DP_CLOCK (1)	Wait for cycle clock on the DP-Slave interface
DP_CLOCk_DETECTED (2)	Cycle clock on the DP-Slave interface was recognized
DP_INTERFACES_SYNCHRONIZED (3)	DP-Master interface is synchronized to the cycle clock on the DP-Slave interface

System variable 'stateOfDpInterfaceSynchronization' shows the current internal status of the interface synchronization.
The variable is a read-only variable.

3.20 stateOfDpSlaveSynchronization

Current state of the DP-Slave synchronization

Effective:	read only
Data type:	EnumDpSlaveSyncState
Update:	For change

Additional references

Further information about the synchronization of a SIMOTION device without isochronous DP master interface can be found at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- Further information about the synchronization of the interfaces (distributed synchronous operation) can be found at:
- Technology Objects Synchronous Operation, Cam Function Manual
- or in the online help

Description

EnumDpSlaveSyncState

DP_SLAVE_SYNCHRONIZED (0)	The DP-Slave and the cycle clock are synchronous to each other on the DP-Slave interface
DP_SLAVE_NOT_SYNCHRONIZED (1)	The DP-Slave and the cycle clock are not synchronous to each other on the DP-Slave interface

System variable 'stateOfDpSlaveSynchronization' shows the current internal status of the DP slave synchronization.
The variable is a read-only variable.

The following applies for P320-4/C230:

'stateOfDpSlaveSynchronization' is used if an isochronous slave interface is available and the master interface is not isochronous.

The following applies to D4xx:

'stateOfDpSlaveSynchronization' is irrelevant to D4xx. This is because an isochronous master is available and, as a result, system variable 'stateOfDpInterfaceSynchronization' is always used.

3.21 persistentDataPowerMonitoring

Buffer voltage monitoring for persistent data

Effective:	immediately
Data type:	structPersistentDataPowerMonitoringtype
Update:	For change

Additional references

For further information about the system variable, refer to the Commissioning and Hardware Installation Manuals:

- SIMOTION D4x5
- SIMOTION D410
- or in the online help

Description

structPersistentDataPowerMonitoringtype
Voltage error states on the SRAM

Structure	Name	Data type	Unit
powerFailure	Voltage error too low, possible data loss	EnumYesNo	-
persistentDataState	Read the persistent data	EnumPersistentDataState	-
warningBatteryVoltageLevel1	Battery voltage under the prewarning level	EnumYesNo	-
warningBatteryVoltageLevel2	Battery voltage under the warning level	EnumYesNo	-
rtcFailure	Time content (RTC) invalid	EnumYesNo	-
retainDataFailure	Retain data content (RTC) invalid	EnumYesNo	-

EnumYesNo

NO (91)	No
YES (173)	Yes

EnumPersistentDataState

FROM_RAM (1)	Data from the SRAM
FROM_FILE (2)	Data from the file (MemCard)
FROM_BACKUP (3)	Data from the backup file (data in the file is invalid)
INVALID (4)	Data in the SRAM and in file/backup are invalid or not present / deleted, system will create the initial values

The structure elements for 'persistentDataPowerMonitoring' show the error states for voltage on the SRAM or in the case of the C230-2 on the NV-RAM.

When 'powerFailure=NO': No error. System default is NO because C230-2 does not require a buffer battery for retentive data.
When 'powerFailure=YES', buffer voltage (super capacitor or battery) is too low following activation. Possible data loss in the persistent data/RTC (realTime, clock) is updated only during startup. Bit remains at YES until it is reset by the user program.

'persistentDataState' specifies with which persistent data the SIMOTION D435 is started; will only be updated in the start up.
FromRam: "Data from the SRAM"
FromFile: Data from file (memory card)
FromBackup: Data from the backup file (data in the file is invalid)
Invalid: Data in the SRAM and in the file/backup is invalid or unavailable/deleted, system is generating initial values.

