XPS Controller

Universal High-Performance Motion Controller/Driver

Programmer's Manual
V2.6.x

Spectra-Physics
A Division of Newport Corporation

Newport®
Experience | Solutions

Confidentiality & Proprietary Rights

Reservation of Title:

The Newport Programs and all materials furnished or produced in connection with them ("Related Materials") contain trade secrets of Newport and are for use only in the manner expressly permitted. Newport claims and reserves all rights and benefits afforded under law in the Programs provided by Newport Corporation.
Newport shall retain full ownership of Intellectual Property Rights in and to all development, process, align or assembly technologies developed and other derivative work that may be developed by Newport. Customer shall not challenge, or cause any third party to challenge, the rights of Newport.

Preservation of Secrecy and Confidentiality and Restrictions to Access:

Customer shall protect the Newport Programs and Related Materials as trade secrets of Newport, and shall devote its best efforts to ensure that all its personnel protect the Newport Programs as trade secrets of Newport Corporation. Customer shall not at any time disclose Newport's trade secrets to any other person, firm, organization, or employee that does not need (consistent with Customer's right of use hereunder) to obtain access to the Newport Programs and Related Materials. These restrictions shall not apply to information (1) generally known to the public or obtainable from public sources; (2) readily apparent from the keyboard operations, visual display, or output reports of the Programs; (3) previously in the possession of Customer or subsequently developed or acquired without reliance on the Newport Programs; or (4) approved by Newport for release without restriction.

2003 Newport Corporation
1791 Deere Ave.
Irvine, CA 92606, USA
(949) 863-3144

Table of Contents

1. TCP/IP COMMUNICATION 8

1.1. DESCRIPTION 8
1.2. FUNCTION DESCRIPTION 9
1.2.1. CloseAllOtherSockets 9
1.2.2. Login 11
1.2.3. OpenConnection 13
1.2.4. TCP_CloseSocket 14
1.2.5. TCP_ConnectToServer 15
1.2.6. TCP_GetError 17
1.2.7. TCP_SetTimeout 19

2. XPS FEATURES 21

2.1. GENERAL FEATURES 21
2.1.1. ControllerMotionKernelTimeLoadGet (NEW) 21
2.1.2. ControllerStatusGet 24
2.1.3. ControllerStatusStringGet 26
2.1.4. DoubleGlobalArrayGet 28
2.1.5. DoubleGlobalArraySet 30
2.1.6. ErrorStringGet 32
2.1.7. ElapsedTimeGet 34
2.1.8. FirmwareVersionGet 36
2.1.9. GroupStatusStringGet 38
2.1.10. GlobalArrayGet 40
2.1.11. GlobalArraySet 42
2.1.12. KillAll 44
2.1.13. Reboot 46
2.1.14. TimerGet 48
2.1.15. TimerSet 50
2.2. POSITIONER 52
2.2.1. Description 52
2.2.2. Object structure 52
2.2.3. Definition of the different positions for a positioner 53
2.2.4. Function description 54
2.2.4.1. PositionerAccelerationAutoScaling 54
2.2.4.2. PositionerAnalogTrackingPositionParametersGet 57
2.2.4.3. PositionerAnalogTrackingPositionParametersSet 60
2.2.4.4. PositionerAnalogTrackingVelocityParametersGet 63
2.2.4.5. PositionerAnalogTrackingVelocityParametersSet 66
2.2.4.6. PositionerBacklashDisable 69
2.2.4.7. PositionerBacklashEnable 71
2.2.4.8. PositionerBacklashGet 73
2.2.4.9. PositionerBacklashSet 75
2.2.4.10. PositionerCorrectorAutoTuning 77
2.2.4.11. PositionerCorrectorNotchFiltersGet 80
2.2.4.12. PositionerCorrectorNotchFiltersSet 83
2.2.4.13. PositionerCorrectorPIDDualFFVoltageGet 86
2.2.4.14. PositionerCorrectorPIDDualFFVoltageSet 89
2.2.4.15. PositionerCorrectorPIDFFAccelerationGet 93
2.2.4.16. PositionerCorrectorPIDFFAccelerationSet 96
2.2.4.17. PositionerCorrectorPIDFFVelocityGet 100
2.2.4.18. PositionerCorrectorPIDFFVelocitySet 103
2.2.4.19. PositionerCorrectorPIPositionGet 107
2.2.4.20. PositionerCorrectorPIPositionSet 110
2.2.4.21. PositionerCorrectorTypeGet 113
2.2.4.22. PositionerCurrentVelocityAccelerationFiltersGet 115

2.2.4.23. PositionerCurrentVelocityAccelerationFiltersSet 117
2.2.4.24. PositionerDriverFiltersGet (NEW) 119
2.2.4.25. PositionerDriverFiltersSet (NEW) 121
2.2.4.26. PositionerDriverPositionOffsetsGet (NEW) 124
2.2.4.27. PositionerDriverStatusGet 126
2.2.4.28. PositionerDriverStatusStringGet 128
2.2.4.29. PositionerEncoderAmplitudeValuesGet 130
2.2.4.30. PositionerEncoderCalibrationParametersGet 133
2.2.4.31. PositionerErrorGet 136
2.2.4.32. PositionerErrorRead 138
2.2.4.33. PositionerErrorStringGet 140
2.2.4.34. PositionerExcitationSignalGet 142
2.2.4.35. PositionerExcitationSignalSet 144
2.2.4.36. PositionerHardInterpolatorFactorGet 147
2.2.4.37. PositionerHardInterpolatorFactorSet 149
2.2.4.38. PositionerHardwareStatusGet 151
2.2.4.39. PositionerHardwareStatusStringGet 153
2.2.4.40. PositionerMaximumVelocityAndAccelerationGet 155
2.2.4.41. PositionerMotionDoneGet 157
2.2.4.42. PositionerMotionDoneSet 160
2.2.4.43. PositionerPositionCompareDisable 163
2.2.4.44. PositionerPositionCompareEnable 165
2.2.4.45. PositionerPositionCompareGet 167
2.2.4.46. PositionerPositionCompareSet 170
2.2.4.47. PositionerPositionComparePulseParametersGet 172
2.2.4.48. PositionerPositionComparePulseParametersSet 174
2.2.4.49. PositionerPositionCompareAquadBAlwaysEnable 176
2.2.4.50. PositionerPositionCompareAquadBWindowedGet 178
2.2.4.51. PositionerPositionCompareAquadBWindowedSet 180
2.2.4.52. PositionerRawEncoderPositionGet 182
2.2.4.53. PositionersEncoderIndexDifferenceGet 184
2.2.4.54. PositionerSGammaExactVelocityAjustedDisplacementGet 186
2.2.4.55. PositionerSGammaParametersSet 188
2.2.4.56. PositionerSGammaParametersGet 191
2.2.4.57. PositionerSGammaPreviousMotionTimesGet 194
2.2.4.58. PositionerStageParameterGet 196
2.2.4.59. PositionerStageParameterSet 198
2.2.4.60. PositionerTimeFlasherDisable 200
2.2.4.61. PositionerTimeFlasherEnable 202
2.2.4.62. PositionerTimeFlasherGet 204
2.2.4.63. PositionerTimeFlasherSet 207
2.2.4.64. PositionerUserTravelLimitsGet 210
2.2.4.65. PositionerUserTravelLimitsSet 212
2.2.5. Configuration files 215
2.3. GROUP 222
2.3.1. Description 222
2.3.2. Object structure 222
2.3.3. Function description 223
2.3.3.1. GroupAccelerationSetpointGet 223
2.3.3.2. GroupAnalogTrackingModeDisable 225
2.3.3.3. GroupAnalogTrackingModeEnable 227
2.3.3.4. GroupCorrectorOutputGet 229
2.3.3.5. GroupCurrentFollowingErrorGet 231
2.3.3.6. GroupInitialize 233
2.3.3.7. GroupInitializeWithEncoderCalibration 235
2.3.3.8. GroupHomeSearch 238
2.3.3.9. GroupHomeSearchAndRelativeMove 241
2.3.3.10. GroupJogModeDisable 244
2.3.3.11. GroupJogModeEnable 246
2.3.3.12. GroupJogCurrentGet 248
2.3.3.13. GroupJogParametersGet 251

2.3.3.14. GroupJogParametersSet 254
2.3.3.15. GroupKill 257
2.3.3.16. GroupMotionDisable 259
2.3.3.17. GroupMotionEnable 261
2.3.3.18. GroupMoveAbort 263
2.3.3.19. GroupMoveAbsolute 265
2.3.3.20. GroupMoveRelative 268
2.3.3.21. GroupPositionCurrentGet 271
2.3.3.22. GroupPositionPCORawEncoderGet 273
2.3.3.23. GroupPositionSetpointGet 276
2.3.3.24. GroupPositionTargetGet 278
2.3.3.25. GroupStatusGet 281
2.3.3.26. GroupReferencingActionExecute 283
2.3.3.27. GroupReferencingStart 286
2.3.3.28. GroupReferencingStop 288
2.3.3.29. GroupVelocityCurrentGet 290
2.4. SINGLEAXIS GROUP 292
2.4.1. Description 292
2.4.2. State diagram 292
2.4.3. Specific function description 293
2.4.3.1. SingleAxisSlaveModeDisable 293
2.4.3.2. SingleAxisSlaveModeEnable 295
2.4.3.3. SingleAxisSlaveParametersGet 297
2.4.3.4. SingleAxisSlaveParametersSet 299
2.4.4. Configuration files 301
2.5. SPINDLE GROUP 302
2.5.1. Description 302
2.5.2. State diagram 302
2.5.3. Specific function description 303
2.5.3.1. GroupSpinCurrentGet 303
2.5.3.2. GroupSpinModeStop 305
2.5.3.3. GroupSpinParametersGet 307
2.5.3.4. GroupSpinParametersSet 309
2.5.3.5. SpindleSlaveModeDisable 311
2.5.3.6. SpindleSlaveModeEnable 313
2.5.3.7. SpindleSlaveParametersGet 315
2.5.3.8. SpindleSlaveParametersSet 317
2.5.4. Configuration files 319
2.6. XY GROUP 320
2.6.1. Description 320
2.6.2. State diagram 320
2.6.3. Specific function description 321
2.6.3.1. GroupPositionCorrectedProfilerGet 321
2.6.3.2. XYLineArcExecution 324
2.6.3.3. XYLineArcParametersGet 327
2.6.3.4. XYLineArcPulseOutputGet 330
2.6.3.5. XYLineArcPulseOutputSet 333
2.6.3.6. XYLineArcVerification 336
2.6.3.7. XYLineArcVerificationResultGet 339
2.6.4. Configuration files 342
2.7. XYZ GROUP 343
2.7.1. Description 343
2.7.2. State diagram 343
2.7.3. Specific function description 344
2.7.3.1. XYZGroupPositionCorrectedProfilerGet 344
2.7.3.2. XYZSplineExecution 347
2.7.3.3. XYZSplineParametersGet 350
2.7.3.4. XYZSplineVerification 353
2.7.3.5. XYZSplineVerificationResultGet 356
2.7.4. Configuration files 359
2.8. MULTIPLEAXES GROUP 361

2.8.1. Description *361*
2.8.2. State diagram *361*
2.8.3. Specific function description *362*
2.8.3.1. MultipleAxesPVTExecution 362
2.8.3.2. MultipleAxesPVTParametersGet 365
2.8.3.3. MultipleAxesPVTPulseOutputGet 367
2.8.3.4. MultipleAxesPVTPulseOutputSet 370
2.8.3.5. MultipleAxesPVTVerification 373
2.8.3.6. MultipleAxesPVTVerificationResultGet 376
2.8.4. Configuration files *379*
2.9. ANALOG AND DIGITAL I/O 380
2.9.1. GPIO name list *380*
2.9.1.1. Digital inputs 380
2.9.1.2. Digital outputs 380
2.9.1.3. Analog inputs 380
2.9.1.4. Analog outputs 380
2.9.2. Function description *381*
2.9.2.1. GPIOAnalogGainGet 381
2.9.2.2. GPIOAnalogGainSet 384
2.9.2.3. GPIOAnalogGet 387
2.9.2.4. GPIOAnalogSet 389
2.9.2.5. GPIODigitalGet 392
2.9.2.6. GPIODigitalSet 394
2.10. GATHERING 396
2.10.1. Function description *396*
2.10.1.1. GatheringConfigurationGet 396
2.10.1.2. GatheringConfigurationSet 398
2.10.1.3. GatheringCurrentNumberGet 402
2.10.1.4. GatheringDataAcquire 404
2.10.1.5. GatheringDataGet 406
2.10.1.6. GatheringDataMultipleLinesGet 408
2.10.1.7. GatheringExternalConfigurationGet 411
2.10.1.8. GatheringExternalConfigurationSet 413
2.10.1.9. GatheringExternalCurrentNumberGet 416
2.10.1.10. GatheringExternalDataGet 418
2.10.1.11. GatheringExternalStopAndSave 420
2.10.1.12. GatheringReset 422
2.10.1.13. GatheringRun 424
2.10.1.14. GatheringRunAppend 426
2.10.1.15. GatheringStop 428
2.10.1.16. GatheringStopAndSave 430
2.11. EVENTS AND ACTIONS 432
2.11.1. Functions description *432*
2.11.1.1. EventExtendedAllGet 432
2.11.1.2. EventExtendedConfigurationActionGet 434
2.11.1.3. EventExtendedConfigurationActionSet 436
2.11.1.4. EventExtendedConfigurationTriggerGet 440
2.11.1.5. EventExtendedConfigurationTriggerSet 442
2.11.1.6. EventExtendedGet 447
2.11.1.7. EventExtendedRemove 449
2.11.1.8. EventExtendedStart 451
2.11.1.9. EventExtendedWait 453
2.11.2. Obsolete Functions Description *456*
2.11.2.1. EventAdd 456
2.11.2.2. EventGet 457
2.11.2.3. EventRemove 458
2.11.2.4. EventWait 459
2.12. TCL PROGRAMMING 460
2.12.1. Function Description *460*
2.12.1.1. TCLScriptExecute 460
2.12.1.2. TCLScriptExecuteAndWait 462

2.12.1.3. TCLScriptExecuteWithPriority 464
2.12.1.4. TCLScriptKill 467
2.13. OPTIONAL MODULE PROGRAMMING 469
2.13.1. Function Description *469*
2.13.1.1. OptionalModuleExecute 469
2.13.1.2. OptionalModuleKill 472
2.14. HARDWARE DATE AND TIME SETTING 474
2.14.1. Function Description *474*
2.14.1.1. HardwareDateAndTimeGet 474
2.14.1.2. HardwareDateAndTimeSet 476
2.15. VERSION 478
2.15.1. Function Description *478*
2.15.1.1. GetLibraryVersion 478
2.16. POSITIONER ERROR LIST 480
2.17. POSITIONER HARDWARE STATUS LIST 480
2.18. POSITIONER DRIVER STATUS LIST 483
2.19. GROUP STATUS LIST 484
2.20. ERROR LIST 486
2.21. CONTROLLER STATUS LIST 488
2.22. FUNCTION LIST CLASSED IN CATEGORIES 489

3. PROCESS EXAMPLES 493

3.1. MANAGEMENT OF THE ERRORS 493
3.2. FIRMWARE VERSION 494
3.3. GATHERING WITH MOTION 495
3.4. EXTERNAL GATHERING 497
3.5. OUTPUT COMPARE 499
3.6. SLAVE-MASTER MODE 501
3.7. JOGGING 503
3.8. TRACKING 505
3.9. BACKLASH 506
3.10. TIMER EVENT AND GLOBAL VARIABLES 508
3.11. RUNNING SIMULTANEOUSLY SEVERAL MOTION PROCESSES 510

1. TCP/IP communication

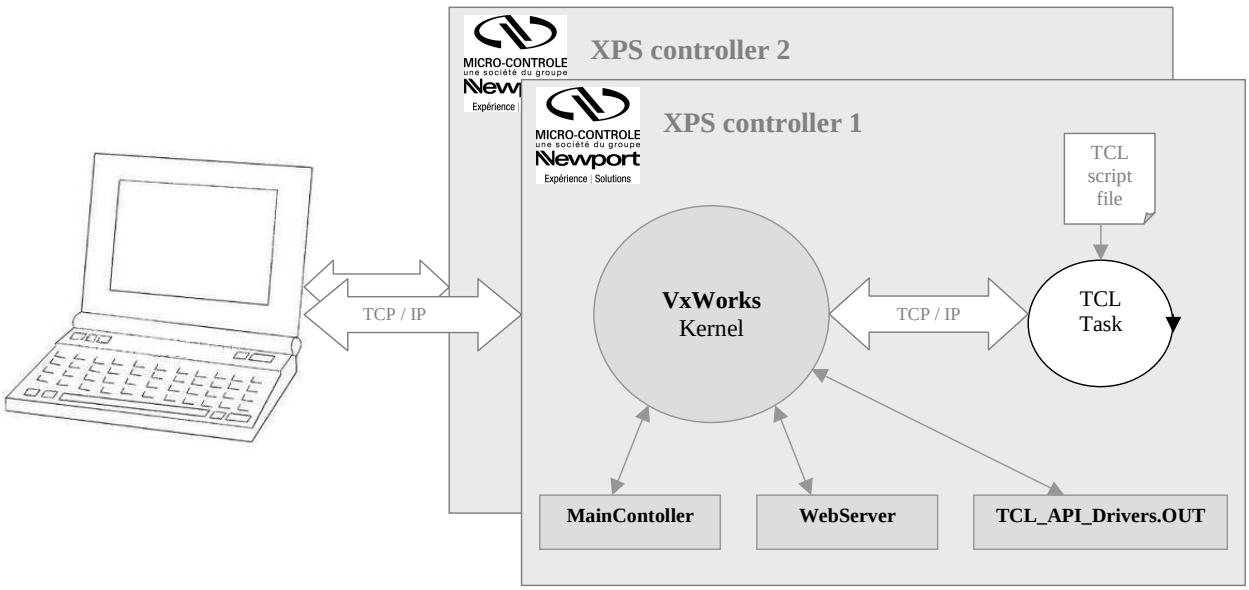
1.1. Description

XPS is based on a 10/100 Base-T Ethernet communication link with TCP/IP protocol and uses a web site approach for all software tools and a FTP server for file transfer. This makes the XPS controller most independent from the operating system of the user. When networked, Unix, Linux or Windows users can access the same controller from any place in the world for remote control, code development, file transfer or diagnostics. The completely object oriented approach of the XPS firmware with powerful, multi-parameter Function's (commands) is also much more self-consistent and intuitive to use than old-style mnemonic commands.

To connect to the XPS controller you must open a socket with the "OpenConnection()" Function. Communication through this socket is done by specifying the socket identifier (socketID) with the Function.

Each Function returns a completion or error message. In case of a successful completion, the return is 0 (zero). In case of an error, the returned error code can be used for diagnosing the problem by the Function ErrorStringGet().

The function call is blocked until a reply is sent by the XPS, or until the timeout value has been reached. For running several processes in parallel (for instance for asking the position while a stage is moving), several sockets can be used in parallel. When using the XPS controller with programming languages that do not support multiple sockets, the timeout value of the Function can be also set to a low value (20 ms).



1.2. Function description

1.2.1. CloseAllOtherSockets

NAME

CloseAllOtherSockets – Closes all sockets beside the one used.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check “Administrator” rights: ERR_NEED_ADMINISTRATOR_RIGHTS (-9)

DESCRIPTION

This function allows an administrator to close all sockets beside the one used to call this function.

All used sockets are closed. So, the ERR_SOCKET_CLOSED_BY_ADMIN error is sent to each function in running before to close the used socket.

NOTE:
Call the “Login” function to identify the user as “Administrator”.

If some TCL scripts are in progress (after a “TCLScriptExecute” function or a “TCLScriptExecuteAndWait” function), don’t use this function before to kill these TCL scripts. So, you must call the “TCLScriptKill” function to stop the TCL execution and only next you can use the **“**CloseAllOtherSockets” function to close sockets.

RETURN

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_NEED_ADMINISTRATOR_RIGHTS (-107)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0): No error

TCL

Prototype
CloseAllOtherSockets

Input parameters
None

Output parameters
None

Return
Error.....integer.....Function error code

C / C++

Prototype
int **CloseAllOtherSockets** ()

Input parameters
None

Output parameters
None

Return
Error.....int.....Function error code

VISUAL BASIC

Prototype
Long **CloseAllOtherSockets** ()

Input parameters
None

Output parameters
None

Return
Error.....Long.....Function error code

MATLAB

Prototype
[Error] **CloseAllOtherSockets** ()

Input parameters
None

Return
Error.....int32.....Function error code

PYTHON

Prototype
[Error] **CloseAllOtherSockets** ()

Input parameters
None

Return
Error.....integer.....Function error code

1.2.2. Login

NAME

Login – Self-identification.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Check command format: ERR_WRONG_FORMAT (-7)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the user name and the password: ERR_WRONG_USERNAME_OR_PASSWORD (-106)

DESCRIPTION

This function allows a user to identify himself as “SuperUser”, “Administrator” or “User”.
The user account must be exited else the ERR_WRONG_USERNAME_OR_PASSWORD (-106) error is returned.

NOTE:
To add a new user account, you must use the XPS web site with “Administrator” rights. In the main menu, select “CONTROLLER CONFIGURATION” and go to the “Users management” page.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_USERNAME_OR_PASSWORD (-106)
SUCCESS (0): no error

TCL

Prototype

Login $SocketID $UserName $Password

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
UserName	string	user name
Password	string	password

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **Login** (int SocketID, char *UserName, char *Password)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
UserName	char *	user name
Password	char *	password

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **Login** (ByVal SocketID As Long, ByVal UserName As String, ByVal Password As String)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
UserName ..... String ..... user name
Password ..... String ..... password

Output parameters

None

Return

Function error code

MATLAB

Prototype

int32 **Login** (int32 SocketID, cstring UserName, cstring Password)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
UserName ..... cstring ..... user name
Password ..... cstring ..... password

Return

Function error code

PYTHON

Prototype

integer **Login** (integer SocketID, string UserName, string Password)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
UserName ..... string ..... user name
Password ..... string ..... password

Return

Function error code

1.2.3. OpenConnection

NAME

OpenConnection – opens a socket to connect TCP server (local).

INPUT TESTS

- Check number of used sockets (Max = 100): if no free socket then the SocketID is affected to -1

DESCRIPTION

This function allows to open a socket in a TCL script located in the “Scripts” directory from XPS controller.

The TCP/IP communication is configured as like:
- Local Host Address = 127.0.0.1
- IP Port = 5001

This function returns a socket identifier to use for each function call. The socket identifier is defined between 0 to 99. If the TCP/IP connection failed then the “SocketID” value is –1.

RETURN

Socket identifier used in each function

TCL

Prototype

OpenConnection $TimeOut SocketID

Input parameters

TimeOut.....floating point.....Timeout in seconds used for each Function execution

Output parameters

SocketID .....integer .....Socket identifier used in each function

Return

Error code

1.2.4. TCP_CloseSocket

NAME

TCP_CloseSocket – Closes a socket.

INPUT TESTS

- Check socket identifier (Max = 100).
- Socket must be used.

DESCRIPTION

Close the opened TCP/IP communication defined by the given socket identifier. If the socket is undefined or is not used then do nothing.

RETURN

None

TCL

Prototype

TCP_CloseSocket $SocketID

Input parameters

SocketID ..... integer ..... Socket identifier used in each function

Output parameters

None

C / C++

Prototype

void **TCP_CloseSocket** (int SocketID)

Input parameters

SocketID ..... int ..... Socket identifier used in each function

Output parameters

None

VISUAL BASIC

Prototype

TCP_CloseSocket (ByVal SocketID As Long)

Input parameters

SocketID ..... Long ..... Socket identifier used in each function

Output parameters

None

MATLAB

Prototype

TCP_CloseSocket (int32 SocketID)

Input parameters

SocketID ..... int32 ..... Socket identifier used in each function

PYTHON

Prototype

TCP_CloseSocket (integer SocketID)

Input parameters

SocketID ..... integer ..... Socket identifier used in each function

1.2.5. TCP_ConnectToServer

NAME

TCP_ConnectToServer – Configures the TCP/IP communication and opens a socket.

INPUT TESTS

- Check number of used sockets (Max = 100): if no free socket then the SocketID is affected to -1

DESCRIPTION

Configurate the TCP/IP communication and open a socket to connect TCP server.
This function returns a socket identifier to use with each function call. The socket identifier is defined between 0 to 99. If the TCP/IP connection failed then the "SocketID" value is –1.

NOTE:
OpenConnection function is used when users are in local, it only needs the timeout and socket number to open the connection with the XPS controller. TCP_ConnectToServer function needs more information like the port number and the IP address. This function is called with the DLL.