If 'warningBatteryVoltageLevel1=NO': No error. System default is NO because C230-2 does not require a buffer battery for retentive data.
When 'warningBatteryVoltageLevel1=YES': Battery voltage has fallen below the early warning level; when this occurs, it is updated cyclically.

When 'warningBatteryVoltageLevel2=NO': No error. System default is NO because C230-2 does not require a buffer battery for retentive data.
When 'warningBatteryVoltageLevel2=YES': Battery voltage has fallen below the warning level; when this occurs, it is updated cyclically.

3.22 _upsData

Status of a connected UPS

Effective:	read only
Data type:	StructDeviceUpsData
Update:	For change

Description

StructDeviceUpsData
UPS states

Structure	Name	Data type	Unit
state	Current state of the UPS	EnumUpsState	-
batteryState	Current state of the UPS battery	EnumUpsBatteryState	-

EnumUpsState

BUFFERED (1)	Input voltage of the UPS missing, waiting period running
FAILED (2)	Input voltage of the UPS missing, waiting period expired; system will be shut down
NOTSUPP (3)	UPS functionality not present
O_K_ (102)	Input voltage at the UPS OK

EnumUpsBatteryState

WARNING (1)	Batteries should be replaced
CRITICAL (2)	Batteries are no longer operational
NOTSUPP (3)	UPS functionality not present
O_K_ (102)	Batteries are OK

The '_upsData' variable shows the state of a connected UPS.
'state' returns the state of the UPS.
'batteryState' returns the state of the UPS batteries.

Note

SIMOTION P320-4 no longer supports the _upsData.state and _upsData.batterystate system variables as of version V4.5.
Both variables are set permanently to NOT_SUPPORTED.

3.23 _cpuData

CPU-relevant data

Effective:	read only
Data type:	StructDeviceCpuData
Update:	For change

Additional references

For further information about the system variable, refer to the Commissioning and Hardware Installation Manuals:

- SIMOTION P
- SIMOTION D
- or in the online help

Description

StructDeviceCpuData
CPU states (read-only)

Structure	Name	Data type	Unit
fanRpm	Current fan speed	DINT	-
temperature	Current CPU temperature	DINT	-
moduletemperature	Current temperature in the HW module	DINT	-

The '_cpuData' variable shows the following values:
'fanRpm' shows the current constant fan speed (applies to SIMOTION D410-2).

'temperature' shows the current temperature of the processor (applies to SIMOTION P320-4).
'moduletemperature' shows the current temperature inside the HW module (applies to SIMOTION C/D/P).

The variable is a read-only variable.

3.24 _cpuDataRW

CPU-relevant data

Effective:	immediately
Data type:	StructDeviceCpuDataRW
Update:	For change

Description

StructDeviceCpuDataRW
CPU states

Structure	Name	Data type	Unit
fanWarning	Current state of the processor fan	EnumYesNo	-
redundantFanWarning	Current state of the redundant processor fan	EnumYesNo	-

EnumYesNo

NO (91)	No
YES (173)	Yes

The 'cpuData' variable shows the CPU-relevant data.
'fanWarning' indicates whether or not the current processor fan speed has fallen below a minimum (HW dependent) value (value changes to YES). The value must be reset to NO by the application.

The variables can be read and written.

3.25 fanbattery

Battery and fan states (read-write)

Effective:	immediately
Data type:	StructDeviceFanBattery
Update:	For change

Description

StructDeviceFanBattery
Battery and fan states (read-only)

Structure	Name	Data type	Unit
fannecessary	Fan required	EnumFanBattery	-
fanexisting	Fan present	EnumFanBattery	-
batterynecessary	Battery required	EnumFanBattery	-
batteryexisting	Battery present	EnumFanBattery	-

EnumFanBattery

MANDATORY (0)	Required
OPTIONAL (1)	Optional
OPTIONAL_RTC (2)	Optional for the RTC buffering
SINGLE (3)	Present once
REDUNDANT (4)	Present redundant
EXISTING (5)	Present
NOT_EXISTING (6)	Not present
NOT_MANDATORY (7)	Not required

The variable 'fanbattery' indicates whether the battery and the fan are required and whether they are installed.