RETURN

Socket identifier used in each function

TCL

Prototype

TCP_ConnectToServer $IP_Address $IP_Port $TimeOut SocketID

Input parameters

IP_Address	string	TCP IP address : 195.168.33.xxx or another
IP_Port	interger	TCP IP port : 5001 for XPS controller
TimeOut	floating point	Timeout in seconds used for each Function execution

Output parameters

None

Return

SocketID	integer	Socket identifier used in each function

C / C++

Prototype

int **TCP_ConnectToServer** (char * IP_Address, int IP_Port, double TimeOut)

Input parameters

IP_Address	char *	TCP IP address : 195.168.33.xxx or another
IP_Port	int	TCP IP port : 5001 for XPS controller
TimeOut	double	Timeout in seconds used for each Function execution

Output parameters

None

Return

SocketID	int	Socket identifier used in each function

VISUAL BASIC

Prototype

Long **TCP_ConnectToServer** (ByVal IP_Address As String, ByVal IP_Port As Long, ByVal TimeOut As Double)

Input parameters

IP_Address	String	TCP IP address : 195.168.33.xxx or another
IP_Port	Long	TCP IP port : 5001 for XPS controller
TimeOut	Double	Timeout in seconds used for each Function execution

Output parameters

None

Return

SocketID	Long	Socket identifier used in each function

MATLAB

Prototype

int32 **TCP_ConnectToServer** (cstring IP_Address, int32 IP_Port, double TimeOut)

Input parameters

IP_Address	cstring	TCP IP address : 195.168.33.xxx or another
IP_Port	int32	TCP IP port : 5001 for XPS controller
TimeOut	double	Timeout in seconds used for each Function execution

Return

SocketID	int32	Socket identifier used in each function

PYTHON

Prototype

integer **TCP_ConnectToServer** (string IP_Address, integer IP_Port, double TimeOut)

Input parameters

IP_Address	string	TCP IP address : 195.168.33.xxx or another
IP_Port	integer	TCP IP port : 5001 for XPS controller
TimeOut	double	Timeout in seconds used for each Function execution

Return

SocketID	integer	Socket identifier used in each function

1.2.6. TCP_GetError

NAME

TCP_GetError – Gets the last error about socket.

INPUT TESTS

- Check socket identifier (Max = 100).
- Socket must be used.
-

DESCRIPTION

Gets the last error from the socket defined by the given socket identifier. If the socket is undefined or is not used, the error description is empty.

RETURN

None

TCL

Prototype

TCP_GetError $SocketID ErrorString

Input parameters

SocketID ..... integer ..... Socket identifier used in each function

Output parameters

ErrorString ..... string ..... Last error description

Return

Last error description.

C / C++

Prototype

void **TCP_GetError** (int SocketID, char * ErrorString)

Input parameters

SocketID ..... int ..... Socket identifier used in each function

Output parameters

ErrorString ..... char * ..... Last error description

VISUAL BASIC

Prototype

TCP_GetError (ByVal SocketID As Long, ErrorString As String)

Input parameters

SocketID ..... Long ..... Socket identifier used in each function

Output parameters

ErrorString ..... String ..... Last error description

MATLAB

Prototype

[ErrorString] **TCP_GetError** (int32 SocketID)

Input parameters

SocketID ..... int32 ..... Socket identifier used in each function

Return

ErrorString ..... cstring ..... Last error description

PYTHON

Prototype

[ErrorString] **TCP_GetError** (integer SocketID)

Input parameters

SocketID ..... integer ..... Socket identifier used in each function

Return

ErrorString ..... string ..... Last error description

1.2.7. TCP_SetTimeout

NAME

TCP_SetTimeout – Configures the timeout for TCP/IP communication.

INPUT TESTS

- Check number of used sockets (Maximum number = 100).
- Socket must be used.
- Timeout value must be positive.

DESCRIPTION

Sets a new timeout value in seconds for the opened TCP/IP communication defined by a socket identifier.
If the timeout is inferior to 0.001, the timeout value is setting to 0.001.
If the socket is undefined or is not used then nothing is doing.

RETURN

None

TCL

Prototype

TCP_SetTimeout $SocketID $TimeOut

Input parameters

SocketID ..... integer ..... Socket identifier used in each function
TimeOut ..... floating point ..... Timeout in seconds used for each Function execution

Output parameters

None

Return

TCL error (0 = success or 1 = syntax error)

C / C++

Prototype

void **TCP_SetTimeout** (int SocketID, double TimeOut)

Input parameters

SocketID ..... int ..... Socket identifier used in each function
TimeOut ..... double ..... Timeout in seconds used for each Function execution

Output parameters

None

Return

None

VISUAL BASIC

Prototype

TCP_SetTimeout (ByVal SocketID As Long, ByVal TimeOut As Double)

Input parameters

SocketID ..... Long ..... Socket identifier used in each function
TimeOut ..... Double ..... Timeout in seconds used for each Function execution

Output parameters

None

Return

None

MATLAB

Prototype

int32 **TCP_SetTimeout** (int32 SocketID, double TimeOut)

Input parameters

SocketID	int32	Socket identifier used in each function
TimeOut	double	Timeout in seconds used for each Function execution

Return

None

PYTHON

Prototype

integer **TCP_SetTimeout** (integer SocketID, double TimeOut)

Input parameters

SocketID	integer	Socket identifier used in each function
TimeOut	double	Timeout in seconds used for each Function execution

Return

None

2. XPS features

2.1. General features

2.1.1. ControllerMotionKernelTimeLoadGet (NEW)

NAME

ControllerMotionKernelTimeLoadGet – Returns controller motion kernel time load.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)
- Valid group name: ERR_GROUP_NAME (-19)

DESCRIPTION

This function allows to get the last exact value of controller motion kernel time load (total , corrector, profier and servitudes calculation time).

CorrectorTimeLoad = CorrectorCalculationTime / CorrectorISRPeriod
ProfilerTimeLoad = ProfilerCalculationTime / CorrectorISRPeriod / ProfileGeneratorISRRatio
ServitudesTimeLoad = ServitudesCalculationTime / CorrectorISRPeriod / ServitudesISRRatio
TotalTimeLoad = CorrectorTimeLoad + ProfilerTimeLoad + ServitudesTimeLoad

Note: Refer to *system.ref* file to get *CorrectorISRPeriod*, *ProfileGeneratorISRRatio* and *ServitudesISRRatio*.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

ControllerMotionKernelTimeLoadGet SocketID CPUTotalLoadRatio CPUCorrectorLoadRatio CPUProfilerLoadRatio CPUServitudesLoadRatio

Input parameters

SocketID ..... integer ..... Socket identifier gets by the “TCP_ConnectToServer” function

Output parameters

CPUTotalLoadRatio ..... double ..... controller motion kernel total CPU time load
CPUCorrectorLoadRatio.. double ..... controller motion kernel corrector CPU time load
CPUProfilerLoadRatio..... double ..... controller motion kernel profiler CPU time load
CPUServitudesLoadRatio double ..... controller motion kernel servitudes CPU time load

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **ControllerMotionKernelTimeLoadGet** (int SocketID, double * CPUTotalLoadRatio, double * CPUCorrectorLoadRatio, double * CPUProfilerLoadRatio, double * CPUServitudesLoadRatio)

Input parameters

SocketID ..... int ..... Socket identifier gets by the “TCP_ConnectToServer”function

Output parameters

CPUTotalLoadRatio ..... double * ..... controller motion kernel total CPU time load
CPUCorrectorLoadRatio.. double * ..... controller motion kernel corrector CPU time load
CPUProfilerLoadRatio..... double * ..... controller motion kernel profiler CPU time load
CPUServitudesLoadRatio double * ..... controller motion kernel servitudes CPU time load

Return

Function error code

VISUAL BASIC

Prototype

Long **ControllerMotionKernelTimeLoadGet** (ByVal SocketID As Long, CPUTotalLoadRatio As Double, CPUCorrectorLoadRatio As Double, CPUProfilerLoadRatio As Double, CPUServitudesLoadRatio As Double)

Input parameters

SocketID ..... Long..... Socket identifier gets by the “TCP_ConnectToServer” function

Output parameters

CPUTotalLoadRatio ..... Double ..... controller motion kernel total CPU time load
CPUCorrectorLoadRatio.. Double ..... controller motion kernel corrector CPU time load
CPUProfilerLoadRatio..... Double ..... controller motion kernel profiler CPU time load
CPUServitudesLoadRatio Double ..... controller motion kernel servitudes CPU time load

Return

Function error code

MATLAB

Prototype

[Error, CPUTotalLoadRatio, CPUCorrectorLoadRatio, CPUProfilerLoadRatio, CPUServitudesLoadRatio] **ControllerMotionKernelTimeLoadGet** (int32 SocketID)

Input parameters

SocketID ..... int32 ..... Socket identifier gets by the “TCP_ConnectToServer”function

Return

Error..... int32 ..... Function error code
CPUTotalLoadRatio ..... double ..... controller motion kernel total CPU time load
CPUCorrectorLoadRatio.. double ..... controller motion kernel corrector CPU time load
CPUProfilerLoadRatio..... double ..... controller motion kernel profiler CPU time load
CPUServitudesLoadRatio double ..... controller motion kernel servitudes CPU time load

PYTHON

Prototype

[Error, CPUTotalLoadRatio, CPUCorrectorLoadRatio, CPUProfilerLoadRatio, CPUServitudesLoadRatio] **ControllerMotionKernelTimeLoadGet** (integer SocketID)

Input parameters

SocketID	integer	Socket identifier gets by the “TCP_ConnectToServer”function

Return

Error	integer	Function error code
CPUTotalLoadRatio	double	controller motion kernel total CPU time load
CPUCorrectorLoadRatio	double	controller motion kernel corrector CPU time load
CPUProfilerLoadRatio	double	controller motion kernel profiler CPU time load
CPUServitudesLoadRatio	double	controller motion kernel servitudes CPU time load

2.1.2. ControllerStatusGet

NAME

ControllerStatusGet – Returns the controller status code.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid output parameter type: ERR_WRONG_TYPE_INT (-15)
- Valid group name: ERR_GROUP_NAME (-19)

DESCRIPTION

Returns the controller status code. The controller status codes are listed in the “Controller status list” § 2.193.
The description of the controller status code can be get with the “ControllerStatusStringGet” function.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

ControllerStatusGet SocketID ControllerStatus

Input parameters

SocketID ..... integer ..... Socket identifier gets by the “TCP_ConnectToServer” function

Output parameters

ControllerStatus ..... interger ..... Status of the controller.

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **ControllerStatusGet** (int SocketID, int *ControllerStatus)

Input parameters

SocketID ..... int ..... Socket identifier gets by the “TCP_ConnectToServer”function

Output parameters

ControllerStatus ..... int * ..... Status of the controller

Return

Function error code

VISUAL BASIC

Prototype

Long **ControllerStatusGet** (ByVal SocketID As Long, ControllerStatus As Long)

Input parameters

SocketID ..... Long ..... Socket identifier gets by the “TCP_ConnectToServer” function

Output parameters

ControllerStatus ..... Long ..... Status of the controller

Return

Function error code

MATLAB

Prototype

[Error, ControllerStatus] **ControllerStatusGet** (int32 SocketID)

Input parameters

SocketID ..... int32 ..... Socket identifier gets by the “TCP_ConnectToServer”function

Return

Error ..... int32 ..... Function error code
ControllerStatus ..... int32 ..... Status of the controller

PYTHON

Prototype

[Error, ControllerStatus] **ControllerStatusGet** (integer SocketID)

Input parameters

SocketID ..... integer ..... Socket identifier gets by the “TCP_ConnectToServer”function

Return

Error ..... integer ..... Function error code
ControllerStatus ..... integer ..... Status of the controller

2.1.3. ControllerStatusStringGet

NAME

ControllerStatusStringGet – Get the controller status description from a controller status code.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check input parameter type: ERR_WRONG_TYPE_INT (-15)
- Check output parameter type: ERR_WRONG_TYPE_CHAR (-13)

DESCRIPTION

This function returns the controller status description corresponding to a controller status code (see §2.19 controller status list).
If the status code is not referenced then the “Unknown controller status code” message will be returned.

ERROR CODES

ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

ControllerStatusStringGet $SocketID $ControllerStatusCode ControllerStatusString

Input parameters

SocketID	integer	Socket identifier gets by the “TCP_ConnectToServer” function
ControllerStatusCode	integer	Controller status code

Output parameters

ControllerStatusString	string	Controller status description

Return

Error	integer	TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **ControllerStatusStringGet** (int SocketID, int ControllerStatusCode, char * ControllerStatusString)

Input parameters

SocketID	int	Socket identifier gets by the “TCP_ConnectToServer” function
ControllerStatusCode	int	Controller status code

Output parameters

ControllerStatusString	char *	Controller status description

Return

Error	int	Function error code

VISUAL BASIC

Prototype

Long **ControllerStatusStringGet** (ByVal SocketID As Long, ControllerStatusCode As Integer, ByVal ControllerStatusString As String)

Input parameters

SocketID ..... Long ..... Socket identifier gets by the “TCP_ConnectToServer” function
ControllerStatusCode ..... Integer ..... Controller status code

Output parameters

ControllerStatusString ..... String ..... Controller status description

Return

Error ..... Long ..... Function error code

MATLAB

Prototype

[Error, ControllerStatusString] **ControllerStatusStringGet** (int32 SocketID, int32 ControllerStatusCode)

Input parameters

SocketID ..... int32 ..... Socket identifier gets by the “TCP_ConnectToServer” function
ControllerStatusCode ..... int32 ..... Controller status code

Return

Error ..... int32 ..... Function error code
ControllerStatusString ..... cstring ..... Controller status description

PYTHON

Prototype

[Error, ControllerStatusString] **ControllerStatusStringGet** (integer SocketID, integer ControllerStatusCode)

Input parameters

SocketID ..... integer ..... Socket identifier gets by the “TCP_ConnectToServer” function
ControllerStatusCode ..... integer ..... Controller status code

Return

Error ..... integer ..... Function error code
ControllerStatusString ..... string ..... Controller status description

2.1.4. DoubleGlobalArrayGet

NAME

DoubleGlobalArrayGet – Get a value from the global array of type “double”.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check input parameter type: ERR_WRONG_TYPE_INT (-15)
- Check output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)
- Verify the index number [0:1000[: ERR_PARAMETER_OUT_OF_RANGE (-17)

DESCRIPTION

This function gets the variable value from the global array of type “double”, located by a “Number” index. So, the first variable value from the global array is located to the index “0”.

The returned value is returned in a double.

NOTE:
The number of datas in the global array of type “double” is limited to 1000.

ERROR CODES

ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

DoubleGlobalArrayGet $SocketID $Number DoubleValue

Input parameters

SocketID .....integer.....Socket identifier got from “TCP_ConnectToServer” function
Number .....integer.....Index in the global array

Output parameters

DoubleValue .....floating point ... Variable value

Return

Error.....integer.....TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **DoubleGlobalArrayGet** (int SocketID, int Number, double * DoubleValue)

Input parameters

SocketID .....int ..... Socket identifier got from “TCP_ConnectToServer” function
Number .....int ..... Index in the global array

Output parameters

DoubleValue .....double *..... Variable value

Return

Error.....int ..... Function error code

VISUAL BASIC

Prototype

Long **DoubleGlobalArrayGet** (ByVal SocketID As Long, Number As Integer, ByVal DoubleValue As Double)

Input parameters

SocketID ..... Long..... Socket identifier got from “TCP_ConnectToServer” function
Number ..... Integer..... Index in the global array

Output parameters

DoubleValue ..... Double ..... Variable value

Return

Error..... Long..... Function error code

MATLAB

Prototype

[Error, DoubleValue] **DoubleGlobalArrayGet** (int32 SocketID, int32 Number)

Input parameters

SocketID ..... int32..... Socket identifier got from “TCP_ConnectToServer” function
Number ..... int32..... Index in the global array

Return

Error..... int32..... Function error code
DoubleValue ..... double ..... Variable value

PYTHON

Prototype

[Error, DoubleValue] **DoubleGlobalArrayGet** (integer SocketID, integer Number)

Input parameters

SocketID ..... integer..... Socket identifier got from “TCP_ConnectToServer” function
Number ..... integer..... Index in the global array

Return

Error..... integer..... Function error code
DoubleValue ..... double ..... Variable value

2.1.5. DoubleGlobalArraySet

NAME

DoubleGlobalArraySet – Set a value from the global array of type “double”.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check input parameter type: ERR_WRONG_TYPE_INT (-15), ERR_WRONG_TYPE_DOUBLE (-14)
- Verify the index number [0:1000[: ERR_PARAMETER_OUT_OF_RANGE (-17)

DESCRIPTION

This function allows to set a new value in the global array located at the “Number” index and the new value is setting in a double.

NOTE:
The first variable value from the global array is always located to the index “0”.
The number of datas in the global array is limited to 1000, so the last index is “999”.

ERROR CODES

ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

DoubleGlobalArraySet $SocketID $Number $DoubleValue

Input parameters

SocketID .....integer.....Socket identifier got from “TCP_ConnectToServer” function
Number .....integer.....Index in the global array
DoubleValue .....floating point ...Variable value

Output parameters

None

Return

Error.....integer.....TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **DoubleGlobalArraySet** (int SocketID, int Number, char * DoubleValue)

Input parameters

SocketID .....int .....Socket identifier got from “TCP_ConnectToServer” function
Number .....int .....Index in the global array
DoubleValue .....double.....Variable value

Output parameters

None

Return

Error.....int .....Function error code

VISUAL BASIC

Prototype

Long **DoubleGlobalArraySet** (ByVal SocketID As Long, Number As Integer, ByVal DoubleValue As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
Number	Integer	Index in the global array
DoubleValue	Double	Variable value

Output parameters

None

Return

Error	Long	Function error code

MATLAB

Prototype

[Error] **DoubleGlobalArraySet** (int32 SocketID, int32 Number, double DoubleValue)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
Number	int32	Index in the global array
DoubleValue	double	Variable value

Return

Error	int32	Function error code

PYTHON

Prototype

[Error] **DoubleGlobalArraySet** (integer SocketID, integer Number, Double DoubleValue)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
Number	integer	Index in the global array
DoubleValue	Double	Variable value

Return

Error	integer	Function error code

2.1.6. ErrorStringGet

NAME

ErrorStringGet – Get the error description from a function error code.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check input parameter type: ERR_WRONG_TYPE_INT (-15)
- Check output parameter type: ERR_WRONG_TYPE_CHAR (-13)

DESCRIPTION

The function returns the error description corresponding to a function error code (see §2.20 Error list).
If the error code is not referenced then the “Unknown error code” message will be returned.

ERROR CODES

ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

ErrorStringGet $SocketID $ErrorCode ErrorString

Input parameters

SocketID .....integer.....Socket identifier got from “TCP_ConnectToServer” function
ErrorCode .....integer.....Error code

Output parameters

ErrorString .....string .....Error description

Return

Error.....integer.....TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **ErrorStringGet** (int SocketID, int ErrorCode, char *ErrorString)

Input parameters

SocketID .....int .....Socket identifier got from “TCP_ConnectToServer” function
ErrorCode .....int .....Error code

Output parameters

ErrorString .....char *.....Error description

Return

Error.....int .....Function error code

VISUAL BASIC

Prototype

Long **ErrorStringGet** (ByVal SocketID As Long, ErrorCode As Integer, ByVal ErrorString As String)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
ErrorCode	Integer	Error code

Output parameters

ErrorString	String	Error description

Return

Error	Long	Function error code

MATLAB

Prototype

[Error, ErrorString] **ErrorStringGet** (int32 SocketID, int32 ErrorCode)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
ErrorCode	int32	Error code

Return

Error	int32	Function error code
ErrorString	cstring	Error description

PYTHON

Prototype

[Error, ErrorString] **ErrorStringGet** (integer SocketID, integer ErrorCode)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
ErrorCode	integer	Error code

Return

Error	integer	Function error code
ErrorString	string	Error description

2.1.7. ElapsedTimeGet

NAME

ElapsedTimeGet – Get the elapsed time since the controller power on.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function returns the time in seconds that elapsed since the controller power on.

ERROR CODES

ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

ElapsedTimeGet $SocketID ElapsedTime

Input parameters

SocketID .....integer.....Socket identifier got from “TCP_ConnectToServer” function

Output parameters

ErrorString .....double.....Elapsed time (seconds)

Return

Error.....integer.....TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **ElapsedTimeGet** (int SocketID, double * ElapsedTime)

Input parameters

SocketID .....int .....Socket identifier got from “TCP_ConnectToServer” function

Output parameters

ElapsedTime .....double *.....Elapsed time (seconds)

Return

Error.....int .....Function error code

VISUAL BASIC

Prototype

Long **ElapsedTimeGet** (ByVal SocketID As Long, ErrorCode As Integer, ByVal ElapsedTime As Double)

Input parameters

SocketID .....Long.....Socket identifier got from “TCP_ConnectToServer” function

Output parameters

ElapsedTime .....Double .....Elapsed time (seconds)

Return

Error.....Long.....Function error code

MATLAB

Prototype

[Error, ElapsedTime] **ElapsedTimeGet** (int32 SocketID)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function

Return

Error	int32	Function error code
ElapsedTime	double	Elapsed time (seconds)

PYTHON

Prototype

[Error, ElapsedTime] **ElapsedTimeGet** (integer SocketID)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function

Return

Error	integer	Function error code
ElapsedTime	double	Elapsed time (seconds)

2.1.8. FirmwareVersionGet

NAME

FirmwareVersionGet – Gets the version of the firmware inside the controller.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check output parameter type: ERR_WRONG_TYPE_CHAR (-13)
-

DESCRIPTION

This function gets the controller name and the firmware version number.

Example of returned version string :
"XPS-C8 Firmware V2.1.0"

- Controller name is **XPS-C8**
- Firmware version is **V2.1.0**

ERROR CODES

ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
SUCCESS (0) : no error

TCL

Prototype

FirmwareVersionGet $SocketID Version

Input parameters

SocketID .....integer.....Socket identifier got from "TCP_ConnectToServer" function

Output parameters

Version.....string .....Controller version

Return

Error.....integer.....TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **FirmwareVersionGet** (int SocketID, char * Version)

Input parameters

SocketID .....int .....Socket identifier got from "TCP_ConnectToServer" function

Output parameters

Version.....char *.....Controller version

Return

Error.....int .....Function error code

VISUAL BASIC

Prototype

Long **FirmwareVersionGet** (ByVal SocketID As Long, ByVal ErrorString As String)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

Version ..... String ..... Controller version

Return

Error ..... Long ..... Function error code

MATLAB

Prototype

[Error, Version] **FirmwareVersionGet** (int32 SocketID)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function

Return

Error ..... int32 ..... Function error code
Version ..... cstring ..... Controller version

PYTHON

Prototype

[Error, Version] **FirmwareVersionGet** (integer SocketID)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function

Return

Error ..... integer ..... Function error code
Version ..... string ..... Controller version

2.1.9. GroupStatusStringGet

NAME

GroupStatusStringGet – Get the group state description from a group state code.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check input parameter type: ERR_WRONG_TYPE_INT (-15)
- Check output parameter type: ERR_WRONG_TYPE_CHAR (-13)

DESCRIPTION

This function returns the group state description corresponding to a group state code (see § 2.19 Group state list).
If the group state code is not referenced then the “Error : undefined status” message will be returned.

ERROR CODES

ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

GroupStatusStringGet $SocketID $GroupStatusCode GroupStatusString

Input parameters

SocketID .....integer.....Socket identifier got from “TCP_ConnectToServer” function
GroupStatusCode .....integer.....Group status code

Output parameters

GroupStatusString....string .....Group status description

Return

Error.....integer.....TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **GroupStatusStringGet** (int SocketID, int GroupStatusCode, char * GroupStatusString)

Input parameters

SocketID .....int .....Socket identifier got from “TCP_ConnectToServer” function
GroupStatusCode .....int .....Group status code

Output parameters

GroupStatusString....char *.....Group status description

Return

Error.....int .....Function error code

VISUAL BASIC

Prototype

Long **GroupStatusStringGet** (ByVal SocketID As Long, GroupStatusCode As Integer, ByVal GroupStatusString As String)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GroupStatusCode	Integer	Group status code

Output parameters

GroupStatusString	String	Group status description

Return

Error	Long	Function error code

MATLAB

Prototype

[Error, GroupStatusString] **GroupStatusStringGet** (int32 SocketID, int32 GroupStatusCode)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupStatusCode	int32	Group status code

Return

Error	int32	Function error code
GroupStatusString	cstring	Group status description

PYTHON

Prototype

[Error, GroupStatusString] **GroupStatusStringGet** (integer SocketID, integer GroupStatusCode)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupStatusCode	integer	Group status code

Return

Error	integer	Function error code
GroupStatusString	string	Group status description

2.1.10. GlobalArrayGet

NAME

GlobalArrayGet – Get a value from the global array.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check input parameter type: ERR_WRONG_TYPE_INT (-15)
- Check output parameter type: ERR_WRONG_TYPE_CHAR (-13)
- Verify the index number [0:100[: ERR_PARAMETER_OUT_OF_RANGE (-17)

DESCRIPTION

This function gets the variable value from the global array, located by a “Number” index. So, the first variable value from the global array is located to the index “0”.