3.26 _startupData

Settings for startup behavior - including operating mode after a power on / restart of SIMOTION

Effective:	immediately
Data type:	StructDeviceStartUp
Update:	For change

Description

StructDeviceStartUp
Settings for startup behavior - including operating mode after a power on/restart of SIMOTION

Structure	Name	Data type	Unit
operationMode	Operating state after power on / SIMOTION restart	EnumDeviceStartupOperationMode	-

EnumDeviceStartupOperationMode

LAST_OPERATION_MODE (0)	Last operating state in SIMOTION
RUN (1)	Automatically switch SIMOTION to RUN

The '_startup' (structure) system variable contains the configuration of the device functionality in SIMOTION.

The variable can be changed in SIMOTION SCOUT when it is in the "Offline" mode.
The values set there and stored in the project are downloaded to the SIMOTION device.
Write accesses (from the user program or via online accesses from SIMOTION SCOUT/OP/HMI) do not change the contents of the variable; no exceptions occur either.

3.27 _configurationManagement

Device functionality of SIMOTION configuration

Effective:	immediately
Data type:	StructDeviceConfigurationManagement
Update:	For change

Additional references

Further information about the retention of permanent data for a configuration change can be found at:

- SIMOTION Basic Functions for Modular Machines Function Manual
- or in the online help

Description

StructDeviceConfigurationManagement
Configure device behavior structure

Structure	Name	Data type	Unit
preserveToRetainData	Test whether RetainData TO after loading new Configurations TO must be initialized to Names TO	EnumYesNo	-

EnumYesNo

NO (91)	No
YES (173)	Yes

The '_configurationManagement' (structure) system variable contains the configuration of the device functionality in SIMOTION.

The variable can be changed in SIMOTION SCOUT when it is in the "Offline" mode. The values set there and stored in the project are downloaded to the SIMOTION device.

Write accesses (from the user program or via online accesses from SIMOTION SCOUT/OP/HMI) do not change the contents of the variable; no exceptions occur either.

3.28 modeOfOperation

Indication of the operating state

Effective:	immediately
Data type:	EnumDeviceModeOfOperation
Update:	For change

Additional references

Further information about CPU entering STOP - display and change mode can be found at:

- SIMOTION Basic Functions Function Manual
- or in the online help

Description

EnumDeviceModeOfOperation

_STOP (0)	STOP mode
_STOPU (1)	STOPU mode
_STARTUP (2)	STARTUP mode
_RUN (3)	RUN mode
_SHUTDOWN (4)	SHUTDOWN mode
_MRES (5)	MRES mode
_SERVICE (6)	SERVICE mode

The 'modeOfOperation' system variable indicates the current operating mode.
The mode can be changed by writing to the variables.
The values _STOP, _STOPU, and _RUN are permitted.

3.29 dccAuxClock

DccAux clock

Effective:	read only
Data type:	DINT
Update:	For change

Description

The 'dccAuxClock' variable shows the DccAux clock. The variable is shown in the µsec. unit.
The variable is a read-only variable.

3.30 dccAux_2Clock

Display of the DccAux_2 clock

Effective:	read only
Data type:	DINT
Update:	For change

Description

The 'dccAux_2 clock' variable shows dccAux clock 2. The variable is shown in the µsec. unit.
The variable is a read-only variable.

3.31 _imData

IM data of device

Effective:	read only
Data type:	StructDeviceImData
Update:	For change

Description

StructDeviceImData
IM parameter on device

Structure	Name	Data type	Unit
im0	IM0 parameter	StructDeviceIm0	-

StructDeviceIm0
IM0 parameter on device

Structure	Name	Data type	Unit
manufacturerId	Manufacturer ID	UINT	-
hardwareRevision	HW version	UINT	-
softwareRevision	SW version	StructDeviceIm0Softwareversion	-

revisionCounter	CM stamp	UDINT	-
imVersion	Specification version	StructDeviceIm0Version	-
imSupported	IM support	UINT	-

The 'hardwareRevision' variable contains the product version of the SIMOTION device.

The product versions are assigned as follows:

A->1, B->2, C->3, ...

The 'hardwareRevision' system variable can only be specified for SIMOTION D4xx-2 and SIMOTION C.

StructDeviceIm0Softwareversion
IM0 parameter on device

Structure	Name	Data type	Unit
typeRecognition	1st letter in the version name, e.g. 'V' as ASCII	USINT	-
functionalEnhancement	Major version number	USINT	-
bugfix	Minor version number	USINT	-
internalChange	Internal number	USINT	-

StructDeviceIm0Version
IM0 version parameter on device

Structure	Name	Data type	Unit
majorVersionNumber	majorVersionNumber	USINT	-
minorVersionNumber	minorVersionNumber	USINT	-

The variables are read-only variables.

3.32 _driveStates

Status of the drives

Effective:	read only
Data type:	StructCpuDriveStates
Update:	For change

Description

StructCpuDriveStates
Status of the drives (read-only)

Structure	Name	Data type	Unit
allClocksSynchronized	Status of the clock synchronization for the drives	EnumYesNo	-

EnumYesNo

NO (91)	No
YES (173)	Yes

The '_driveStates' variable shows the following values.
'allClocksSynchronized' indicates that the clock synchronization for all drives is active. This is relevant only for SINAMICS drives.

The variable is a read-only variable.

3.33 _automaticRestart

As long as this variable is not equal to zero, the device will execute an 'automatic restart' when crashes are identified ('general Protection', for example).

Effective:	immediately
Data type:	UDINT
Update:	For change

Additional references

Additional information is available at:

- SIMOTION D410-2 Commissioning and Hardware Installation Manual
  Section: Automatic restart after FAULT state
- or in the online help

Description

System settings _automaticRestart

Parameter name	min. value	Factory setting	max. value	Unit	Axis type
	0	0	0xffffffff	-	-

Note

System variable _automaticRestart is visible on all SIMOTION devices, but the functionality is only supported by SIMOTION D410-2.

Note

Restart cannot be ensured in all cases, for example, if a hardware fault occurs.

Note

System variable _automaticRestart is not backed up with the system function _savePersistentMemoryData

An automatic restart can be caused by hardware problems. This cause is not to be transferred to the new module if a module is replaced.

The user program decides for itself under what condition the system variable _automaticRestart will be set to a certain value.

The system variable controls the 'automatic restart' functionality after the occurrence of the FAULT operating mode.

16#00000000	Automatic restart after FAULT state is not active. (default)
16#00000000 < n < 16#FFFFFFFF	Automatic restart after FAULT state is active until the value 0 is reached. On a restart after FAULT state, the value is decremented by 1.
16#FFFFFFFF	Automatic restart after FAULT state. The value remains unchanged.

The variable can be written and read.

3.34 _simulation

Detection of simulated modules

Effective:	read only
Data type:	EnumActiveInactive
Update:	For change

Description

EnumActiveInactive

ACTIVE (4)	Active
INACTIVE (61)	Inactive

The _simulation system variable is used to detect simulated SIMOTION modules. On simulated modules, this variable has the value ACTIVE, on real modules, the value INACTIVE.

The variable is a read-only variable.

3.35 PCShutdown

Shutdown PC

Effective:	immediately
Data type:	EnumYesNo
Update:	For change

Description

EnumYesNo

NO (91)	No
YES (173)	Yes

If the variable is set, the PC is shut down.
This system variable does not apply to the SIMOTION P.