The returned value is returned in a string.

NOTE:
The number of datas in the global array is limited to 100.

ERROR CODES

ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

GlobalArrayGet $SocketID $Number StringValue

Input parameters

SocketID .....integer.....Socket identifier got from “TCP_ConnectToServer” function
Number .....integer.....Index in the global array

Output parameters

StringValue .....string .....Variable value

Return

Error.....integer.....TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **GlobalArrayGet** (int SocketID, int Number, char * StringValue)

Input parameters

SocketID .....int .....Socket identifier got from “TCP_ConnectToServer” function
Number .....int .....Index in the global array

Output parameters

StringValue .....char *.....Variable value

Return

Error.....int .....Function error code

VISUAL BASIC

Prototype

Long **GlobalArrayGet** (ByVal SocketID As Long, Number As Integer, ByVal StringValue As String)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
Number ..... Integer ..... Index in the global array

Output parameters

StringValue ..... String ..... Variable value

Return

Error ..... Long ..... Function error code

MATLAB

Prototype

[Error, StringValue] **GlobalArrayGet** (int32 SocketID, int32 Number)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
Number ..... int32 ..... Index in the global array

Return

Error ..... int32 ..... Function error code
StringValue ..... cstring ..... Variable value

PYTHON

Prototype

[Error, StringValue] **GlobalArrayGet** (integer SocketID, integer Number)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
Number ..... integer ..... Index in the global array

Return

Error ..... integer ..... Function error code
StringValue ..... string ..... Variable value

2.1.11. GlobalArraySet

NAME

GlobalArraySet – Set a value from the global array.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check input parameter type: ERR_WRONG_TYPE_INT (-15), ERR_WRONG_TYPE_CHAR (-13)
- Verify the index number [0:100[: ERR_PARAMETER_OUT_OF_RANGE (-17)

DESCRIPTION

This function allows to set a new value in the global array located at the “Number” index and the new value is setting in a string.

NOTE:
The first variable value from the global array is always located to the index “0”.
The number of datas in the global array is limited to 100, so the last index is “99”.

ERROR CODES

ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

GlobalArraySet $SocketID $Number $StringValue

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
Number	integer	Index in the global array
StringValue	string	Variable value

Output parameters

None

Return

Error	integer	TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **GlobalArraySet** (int SocketID, int Number, char * StringValue)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
Number	int	Index in the global array
StringValue	char *	Variable value

Output parameters

None

Return

Error	int	Function error code

VISUAL BASIC

Prototype

Long **GlobalArraySet** (ByVal SocketID As Long, Number As Integer, ByVal StringValue As String)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
Number	Integer	Index in the global array
StringValue	String	Variable value

Output parameters

None

Return

Error	Long	Function error code

MATLAB

Prototype

[Error] **GlobalArraySet** (int32 SocketID, int32 Number, cstring StringValue)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
Number	int32	Index in the global array
StringValue	cstring	Variable value

Return

Error	int32	Function error code

PYTHON

Prototype

[Error] **GlobalArraySet** (integer SocketID, integer Number, string StringValue)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
Number	integer	Index in the global array
StringValue	string	Variable value

Return

Error	integer	Function error code

2.1.12. KillAll

NAME

KillAll – Kills all groups.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)

DESCRIPTION

This function allows to kill and to reset all groups.
This function resets all analog and digital I/O too.

The differents steps to kill all groups are:

1) An “emergency stop” is done if the group state is defined as:
   - HOMING
   - REFERENCING
   - MOVING
   - JOGGING
   - ANALOG_TRACKING

2) The motor is turned off, the motion done is stopped and the control loop is stopped.

3) An “ERR_EMERGENCY_SIGNAL” error is returned by each function in progress, where the group state is:
   - MOTOR_INIT
   - ENCODER_CALIBRATING
   - HOMING
   - REFERENCING
   - MOVING
   - TRAJECTORY
   - ERR_EMERGENCY_SIGNAL

4) At end, the group state must become “NOT INITIALIZED” for all groups.

ERROR CODES

ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype

KillAll $SocketID

Input parameters

SocketID .....integer.....Socket identifier got from “TCP_ConnectToServer” function

Output parameters

None

Return

Error.....integer.....TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **KillAll** (int SocketID)

Input parameters

SocketID .....int ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

None

Return

Error.....int ..... Function error code

VISUAL BASIC

Prototype

Long **KillAll** (ByVal SocketID As Long)

Input parameters

SocketID ..... Long..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

None

Return

Error..... Long..... Function error code

MATLAB

Prototype

[Error] **KillAll** (int32 SocketID)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function

Return

Error..... int32 ..... Function error code

PYTHON

Prototype

[Error] **KillAll** (integer SocketID)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function

Return

Error..... integer ..... Function error code

2.1.13. Reboot

NAME

Reboot – Reboots the controller.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)

DESCRIPTION

This function allows to reboot the controller.
Notes that this function is not a hardware reboot (power off / on), it's a firmware reboot.

NOTE:
If an FTP client is connected, this function is not allowed and the ERR_NOT_ALLOWED_ACTION (-22) error is returned.

ERROR CODES

ERR_NOT_ALLOWED_ACTION (-22)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype
Reboot $SocketID

Input parameters
SocketID .....integer.....Socket identifier got from "TCP_ConnectToServer" function

Output parameters
None

Return
Error.....integer.....TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype
int **Reboot** (int SocketID)

Input parameters
SocketID .....int .....Socket identifier got from "TCP_ConnectToServer" function

Output parameters
None

Return
Error.....int .....Function error code

VISUAL BASIC

Prototype
Long **Reboot** (ByVal SocketID As Long)

Input parameters
SocketID .....Long.....Socket identifier got from "TCP_ConnectToServer" function

Output parameters
None

Return
Error.....Long.....Function error code

MATLAB

Prototype

[Error] **Reboot** (int32 SocketID)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function

Return

Error	int32	Function error code

PYTHON

Prototype

[Error] **Reboot** (integer SocketID)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function

Return

Error	integer	Function error code

2.1.14. TimerGet

NAME

TimerGet – Gets the number of frequency ticks for the selected timer.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check input parameter mnemonic: ERR_WRONG_TYPE (-10)
- Check output parameter type: ERR_WRONG_TYPE_INT (-15)

DESCRIPTION

This function returns the number of frequency ticks configured for the selected timer.

The “TimerName” can be defined as:
- Timer1
- Timer2
- Timer3
- Timer4
- Timer5

The “FrequencyTicks” allows to defined the frequency of the timer:
- One frequency tick represents a corrector period => 0.0001ms => 10 Khz
- N frequency ticks represent N corrector periods => N * 0.0001ms => $\frac{10}{N}$ KHz

NOTE:
The NULL “FrequencyTicks” (=0) means that the timer is disabled.

ERROR CODES

ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE (-10)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

TimerGet $SocketID $TimerName FrequencyTicks

Input parameters

SocketID .....integer.....Socket identifier got from “TCP_ConnectToServer” function
TimerName .....string .....Name of timer

Output parameters

FrequencyTicks.....integer.....Number of frequency ticks

Return

Error.....integer.....TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **TimerGet** (int SocketID, char *TimerName, int* FrequencyTicks)

Input parameters

SocketID .....int ..... Socket identifier got from “TCP_ConnectToServer” function
TimerName .....char *..... Name of timer

Output parameters

FrequencyTicks.....int * ..... Number of frequency ticks

Return

Error.....int ..... Function error code

VISUAL BASIC

Prototype

Long **TimerGet** (ByVal SocketID As Long, ByVal TimerName As String, FrequencyTicks As Integer)

Input parameters

SocketID ..... Long..... Socket identifier got from “TCP_ConnectToServer” function
TimerName ..... String ..... Name of timer

Output parameters

FrequencyTicks..... Integer..... Number of frequency ticks

Return

Error..... Long..... Function error code

MATLAB

Prototype

[Error, FrequencyTicks] **TimerGet** (int32 SocketID, cstring TimerName)

Input parameters

SocketID ..... int32..... Socket identifier got from “TCP_ConnectToServer” function
TimerName ..... cstring ..... Name of timer

Return

Error..... int32..... Function error code
FrequencyTicks..... int32..... Number of frequency ticks

PYTHON

Prototype

[Error, FrequencyTicks] **TimerGet** (integer SocketID, string TimerName)

Input parameters

SocketID ..... integer..... Socket identifier got from “TCP_ConnectToServer” function
TimerName ..... string..... Name of timer

Return

Error..... integer..... Function error code
FrequencyTicks..... integer..... Number of frequency ticks

2.1.15. TimerSet

NAME

TimerSet – Sets the number of frequency ticks for the selected timer.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check input parameter mnemonic: ERR_WRONG_TYPE (-10)
- Check output parameter type: ERR_WRONG_TYPE_INT (-15)

DESCRIPTION

This function sets the number of frequency ticks for the selected timer to activates it.

The “TimerName” can be defined as:
- Timer1
- Timer2
- Timer3
- Timer4
- Timer5

The “FrequencyTicks” allows to defined the frequency of the timer:
- One frequency tick represents a corrector period => 0.0001ms => 10 Khz
- N frequency ticks represent N corrector periods => N * 0.0001ms => $\frac{10}{N}$ KHz

NOTE:
If the “FrequencyTicks” is null (0) then the timer is disabled.

ERROR CODES

ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE (-10)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

TimerSet $SocketID $TimerName $FrequencyTicks

Input parameters

SocketID .....integer.....Socket identifier got from “TCP_ConnectToServer” function
TimerName .....string .....Name of timer
FrequencyTicks.....integer.....Number of frequency ticks

Output parameters

None

Return

Error.....integer.....TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **TimerSet** (int SocketID, char *TimerName, int FrequencyTicks)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
TimerName	char *	Name of timer
FrequencyTicks	int	Number of frequency ticks

Output parameters

None

Return

Error	int	Function error code

VISUAL BASIC

Prototype

Long **TimerSet** (ByVal SocketID As Long, ByVal TimerName As String, ByVal FrequencyTicks As Integer)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
TimerName	String	Name of timer
FrequencyTicks	Integer	Number of frequency ticks

Output parameters

None

Return

Error	Long	Function error code

MATLAB

Prototype

[Error] **TimerSet** (int32 SocketID, cstring TimerName, int32 FrequencyTicks)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
TimerName	cstring	Name of timer
FrequencyTicks	int32	Number of frequency ticks

Return

Error	int32	Function error code

PYTHON

Prototype

[Error, FrequencyTicks] **TimerSet** (integer SocketID, string TimerName, integer FrequencyTicks)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
TimerName	string	Name of timer
FrequencyTicks	integer	Number of frequency ticks

Return

Error	integer	Function error code

2.2. Positioner

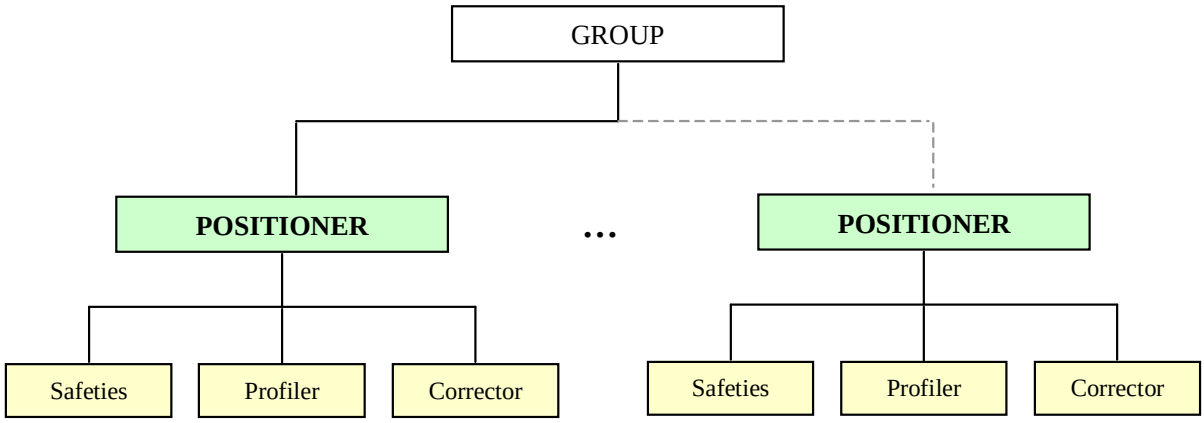
2.2.1. Description

Positioner objects are used to define all motion specific configuration parameters.

The positioner includes a mapping correction : X = f (X)
The positioner includes the SGamma profile.

The maximum number of positioners is limited to **8**.

2.2.2. Object structure



To use a **positioner**, it must belong to a motion group. Positioners are defined by its **full positioner name**. The full positioner name is composed of the group name and the positioner name seperated by a dot.

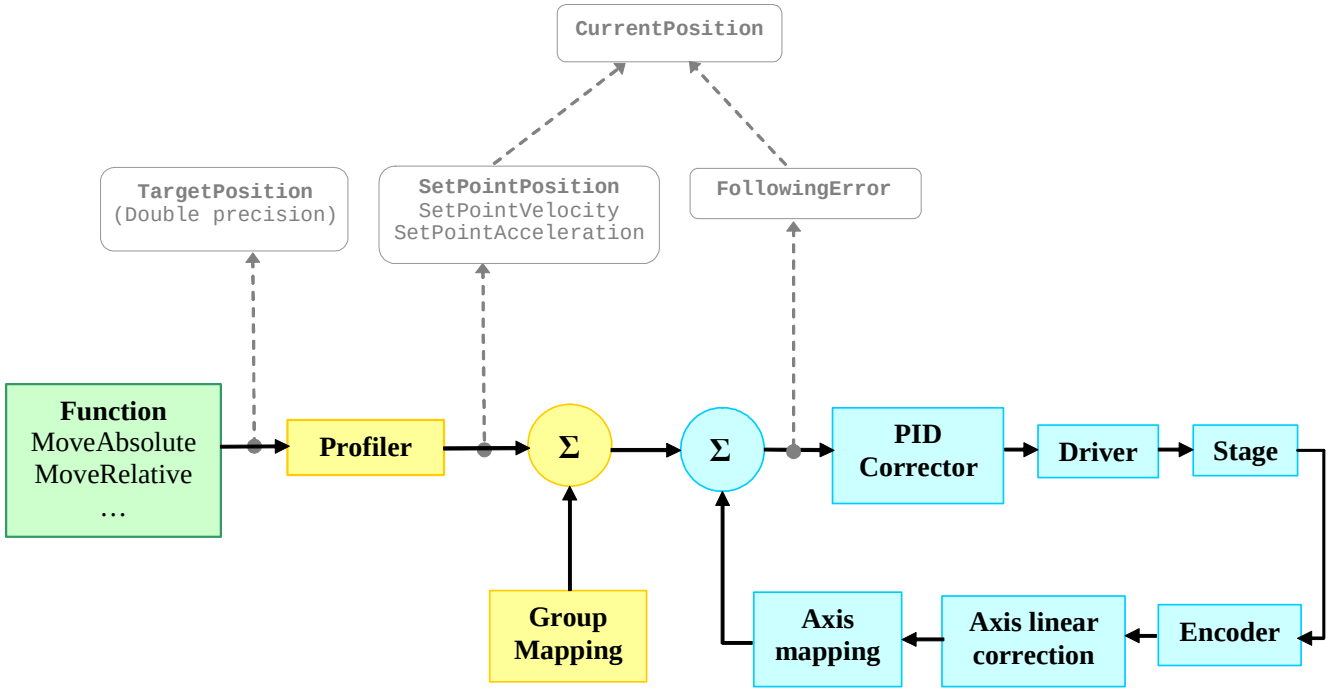
Example:
GroupName.PositionerName

2.2.3. Definition of the different positions for a positioner

For each positioner, three different positions can be called:

- ✓ The **SetpointPosition** is the profiler position. This is the position where the positioner should be according to his motion profile.
- ✓ The **CurrentPosition** is the encoder position of the stage after mapping corrections. This is the actual position of the positioner
- ✓ The **TargetPosition** is the final targeted position of the displacement.

The difference between the SetpointPosition and the CurrentPosition is called the following error.



For instance, during a motion from the position 0 (units) to 100 (units), we could have the following results:

SetpointPosition = 50
CurrentPosition = 49.998 (FollowingError = 50 – 49.998 = 0.002 unit)
TargetPosition = 100.

2.2.4. Function description

2.2.4.1. PositionerAccelerationAutoScaling

NAME

PositionerAccelerationAutoScaling – Auto-scaling process for stage scaling acceleration determination.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)
- Positioner must not be a “Secondary Positioner”: ERR_WRONG_OBJECT_TYPE (-8)
- Check positioner name: ERR_POSITIONER_NAME (-18)
- Check group type: ERR_WRONG_OBJECT_TYPE (-8)
- Control loop type must be “PIDFFAcceleration”: ERR_UNCOMPATIBLE (-24)
- Group status must be “NOT_INITIALIZED”: ERR_NOT_ALLOWED_ACTION (-22)

DESCRIPTION

The function executes an auto-scaling process and returns the new calculated scaling acceleration. The selected group must be in “NOT INITIALIZED” state, else the ERR_NOT_ALLOWED_ACTION (-22) error is returned. This function works only if the positioner control loop type is “PIDFFAcceleration” (acceleration control), else it returns the ERR_UNCOMPATIBLE error.

This function begins to check the positioner error. If an error is detected, the hardware status register is reset (motor on) and the positioner error is cleared before to check it again. If a positioner error is always present, the motor is turn off, the ERR_POSITIONER_ERROR (-5) error is returned and the group status becomes “NOT INITIALIZED”.

If no positioner error then the master-slave error is cleared, the encoder is preset (update encoder position) and the user travel limits are checked. If a travel limit error is detected then the motor is turn off, the ERR_TRAVEL_LIMITS (-35) error is returned and the group status becomes “NOT INITIALIZED”.

If no error, the motor is now to be initialized in the case of stage acceleration control. If the motor initialization failed then the error ERR_MOTOR_INITIALIZATION_ERROR (-50) is returned and the group status becomes “NOT INITIALIZED”.

If successful, the positions are preset, the motion is enabled (the motor is powered) permitting the process of auto-scaling. If the motion can not be enabled, the ERR_NOT_ALLOWED_ACTION (-22) is returned.

During auto-scaling, if the auto-scaling fails the ERR_SCALING_CALIBRATION (-105) error is returned or if the motion becomes disabled then the ERR_EMERGENCY_SIGNAL (-26) error is returned.

The auto-scaling process is executed in 5 periods. At the end of each period, the auto-tuning process estimates the auto-tuning quality by calculating the noise/signal ratio. If the noise/signal ratio is very closed to zero (it means no oscillation) the ERR_RELAY_FEEDBACK_TEST_NO_OSCILLATION (-101) error is returned. Elsewhere if the noise ratio > MaximumNoiseRatio (normally between 0.1 and 0.2, exact value defined in system.ref) then the ERR_RELAY_FEEDBACK_TEST_SIGNAL_NOISY (-102) is returned.

If the number of acquired data points (minimum = 9) or the number of acquired signal periods (minimum = 5) is not enough for a good estimation then the ERR_SIGNAL_POINTS_NOT_ENOUGH (-103) error is returned.

At end of this function, the new value of scaling acceleration is returned and the group status becomes “NOT INITIALIZED” once again.

ERROR CODES

ERR_FATAL_INIT (-20)

ERR_IN_INITIALIZATION (-21)
ERR_MOTOR_INITIALIZATION_ERROR (-50)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_POSITIONER_ERROR (-5)
ERR_POSITIONER_NAME (-18)
ERR_RELAY_FEEDBACK_TEST_NO_OSCILLATION (-101)
ERR_RELAY_FEEDBACK_TEST_SIGNAL_NOISY (-102)
ERR_SCALING_CALIBRATION (-105)
ERR_SIGNAL_POINTS_NOT_ENOUGH (-103)
ERR_TRAVEL_LIMITS (-35)
ERR_UNCOMPATIBLE (-24)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerAccelerationAutoScaling $SocketID $PositionerName Scaling

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
PositionerName	string	Name of a positioner

Output parameters

Scaling	floating point	Calculated scaling acceleration value

Return

Error	integer	TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **PositionerAccelerationAutoScaling** (int SocketID, char * PositionerName, double *Scaling)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
PositionerName	char *	Name of a positioner

Output parameters

Scaling	double *	Calculated scaling acceleration value

Return

Error	int	Function error code

VISUAL BASIC

Prototype

Long **PositionerAccelerationAutoScaling** (ByVal SocketID As Long, ByVal PositionerName As String, Scaling As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
PositionerName	String	Name of a positioner

Output parameters

Scaling	Double	Calculated scaling acceleration value

Return

Error	Long	Function error code

MATLAB

Prototype

[Error, Scaling] **PositionerAccelerationAutoScaling** (int32 SocketID, cstring PositionerName)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
PositionerName	cstring	Name of a positioner

Return

Error	int32	Function error code
Scaling	double	Calculated scaling acceleration value

PYTHON

Prototype

[Error, Scaling] **PositionerAccelerationAutoScaling** (integer SocketID, string PositionerName, string Password)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
PositionerName	string	Name of a positioner

Return

Error	integer	Function error code
Scaling	double	Calculated scaling acceleration value

2.2.4.2. PositionerAnalogTrackingPositionParametersGet

NAME

PositionerAnalogTrackingPositionParametersGet – Gets the parameters of the current tracking position mode.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check input parameter: ERR_WRONG_OBJECT_TYPE (-8), ERR_POSITIONER_NAME (-18)
- Check output parameter type: ERR_WRONG_TYPE_CHAR (-13), ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function returns the current analog input name, the current offset and the current scale used by analog tracking position mode. For a more thorough description of the analog tracking mode, please refer to the XPS Motion Tutorial, section named Motion/Analog tracking.

NOTE :
"velocity" and "acceleration" define the maximum velocity and acceleration used in the position tracking mode.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerAnalogTrackingPositionParametersGet $SocketID $FullPositionerName GPIOName Offset Scale velocity acceleration

Input parameters

SocketID ..... integer ..... Socket identifier got from "TCP_ConnectToServer" function
FullPositionerName ..... string ..... Positioner name

Output parameters

GPIOName..... string ..... Analog input name (ADC)
Offset ..... double ..... Offset (volts)
Scale..... double ..... Scale (Units / Volts)
velocity ..... double ..... velocity (Units / s)
acceleration ..... double ..... acceleration (Units / s^2)

Return

Error..... integer .....TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerAnalogTrackingPositionParametersGet** (int SocketID, char FullPositionerName [250], char *GPIOName, double *Offset, double *Scale, double *velocity, double *acceleration)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	char *	Positioner name

Output parameters

GPIOName	char *	Analog input name (ADC)
Offset	double *	offset in volts
Scale	double *	Scale (Units / Volts)
velocity	double *	velocity (Units / s)
acceleration	double *	acceleration (Units / s²)

Return

Error	int	Function error code

VISUAL BASIC

Prototype

Long **PositionerAnalogTrackingPositionParametersGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal GPIOName As String, Offset As Double, Scale As Double, velocity As Double, acceleration As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	String	Positioner name

Output parameters

GPIOName	String	Analog input name (ADC)
Offset	Double	offset in volts
Scale	Double	Scale (Units / Volts)
velocity	Double	velocity (Units / s)
acceleration	Double	acceleration (Units / s²)

Return

Error	Long	Function error code

MATLAB

Prototype

[Error, GPIOName, Offset, Scale, velocity, acceleration] **PositionerAnalogTrackingPositionParametersGet** (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	cstring	Positioner name

Return

Error	int32	Function error code
GPIOName	cstring	Analog input name (ADC)
Offset	double	offset in volts
Scale	double	Scale (Units / Volts)
velocity	double	velocity (Units / s)
acceleration	double	acceleration (Units / s²)

PYTHON

Prototype

[Error, GPIOName, Offset, Scale, velocity, acceleration] **PositionerAnalogTrackingPositionParametersGet** (integer SocketID, string FullPositionerName)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name

Return

Error	integer	Function error code
GPIOName	string	Analog input name (ADC)
Offset	double	offset in volts
Scale	double	Scale (Units / Volts)
velocity	double	velocity (Units / s)
acceleration	double	acceleration (Units / s^2)

2.2.4.3. PositionerAnalogTrackingPositionParametersSet

NAME

PositionerAnalogTrackingPositionParametersSet – Sets the parameters of the current tracking position mode.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check input parameter: ERR_POSITIONER_NAME (-18)
- Check output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)
- Check Positioner and GPIO type (ADC): ERR_WRONG_OBJECT_TYPE (-8)
- Check velocity and acceleration: ERR_PARAMETER_OUT_OF_RANGE (-17)

DESCRIPTION

This function allows modify the analog input name, the offset and the scale used by the analog tracking position mode. To use this function, the group state must be READY else the ERR_NOT_ALLOWED_ACTION error is returned.

The “Offset” and the “Scale” parameters are used to calculate the target tracking position:
*TrackingPosition = InitialPosition + (AnalogValue - **Offset**) * **Scale***
The “velocity” and “acceleration” parameters define the maximum velocity and acceleration used in the position tracking mode.

NOTE :
The parameters of analog tracking position mode can be reset if the “GPIOName” parameter is empty.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerAnalogTrackingPositionParametersSet $SocketID $FullPositionerName $GPIOName $Offset $Scale $velocity $acceleration

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name
GPIOName	string	Analog input name (ADC)
Offset	double	Offset (volts)
Scale	double	Scale (Units / Volts)
velocity	double	velocity (Units / s)
acceleration	double	acceleration (Units / s²)

Output parameters

None

Return

Error	integer	TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerAnalogTrackingPositionParametersSet** (int SocketID, char FullPositionerName [250], char *GPIOName, double Offset, double Scale, double velocity, double acceleration)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	char *	Positioner name
GPIOName	char *	Analog input name (ADC)
Offset	double	offset in volts
Scale	double	Scale (Units / Volts)
velocity	double	velocity (Units / s)
acceleration	double	acceleration (Units / s²)

Output parameters

None

Return

Error	int	Function error code

VISUAL BASIC

Prototype

Long **PositionerAnalogTrackingPositionParametersSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal GPIOName As String, ByVal Offset As Double, ByVal Scale As Double, ByVal velocity As Double, ByVal acceleration As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	String	Positioner name
GPIOName	String	Analog input name (ADC)
Offset	Double	offset in volts
Scale	Double	Scale (Units / Volts)
velocity	Double	velocity (Units / s)
acceleration	Double	acceleration (Units / s²)

Output parameters

None

Return

Error	Long	Function error code

MATLAB

Prototype

[Error] **PositionerAnalogTrackingPositionParametersSet** (int32 SocketID, cstring FullPositionerName, cstring GPIOName, double Offset, double Scale, double velocity, double acceleration)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	cstring	Positioner name
GPIOName	cstring	Analog input name (ADC)
Offset	double	offset in volts
Scale	double	Scale (Units / Volts)
velocity	double	velocity (Units / s)
acceleration	double	acceleration (Units / s²)

Return

Error	int32	Function error code

PYTHON

Prototype

[Error] **PositionerAnalogTrackingPositionParametersSet** (integer SocketID, string FullPositionerName, string GPIOName, double Offset, double Scale, double velocity, double acceleration)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name
GPIOName	string	Analog input name (ADC)
Offset	double	offset in volts
Scale	double	Scale (Units / Volts)
velocity	double	velocity (Units / s)
acceleration	double	acceleration (Units / s^2)

Return

Error	integer	Function error code

2.2.4.4. PositionerAnalogTrackingVelocityParametersGet

NAME

PositionerAnalogTrackingVelocityParametersGet – Gets the parameters of the current tracking velocity mode.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check input parameter: ERR_WRONG_OBJECT_TYPE (-8), ERR_POSITIONER_NAME (-18)
- Check output parameter type: ERR_WRONG_TYPE_CHAR (-13), ERR_WRONG_TYPE_DOUBLE (-14), ERR_WRONG_TYPE_INT (-15)

DESCRIPTION

This function returns the analog input name, the offset, the scale, the deadband threshold and the order used by analog traking velocity mode. For a more thorough description of the analog tracking mode, please refer to the XPS Motion Tutorial, section named Motion / Analog tracking.

Note :
“velocity” and “acceleration” define the maximum velocity and acceleration used in the velocity tracking mode.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
ERR_WRONG_TYPE_DOUBLE (-14)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

PositionerAnalogTrackingVelocityParametersGet $SocketID $FullPositionerName GPIOName Offset Scale DeadBandThreshold Order velocity acceleration

Input parameters

SocketID ..... integer .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string .....Positioner name

Output parameters

GPIOName..... string .....Analog input name (ADC)
Offset ..... double .....Offset (volts)
Scale..... double .....Scale (Units / Volts)
DeadBandThreshold ..... double .....Dead band threshold (Volts)
Order..... integer .....Order (No unit)
velocity ..... double .....velocity (Units / s)
acceleration ..... double .....acceleration (Units / s^2)

Return

Error..... integer .....TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerAnalogTrackingVelocityParametersGet** (int SocketID, char FullPositionerName [250], char *GPIOName, double *Offset, double *Scale, double *DeadBandThreshold, int *Order, double *velocity, double *acceleration)

Input parameters

SocketID ..... int .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... char * .....Positioner name

Output parameters

GPIOName..... char * .....Analog input name (ADC)
Offset ..... double * .....offset in volts
Scale..... double * .....Scale (Units / Volts)
DeadBandThreshold ..... double * .....Dead band threshold (Volts)
Order..... integer * .....Order (No unit)
velocity ..... double * .....velocity (Units / s)
acceleration ..... double * .....acceleration (Units / s^2)

Return

Error..... int.....Function error code

VISUAL BASIC

Prototype

Long **PositionerAnalogTrackingVelocityParametersGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal GPIOName As String, Offset As Double, Scale As Double, DeadBandThreshold As Double, Order As Integer, velocity As Double, acceleration As Double)

Input parameters

SocketID ..... Long..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String ..... Positioner name

Output parameters

GPIOName..... String ..... Analog input name (ADC)
Offset ..... Double ..... offset in volts
Scale..... Double ..... Scale (Units / Volts)
DeadBandThreshold ..... Double ..... Dead band threshold (Volts)
Order..... Integer..... Order (No unit)
velocity ..... Double ..... velocity (Units / s)
acceleration ..... Double ..... acceleration (Units / s^2)

Return

Error..... Long..... Function error code

MATLAB

Prototype

[Error, GPIOName, Offset, Scale, DeadBandThreshold, Order , velocity, acceleration]
PositionerAnalogTrackingVelocityParametersGet (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring ..... Positioner name

Return

Error..... int32 ..... Function error code
GPIOName..... cstring ..... Analog input name (ADC)
Offset ..... double ..... offset in volts
Scale..... double ..... Scale (Units / Volts)
DeadBandThreshold ..... double ..... Dead band threshold (Volts)
Order..... int32 ..... Order (No unit)
velocity ..... double ..... velocity (Units / s)
acceleration ..... double ..... acceleration (Units / s^2)

PYTHON

Prototype

[Error, GPIOName, Offset, Scale, DeadBandThreshold, Order , velocity, acceleration]
PositionerAnalogTrackingVelocityParametersGet (integer SocketID, string FullPositionerName)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name

Return

Error	integer	Function error code
GPIOName	string	Analog input name (ADC)
Offset	double	offset in volts
Scale	double	Scale (Units / Volts)
DeadBandThreshold	double	Dead band threshold (Volts)
Order	integer	Order (No unit)
velocity	double	velocity (Units / s)
acceleration	double	acceleration (Units / s^2)

2.2.4.5. PositionerAnalogTrackingVelocityParametersSet

NAME

PositionerAnalogTrackingVelocityParametersSet – Sets the parameters of the current tracking velocity mode.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check Positioner: ERR_POSITIONER_NAME (-18), ERR_WRONG_OBJECT_TYPE (-8)
- Check output parameter type: ERR_WRONG_TYPE_DOUBLE (-14), ERR_WRONG_TYPE_INT (-15)
- Check GPIO type (ADC): ERR_WRONG_OBJECT_TYPE (-8)
- Check velocity and acceleration: ERR_PARAMETER_OUT_OF_RANGE (-17)

DESCRIPTION

This function allows modifying the GPIO name, the offset, the scale, the deadband threshold and the order used by the analog tracking velocity mode. To use this function the group state must be READY else the ERR_NOT_ALLOWED_ACTION (-22) error is returned.

The target tracking velocity is defined as like:

```
InputValue = GPIOAnalogInput – Offset
MaxADCAmplitude = 10/GPIOAnalogGain

if  (InputValue >= 0) then
     InputValue = InputValue - DeadBandThreshold
     if (InputValue < 0) then InputValue = 0
else
     InputValue = AnalogInputValue + DeadBandThreshold
     if (InputValue > 0) then InputValue = 0

OutputValue = (|InputValue| / MaxADCAmplitude)^Order

TrackingVelocity = Sign(InputValue) * OutputValue * Scale * MaxADCAmplitude
```

The “velocity” and “acceleration” define the maximum velocity and acceleration used in the velocity tracking mode.

NOTE :
The analog tracking velocity mode can be reset if the “GPIOName” parameter is empty.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

PositionerAnalogTrackingVelocityParametersSet $SocketID $FullPositionerName $GPIOName $Offset $Scale $DeadBandThreshold $Order $velocity $acceleration

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name
GPIOName	string	Analog input name (ADC)
Offset	double	Offset (volts)
Scale	double	Scale (Units / Volts)
DeadBandThreshold	double	Dead band threshold (Volts)
Order	integer	Order (No unit)
velocity	double	velocity (Units / s)
acceleration	double	acceleration (Units / s²)

Output parameters

None

Return

Error	integer	TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerAnalogTrackingVelocityParametersSet** (int SocketID, char FullPositionerName [250], char *GPIOName, double Offset, double Scale, double DeadBandThreshold, int Order, double velocity, double acceleration)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	char *	Positioner name
GPIOName	char *	Analog input name (ADC)
Offset	double	offset in volts
DeadBandThreshold	double	Dead band threshold (Volts)
Order	int	Order (No unit)
Scale	double	Scale (Units / Volts)
velocity	double	velocity (Units / s)
acceleration	double	acceleration (Units / s²)

Output parameters

None

Return

Error	int	Function error code

VISUAL BASIC

Prototype

Long **PositionerAnalogTrackingVelocityParametersSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal GPIOName As String, ByVal Offset As Double, ByVal Scale As Double, ByVal DeadBandThreshold As Double, ByVal Order As Integer, ByVal velocity As Double, ByVal acceleration As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	String	Positioner name
GPIOName	String	Analog input name (ADC)
Offset	Double	offset in volts
DeadBandThreshold	Double	Dead band threshold (Volts)
Order	Integer	Order (No unit)
Scale	Double	Scale (Units / Volts)
velocity	Double	velocity (Units / s)
acceleration	Double	acceleration (Units / s²)

Output parameters

None

Return

Error	Long	Function error code

MATLAB

Prototype

[Error] **PositionerAnalogTrackingVelocityParametersSet** (int32 SocketID, cstring FullPositionerName, cstring GPIOName, double Offset, double Scale, double DeadBandThreshold, int32 Order, double velocity, double acceleration)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	cstring	Positioner name
GPIOName	cstring	Analog input name (ADC)
Offset	double	offset in volts
DeadBandThreshold	double	Dead band threshold (Volts)
Order	int32	Order (No unit)
Scale	double	Scale (Units / Volts)
velocity	double	velocity (Units / s)
acceleration	double	acceleration (Units / s^2)

Return

Error	int32	Function error code

PYTHON

Prototype

[Error] **PositionerAnalogTrackingVelocityParametersSet** (integer SocketID, string FullPositionerName, string GPIOName, double Offset, double Scale, double DeadBandThreshold, integer Order, double velocity, double acceleration)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name
GPIOName	string	Analog input name (ADC)
Offset	double	offset in volts
DeadBandThreshold	double	Dead band threshold (Volts)
Order	integer	Order (No unit)
Scale	double	Scale (Units / Volts)
velocity	double	velocity (Units / s)
acceleration	double	acceleration (Units / s^2)

Return

Error	integer	Function error code

2.2.4.6. PositionerBacklashDisable

NAME

PositionerBacklashDisable – Disables the backlash compensation.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type (must not be a secondary positioner): ERR_WRONG_OBJECT_TYPE (-8)

DESCRIPTION

This function disables the backlash compensation. For a more thorough description of the backlash compensation, please refer to the XPS Motion Tutorial, section named Compensation / Backlash compensation.

In the “stages.ini” file the parameter “Backlash” allows to enable or disable the backlash compensation:

- Backlash = 0 → Disable backlash
- Backlash > 0 → Enable backlash

NOTE:
The backlash compensation is not allowed with a secondary positioner (gantry mode).
The backlash must be disabled to execute a trajectory, to use the jog mode or to use the analog tracking mode.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_OBJECT_TYPE (-8)
SUCCESS (0) : no error

TCL

Prototype

PositionerBacklashDisable $SocketID $FullPositionerName

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string ..... Positioner name (maximum size = 250)

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **PositionerBacklashDisable** (int SocketID, char * FullPositionerName)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... char * ..... Positioner name (maximum size = 250)

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **PositionerBacklashDisable** (ByVal SocketID As Long, ByVal FullPositionerName As String)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String ..... Positioner name (maximum size = 250)

Output parameters

None

Return

Function error code

MATLAB

Prototype

int32 **PositionerBacklashDisable** (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring ..... Positioner name (maximum size = 250)

Return

Function error code

PYTHON

Prototype

integer **PositionerBacklashDisable** (integer SocketID, string FullPositionerName)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string ..... Positioner name (maximum size = 250)

Return

Function error code

2.2.4.7. PositionerBacklashEnable

NAME

PositionerBacklashEnable – Enables the backlash compensation.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Group status must be “NOT INITIALIZED”: ERR_NOT_ALLOWED_ACTION (-22)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type (must not be a secondary positioner): ERR_WRONG_OBJECT_TYPE (-8)

DESCRIPTION

This function enables the backlash compensation defined in the “stages.ini” file or defined by the “PositionerBacklashSet” function. If the backlash compensation value is null then this function could not enable the backlash mode. For a more thorough description of the backlash compensation, please refer to the XPS Motion Tutorial, section named Compensation / Backlash compensation.

The group state must be NOT INITIALIZED to enable the backlash compensation. If it is not the case then the ERR_NOT_ALLOWED_ACTION (-22) error is returned.

In the “stages.ini” file the parameter “Backlash” allows to enable or disable the backlash compensation:

- Backlash = 0 → Disable backlash
- Backlash > 0 → Enable backlash

NOTE:
The backlash must be disabled to execute a trajectory, to use the jog mode or to use the analog tracking mode.

CAUTION:
It is not possible to use backlash compensation with positioners that have a “HomeSearchSequenceType” defined as “CurrentPositionAsHome” or that have a “PositionerMappingFileName” defined in the stages.ini file.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_OBJECT_TYPE (-8)
SUCCESS (0) : no error

TCL

Prototype
PositionerBacklashEnable $SocketID $FullPositionerName

Input parameters
SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string ..... Positioner name (maximum size = 250)

Output parameters
None

Return
TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **PositionerBacklashEnable** (int SocketID, char * FullPositionerName)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... char * ..... Positioner name (maximum size = 250)

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **PositionerBacklashEnable** (ByVal SocketID As Long, ByVal FullPositionerName As String)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String ..... Positioner name (maximum size = 250)

Output parameters

None

Return

Function error code

MATLAB

Prototype

int32 **PositionerBacklashEnable** (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring ..... Positioner name (maximum size = 250)

Return

Function error code

PYTHON

Prototype

integer **PositionerBacklashEnable** (integer SocketID, string FullPositionerName)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string ..... Positioner name (maximum size = 250)

Return

Function error code

2.2.4.8. PositionerBacklashGet

NAME

PositionerBacklashGet – Gets the backlash compensation value.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check output parameter type: ERR_WRONG_TYPE_CHAR (-13), ERR_WRONG_TYPE_DOUBLE (-14)
- Check the positioner type (must not be a secondary positioner): ERR_WRONG_OBJECT_TYPE (-8)

DESCRIPTION

This function returns the backlash compensation value, defined in the “stages.ini” file or defined by the “PositionerBacklashSet” function, and the backlash status (“Enable” or “Disable”). For a more thorough description of the backlash compensation, please refer to the XPS Motion Tutorial, section named Compensation / Backlash compensation.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerBacklashGet $SocketID $FullPositionerName BacklashValue Status

Input parameters

SocketID ..... integer .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string .....Positioner name

Output parameters

BacklashValue ..... double .....Backlash compensation value (units)
Status..... string .....Backlash status (“Enable” or “Disable”)

Return

Error..... integer .....TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerBacklashGet** (int SocketID, char FullPositionerName [250], double * BacklashValue, char * Status)

Input parameters

SocketID ..... int .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... char * .....Positioner name

Output parameters

BacklashValue ..... double * .....Backlash compensation value (units)
Status..... char * .....Backlash status (“Enable” or “Disable”)

Return

Error..... int .....Function error code

VISUAL BASIC

Prototype

Long **PositionerBacklashGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, BacklashValue As Double, ByVal Status As String)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String ..... Positioner name

Output parameters

BacklashValue ..... Double ..... Backlash compensation value (units)
Status..... String ..... Backlash status (“Enable” or “Disable”)

Return

Error..... Long..... Function error code

MATLAB

Prototype

[Error, BacklashValue, Status] **PositionerBacklashGet** (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring ..... Positioner name

Return

Error..... int32 ..... Function error code
BacklashValue ..... double ..... Backlash compensation value (units)
Status..... cstring ..... Backlash status (“Enable” or “Disable”)

PYTHON

Prototype

[Error, BacklashValue, Status] **PositionerBacklashGet** (integer SocketID, string FullPositionerName)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string ..... Positioner name

Return

Error..... integer ..... Function error code
BacklashValue ..... double ..... Backlash compensation value (units)
Status..... string ..... Backlash status (“Enable” or “Disable”)

2.2.4.9. PositionerBacklashSet

NAME

PositionerBacklashSet – Sets the backlash compensation value.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type (must not be a secondary positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Check input parameter: ERR_WRONG_TYPE_DOUBLE (-14)
- The “BacklashValue” must be positive : ERR_PARAMETER_OUT_OF_RANGE (-17)

DESCRIPTION

This function changes the backlash compensation value. For a more thorough description of the backlash compensation, please refer to the XPS Motion Tutorial, section named Compensation / Backlash compensation.

NOTE :
This function can be used only if a backlash compensation is defined in the “stages.ini” file (Backlash > 0) else the ERR_NOT_ALLOWED_ACTION (-22) error is returned.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerBacklashSet $SocketID $FullPositionerName $BacklashValue

Input parameters

SocketID ..... integer .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string .....Positioner name
BacklashValue ..... double .....Backlash compensation value (units)

Output parameters (None)

Return

Error..... integer .....TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerBacklashSet** (int SocketID, char FullPositionerName [250], double BacklashValue)

Input parameters

SocketID ..... int .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... char * .....Positioner name
BacklashValue ..... double .....Backlash compensation value (units)

Output parameters (None)

Return

Error..... int.....Function error code

VISUAL BASIC

Prototype

Long **PositionerBacklashSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal BacklashValue As Double)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String ..... Positioner name
BacklashValue ..... Double ..... Backlash compensation value (units)

Output parameters

None

Return

Error ..... Long ..... Function error code

MATLAB

Prototype

[Error] **PositionerBacklashSet** (int32 SocketID, cstring FullPositionerName, double BacklashValue)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring ..... Positioner name
BacklashValue ..... double ..... Backlash compensation value (units)

Return

Error ..... int32 ..... Function error code

PYTHON

Prototype

[Error] **PositionerBacklashSet** (integer SocketID, string FullPositionerName, double BacklashValue)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string ..... Positioner name
BacklashValue ..... double ..... Backlash compensation value (units)

Return

Error ..... integer ..... Function error code

2.2.4.10. PositionerCorrectorAutoTuning

NAME

PositionerCorrectorAutoTuning – Auto-tuning process for position control loop PID values determination.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check input parameter type: ERR_WRONG_TYPE_INT (-15)
- Check output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)
- Positioner must not be a “Secondary Positioner”: ERR_WRONG_OBJECT_TYPE (-8)
- Check positioner name: ERR_POSITIONER_NAME (-18)
- Check group type: ERR_WRONG_OBJECT_TYPE (-8)
- Control loop type must be “PIDFFVelocity”, “PIDDualFFVoltage” or “PIDFFAcceleration”: ERR_UNCOMPATIBLE (-24)
- Group status must be “READY”: ERR_NOT_ALLOWED_ACTION (-22)

DESCRIPTION

The function executes an auto-tuning process and returns the new calculated PID setting (KP, KI and KD values). The selected group must be in “READY” state, else the ERR_NOT_ALLOWED_ACTION (-22) error is returned.

This function works only if the positioner control loop type is “PIDFFVelocity” (velocity control), “PIDDualFFVoltage” (voltage control) or “PIDFFAcceleration” (acceleration control), else it returns the ERR_UNCOMPATIBLE error.

If the function is called when the positioner is not in READY state, the ERR_NOT_ALLOWED_ACTION (-22) error will be returned.

The “Mode” input value indicates the control mode of the position loop (**Short Settle** or **High Robustness**).

1. In the **Short Settle** mode, the PID values are adjusted for have the high motion performance (short settling time, less following errors).
2. The **High Robustness** mode is used to have a relatively good performance in motion, but guarantees the robustness (stable) for all stage situations (positions, velocities, accelerations).

If every thing is OK, the auto-tuning goes in execution. During the auto-tuning, if the auto-tuning initialization fails the ERR_PID_TUNING_INITIALIZATION (-104) error is returned, or if the motion becomes disabled then the ERR_EMERGENCY_SIGNAL (-26) error is returned.

The auto-tuning process is executed in 5 periods. At the end of each period, the auto-tuning process estimates the auto-tuning quality by calculating the noise/signal ratio. If the noise/signal ratio is very closed to zero (it means no oscillation) the ERR_RELAY_FEEDBACK_TEST_NO_OSCILLATION (-101) error is returned. Elsewhere if the noise ratio > MaximumNoiseRatio (normally between 0.1 and 0.2, exact value defined in system.ref) then the ERR_RELAY_FEEDBACK_TEST_SIGNAL_NOISY (-102) is returned.

If the number of acquired data points (minimum = 9) or the number of acquired signal periods (minimum = 5) is not enough for a good estimation then the ERR_SIGNAL_POINTS_NOT_ENOUGH (-103) error is returned.

At end of this function, the new PID setting is returned and the group status becomes “READY” once again.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_POSITIONER_ERROR (-5)
ERR_POSITIONER_NAME (-18)

ERR_RELAY_FEEDBACK_TEST_NO_OSCILLATION (-101)
ERR_RELAY_FEEDBACK_TEST_SIGNAL_NOISY (-102)
ERR_PID_TUNING_INITIALIZATION (-104)
ERR_SIGNAL_POINTS_NOT_ENOUGH (-103)
ERR_UNCOMPATIBLE (-24)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
ERR_WRONG_TYPE_DOUBLE (-14)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

PositionerCorrectorAutoTuning $SocketID $PositionerName $Mode KP KI KD

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
PositionerName	string	Name of a positioner
Mode	.integer.	Loop control mode (0 = short settle, or 1 = robust)

Output parameters

KP	double *	Calculated KP value
KI	double *	Calculated KI value
KD	double *	Calculated KD value

Return

Error	integer	TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerCorrectorAutoTuning** (int SocketID, char * PositionerName, int Mode, double *KP, double *KI, double *KD)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
PositionerName	char *	Positioner name
Mode	int	Loop control mode (0 = short settle, or 1 = robust)

Output parameters

KP	double *	Calculated KP value
KI	double *	Calculated KI value
KD	double *	Calculated KD value

Return

Error	int	Function error code

VISUAL BASIC

Prototype

Long **PositionerCorrectorAutoTuning** (ByVal SocketID As Long, ByVal PositionerName As String, Mode As Integer, KP As Double, KI As Double, KD As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
PositionerName	String	Positioner name
Mode	Integer	Loop control mode (0 = short settle, or 1 = robust)

Output parameters

KP	Double	Calculated KP value
KI	Double	Calculated KI value
KD	Double	Calculated KD value

Return

Error	Long	Function error code

MATLAB

Prototype

[Error, KP, KI, KD] **PositionerCorrectorAutoTuning** (int32 SocketID, cstring PositionerName, int32 Mode)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
PositionerName	cstring	Positioner name
Mode	int32	Loop control mode (0 = short settle, or 1 = robust)

Return

Error	int32	Function error code
KP	double	Calculated KP value
KI	double	Calculated KI value
KD	double	Calculated KD value

PYTHON

Prototype

[Error, KP, KI, KD] **PositionerCorrectorAutoTuning** (integer SocketID, string PositionerName, integer Mode**,** string Password)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
PositionerName	string	Positioner name
Mode	.integer	Loop control mode (0 = short settle, or 1 = robust)

Return

Error	integer	Function error code
KP	double	Calculated KP value
KI	double	Calculated KI value
KD	double	Calculated KD value

2.2.4.11. PositionerCorrectorNotchFiltersGet

NAME

PositionerCorrectorNotchFiltersGet – Gets the notch filter parameters.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)
- Check the positioner type: ERR_WRONG_OBJECT_TYPE (-8)

DESCRIPTION

This functions returns the parameters defined for two notch filters.

First notch filter parameters:

- UserNotchFrequency1
- UserNotchBandwidth1
- UserNotchGain1
-

Second notch filter parameters:

- UserNotchFrequency2
- UserNotchBandwidth2
- UserNotchGain2.

NOTE:
If the corrector type is "NoEncoderPositionCorrector" then the ERR_UNCOMPATIBLE (-24) error is returned.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_UNCOMPATIBLE (-24)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerCorrectorNotchFiltersGet $SocketID $FullPositionerName NotchFrequency1 NotchBandwith1 NotchGain1 NotchFrequency2 NotchBandwith2 NotchGain2

Input parameters

SocketID ..... integer .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string .....Positioner name

Output parameters

NotchFrequency1 ..... double .....Frequency (Herz) for notch filter #1
NotchBandwith1 ..... double .....Band width (Herz) for notch filter #1
NotchGain1 ..... double .....Gain for notch filter #1
NotchFrequency2 ..... double .....Frequency (Herz) for notch filter #2
NotchBandwith2 ..... double .....Band width (Herz) for notch filter #2
NotchGain2 ..... double .....Gain for notch filter #2

Return

Error ..... integer .....TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerCorrectorNotchFiltersGet** (int SocketID, char FullPositionerName [250], double* NotchFrequency1,double* NotchBandwith1, double* NotchGain1, double* NotchFrequency2, double* NotchBandwith2, double* NotchGain2)

Input parameters

SocketID ..... int .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... char * .....Positioner name

Output parameters

NotchFrequency1 ..... double * .....Frequency (Herz) for notch filter #1
NotchBandwith1 ..... double * .....Band width (Herz) for notch filter #1
NotchGain1 ..... double * .....Gain for notch filter #1
NotchFrequency2 ..... double * .....Frequency (Herz) for notch filter #2
NotchBandwith2 ..... double * .....Band width (Herz) for notch filter #2
NotchGain2 ..... double * .....Gain for notch filter #2

Return

Error ..... int .....Function error code

VISUAL BASIC

Prototype

Long **PositionerCorrectorNotchFiltersGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, NotchFrequency1 As Double, NotchBandwith1 As Double, NotchGain1 As Double, NotchFrequency2 As Double, NotchBandwith2 As Double, NotchGain2 As Double)

Input parameters

SocketID ..... Long..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String ..... Positioner name

Output parameters

NotchFrequency1 ..... Double ..... Frequency (Herz) for notch filter #1
NotchBandwith1 ..... Double ..... Band width (Herz) for notch filter #1
NotchGain1 ..... Double ..... Gain for notch filter #1
NotchFrequency2 ..... Double ..... Frequency (Herz) for notch filter #2
NotchBandwith2 ..... Double ..... Band width (Herz) for notch filter #2
NotchGain2 ..... Double ..... Gain for notch filter #2

Return

Error ..... Long..... Function error code

MATLAB

Prototype

[Error, NotchFrequency1, NotchBandwith1, NotchGain1, NotchFrequency2, NotchBandwith2, NotchGain2] **PositionerCorrectorNotchFiltersGet** (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID ..... int32..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring ..... Positioner name

Return

Error ..... int32..... Function error code
NotchFrequency1 ..... double ..... Frequency (Herz) for notch filter #1
NotchBandwith1 ..... double ..... Band width (Herz) for notch filter #1
NotchGain1 ..... double ..... Gain for notch filter #1
NotchFrequency2 ..... double ..... Frequency (Herz) for notch filter #2
NotchBandwith2 ..... double ..... Band width (Herz) for notch filter #2
NotchGain2 ..... double ..... Gain for notch filter #2

PYTHON

Prototype

[Error, NotchFrequency1, NotchBandwith1, NotchGain1, NotchFrequency2, NotchBandwith2, NotchGain2]
PositionerCorrectorNotchFiltersGet (integer SocketID, string FullPositionerName)

Input parameters

SocketID ..... integer .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string .....Positioner name

Return

Error ..... integer .....Function error code
NotchFrequency1 ..... double .....Frequency (Herz) for notch filter #1
NotchBandwith1 ..... double .....Band width (Herz) for notch filter #1
NotchGain1 ..... double .....Gain for notch filter #1
NotchFrequency2 ..... double .....Frequency (Herz) for notch filter #2
NotchBandwith2 ..... double .....Band width (Herz) for notch filter #2
NotchGain2 ..... double .....Gain for notch filter #2

2.2.4.12. PositionerCorrectorNotchFiltersSet

NAME

PositionerCorrectorNotchFiltersSet – Sets the notch filter parameters.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)
- Check the positioner type: ERR_WRONG_OBJECT_TYPE (-8)
- Check parameter values: ERR_PARAMETER_OUT_OF_RANGE (-17)

$$\text{NotchFrequency} \in \left[0 : \frac{0.5}{\text{CorrectorPeriod}}\right] \text{ avec } \text{CorrectorPeriod} = 0.0001 \text{ s } (10 \text{ KHz}) \Rightarrow [0 : 5000]$$

$$\text{NotchBandwidth} \in \left[0 : \frac{0.5}{\text{CorrectorPeriod}}\right] \text{ avec } \text{CorrectorPeriod} = 0.0001 \text{ s } (10 \text{ KHz}) \Rightarrow [0 : 5000]$$

$$\text{NotchGain} \in [0 : 100]$$

DESCRIPTION

This functions configures the parameters defined for two notch filters. If the “NotchFrequency” value is NULL or the “NotchGain” value is NULL then the notch filter is not activated.

First notch filter parameters:

- NotchFrequency1
- NotchBandwidth1
- NotchGain1

Second notch filter parameters:

- NotchFrequency2
- NotchBandwidth2
- NotchGain2.

NOTE:
If the corrector type is “NoEncoderPositionCorrector" then the ERR_UNCOMPATIBLE (-24) error is returned.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_UNCOMPATIBLE (-24)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerCorrectorNotchFiltersSet $SocketID $FullPositionerName $NotchFrequency1 $NotchBandwith1 $NotchGain1 $NotchFrequency2 $NotchBandwith2 $NotchGain2

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name
NotchFrequency1	double	Frequency (Herz) for notch filter #1
NotchBandwith1	double	Band width (Herz) for notch filter #1
NotchGain1	double	Gain for notch filter #1
NotchFrequency2	double	Frequency (Herz) for notch filter #2
NotchBandwith2	double	Band width (Herz) for notch filter #2
NotchGain2	double	Gain for notch filter #2

Output parameters (None)

Return

Error	integer	TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerCorrectorNotchFiltersSet** (int SocketID, char FullPositionerName [250], double* NotchFrequency1,double* NotchBandwith1, double* NotchGain1, double* NotchFrequency2, double* NotchBandwith2, double* NotchGain2)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	char *	Positioner name
NotchFrequency1	double	Frequency (Herz) for notch filter #1
NotchBandwith1	double	Band width (Herz) for notch filter #1
NotchGain1	double	Gain for notch filter #1
NotchFrequency2	double	Frequency (Herz) for notch filter #2
NotchBandwith2	double	Band width (Herz) for notch filter #2
NotchGain2	double	Gain for notch filter #2

Output parameters (None)

Return

Error	int	Function error code

VISUAL BASIC

Prototype

Long **PositionerCorrectorNotchFiltersSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal NotchFrequency1 As Double, ByVal NotchBandwith1 As Double, ByVal NotchGain1 As Double, ByVal NotchFrequency2 As Double, ByVal NotchBandwith2 As Double, ByVal NotchGain2 As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	String	Positioner name
NotchFrequency1	Double	Frequency (Herz) for notch filter #1
NotchBandwith1	Double	Band width (Herz) for notch filter #1
NotchGain1	Double	Gain for notch filter #1
NotchFrequency2	Double	Frequency (Herz) for notch filter #2
NotchBandwith2	Double	Band width (Herz) for notch filter #2
NotchGain2	Double	Gain for notch filter #2

Output parameters (None)

Return

Error	Long	Function error code

MATLAB

Prototype

[Error] **PositionerCorrectorNotchFiltersSet** (int32 SocketID, cstring FullPositionerName, double NotchFrequency1, double NotchBandwith1, double NotchGain1, double NotchFrequency2, double NotchBandwith2, double NotchGain2)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring ..... Positioner name
NotchFrequency1 ..... double ..... Frequency (Herz) for notch filter #1
NotchBandwith1 ..... double ..... Band width (Herz) for notch filter #1
NotchGain1 ..... double ..... Gain for notch filter #1
NotchFrequency2 ..... double ..... Frequency (Herz) for notch filter #2
NotchBandwith2 ..... double ..... Band width (Herz) for notch filter #2
NotchGain2 ..... double ..... Gain for notch filter #2

Return

Error ..... int32 ..... Function error code

PYTHON

Prototype

[Error] **PositionerCorrectorNotchFiltersSet** (integer SocketID, string FullPositionerName, double NotchFrequency1, double NotchBandwith1, double NotchGain1, double NotchFrequency2, double NotchBandwith2, double NotchGain2)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string ..... Positioner name
NotchFrequency1 ..... double ..... Frequency (Herz) for notch filter #1
NotchBandwith1 ..... double ..... Band width (Herz) for notch filter #1
NotchGain1 ..... double ..... Gain for notch filter #1
NotchFrequency2 ..... double ..... Frequency (Herz) for notch filter #2
NotchBandwith2 ..... double ..... Band width (Herz) for notch filter #2
NotchGain2 ..... double ..... Gain for notch filter #2

Return

Error ..... integer ..... Function error code

2.2.4.13. PositionerCorrectorPIDDualFFVoltageGet

NAME

PositionerCorrectorPIDDualFFVoltageGet – Gets the corrector "PIDDualFFVoltage" parameters.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type and the corrector type: ERR_WRONG_OBJECT_TYPE (-8)
- Check output parameter type: ERR_WRONG_TYPE_BOOL (-12), ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function allow to return the corrector parameter values used by a PID dual feed-forward with a motor voltage output.

NOTE:
The "CorrectorType" must be "PIDDualFFVoltage" in the stages.ini file. This servo loop type is used when the position servo loop drives directly the voltage applied to the motor.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_BOOL (-12)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerCorrectorPIDDualFFVoltageGet $SocketID $FullPositionerName ClosedLoopStatus KP KI KD KS IntegrationTime DerivativeFilterCutOffFrequency GKP GKI GKD KForm FeedForwardGainVelocity FeedForwardGainAcceleration Friction

Input parameters

SocketID	integer	Socket identifier got from "TCP_ConnectToServer" function
FullPositionerName	string	Positioner name

Output parameters

ClosedLoopStatus	bool	Position servo loop status (true=closed and false=opened)
KP	double	PID servo loop proportional gain
KI	double	PID servo loop integral gain
KD	double	PID servo loop derivative gain
KS	double	PID integral saturation value (0 to 1)
IntegrationTime	double	PID integration time (seconds)
DerivativeFilterCutOffFrequency	double	PID derivative filter cut off frequency (Hz)
GKP	double	variable PID proportional gain multiplier
GKI	double	variable PID integral gain multiplier
GKD	double	variable PID derivative gain multiplier
KForm	double	variable PID form coefficient
FeedForwardGainVelocity	double	Velocity feedforward gain (units)
FeedForwardGainAcceleration	double	Acceleration feedforward gain (units)
Friction	double	friction compensation

Return

Error	integer	TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerCorrectorPIDDualFFVoltageGet** (int SocketID, char FullPositionerName[250], bool* ClosedLoopStatus, double* KP, double* KI, double* KD, double* KS, double* IntegrationTime, double* DerivativeFilterCutOffFrequency, double* GKP, double* GKI, double* GKD, double* KForm, double* FeedForwardGainVelocity, double* FeedForwardGainAcceleration, double* Friction)

Input parameters

SocketID ..... int .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... char * .....Positioner name

Output parameters

ClosedLoopStatus .....bool * ..... Position servo loop status (true=closed and false=opened)
KP .....double * ..... PID servo loop proportional gain
KI .....double * ..... PID servo loop integral gain
KD.....double * ..... PID servo loop derivative gain
KS .....double * ..... PID integral saturation value (0 to 1)
IntegrationTime .....double * ..... PID integration time (seconds)
DerivativeFilterCutOffFrequency.....double * ..... PID derivative filter cut off frequency (Hz)
GKP .....double * ..... variable PID proportional gain multiplier
GKI .....double * ..... variable PID integral gain multiplier
GKD.....double * ..... variable PID derivative gain multiplier
KForm.....double * ..... variable PID form coefficient
FeedForwardGainVelocity.....double * ..... Velocity feedforward gain (units)
FeedForwardGainAcceleration .....double * ..... Acceleration feedforward gain (units)
Friction.....double * ..... friction compensation

Return

Error.....int ..... Function error code

VISUAL BASIC

Prototype

Long **PositionerCorrectorPIDDualFFVoltageGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ClosedLoopStatus As Boolean, KP As Double, KI As Double, KD As Double, KS As Double, IntegrationTime As Double, DerivativeFilterCutOffFrequency As Double, GKP As Double, GKI As Double, GKD As Double, KForm As Double, FeedForwardGainVelocity As Double, FeedForwardGainAcceleration As Double, Friction As Double)

Input parameters

SocketID ..... Long.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String .....Positioner name

Output parameters

ClosedLoopStatus .....Boolean..... Position servo loop status (true=closed and false=opened)
KP .....Double..... PID servo loop proportional gain
KI .....Double..... PID servo loop integral gain
KD.....Double..... PID servo loop derivative gain
KS .....Double..... PID integral saturation value (0 to 1)
IntegrationTime .....Double..... PID integration time (seconds)
DerivativeFilterCutOffFrequency.....Double..... PID derivative filter cut off frequency (Hz)
GKP .....Double..... variable PID proportional gain multiplier
GKI .....Double..... variable PID integral gain multiplier
GKD.....Double..... variable PID derivative gain multiplier
KForm.....Double..... variable PID form coefficient
FeedForwardGainVelocity.....Double..... Velocity feedforward gain (units)
FeedForwardGainAcceleration .....Double..... Acceleration feedforward gain (units)
Friction.....Double..... friction compensation

Return

Error.....Long..... Function error code

MATLAB

Prototype

[Error, ClosedLoopStatus, KP, KI, KD, KS, IntegrationTime, DerivativeFilterCutOffFrequency, GKP, GKI, GKD, KForm, FeedForwardGainVelocity, FeedForwardGainAcceleration, Friction]
PositionerCorrectorPIDDualFFVoltageGet (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID .....int32 .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....cstring.....Positioner name

Return

Error.....int32 ..... Function error code
ClosedLoopStatus .....boolean..... Position servo loop status (true=closed and false=opened)
KP .....double ..... PID servo loop proportional gain
KI .....double ..... PID servo loop integral gain
KD.....double ..... PID servo loop derivative gain
KS .....double ..... PID integral saturation value (0 to 1)
IntegrationTime .....double ..... PID integration time (seconds)
DerivativeFilterCutOffFrequency.....double ..... PID derivative filter cut off frequency (Hz)
GKP .....double ..... variable PID proportional gain multiplier
GKI .....double ..... variable PID integral gain multiplier
GKD.....double ..... variable PID derivative gain multiplier
KForm.....double ..... variable PID form coefficient
FeedForwardGainVelocity.....double ..... Velocity feedforward gain (units)
FeedForwardGainAcceleration .....double ..... Acceleration feedforward gain (units)
Friction.....double ..... friction compensation

PYTHON

Prototype

[Error, ClosedLoopStatus, KP, KI, KD, KS, IntegrationTime, DerivativeFilterCutOffFrequency, GKP, GKI, GKD, KForm, FeedForwardGainVelocity, FeedForwardGainAcceleration, Friction]
PositionerCorrectorPIDDualFFVoltageGet (integer SocketID, string FullPositionerName)

Input parameters

SocketID .....integer .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....string .....Positioner name

Return

Error.....integer ..... Function error code
ClosedLoopStatus .....boolean..... Position servo loop status (true=closed and false=opened)
KP .....double ..... PID servo loop proportional gain
KI .....double ..... PID servo loop integral gain
KD.....double ..... PID servo loop derivative gain
KS .....double ..... PID integral saturation value (0 to 1)
IntegrationTime .....double ..... PID integration time (seconds)
DerivativeFilterCutOffFrequency.....double ..... PID derivative filter cut off frequency (Hz)
GKP .....double ..... variable PID proportional gain multiplier
GKI .....double ..... variable PID integral gain multiplier
GKD.....double ..... variable PID derivative gain multiplier
KForm.....double ..... variable PID form coefficient
FeedForwardGainVelocity.....double ..... Velocity feedforward gain (units)
FeedForwardGainAcceleration .....double ..... Acceleration feedforward gain (units)
Friction.....double ..... friction compensation

2.2.4.14. PositionerCorrectorPIDDualFFVoltageSet

NAME

PositionerCorrectorPIDDualFFVoltageSet – Configures the corrector "PIDDualFFVoltage" parameters.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type and the corrector type: ERR_WRONG_OBJECT_TYPE (-8)
- Check input parameter type: ERR_WRONG_TYPE_BOOL (-12), ERR_WRONG_TYPE_DOUBLE (-14)
- Check parameter value: ERR_PARAMETER_OUT_OF_RANGE (-17)

$KP \geq 0$
$KI \geq 0$
$KD \geq 0$
$0 \leq KS \leq 1$
$IntegrationTime \geq CorrectorPeriod\ (0.0001\ s)$
$GKP > -1$
$GKI > -1$
$GKD > -1$
$KForm \geq 0$
$KFeedForwardVelocity \geq 0$
$KFeedForwardAcceleration \geq 0$
$Friction \geq 0$

$$DerivativeFilterCutOffFrequency \in \left[0 : \frac{0.5}{\text{CorrectorPeriod}}\right] => [0 : 5000] \text{ with } CorrectorPeriod = 0.0001\ s$$

DESCRIPTION

This function allows to configure the "PIDDualFFVoltage" corrector parameters. The "CorrectorType" must be "PIDDualFFVoltage" in the stages.ini file else the ERR_WRONG_OBJECT_TYPE (-8) error is returned.

NOTE:
This servo loop type is used when the position servo loop drives directly the voltage applied to the motor.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_BOOL (-12)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerCorrectorPIDDualFFVoltageSet $SocketID $FullPositionerName $ClosedLoopStatus $KP $KI $KD $KS $IntegrationTime $DerivativeFilterCutOffFrequency $GKP $GKI $GKD $KForm $FeedForwardGainVelocity $FeedForwardGainAcceleration $Friction

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name
ClosedLoopStatus	bool	Position servo loop status (true=closed and false=opened)
KP	double	PID servo loop proportional gain
KI	double	PID servo loop integral gain
KD	double	PID servo loop derivative gain
KS	double	PID integral saturation value (0 to 1)
IntegrationTime	double	PID integration time (seconds)
DerivativeFilterCutOffFrequency	double	PID derivative filter cut off frequency (Hz)
GKP	double	variable PID proportional gain multiplier
GKI	double	variable PID integral gain multiplier
GKD	double	variable PID derivative gain multiplier
KForm	double	variable PID form coefficient
FeedForwardGainVelocity	double	Velocity feedforward gain (units)
FeedForwardGainAcceleration	double	Acceleration feedforward gain (units)
Friction	double	friction compensation

Output parameters

None

Return

Error	integer	TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerCorrectorPIDDualFFVoltageSet** (int SocketID, char FullPositionerName[250], bool ClosedLoopStatus, double KP, double KI, double KD, double KS, double IntegrationTime, double DerivativeFilterCutOffFrequency, double GKP, double GKI, double GKD, double KForm, double FeedForwardGainVelocity, double FeedForwardGainAcceleration, double Friction)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	char *	Positioner name
ClosedLoopStatus	bool	Position servo loop status (true=closed and false=opened)
KP	double	PID servo loop proportional gain
KI	double	PID servo loop integral gain
KD	double	PID servo loop derivative gain
KS	double	PID integral saturation value (0 to 1)
IntegrationTime	double	PID integration time (seconds)
DerivativeFilterCutOffFrequency	double	PID derivative filter cut off frequency (Hz)
GKP	double	variable PID proportional gain multiplier
GKI	double	variable PID integral gain multiplier
GKD	double	variable PID derivative gain multiplier
KForm	double	variable PID form coefficient
FeedForwardGainVelocity	double	Velocity feedforward gain (units)
FeedForwardGainAcceleration	double	Acceleration feedforward gain (units)
Friction	double	friction compensation

Output parameters

None

Return

Error	int	Function error code

VISUAL BASIC

Prototype

Long **PositionerCorrectorPIDDualFFVoltageSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal ClosedLoopStatus As Boolean, ByVal KP As Double, ByVal KI As Double, ByVal KD As Double, ByVal KS As Double, ByVal IntegrationTime As Double, ByVal DerivativeFilterCutOffFrequency As Double, ByVal GKP As Double, ByVal GKI As Double, ByVal GKD As Double, ByVal KForm As Double, ByVal FeedForwardGainVelocity As Double, ByVal FeedForwardGainAcceleration As Double, ByVal Friction As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	String	Positioner name
ClosedLoopStatus	Boolean	Position servo loop status (true=closed and false=opened)
KP	Double	PID servo loop proportional gain
KI	Double	PID servo loop integral gain
KD	Double	PID servo loop derivative gain
KS	Double	PID integral saturation value (0 to 1)
IntegrationTime	Double	PID integration time (seconds)
DerivativeFilterCutOffFrequency	Double	PID derivative filter cut off frequency (Hz)
GKP	Double	variable PID proportional gain multiplier
GKI	Double	variable PID integral gain multiplier
GKD	Double	variable PID derivative gain multiplier
KForm	Double	variable PID form coefficient
FeedForwardGainVelocity	Double	Velocity feedforward gain (units)
FeedForwardGainAcceleration	Double	Acceleration feedforward gain (units)
Friction	Double	friction compensation

Output parameters

None

Return

Error	Long	Function error code

MATLAB

Prototype

[Error] **PositionerCorrectorPIDDualFFVoltageSet** (int32 SocketID, cstring FullPositionerName, bool ClosedLoopStatus, double KP, double KI, double KD, double KS, double IntegrationTime, double DerivativeFilterCutOffFrequency, double GKP, double GKI, double GKD, double KForm, double FeedForwardGainVelocity, double FeedForwardGainAcceleration, double Friction)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	cstring	Positioner name
ClosedLoopStatus	boolean	Position servo loop status (true=closed and false=opened)
KP	double	PID servo loop proportional gain
KI	double	PID servo loop integral gain
KD	double	PID servo loop derivative gain
KS	double	PID integral saturation value (0 to 1)
IntegrationTime	double	PID integration time (seconds)
DerivativeFilterCutOffFrequency	double	PID derivative filter cut off frequency (Hz)
GKP	double	variable PID proportional gain multiplier
GKI	double	variable PID integral gain multiplier
GKD	double	variable PID derivative gain multiplier
KForm	double	variable PID form coefficient
FeedForwardGainVelocity	double	Velocity feedforward gain (units)
FeedForwardGainAcceleration	double	Acceleration feedforward gain (units)
Friction	double	friction compensation

Return

Error	int32	Function error code

PYTHON

Prototype

[Error] **PositionerCorrectorPIDDualFFVoltageSet** (integer SocketID, string FullPositionerName, bool ClosedLoopStatus, double KP, double KI, double KD, double KS, double IntegrationTime, double DerivativeFilterCutOffFrequency, double GKP, double GKI, double GKD, double KForm, double FeedForwardGainVelocity, double FeedForwardGainAcceleration, double Friction)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string ..... Positioner name
ClosedLoopStatus ..... boolean .... Position servo loop status (true=closed and false=opened)
KP ..... double ..... PID servo loop proportional gain
KI ..... double ..... PID servo loop integral gain
KD ..... double ..... PID servo loop derivative gain
KS ..... double ..... PID integral saturation value (0 to 1)
IntegrationTime ..... double ..... PID integration time (seconds)
DerivativeFilterCutOffFrequency ..... double ..... PID derivative filter cut off frequency (Hz)
GKP ..... double ..... variable PID proportional gain multiplier
GKI ..... double ..... variable PID integral gain multiplier
GKD ..... double ..... variable PID derivative gain multiplier
KForm ..... double ..... variable PID form coefficient
FeedForwardGainVelocity ..... double ..... Velocity feedforward gain (units)
FeedForwardGainAcceleration ..... double ..... Acceleration feedforward gain (units)
Friction ..... double ..... friction compensation

Return

Error ..... integer ..... Function error code

2.2.4.15. PositionerCorrectorPIDFFAccelerationGet

NAME

PositionerCorrectorPIDFFAccelerationGet – Gets the corrector "PIDFFAcceleration" parameters.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type and the corrector type: ERR_WRONG_OBJECT_TYPE (-8)
- Check output parameter type: ERR_WRONG_TYPE_BOOL (-12), ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function allow to return the corrector parameter values used by a PID feed-forward with an acceleration output.

NOTE:
The "CorrectorType" must be "PIDFFAcceleration" in the stages.ini file. This servo loop type is used when a constant value applied to the driver results in a constant acceleration of the stage.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_BOOL (-12)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerCorrectorPIDFFAccelerationGet $SocketID $FullPositionerName ClosedLoopStatus KP KI KD KS IntegrationTime DerivativeFilterCutOffFrequency GKP GKI GKD KForm FeedForwardGainAcceleration

Input parameters

SocketID	integer	Socket identifier got from "TCP_ConnectToServer" function
FullPositionerName	string	Positioner name

Output parameters

ClosedLoopStatus	bool	Position servo loop status (true=closed and false=opened)
KP	double	PID servo loop proportional gain
KI	double	PID servo loop integral gain
KD	double	PID servo loop derivative gain
KS	double	PID integral saturation value (0 to 1)
IntegrationTime	double	PID integration time (seconds)
DerivativeFilterCutOffFrequency	double	PID derivative filter cut off frequency (Hz)
GKP	double	variable PID proportional gain multiplier
GKI	double	variable PID integral gain multiplier
GKD	double	variable PID derivative gain multiplier
KForm	double	variable PID form coefficient
FeedForwardGainAcceleration	double	Acceleration feedforward gain (units)

Return

Error	integer	TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerCorrectorPIDFFAccelerationGet** (int SocketID, char FullPositionerName[250], bool* ClosedLoopStatus, double* KP, double* KI, double* KD, double* KS, double* IntegrationTime, double* DerivativeFilterCutOffFrequency, double* GKP, double* GKI, double* GKD, double* KForm, double* FeedForwardGainAcceleration)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	char *	Positioner name

Output parameters

ClosedLoopStatus	bool *	Position servo loop status (true=closed and false=opened)
KP	double *	PID servo loop proportional gain
KI	double *	PID servo loop integral gain
KD	double *	PID servo loop derivative gain
KS	double *	PID integral saturation value (0 to 1)
IntegrationTime	double *	PID integration time (seconds)
DerivativeFilterCutOffFrequency	double *	PID derivative filter cut off frequency (Hz)
GKP	double *	variable PID proportional gain multiplier
GKI	double *	variable PID integral gain multiplier
GKD	double *	variable PID derivative gain multiplier
KForm	double *	variable PID form coefficient
FeedForwardGainAcceleration	double *	Acceleration feedforward gain (units)

Return

Error	int	Function error code

VISUAL BASIC

Prototype

Long **PositionerCorrectorPIDFFAccelerationGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ClosedLoopStatus As Boolean, KP As Double, KI As Double, KD As Double, KS As Double, IntegrationTime As Double, DerivativeFilterCutOffFrequency As Double, GKP As Double, GKI As Double, GKD As Double, KForm As Double, FeedForwardGainAcceleration As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	String	Positioner name

Output parameters

ClosedLoopStatus	Boolean	Position servo loop status (true=closed and false=opened)
KP	Double	PID servo loop proportional gain
KI	Double	PID servo loop integral gain
KD	Double	PID servo loop derivative gain
KS	Double	PID integral saturation value (0 to 1)
IntegrationTime	Double	PID integration time (seconds)
DerivativeFilterCutOffFrequency	Double	PID derivative filter cut off frequency (Hz)
GKP	Double	variable PID proportional gain multiplier
GKI	Double	variable PID integral gain multiplier
GKD	Double	variable PID derivative gain multiplier
KForm	Double	variable PID form coefficient
FeedForwardGainAcceleration	Double	Acceleration feedforward gain (units)

Return

Error	Long	Function error code

MATLAB

Prototype

[Error, ClosedLoopStatus, KP, KI, KD, KS, IntegrationTime, DerivativeFilterCutOffFrequency, GKP, GKI, GKD, KForm, FeedForwardGainAcceleration] **PositionerCorrectorPIDFFAccelerationGet** (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	cstring	Positioner name

Return

Error	int32	Function error code
ClosedLoopStatus	boolean	Position servo loop status (true=closed and false=opened)
KP	double	PID servo loop proportional gain
KI	double	PID servo loop integral gain
KD	double	PID servo loop derivative gain
KS	double	PID integral saturation value (0 to 1)
IntegrationTime	double	PID integration time (seconds)
DerivativeFilterCutOffFrequency	double	PID derivative filter cut off frequency (Hz)
GKP	double	variable PID proportional gain multiplier
GKI	double	variable PID integral gain multiplier
GKD	double	variable PID derivative gain multiplier
KForm	double	variable PID form coefficient
FeedForwardGainAcceleration	double	Acceleration feedforward gain (units)

PYTHON

Prototype

[Error, ClosedLoopStatus, KP, KI, KD, KS, IntegrationTime, DerivativeFilterCutOffFrequency, GKP, GKI, GKD, KForm, FeedForwardGainAcceleration] **PositionerCorrectorPIDFFAccelerationGet** (integer SocketID, string FullPositionerName)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name

Return

Error	integer	Function error code
ClosedLoopStatus	boolean	Position servo loop status (true=closed and false=opened)
KP	double	PID servo loop proportional gain
KI	double	PID servo loop integral gain
KD	double	PID servo loop derivative gain
KS	double	PID integral saturation value (0 to 1)
IntegrationTime	double	PID integration time (seconds)
DerivativeFilterCutOffFrequency	double	PID derivative filter cut off frequency (Hz)
GKP	double	variable PID proportional gain multiplier
GKI	double	variable PID integral gain multiplier
GKD	double	variable PID derivative gain multiplier
KForm	double	variable PID form coefficient
FeedForwardGainAcceleration	double	Acceleration feedforward gain (units)

2.2.4.16. PositionerCorrectorPIDFFAccelerationSet

NAME

PositionerCorrectorPIDFFAccelerationSet – Gets the corrector “PIDFFAcceleration” parameters.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type and the corrector type: ERR_WRONG_OBJECT_TYPE (-8)
- Check input parameter type: ERR_WRONG_TYPE_BOOL (-12), ERR_WRONG_TYPE_DOUBLE (-14)
- Check parameter value: ERR_PARAMETER_OUT_OF_RANGE (-17)

$KP \geq 0$
$KI \geq 0$
$KD \geq 0$
$0 \leq KS \leq 1$
$IntegrationTime \geq CorrectorPeriod\ (0.0001\ s)$
$GKP > -1$
$GKI > -1$
$GKD > -1$
$KForm \geq 0$
$KFeedForwardAcceleration \geq 0$

$$DerivativeFilterCutOffFrequency \in \left[0 : \frac{0.5}{CorrectorPeriod}\right] => [0 : 5000] \ with\ CorrectorPeriod = 0.0001\ s$$

DESCRIPTION

This function allows to configure the “PIDFFAcceleration” corrector parameters.

NOTE:
The “CorrectorType” parameter must be defined as “PIDFFAcceleration” in the “stages.ini” file else the ERR_WRONG_OBJECT_TYPE (-8) error is returned. This servo loop type is used when a constant value applied to the driver results in a constant acceleration of the stage.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_BOOL (-12)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerCorrectorPIDFFAccelerationSet $SocketID $FullPositionerName $ClosedLoopStatus $KP $KI $KD $KS $IntegrationTime $DerivativeFilterCutOffFrequency $GKP $GKI $GKD $KForm $FeedForwardGainAcceleration

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name
ClosedLoopStatus	bool	Position servo loop status (true=closed and false=opened)
KP	double	PID servo loop proportional gain
KI	double	PID servo loop integral gain
KD	double	PID servo loop derivative gain
KS	double	PID integral saturation value (0 to 1)
IntegrationTime	double	PID integration time (seconds)
DerivativeFilterCutOffFrequency	double	PID derivative filter cut off frequency (Hz)
GKP	double	variable PID proportional gain multiplier
GKI	double	variable PID integral gain multiplier
GKD	double	variable PID derivative gain multiplier
KForm	double	variable PID form coefficient
FeedForwardGainAcceleration	double	Acceleration feedforward gain (units)

Output parameters

None

Return

Error	integer	TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerCorrectorPIDFFAccelerationSet** (int SocketID, char FullPositionerName[250], bool ClosedLoopStatus, double KP, double KI, double KD, double KS, double IntegrationTime, double DerivativeFilterCutOffFrequency, double GKP, double GKI, double GKD, double KForm, double FeedForwardGainAcceleration)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	char *	Positioner name
ClosedLoopStatus	bool	Position servo loop status (true=closed and false=opened)
KP	double	PID servo loop proportional gain
KI	double	PID servo loop integral gain
KD	double	PID servo loop derivative gain
KS	double	PID integral saturation value (0 to 1)
IntegrationTime	double	PID integration time (seconds)
DerivativeFilterCutOffFrequency	double	PID derivative filter cut off frequency (Hz)
GKP	double	variable PID proportional gain multiplier
GKI	double	variable PID integral gain multiplier
GKD	double	variable PID derivative gain multiplier
KForm	double	variable PID form coefficient
FeedForwardGainAcceleration	double	Acceleration feedforward gain (units)

Output parameters

None

Return

Error	int	Function error code

VISUAL BASIC

Prototype

Long **PositionerCorrectorPIDFFAccelerationSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal ClosedLoopStatus As Boolean, ByVal KP As Double, ByVal KI As Double, ByVal KD As Double, ByVal KS As Double, ByVal IntegrationTime As Double, ByVal DerivativeFilterCutOffFrequency As Double, ByVal GKP As Double, ByVal GKI As Double, ByVal GKD As Double, ByVal KForm As Double, ByVal FeedForwardGainAcceleration As Double

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	String	Positioner name
ClosedLoopStatus	Boolean	Position servo loop status (true=closed and false=opened)
KP	Double	PID servo loop proportional gain
KI	Double	PID servo loop integral gain
KD	Double	PID servo loop derivative gain
KS	Double	PID integral saturation value (0 to 1)
IntegrationTime	Double	PID integration time (seconds)
DerivativeFilterCutOffFrequency	Double	PID derivative filter cut off frequency (Hz)
GKP	Double	variable PID proportional gain multiplier
GKI	Double	variable PID integral gain multiplier
GKD	Double	variable PID derivative gain multiplier
KForm	Double	variable PID form coefficient
FeedForwardGainAcceleration	Double	Acceleration feedforward gain (units)

Output parameters

None

Return

Error	Long	Function error code

MATLAB

Prototype

[Error] **PositionerCorrectorPIDFFAccelerationSet** (int32 SocketID, cstring FullPositionerName, bool ClosedLoopStatus, double KP, double KI, double KD, double KS, double IntegrationTime, double DerivativeFilterCutOffFrequency, double GKP, double GKI, double GKD, double KForm, double FeedForwardGainAcceleration)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	cstring	Positioner name
ClosedLoopStatus	boolean	Position servo loop status (true=closed and false=opened)
KP	double	PID servo loop proportional gain
KI	double	PID servo loop integral gain
KD	double	PID servo loop derivative gain
KS	double	PID integral saturation value (0 to 1)
IntegrationTime	double	PID integration time (seconds)
DerivativeFilterCutOffFrequency	double	PID derivative filter cut off frequency (Hz)
GKP	double	variable PID proportional gain multiplier
GKI	double	variable PID integral gain multiplier
GKD	double	variable PID derivative gain multiplier
KForm	double	variable PID form coefficient
FeedForwardGainAcceleration	double	Acceleration feedforward gain (units)

Return

Error	int32	Function error code

PYTHON

Prototype

[Error] **PositionerCorrectorPIDFFAccelerationSet** (integer SocketID, string FullPositionerName, bool ClosedLoopStatus, double KP, double KI, double KD, double KS, double IntegrationTime, double DerivativeFilterCutOffFrequency, double GKP, double GKI, double GKD, double KForm, double FeedForwardGainAcceleration)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name
ClosedLoopStatus	boolean	Position servo loop status (true=closed and false=opened)
KP	double	PID servo loop proportional gain
KI	double	PID servo loop integral gain
KD	double	PID servo loop derivative gain
KS	double	PID integral saturation value (0 to 1)
IntegrationTime	double	PID integration time (seconds)
DerivativeFilterCutOffFrequency	double	PID derivative filter cut off frequency (Hz)
GKP	double	variable PID proportional gain multiplier
GKI	double	variable PID integral gain multiplier
GKD	double	variable PID derivative gain multiplier
KForm	double	variable PID form coefficient
FeedForwardGainAcceleration	double	Acceleration feedforward gain (units)

Return

Error	integer	Function error code

2.2.4.17. PositionerCorrectorPIDFFVelocityGet

NAME

PositionerCorrectorPIDFFVelocityGet – Gets the corrector "PIDFFVelocity" parameters.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type and the corrector type: ERR_WRONG_OBJECT_TYPE (-8)
- Check output parameter type: ERR_WRONG_TYPE_BOOL (-12), ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function allow to return the corrector parameter values used by a PID with a velocity output: ClosedLoopStatus, KP, KI, KD, KS, IntegrationTime, DerivativeFilterCutOffFrequency, GKP, GKI, GKD, Kform and FeedForwardGainVelocity.

NOTE:
The "CorrectorType" must be "PIDFFVelocity" in the stages.ini file. This servo loop type is used when a constant value applied to the driver results in a constant velocity of the stage.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_BOOL (-12)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerCorrectorPIDFFVelocityGet $SocketID $FullPositionerName ClosedLoopStatus KP KI KD KS IntegrationTime DerivativeFilterCutOffFrequency GKP GKI GKD KForm FeedForwardGainVelocity

Input parameters

SocketID	integer	Socket identifier got from "TCP_ConnectToServer" function
FullPositionerName	string	Positioner name

Output parameters

ClosedLoopStatus	bool	Position servo loop status (true=closed and false=opened)
KP	double	PID servo loop proportional gain
KI	double	PID servo loop integral gain
KD	double	PID servo loop derivative gain
KS	double	PID integral saturation value (0 to 1)
IntegrationTime	double	PID integration time (seconds)
DerivativeFilterCutOffFrequency	double	PID derivative filter cut off frequency (Hz)
GKP	double	variable PID proportional gain multiplier
GKI	double	variable PID integral gain multiplier
GKD	double	variable PID derivative gain multiplier
KForm	double	variable PID form coefficient
FeedForwardGainVelocity	double	Velocity feedforward gain (units)

Return

Error	integer	TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerCorrectorPIDFFVelocityGet** (int SocketID, char FullPositionerName[250], bool* ClosedLoopStatus, double* KP, double* KI, double* KD, double* KS, double* IntegrationTime, double* DerivativeFilterCutOffFrequency, double* GKP, double* GKI, double* GKD, double* KForm, double* FeedForwardGainVelocity)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	char *	Positioner name

Output parameters

ClosedLoopStatus	bool *	Position servo loop status (true=closed and false=opened)
KP	double *	PID servo loop proportional gain
KI	double *	PID servo loop integral gain
KD	double *	PID servo loop derivative gain
KS	double *	PID integral saturation value (0 to 1)
IntegrationTime	double *	PID integration time (seconds)
DerivativeFilterCutOffFrequency	double *	PID derivative filter cut off frequency (Hz)
GKP	double *	variable PID proportional gain multiplier
GKI	double *	variable PID integral gain multiplier
GKD	double *	variable PID derivative gain multiplier
KForm	double *	variable PID form coefficient
FeedForwardGainVelocity	double *	Velocity feedforward gain (units)

Return

Error	int	Function error code

VISUAL BASIC

Prototype

Long **PositionerCorrectorPIDFFVelocityGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ClosedLoopStatus As Boolean, KP As Double, KI As Double, KD As Double, KS As Double, IntegrationTime As Double, DerivativeFilterCutOffFrequency As Double, GKP As Double, GKI As Double, GKD As Double, KForm As Double, FeedForwardGainVelocity As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	String	Positioner name

Output parameters

ClosedLoopStatus	Boolean	Position servo loop status (true=closed and false=opened)
KP	Double	PID servo loop proportional gain
KI	Double	PID servo loop integral gain
KD	Double	PID servo loop derivative gain
KS	Double	PID integral saturation value (0 to 1)
IntegrationTime	Double	PID integration time (seconds)
DerivativeFilterCutOffFrequency	Double	PID derivative filter cut off frequency (Hz)
GKP	Double	variable PID proportional gain multiplier
GKI	Double	variable PID integral gain multiplier
GKD	Double	variable PID derivative gain multiplier
KForm	Double	variable PID form coefficient
FeedForwardGainVelocity	Double	Velocity feedforward gain (units)

Return

Error	Long	Function error code

MATLAB

Prototype

[Error, ClosedLoopStatus, KP, KI, KD, KS, IntegrationTime, DerivativeFilterCutOffFrequency, GKP, GKI, GKD, KForm, FeedForwardGainVelocity] **PositionerCorrectorPIDFFVelocityGet** (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring ..... Positioner name

Return

Error ..... int32 ..... Function error code
ClosedLoopStatus ..... boolean ..... Position servo loop status (true=closed and false=opened)
KP ..... double ..... PID servo loop proportional gain
KI ..... double ..... PID servo loop integral gain
KD ..... double ..... PID servo loop derivative gain
KS ..... double ..... PID integral saturation value (0 to 1)
IntegrationTime ..... double ..... PID integration time (seconds)
DerivativeFilterCutOffFrequency ..... double ..... PID derivative filter cut off frequency (Hz)
GKP ..... double ..... variable PID proportional gain multiplier
GKI ..... double ..... variable PID integral gain multiplier
GKD ..... double ..... variable PID derivative gain multiplier
KForm ..... double ..... variable PID form coefficient
FeedForwardGainVelocity ..... double ..... Velocity feedforward gain (units)

PYTHON

Prototype

[Error, ClosedLoopStatus, KP, KI, KD, KS, IntegrationTime, DerivativeFilterCutOffFrequency, GKP, GKI, GKD, KForm, FeedForwardGainVelocity] **PositionerCorrectorPIDFFVelocityGet** (integer SocketID, string FullPositionerName)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string ..... Positioner name

Return

Error ..... integer ..... Function error code
ClosedLoopStatus ..... boolean ..... Position servo loop status (true=closed and false=opened)
KP ..... double ..... PID servo loop proportional gain
KI ..... double ..... PID servo loop integral gain
KD ..... double ..... PID servo loop derivative gain
KS ..... double ..... PID integral saturation value (0 to 1)
IntegrationTime ..... double ..... PID integration time (seconds)
DerivativeFilterCutOffFrequency ..... double ..... PID derivative filter cut off frequency (Hz)
GKP ..... double ..... variable PID proportional gain multiplier
GKI ..... double ..... variable PID integral gain multiplier
GKD ..... double ..... variable PID derivative gain multiplier
KForm ..... double ..... variable PID form coefficient
FeedForwardGainVelocity ..... double ..... Velocity feedforward gain (units)

2.2.4.18. PositionerCorrectorPIDFFVelocitySet

NAME

PositionerCorrectorPIDFFVelocitySet – Configures the corrector “PIDFFVelocity” parameters.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type and the corrector type: ERR_WRONG_OBJECT_TYPE (-8)
- Check input parameter type: ERR_WRONG_TYPE_BOOL (-12), ERR_WRONG_TYPE_DOUBLE (-14)
- Check parameter value: ERR_PARAMETER_OUT_OF_RANGE (-17)

$KP \geq 0$
$KI \geq 0$
$KD \geq 0$
$0 \leq KS \leq 1$
$IntegrationTime \geq CorrectorPeriod\ (0.0001\ s)$
$GKP > -1$
$GKI > -1$
$GKD > -1$
$KForm \geq 0$
$KFeedForwardVelocity \geq 0$

$$DerivativeFilterCutOffFrequency \in \left[0 : \frac{0.5}{\text{CorrectorPeriod}}\right] => [0 : 5000] \text{ with } CorrectorPeriod = 0.0001\ s$$

DESCRIPTION

This function allows to configure the “PIDFFVelocity” corrector parameters.

NOTE:
The “CorrectorType” parameter must be defined as “PIDFFVelocity” in the stages.ini file else the ERR_WRONG_OBJECT_TYPE (-8) error is returned. This servo loop type is used when a constant value applied to the driver results in a constant velocity of the stage.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_BOOL (-12)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerCorrectorPIDFFVelocitySet $SocketID $FullPositionerName $ClosedLoopStatus $KP $KI $KD $KS $IntegrationTime $DerivativeFilterCutOffFrequency $GKP $GKI $GKD $KForm $FeedForwardGainVelocity

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name
ClosedLoopStatus	bool	Position servo loop status (true=closed and false=opened)
KP	double	PID servo loop proportional gain
KI	double	PID servo loop integral gain
KD	double	PID servo loop derivative gain
KS	double	PID integral saturation value (0 to 1)
IntegrationTime	double	PID integration time (seconds)
DerivativeFilterCutOffFrequency	double	PID derivative filter cut off frequency (Hz)
GKP	double	variable PID proportional gain multiplier
GKI	double	variable PID integral gain multiplier
GKD	double	variable PID derivative gain multiplier
KForm	double	variable PID form coefficient
FeedForwardGainVelocity	double	Velocity feedforward gain (units)

Output parameters

None

Return

Error	integer	TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerCorrectorPIDFFVelocitySet** (int SocketID, char FullPositionerName[250], bool ClosedLoopStatus, double KP, double KI, double KD, double KS, double IntegrationTime, double DerivativeFilterCutOffFrequency, double GKP, double GKI, double GKD, double KForm, double FeedForwardGainVelocity)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	char *	Positioner name
ClosedLoopStatus	bool	Position servo loop status (true=closed and false=opened)
KP	double	PID servo loop proportional gain
KI	double	PID servo loop integral gain
KD	double	PID servo loop derivative gain
KS	double	PID integral saturation value (0 to 1)
IntegrationTime	double	PID integration time (seconds)
DerivativeFilterCutOffFrequency	double	PID derivative filter cut off frequency (Hz)
GKP	double	variable PID proportional gain multiplier
GKI	double	variable PID integral gain multiplier
GKD	double	variable PID derivative gain multiplier
KForm	double	variable PID form coefficient
FeedForwardGainVelocity	double	Velocity feedforward gain (units)

Output parameters

None

Return

Error	int	Function error code

VISUAL BASIC

Prototype

Long **PositionerCorrectorPIDFFVelocitySet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal ClosedLoopStatus As Boolean, ByVal KP As Double, ByVal KI As Double, ByVal KD As Double, ByVal KS As Double, ByVal IntegrationTime As Double, ByVal DerivativeFilterCutOffFrequency As Double, ByVal GKP As Double, ByVal GKI As Double, ByVal GKD As Double, ByVal KForm As Double, ByVal FeedForwardGainVelocity As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	String	Positioner name
ClosedLoopStatus	Boolean	Position servo loop status (true=closed and false=opened)
KP	Double	PID servo loop proportional gain
KI	Double	PID servo loop integral gain
KD	Double	PID servo loop derivative gain
KS	Double	PID integral saturation value (0 to 1)
IntegrationTime	Double	PID integration time (seconds)
DerivativeFilterCutOffFrequency	Double	PID derivative filter cut off frequency (Hz)
GKP	Double	variable PID proportional gain multiplier
GKI	Double	variable PID integral gain multiplier
GKD	Double	variable PID derivative gain multiplier
KForm	Double	variable PID form coefficient
FeedForwardGainVelocity	Double	Velocity feedforward gain (units)

Output parameters

None

Return

Error	Long	Function error code

MATLAB

Prototype

[Error] **PositionerCorrectorPIDFFVelocitySet** (int32 SocketID, cstring FullPositionerName, bool ClosedLoopStatus, double KP, double KI, double KD, double KS, double IntegrationTime, double DerivativeFilterCutOffFrequency, double GKP, double GKI, double GKD, double KForm, double FeedForwardGainVelocity)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	cstring	Positioner name
ClosedLoopStatus	boolean	Position servo loop status (true=closed and false=opened)
KP	double	PID servo loop proportional gain
KI	double	PID servo loop integral gain
KD	double	PID servo loop derivative gain
KS	double	PID integral saturation value (0 to 1)
IntegrationTime	double	PID integration time (seconds)
DerivativeFilterCutOffFrequency	double	PID derivative filter cut off frequency (Hz)
GKP	double	variable PID proportional gain multiplier
GKI	double	variable PID integral gain multiplier
GKD	double	variable PID derivative gain multiplier
KForm	double	variable PID form coefficient
FeedForwardGainVelocity	double	Velocity feedforward gain (units)

Return

Error	int32	Function error code

PYTHON

Prototype

[Error] **PositionerCorrectorPIDFFVelocitySet** (integer SocketID, string FullPositionerName, bool ClosedLoopStatus, double KP, double KI, double KD, double KS, double IntegrationTime, double DerivativeFilterCutOffFrequency, double GKP, double GKI, double GKD, double KForm, double FeedForwardGainVelocity)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name
ClosedLoopStatus	boolean	Position servo loop status (true=closed and false=opened)
KP	double	PID servo loop proportional gain
KI	double	PID servo loop integral gain
KD	double	PID servo loop derivative gain
KS	double	PID integral saturation value (0 to 1)
IntegrationTime	double	PID integration time (seconds)
DerivativeFilterCutOffFrequency	double	PID derivative filter cut off frequency (Hz)
GKP	double	variable PID proportional gain multiplier
GKI	double	variable PID integral gain multiplier
GKD	double	variable PID derivative gain multiplier
KForm	double	variable PID form coefficient
FeedForwardGainVelocity	double	Velocity feedforward gain (units)

Return

Error	integer	Function error code

2.2.4.19. PositionerCorrectorPIPositionGet

NAME

PositionerCorrectorPIPositionGet – Gets the corrector “PIPosition” parameters.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type and the corrector type: ERR_WRONG_OBJECT_TYPE (-8)
- Check output parameter type: ERR_WRONG_TYPE_BOOL (-12), ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function allow to return the corrector parameter values used by a PI with a position output: ClosedLoopStatus, KP, KI and IntegrationTime.

NOTE:
The “CorrectorType” must be “PIPosition” in the stages.ini file. This servo loop type is used when the position servo loop outputs directly a position value.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_BOOL (-12)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerCorrectorPIPositionGet $SocketID $FullPositionerName ClosedLoopStatus KP KI IntegrationTime

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name

Output parameters

ClosedLoopStatus	bool	Position servo loop status (true=closed and false=opened)
KP	double	PID servo loop proportional gain
KI	double	PID servo loop integral gain
IntegrationTime	double	PID integration time (seconds)

Return

Error	integer	TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerCorrectorPIPositionGet** (int SocketID, char FullPositionerName[250], bool* ClosedLoopStatus, double* KP, double* KI, double* IntegrationTime)

Input parameters

SocketID .....int ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....char *..... Positioner name

Output parameters

ClosedLoopStatus .....bool * ..... Position servo loop status (true=closed and false=opened)
KP .....double *..... PID servo loop proportional gain
KI .....double *..... PID servo loop integral gain
IntegrationTime .....double *..... PID integration time (seconds)

Return

Error..... int..... Function error code

VISUAL BASIC

Prototype

Long **PositionerCorrectorPIPositionGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ClosedLoopStatus As Boolean, KP As Double, KI As Double, IntegrationTime As Double)

Input parameters

SocketID .....Long ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....String..... Positioner name

Output parameters

ClosedLoopStatus .....Boolean ..... Position servo loop status (true=closed and false=opened)
KP .....Double..... PID servo loop proportional gain
KI .....Double..... PID servo loop integral gain
IntegrationTime .....Double..... PID integration time (seconds)

Return

Error.....Long ..... Function error code

MATLAB

Prototype

[Error, ClosedLoopStatus, KP, KI, IntegrationTime] **PositionerCorrectorPIPositionGet** (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID .....int32 ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....cstring..... Positioner name

Return

Error.....int32 ..... Function error code
ClosedLoopStatus .....boolean..... Position servo loop status (true=closed and false=opened)
KP .....double..... PID servo loop proportional gain
KI .....double..... PID servo loop integral gain
IntegrationTime .....double..... PID integration time (seconds)

PYTHON

Prototype

[Error, ClosedLoopStatus, KP, KI, IntegrationTime] **PositionerCorrectorPIPositionGet** (integer SocketID, string FullPositionerName)

Input parameters

SocketID .....integer .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....string .....Positioner name

Return

Error.....integer .....Function error code
ClosedLoopStatus .....boolean.....Position servo loop status (true=closed and false=opened)
KP .....double.....PID servo loop proportional gain
KI .....double.....PID servo loop integral gain
IntegrationTime .....double.....PID integration time (seconds)

2.2.4.20. PositionerCorrectorPIPositionSet

NAME

PositionerCorrectorPIPositionSet – Configures the corrector "PIPosition" parameters.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type and the corrector type: ERR_WRONG_OBJECT_TYPE (-8)
- Check input parameter type: ERR_WRONG_TYPE_BOOL (-12), ERR_WRONG_TYPE_DOUBLE (-14)
- Check parameter value: ERR_PARAMETER_OUT_OF_RANGE (-17)

KP ≥ 0
KI ≥ 0
IntegrationTime ≥ CorrectorPeriod (0.0001 s)

DESCRIPTION

This function allows to configure the "PIPosition" corrector parameters.

NOTE:
The "CorrectorType" parameter must be defined as "PIPosition" in the stages.ini file else the ERR_WRONG_OBJECT_TYPE (-8) error is returned. This servo loop type is used when the position servo loop outputs directly a position value.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_BOOL (-12)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype
PositionerCorrectorPIPositionSet $SocketID $FullPositionerName $ClosedLoopStatus $KP $KI $IntegrationTime

Input parameters

SocketID	integer	Socket identifier got from "TCP_ConnectToServer" function
FullPositionerName	string	Positioner name
ClosedLoopStatus	bool	Position servo loop status (true=closed and false=opened)
KP	double	PID servo loop proportional gain
KI	double	PID servo loop integral gain
IntegrationTime	double	PID integration time (seconds)

Output parameters
None

Return

Error	integer	TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerCorrectorPIPositionSet** (int SocketID, char FullPositionerName[250], bool ClosedLoopStatus, double KP, double KI, double IntegrationTime)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	char *	Positioner name
ClosedLoopStatus	bool	Position servo loop status (true=closed and false=opened)
KP	double	PID servo loop proportional gain
KI	double	PID servo loop integral gain
IntegrationTime	double	PID integration time (seconds)

Output parameters

None

Return

Error	int	Function error code

VISUAL BASIC

Prototype

Long **PositionerCorrectorPIPositionSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal ClosedLoopStatus As Boolean, ByVal KP As Double, ByVal KI As Double, ByVal IntegrationTime As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	String	Positioner name
ClosedLoopStatus	Boolean	Position servo loop status (true=closed and false=opened)
KP	Double	PID servo loop proportional gain
KI	Double	PID servo loop integral gain
IntegrationTime	Double	PID integration time (seconds)

Output parameters

None

Return

Error	Long	Function error code

MATLAB

Prototype

[Error] **PositionerCorrectorPIPositionSet** (int32 SocketID, cstring FullPositionerName, bool ClosedLoopStatus, double KP, double KI, double IntegrationTime)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	cstring	Positioner name
ClosedLoopStatus	boolean	Position servo loop status (true=closed and false=opened)
KP	double	PID servo loop proportional gain
KI	double	PID servo loop integral gain
IntegrationTime	double	PID integration time (seconds)

Return

Error	int32	Function error code

PYTHON

Prototype

[Error] **PositionerCorrectorPIPositionSet** (integer SocketID, string FullPositionerName, bool ClosedLoopStatus, double KP, double KI, double IntegrationTime)

Input parameters

SocketID .....integer .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....string .....Positioner name
ClosedLoopStatus .....boolean.....Position servo loop status (true=closed and false=opened)
KP .....double.....PID servo loop proportional gain
KI .....double.....PID servo loop integral gain
IntegrationTime .....double.....PID integration time (seconds)

Return

Error.....integer .....Function error code

2.2.4.21. PositionerCorrectorTypeGet

NAME

PositionerCorrectorTypeGet – Returns the corrector type.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type: ERR_WRONG_OBJECT_TYPE (-8)
- Check output parameter type: ERR_WRONG_TYPE_CHAR (-13)

DESCRIPTION

This function returns the corrector type used by the selected positioner.

The corrector type can be one of this list:

1. PositionerCorrectorPIDFFAcceleration
2. PositionerCorrectorPIDFFVelocity
3. PositionerCorrectorPIDDualFFVoltage
4. PositionerCorrectorPIPosition
5. NoCorrector

NOTE:
The corrector type is defined in the stages.ini file with the “CorrectorType” parameter.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
SUCCESS (0) : no error

TCL

Prototype

PositionerCorrectorTypeGet $SocketID $FullPositionerName CorrectorType

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string ..... Positioner name

Output parameters

CorrectorType ..... string ..... Corrector type

Return

Error ..... integer ..... TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerCorrectorTypeGet** (int SocketID, char FullPositionerName[250], char* CorrectorType)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... char * ..... Positioner name

Output parameters

CorrectorType ..... char * ..... Corrector type

Return

Error ..... int ..... Function error code

VISUAL BASIC

Prototype

Long **PositionerCorrectorTypeGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal CorrectorType As String)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String ..... Positioner name

Output parameters

CorrectorType ..... String ..... Corrector type

Return

Error ..... Long ..... Function error code

MATLAB

Prototype

[Error, CorrectorType] **PositionerCorrectorTypeGet** (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring ..... Positioner name

Return

Error ..... int32 ..... Function error code
CorrectorType ..... cstring ..... Corrector type

PYTHON

Prototype

[Error, CorrectorType] **PositionerCorrectorTypeGet** (integer SocketID, string FullPositionerName)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string ..... Positioner name

Return

Error ..... integer ..... Function error code
CorrectorType ..... string ..... Corrector type

2.2.4.22. PositionerCurrentVelocityAccelerationFiltersGet

NAME

PositionerCurrentVelocityAccelerationFiltersGet – Gets the velocity and acceleration filter cut off frequencies.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type: ERR_WRONG_OBJECT_TYPE (-8)
- Check output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function allows to return the current velocity cut off frequency and the current acceleration cut off frequency used by the gathering for the selected positioner.
The gathering uses these parameters to filter the current velocity and the current acceleration. These parameters are defined in the stages.ini file.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerCurrentVelocityAccelerationFiltersGet $SocketID $FullPositionerName VelocityCutOffFrequency AccelerationCutOffFrequency

Input parameters

SocketID ..... integer..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string..... Positioner name

Output parameters

VelocityCutOffFrequency..... double..... Velocity filter cut off frequency (Hz)
AccelerationCutOffFrequency.... double..... Acceleration filter cut off frequency (Hz)

Return

Error..... integer..... TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerCurrentVelocityAccelerationFiltersGet** (int SocketID, char FullPositionerName[250] , double* VelocityCutOffFrequency, double* AccelerationCutOffFrequency)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... char * ..... Positioner name

Output parameters

VelocityCutOffFrequency..... double * ..... Velocity filter cut off frequency (Hz)
AccelerationCutOffFrequency.... double * ..... Acceleration filter cut off frequency (Hz)

Return

Error..... int..... Function error code

VISUAL BASIC

Prototype

Long **PositionerCurrentVelocityAccelerationFiltersGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, VelocityCutOffFrequency As Double, AccelerationCutOffFrequency As Double)

Input parameters

SocketID ..... Long .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String .....Positioner name

Output parameters

VelocityCutOffFrequency..... double .....Velocity filter cut off frequency (Hz)
AccelerationCutOffFrequency.... double .....Acceleration filter cut off frequency (Hz)

Return

Error..... Long .....Function error code

MATLAB

Prototype

[Error, VelocityCutOffFrequency, AccelerationCutOffFrequency]
PositionerCurrentVelocityAccelerationFiltersGet (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID ..... int32.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring.....Positioner name

Return

Error..... int32.....Function error code
VelocityCutOffFrequency..... double .....Velocity filter cut off frequency (Hz)
AccelerationCutOffFrequency.... double .....Acceleration filter cut off frequency (Hz)

PYTHON

Prototype

[Error, VelocityCutOffFrequency, AccelerationCutOffFrequency]
PositionerCurrentVelocityAccelerationFiltersGet (integer SocketID, string FullPositionerName)

Input parameters

SocketID ..... integer.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string.....Positioner name

Return

Error..... integer.....Function error code
VelocityCutOffFrequency..... double .....Velocity filter cut off frequency (Hz)
AccelerationCutOffFrequency.... double .....Acceleration filter cut off frequency (Hz)

2.2.4.23. PositionerCurrentVelocityAccelerationFiltersSet

NAME

PositionerCurrentVelocityAccelerationFiltersSet – Sets the velocity and acceleration filter cut off frequencies.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type: ERR_WRONG_OBJECT_TYPE (-8)
- Check output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)
- Check parameter value: ERR_PARAMETER_OUT_OF_RANGE (-17)

$$VelocityCutOffFrequency \in \left[0 : \frac{0.5}{CorrectorISRPeriod}\right]$$

$$AccelerationCutOffFrequency \in \left[0 : \frac{0.5}{CorrectorISRPeriod}\right]$$

Note: Refer to *system.ref* file to get CorrectorISRPeriod value.

DESCRIPTION

This function allows to set a new velocity cut off frequency and a new acceleration cut-off frequency for the selected positioner.
The gathering uses these parameters to filter the current velocity and the current acceleration. These parameters are defined in the stages.ini file.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerCurrentVelocityAccelerationFiltersSet $SocketID $FullPositionerName $VelocityCutOffFrequency $AccelerationCutOffFrequency

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name
VelocityCutOffFrequency	double	Velocity filter cut off frequency (Hz)
AccelerationCutOffFrequency	double	Acceleration filter cut off frequency (Hz)

Output parameters

None

Return

Error	integer	TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerCurrentVelocityAccelerationFiltersSet** (int SocketID, char FullPositionerName[250] , double VelocityCutOffFrequency, double AccelerationCutOffFrequency)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	char *	Positioner name
VelocityCutOffFrequency	double	Velocity filter cut off frequency (Hz)
AccelerationCutOffFrequency	double	Acceleration filter cut off frequency (Hz)

Output parameters

None

Return

Error	int	Function error code

VISUAL BASIC

Prototype

Long **PositionerCurrentVelocityAccelerationFiltersSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal VelocityCutOffFrequency As Double, ByVal AccelerationCutOffFrequency As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	String	Positioner name
VelocityCutOffFrequency	double	Velocity filter cut off frequency (Hz)
AccelerationCutOffFrequency	double	Acceleration filter cut off frequency (Hz)

Output parameters

None

Return

Error	Long	Function error code

MATLAB

Prototype

[Error] **PositionerCurrentVelocityAccelerationFiltersSet** (int32 SocketID, cstring FullPositionerName, double VelocityCutOffFrequency, double AccelerationCutOffFrequency)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	cstring	Positioner name
VelocityCutOffFrequency	double	Velocity filter cut off frequency (Hz)
AccelerationCutOffFrequency	double	Acceleration filter cut off frequency (Hz)

Return

Error	int32	Function error code

PYTHON

Prototype

[Error] **PositionerCurrentVelocityAccelerationFiltersSet** (integer SocketID, string FullPositionerName, double VelocityCutOffFrequency, double AccelerationCutOffFrequency)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name
VelocityCutOffFrequency	double	Velocity filter cut off frequency (Hz)
AccelerationCutOffFrequency	double	Acceleration filter cut off frequency (Hz)

Return

Error	integer	Function error code

2.2.4.24. PositionerDriverFiltersGet (NEW)

NAME

PositionerDriverFiltersGet – Gets the piezo driver notch and lowpass filters parameters.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type: ERR_WRONG_OBJECT_TYPE (-8)
- Check output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)
- Check driver type, if not PIEZO: ERR_UNCOMPATIBLE (-24)
- If piezo driver, check if driver is not initialized: ERR_NOT_ALLOWED_DRIVER_NOT_INITIALIZED (-118)

DESCRIPTION

This function allows to return current values of the piezo driver filters parameters (KI, notch frequency, notch bandwidth, notch gain, lowpass frequency).

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
ERR_UNCOMPATIBLE (-24)
ERR_NOT_ALLOWED_DRIVER_NOT_INITIALIZED (-118)
SUCCESS (0) : no error

TCL

Prototype

PositionerDriverFiltersGet $SocketID $FullPositionerName KI NotchFrequency NotchBandwidth NotchGain LowpassFrequency

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name

Output parameters

KI	double	Driver KI
NotchFrequency	double	Driver notch frequency (Hz)
NotchBandwidth	double	Driver notch bandwidth (Hz)
NotchGain	double	Driver notch gain
LowpassFrequency	double	Driver lowpass frequency (Hz)

Return

Error	integer	TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerDriverFiltersGet** (int SocketID, char FullPositionerName[250] , double *KI, double* NotchFrequency, double* NotchBandwidth, double* NotchGain, double* LowpassFrequency)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	char *	Positioner name

Output parameters

KI	double *	Driver KI
NotchFrequency	double *	Driver notch frequency (Hz)
NotchBandwidth	double *	Driver notch bandwidth (Hz)
NotchGain	double *	Driver notch gain
LowpassFrequency	double *	Driver lowpass frequency (Hz)

Return

Error	int	Function error code

VISUAL BASIC

Prototype

Long **PositionerDriverFiltersGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, KI As Double, NotchFrequency As Double, NotchBandwidth As Double, NotchGain As Double, LowpassFrequency As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	String	Positioner name

Output parameters

KI	Double	Driver KI
NotchFrequency	Double	Driver notch frequency (Hz)
NotchBandwidth	Double	Driver notch bandwidth (Hz)
NotchGain	Double	Driver notch gain
LowpassFrequency	Double	Driver lowpass frequency (Hz)

Return

Error	Long	Function error code

MATLAB

Prototype

[Error, KI, NotchFrequency, NotchBandwidth, NotchGain, LowpassFrequency] **PositionerDriverFiltersGet** (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	cstring	Positioner name

Return

Error	int32	Function error code
KI	double	Driver KI
NotchFrequency	double	Driver notch frequency (Hz)
NotchBandwidth	double	Driver notch bandwidth (Hz)
NotchGain	double	Driver notch gain
LowpassFrequency	double	Driver lowpass frequency (Hz)

PYTHON

Prototype

[Error, KI, NotchFrequency, NotchBandwidth,NotchGain, LowpassFrequency] **PositionerDriverFiltersGet** (integer SocketID, string FullPositionerName)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name

Return

Error	integer	Function error code
KI	double	Driver KI
NotchFrequency	double	Driver notch frequency (Hz)
NotchBandwidth	double	Driver notch bandwidth (Hz)
NotchGain	double	Driver notch gain
LowpassFrequency	double	Driver lowpass frequency (Hz)

2.2.4.25. PositionerDriverFiltersSet (NEW)

NAME

PositionerDriverFiltersSet – Set the piezo driver filters parameters.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type: ERR_WRONG_OBJECT_TYPE (-8)
- Check output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)
- Check parameter value: ERR_PARAMETER_OUT_OF_RANGE (-17)
- Check driver type, if not PIEZO: ERR_UNCOMPATIBLE (-24)
- If piezo driver, check group and driver status:
  - If the driver is not initialized: ERR_NOT_ALLOWED_DRIVER_NOT_INITIALIZED (-118)
  - If the group state is NOTREF or READY: ERR_FUNCTION_ONLY_ALLOWED_IN_DISABLED_STATE (-117)

$$KI >= 0$$

$$NotchFrequency \in \left[0 : \frac{0.5}{\text{CorrectorISRPeriod}}\right]$$

$$NotchBandwith \in \left[0 : \frac{0.5}{\text{CorrectorISRPeriod}}\right]$$

$$NotchGain \in [0 : 100]$$

$$LowpassFrequency \in \left[0 : \frac{0.5}{\text{CorrectorISRPeriod}}\right]$$

Note: Refer to *system.ref* file to get CorrectorISRPeriod value.

DESCRIPTION

This function allows to set parameters of the driver (KI integral, notch and lowpass filters) for a piezo driver positioner.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
ERR_UNCOMPATIBLE (-24)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_MOTOR_INITIALIZATION_ERROR (-50)
ERR_FUNCTION_ONLY_ALLOWED_IN_DISABLED_STATE (-117)
ERR_NOT_ALLOWED_DRIVER_NOT_INITIALIZED (-118)
SUCCESS (0) : no error

TCL

Prototype

PositionerDriverFiltersSet $SocketID $FullPositionerName $KI $NotchFrequency $NotchBandwidth $NotchGain $LowpassFrequency

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name
KI	double	Driver KI
NotchFrequency	double	Driver notch frequency (Hz)
NotchBandwidth	double	Driver notch bandwidth (Hz)
NotchGain	double	Driver notch gain
LowpassFrequency	double	Driver lowpass frequency (Hz)

Output parameters

None

Return

Error	integer	TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerDriverFiltersSet** (int SocketID, char FullPositionerName[250] , double KI, double NotchFrequency, double NotchBandwidth, double NotchGain, double LowpassFrequency)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	char *	Positioner name
KI	double	Driver KI
NotchFrequency	double	Driver notch frequency (Hz)
NotchBandwidth	double	Driver notch bandwidth (Hz)
NotchGain	double	Driver notch gain
LowpassFrequency	double	Driver lowpass frequency (Hz)

Output parameters

None

Return

Error	int	Function error code

VISUAL BASIC

Prototype

Long **PositionerDriverFiltersSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal KI As Double, ByVal NotchFrequency As Double, ByVal NotchBandwidth As Double, ByVal NotchGain As Double, ByVal LowpassFrequency As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	String	Positioner name
KI	Double	Driver KI
NotchFrequency	Double	Driver notch frequency (Hz)
NotchBandwidth	Double	Driver notch bandwidth (Hz)
NotchGain	Double	Driver notch gain
LowpassFrequency	Double	Driver lowpass frequency (Hz)

Output parameters

None

Return

Error	Long	Function error code

MATLAB

Prototype

[Error] **PositionerDriverFiltersSet** (int32 SocketID, cstring FullPositionerName, double KI, double NotchFrequency, double NotchBandwidth, double NotchGain, double LowpassFrequency)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	cstring	Positioner name
KI	double	Driver KI
NotchFrequency	double	Driver notch frequency (Hz)
NotchBandwidth	double	Driver notch bandwidth (Hz)
NotchGain	double	Driver notch gain
LowpassFrequency	double	Driver lowpass frequency (Hz)

Return

Error	int32	Function error code

PYTHON

Prototype

[Error] **PositionerDriverFiltersSet** (integer SocketID, string FullPositionerName, double KI, double NotchFrequency, double NotchBandwidth, double NotchGain, double LowpassFrequency)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name
KI	double	Driver KI
NotchFrequency	double	Driver notch frequency (Hz)
NotchBandwidth	double	Driver notch bandwidth (Hz)
NotchGain	double	Driver notch gain
LowpassFrequency	double	Driver lowpass frequency (Hz)

Return

Error	integer	Function error code

2.2.4.26. PositionerDriverPositionOffsetsGet (NEW)

NAME

PositionerDriverPositionOffsetsGet – Gets current value of piezo driver stage and gage position offsets.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type: ERR_WRONG_OBJECT_TYPE (-8)
- Check output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)
- Check driver type, if not PIEZO: ERR_UNCOMPATIBLE (-24)
- If piezo driver, check group and driver status:
  - If the driver is not initialized: ERR_NOT_ALLOWED_DRIVER_NOT_INITIALIZED (-118)
  - If the group state is NOTREF or READY: ERR_FUNCTION_ONLY_ALLOWED_IN_DISABLED_STATE (-117)

DESCRIPTION

This function allows to return current value of the piezo driver position offset parameters (stage position offset, gage position offset).

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
ERR_UNCOMPATIBLE (-24)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_FUNCTION_ONLY_ALLOWED_IN_DISABLED_STATE (-117)
ERR_NOT_ALLOWED_DRIVER_NOT_INITIALIZED (-118)
SUCCESS (0) : no error

TCL

Prototype

PositionerDriverPositionOffsetsGet $SocketID $FullPositionerName StagePositionOffset GagePositionOffset

Input parameters

SocketID ..... integer..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string..... Positioner name

Output parameters

StagePositionOffset ..... double..... Driver stage position offset (units)
GagePositionOffset ..... double..... Driver gage position offset (units)

Return

Error..... integer..... TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerDriverPositionOffsetsGet** (int SocketID, char FullPositionerName[250] , double* StagePositionOffset, double* GagePositionOffset)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function

FullPositionerName ..... char * .....Positioner name

Output parameters

StagePositionOffset ..... double * .....Driver stage position offset (units)
GagePositionOffset ..... double * .....Driver gage position offset (units)

Return

Error ..... int .....Function error code

VISUAL BASIC

Prototype

Long **PositionerDriverPositionOffsetsGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, StagePositionOffset As Double, GagePositionOffset As Double)

Input parameters

SocketID ..... Long .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String .....Positioner name

Output parameters

StagePositionOffset ..... Double .....Driver stage position offset (units)
GagePositionOffset ..... Double .....Driver gage position offset (units)

Return

Error ..... Long .....Function error code

MATLAB

Prototype

[Error, StagePositionOffset, GagePositionOffset] **PositionerDriverPositionOffsetsGet** (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID ..... int32.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring .....Positioner name

Return

Error ..... int32.....Function error code
StagePositionOffset ..... double .....Driver stage position offset (units)
GagePositionOffset ..... double .....Driver gage position offset (units)

PYTHON

Prototype

[Error, StagePositionOffset, GagePositionOffset] **PositionerDriverPositionOffsetsGet** (integer SocketID, string FullPositionerName)

Input parameters

SocketID ..... integer.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string.....Positioner name

Return

Error ..... integer.....Function error code
StagePositionOffset ..... double .....Driver stage position offset (units)
GagePositionOffset ..... double .....Driver gage position offset (units)

2.2.4.27. PositionerDriverStatusGet

NAME

PositionerDriverStatusGet – Gets the positioner driver status code.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner (must be not a secondary positioner): ERR_WRONG_OBJECT_TYPE (-8), ERR_UNCOMPATIBLE (-24), ERR_POSITIONER_NAME (-18)
- Check output parameter type: ERR_WRONG_TYPE_INT (-15)

DESCRIPTION

This function allows to return the positioner driver status from the driver board.
Use the “PositionerDriverStatusStringGet” function to get the driver status description.

NOTE:
See the positioner driver status list describes in § 2.18

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_POSITIONER_NAME (-18)
ERR_UNCOMPATIBLE (-24)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

PositionerDriverStatusGet $SocketID $FullPositionerName PositionerDriverStatus

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name

Output parameters

PositionerDriverStatus	integer	Driver status code

Return

Error	integer	TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerDriverStatusGet** (int SocketID, char FullPositionerName[250] , int * PositionerDriverStatus)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	char *	Positioner name

Output parameters

PositionerDriverStatus	int *	Driver status code

Return

Error	int	Function error code

VISUAL BASIC

Prototype

Long **PositionerDriverStatusGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, PositionerDriverStatus As Integer)

Input parameters

SocketID ..... Long .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String .....Positioner name

Output parameters

PositionerDriverStatus ..... Integer.....Driver status code

Return

Error..... Long .....Function error code

MATLAB

Prototype

[Error, PositionerDriverStatus] **PositionerDriverStatusGet** (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID ..... int32.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring .....Positioner name

Return

Error..... int32.....Function error code
PositionerDriverStatus ..... int32.....Driver status code

PYTHON

Prototype

[Error, PositionerDriverStatus] **PositionerDriverStatusGet** (integer SocketID, string FullPositionerName)

Input parameters

SocketID ..... integer.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string.....Positioner name

Return

Error..... integer.....Function error code
PositionerDriverStatus ..... integer.....Driver status code

2.2.4.28. PositionerDriverStatusStringGet

NAME

PositionerDriverStatusStringGet – Gets the positioner driver status description.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check input parameter type: ERR_WRONG_TYPE_INT (-15)
- Check output parameter type: ERR_WRONG_TYPE_CHAR (-13)

DESCRIPTION

This function returns a driver status description from a positioner driver status code.

NOTE:
See the positioner driver status list describes in § 2.18

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

PositionerDriverStatusStringGet $SocketID $FullPositionerName $PositionerDriverStatus PositionerDriverStatusString

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name
PositionerDriverStatus	integer	Driver status code

Output parameters

PositionerDriverStatusString	integer	Driver status description

Return

Error	integer	TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerDriverStatusStringGet** (int SocketID, char FullPositionerName[250] , int PositionerDriverStatus, char * PositionerDriverStatusString)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	char *	Positioner name
PositionerDriverStatus	int *	Driver status code

Output parameters

PositionerDriverStatusString	int *	Driver status description

Return

Error	int	Function error code

VISUAL BASIC

Prototype

Long **PositionerDriverStatusStringGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal PositionerDriverStatus As Integer, ByVal PositionerDriverStatusString As String)

Input parameters

SocketID ..... Long .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String .....Positioner name
PositionerDriverStatus ..... Integer.....Driver status code

Output parameters

PositionerDriverStatusString ..... Integer.....Driver status description

Return

Error..... Long .....Function error code

MATLAB

Prototype

[Error, PositionerDriverStatusString] **PositionerDriverStatusStringGet** (int32 SocketID, cstring FullPositionerName, int32 PositionerDriverStatus)

Input parameters

SocketID ..... int32.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring.....Positioner name
PositionerDriverStatus ..... int32.....Driver status code

Return

Error..... int32.....Function error code
PositionerDriverStatusString ..... cstring.....Driver status description

PYTHON

Prototype

[Error, PositionerDriverStatusString] **PositionerDriverStatusStringGet** (integer SocketID, string FullPositionerName, integer PositionerDriverStatus)

Input parameters

SocketID ..... integer.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string.....Positioner name
PositionerDriverStatus ..... integer.....Driver status code

Return

Error..... integer.....Function error code
PositionerDriverStatusString ..... string.....Driver status description

2.2.4.29. PositionerEncoderAmplitudeValuesGet

NAME

PositionerEncoderAmplitudeValuesGet – Gets the encoder amplitude values.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner: ERR_WRONG_OBJECT_TYPE (-8)
- Check the encoder type (must be “AnalogInterpolated”): ERR_WRONG_OBJECT_TYPE (-8)
- Check output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function returns the maximum and current amplitudes values (in volts) of the used analog encoder input.

CAUTION:
The encoder type must be “AnalogInterpolated” in the stages.ini file (“EncoderType” parameter).

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerEncoderAmplitudeValuesGet $SocketID $FullPositionerName MaxSinusAmplitude CurrentSinusAmplitude MaxCosinusAmplitude CurrentCosinusAmplitude

Input parameters

SocketID .....integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....string ..... Positioner name

Output parameters

MaxSinusAmplitude .....double..... Encoder sinus signal maximum amplitude value (Volts)
CurrentSinusAmplitude .....double..... Encoder sinus signal current amplitude value (Volts)
MaxCosinusAmplitude .....double..... Encoder cosinus signal maximum amplitude value (Volts)
CurrentCosinusAmplitude .....double..... Encoder cosinus signal current amplitude value (Volts)

Return

Error.....integer ..... TCL error code (0=success or 1=syntax error) or function error code

C / C++

Prototype

int **PositionerEncoderAmplitudeValuesGet** (int SocketID, char FullPositionerName[250] , double * MaxSinusAmplitude, double * CurrentSinusAmplitude, double * MaxCosinusAmplitude, double * CurrentCosinusAmplitude)

Input parameters

SocketID .....int ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....char *..... Positioner name

Output parameters

MaxSinusAmplitude .....double *..... Encoder sinus signal maximum amplitude value (Volts)
CurrentSinusAmplitude .....double *..... Encoder sinus signal current amplitude value (Volts)
MaxCosinusAmplitude .....double *..... Encoder cosinus signal maximum amplitude value (Volts)
CurrentCosinusAmplitude .....double *..... Encoder cosinus signal current amplitude value (Volts)

Return

Error.....int ..... Function error code

VISUAL BASIC

Prototype

Long **PositionerEncoderAmplitudeValuesGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, MaxSinusAmplitude As Double, CurrentSinusAmplitude As Double, MaxCosinusAmplitude As Double, CurrentCosinusAmplitude As Double)

Input parameters

SocketID ..... Long .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String .....Positioner name

Output parameters

MaxSinusAmplitude ..... Double .....Encoder sinus signal maximum amplitude value (Volts)
CurrentSinusAmplitude ..... Double .....Encoder sinus signal current amplitude value (Volts)
MaxCosinusAmplitude ..... Double .....Encoder cosinus signal maximum amplitude value (Volts)
CurrentCosinusAmplitude ..... Double .....Encoder cosinus signal current amplitude value (Volts)

Return

Error..... Long .....Function error code

MATLAB

Prototype

[Error, MaxSinusAmplitude, CurrentSinusAmplitude, MaxCosinusAmplitude, CurrentCosinusAmplitude] **PositionerEncoderAmplitudeValuesGet** (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID ..... int32.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring.....Positioner name

Return

Error..... int32.....Function error code
MaxSinusAmplitude ..... double.....Encoder sinus signal maximum amplitude value (Volts)
CurrentSinusAmplitude ..... double.....Encoder sinus signal current amplitude value (Volts)
MaxCosinusAmplitude ..... double.....Encoder cosinus signal maximum amplitude value (Volts)
CurrentCosinusAmplitude ..... double.....Encoder cosinus signal current amplitude value (Volts)

PYTHON

Prototype

[Error, MaxSinusAmplitude, CurrentSinusAmplitude, MaxCosinusAmplitude, CurrentCosinusAmplitude] **PositionerEncoderAmplitudeValuesGet** (integer SocketID, string FullPositionerName)

Input parameters

SocketID ..... integer.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string.....Positioner name

Return

Error..... integer.....Function error code
MaxSinusAmplitude ..... double.....Encoder sinus signal maximum amplitude value (Volts)
CurrentSinusAmplitude ..... double.....Encoder sinus signal current amplitude value (Volts)
MaxCosinusAmplitude ..... double.....Encoder cosinus signal maximum amplitude value (Volts)
CurrentCosinusAmplitude ..... double.....Encoder cosinus signal current amplitude value (Volts)

2.2.4.30. PositionerEncoderCalibrationParametersGet

NAME

PositionerEncoderCalibrationParametersGet – Gets the encoder calibration parameters.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner (must be not a secondary positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Check the encoder type (must be "AnalogInterpolated"): ERR_WRONG_OBJECT_TYPE (-8)
- Check output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

After a calibration of the analog encoder input (by the function "GroupInitializeWithEncoderCalibration"), this function returns the optimum parameters for the analog encoder interface. To take these parameters into account (recommended to achieve best performance), these values must be entered manually in the corresponding section of the stages.ini file. The parameters to set in the stages.ini file are:

```
EncoderSinusOffset = 0          ; Volts
EncoderCosinusOffset = 0        ; Volts
EncoderDifferentialGain = 0
EncoderPhaseCompensation = 0    ; Deg
```

CAUTION:
The encoder type must be "**AnalogInterpolated**" in the stages.ini file ("EncoderType" parameter).

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerEncoderCalibrationParametersGet $SocketID $FullPositionerName SinusOffset CosinusOffset DifferentialGain PhaseCompensation

Input parameters

SocketID ..... integer ..... Socket identifier got from "TCP_ConnectToServer" function
FullPositionerName ..... string ..... Positioner name

Output parameters

SinusOffset ..... double ..... Encoder sinus signal offset (Volts)
CosinusOffset..... double ..... Encoder cosinus signal offset (Volts)
DifferentialGain ..... double ..... Encoder differential gain
PhaseCompensation ..... double ..... Encoder phase compensation (Deg)

Return

Error..... integer ..... TCL error code (0=success or 1=syntax error) or function error code

C / C++

Prototype

int **PositionerEncoderCalibrationParametersGet** (int SocketID, char FullPositionerName[250] , double * SinusOffset, double * CosinusOffset, double * DifferentialGain, double * PhaseCompensation)

Input parameters

SocketID .....int ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....char * ..... Positioner name

Output parameters

SinusOffset .....double * ..... Encoder sinus signal offset (Volts)
CosinusOffset.....double * ..... Encoder cosinus signal offset (Volts)
DifferentialGain .....double * ..... Encoder differential gain
PhaseCompensation .....double * ..... Encoder phase compensation (Deg)

Return

Error.....int ..... Function error code

VISUAL BASIC

Prototype

Long **PositionerEncoderCalibrationParametersGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, SinusOffset As Double, CosinusOffset As Double, DifferentialGain As Double, PhaseCompensation As Double)

Input parameters

SocketID .....Long..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....String..... Positioner name

Output parameters

SinusOffset .....Double..... Encoder sinus signal offset (Volts)
CosinusOffset.....Double..... Encoder cosinus signal offset (Volts)
DifferentialGain .....Double..... Encoder differential gain
PhaseCompensation .....Double..... Encoder phase compensation (Deg)

Return

Error.....Long..... Function error code

MATLAB

Prototype

[Error, SinusOffset, CosinusOffset, DifferentialGain, PhaseCompensation]
PositionerEncoderCalibrationParametersGet (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID .....int32..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....cstring ..... Positioner name

Return

Error.....int32..... Function error code
SinusOffset .....double ..... Encoder sinus signal offset (Volts)
CosinusOffset.....double ..... Encoder cosinus signal offset (Volts)
DifferentialGain .....double ..... Encoder differential gain
PhaseCompensation .....double ..... Encoder phase compensation (Deg)

PYTHON

Prototype

[Error, SinusOffset, CosinusOffset, DifferentialGain, PhaseCompensation]
PositionerEncoderCalibrationParametersGet (integer SocketID, string FullPositionerName)

Input parameters

SocketID .....integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....string ..... Positioner name

Return

Error.....integer ..... Function error code
SinusOffset .....double ..... Encoder sinus signal offset (Volts)
CosinusOffset.....double ..... Encoder cosinus signal offset (Volts)
DifferentialGain.....double ..... Encoder differential gain
PhaseCompensation.....double ..... Encoder phase compensation (Deg)

2.2.4.31. PositionerErrorGet

NAME

PositionerErrorGet – Returns the positioner error code and clears it.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid output parameter type: ERR_WRONG_TYPE_INT (-15)
- Valid positioner name: ERR_POSITIONER _NAME (-18)
- Valid secondary positioner: ERR_UNCOMPATIBLE (-18)

DESCRIPTION

Returns the positioner error code and clears it.
The positioner error codes are listed in the “Positioner error list” § 2.16. The description of the positioner error code can be get with the “GroupPositionerErrorStringGet” function.

NOTE:
The “PositionerErrorRead” function allows read the positioner error without clear it.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_POSITIONER_NAME (-18)
ERR_UNCOMPATIBLE (-24)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype
PositionerErrorGet $SocketID $PositionerName PositionerError

Input parameters
SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
PositionerName..... string ..... Positioner name (maximum size = 250)

Output parameters
PositionerError..... interger..... Positioner error code.

Return
TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype
int **PositionerErrorGet** (int SocketID, char * PositionerName, int * PositionerError)

Input parameters
SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
PositionerName..... char * ..... Positioner name

Output parameters
PositionerError..... int * ..... Positioner error code

Return
Function error code

VISUAL BASIC

Prototype

Long **PositionerErrorGet** (ByVal SocketID As Long, ByVal PositionerName As String, PositionerError As Long)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
PositionerName ..... String ..... Positioner name

Output parameters

PositionerError ..... Long ..... Positioner error code

Return

Function error code

MATLAB

Prototype

[Error, PositionerError] **PositionerErrorGet** (int32 SocketID, cstring PositionerName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
PositionerName ..... cstring ..... Positioner name

Return

Error ..... int32 ..... Function error code
PositionerError ..... int32 ..... Positioner error code

PYTHON

Prototype

[Error, PositionerError] **PositionerErrorGet** (integer SocketID, string PositionerName)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
PositionerName ..... string ..... Positioner name

Return

Error ..... integer ..... Function error code
PositionerError ..... integer ..... Positioner error code

2.2.4.32. PositionerErrorRead

NAME

PositionerErrorRead – Returns the positioner error code without clears it.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid output parameter type: ERR_WRONG_TYPE_INT (-15)
- Valid positioner name: ERR_POSITIONER _NAME (-18)
- Valid secondary positioner: ERR_UNCOMPATIBLE (-18)

DESCRIPTION

Returns the positioner error code without clears it.

The positioner error codes are listed in the “Positioner error list” § 2.16. The description of the positioner error code can be get with the “GroupPositionerErrorStringGet” function.

NOTE:
The “PositionerErrorGet” function allows clear the positioner error.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_POSITIONER_NAME (-18)
ERR_UNCOMPATIBLE (-24)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

PositionerErrorRead $SocketID $PositionerName PositionerError

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
PositionerName..... string ..... Positioner name (maximum size = 250)

Output parameters

PositionerError..... interger Positioner error code.

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **PositionerErrorRead** (int SocketID, char * PositionerName, int * PositionerError)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
PositionerName..... char * ..... Positioner name

Output parameters

PositionerError..... int * ..... Positioner error code

Return

Function error code

VISUAL BASIC

Prototype

Long **PositionerErrorRead** (ByVal SocketID As Long, ByVal PositionerName As String, PositionerError As Long)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
PositionerName ..... String ..... Positioner name

Output parameters

PositionerError ..... Long ..... Positioner error code

Return

Function error code

MATLAB

Prototype

[Error, PositionerError] **PositionerErrorRead** (int32 SocketID, cstring PositionerName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
PositionerName ..... cstring ..... Positioner name

Return

Error ..... int32 ..... Function error code
PositionerError ..... int32 ..... Positioner error code

PYTHON

Prototype

[Error, PositionerError] **PositionerErrorRead** (integer SocketID, string PositionerName)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
PositionerName ..... string ..... Positioner name

Return

Error ..... integer ..... Function error code
PositionerError ..... integer ..... Positioner error code

2.2.4.33. PositionerErrorStringGet

NAME

PositionerErrorStringGet – Gets the positioner error description.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check input parameter type: ERR_WRONG_TYPE_INT (-15)
- Check output parameter type: ERR_WRONG_TYPE_CHAR (-13)

DESCRIPTION

This function returns a positioner error description from a positioner error code.

NOTE:
See the positioner error list describes in § 2.16

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

PositionerErrorStringGet $SocketID $FullPositionerName $PositionerErrorCode PositionerErrorString

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name
PositionerErrorCode	integer	Positioner error code

Output parameters

PositionerErrorString	integer	Positioner error description

Return

Error	integer	TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerErrorStringGet** (int SocketID, char FullPositionerName[250] , int PositionerErrorCode, char * PositionerErrorString)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	char *	Positioner name
PositionerErrorCode	int *	Positioner error code

Output parameters

PositionerErrorString	int *	Positioner error description

Return

Error	int	Function error code

VISUAL BASIC

Prototype

Long **PositionerErrorStringGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal PositionerErrorCode As Integer, ByVal PositionerErrorString As String)

Input parameters

SocketID ..... Long .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String .....Positioner name
PositionerErrorCode ..... Integer.....Positioner error code

Output parameters

PositionerErrorString ..... Integer.....Positioner error description

Return

Error..... Long .....Function error code

MATLAB

Prototype

[Error, PositionerErrorString] **PositionerErrorStringGet** (int32 SocketID, cstring FullPositionerName, int32 PositionerErrorCode)

Input parameters

SocketID ..... int32.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring.....Positioner name
PositionerErrorCode ..... int32.....Positioner error code

Return

Error..... int32.....Function error code
PositionerErrorString ..... cstring.....Positioner error description

PYTHON

Prototype

[Error, PositionerErrorString] **PositionerErrorStringGet** (integer SocketID, string FullPositionerName, integer PositionerErrorCode)

Input parameters

SocketID ..... integer.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string.....Positioner name
PositionerErrorCode ..... integer.....Positioner error code

Return

Error..... integer.....Function error code
PositionerErrorString ..... string.....Positioner error description

2.2.4.34. PositionerExcitationSignalGet

NAME

PositionerExcitationSignalGet – Returns the currently configured parameters of the excitation signal functionality.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid object type: ERR_WRONG_OBJECT_TYPE (-8)
- Valid positioner name: ERR_GROUP_NAME (-19)
- Valid output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)
- Valid output parameter type: ERR_WRONG_TYPE_INT (-15)

DESCRIPTION

This function gets the previously configured excitation signal parameters.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

PositionerExcitationSignalGet SocketID $PositionerName SignalType Frequency Amplitude Time

Input parameters

SocketID ..... integer ..... Socket identifier gets by the “TCP_ConnectToServer” function
PositionerName..... string ..... Positioner name (maximum size = 250)

Output parameters

SignalType..... integer ..... Type of signal
Frequency ..... floating point..... Frequency (Hz)
Amplitude ..... floating point..... Amplitude (acceleration, velocity or voltage unit)
Time..... floating point..... During time (seconds)

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **PositionerExcitationSignalGet** (int SocketID, char *PositionerName, int* SignalType, double* Frequency, double* Amplitude, double* Time)

Input parameters

SocketID ..... int ..... Socket identifier gets by the “TCP_ConnectToServer”function
PositionerName..... char * ..... Positioner name

Output parameters

SignalType	int *	Type of signal
Frequency	double *	Frequency (Hz)
Amplitude	double *	Amplitude (acceleration, velocity or voltage unit)
Time	double *	During time (seconds)

Return

Function error code

VISUAL BASIC

Prototype

Long **PositionerExcitationSignalGet** (ByVal SocketID As Long, ByVal GroupName As String, SignalType As Long, Frequency As Double, Amplitude As Double, Time As Double)

Input parameters

SocketID	Long	Socket identifier gets by the “TCP_ConnectToServer” function
PositionerName	String	Positioner name

Output parameters

SignalType	Long	Type of signal
Frequency	Double	Frequency (Hz)
Amplitude	Double	Amplitude (acceleration, velocity or voltage unit)
Time	Double	During time (seconds)

Return

Function error code

MATLAB

Prototype

[Error, SignalType, Frequency, Amplitude, Time] **PositionerExcitationSignalGet** (int32 SocketID, cstring PositionerName)

Input parameters

SocketID	int32	Socket identifier gets by the “TCP_ConnectToServer”function
PositionerName	cstring	Positioner name

Return

Error	int32	Function error code
SignalType	int32Ptr	Type of signal
Frequency	doublePtr	Frequency (Hz)
Amplitude	doublePtr	Amplitude (acceleration, velocity or voltage unit)
Time	doublePtr	During time (seconds)

PYTHON

Prototype

[Error, SignalType, Frequency, Amplitude, Time] **PositionerExcitationSignalGet** (integer SocketID, string PositionerName)

Input parameters

SocketID	integer	Socket identifier gets by the “TCP_ConnectToServer”function
PositionerName	string	Positioner name

Return

Error	integer	Function error code
SignalType	integerPtr	Type of signal
Frequency	doublePtr	Frequency (Hz)
Amplitude	doublePtr	Amplitude (acceleration, velocity or voltage unit)
Time	doublePtr	During time (seconds)

2.2.4.35. PositionerExcitationSignalSet

NAME

PositionerExcitationSignalSet – Configure and activate the signal of excitation.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid object type: ERR_WRONG_OBJECT_TYPE (-8)
- Secondary positioner or has a secondary positioner: ERR_WRONG_OBJECT_TYPE (-8)
- Valid control loop type: ERR_UNCOMPATIBLE (-24)
- Valid positioner name: ERR_GROUP_NAME (-19)
- Valid output parameter type: ERR_WRONG_TYPE_INT (-15)
- Valid output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)
- Check command status (must be not in progress): ERR_NOT_ALLOWED_ACTION (-22)
- Check type of signal (0, 1, 2 or 3): ERR_PARAMETER_OUT_OF_RANGE (-17)
- Check the time window (>0): ERR_PARAMETER_OUT_OF_RANGE (-17)
- Check the amplitude [-Acceleration (Velocity or Voltage) limit to Acceleration (Velocity or Voltage) limit]: ERR_PARAMETER_OUT_OF_RANGE (-17)

DESCRIPTION

The excitation-signal functionality generates a typical signal (a sine, a blank noise or an echelon signal) that the controller sends to motors to excite the system. In measuring the output signal of the excited system, we can determine some system characteristics, like the system transfert function.

The PID excitation-signal functionality is only available with the stages controlled in acceleration (acceleration control, ex: brushless / linear motors), velocity (velocity control) or in voltage (voltage control). It does not exist with the stages controlled in position (ex: stepper motors).

The excitation-signal function **PositionerExcitationSignalSet** can be executed only when the positioner is in the "READY" state. When the excitation-signal process is in progression, the positioner is in the "ExcitationSignal" state. At the end of the process, the positioner returns to the "READY" state (see group state diagram).

The **PositionerExcitationSignalSet** function sends an excitation command to the motor during a time. This function is allowed for "PIDFFAcceleration", "PIDFFVelocity" or "PIDDualFFVoltage" control loop. The parameters to configure are *signal type* (0:sine, 1:echelon,2:random-amplitude,3:random-pulse-width binary-amplitude, integer), *frequency* (Hz, double), *amplitude* (acceleration, velocity or voltage unit, double) and *during time* (seconds, double).

The function effective parameters for each mode are : (here : Limit means AccelerationLimit, VelocityLimit or VoltageLimit)

- Sine signal mode : *Frequency* (>=1 and <= 5000), *Amplitude* (>0 and <= Limit), *Time*(>0)
- Echelon signal mode : *Amplitude* (>0 and <= Limit, or <0 and >= -Limit), *Time* (>0).
  + During *Time* : Signal = *Amplitude*
  + End of *Time* : Signal = 0
- Random-amplitude signal mode : *Amplitude* (>0 and <= Limit), *Time*(>0), *Frequency* (>= 1 and <= 5000).
  Signal is generated with a random value at every controller base time (Tbase = 0.1 ms), then is filtered with a second order low-pass filter at the cut-off *Frequency* value.
- Random-pulse-width binary-amplitude signal mode :
  Amplitude (>0 and <= Limit), *Time* (>0), *Frequency* (>= 1 and <= 5000).
  Signal is a sequence of pulses (Signal = *Amplitude* or = 0) with pulse randomly varied width (multiple of Tbase).
  Frequency is the controlled system band-width (*cut-off frequency*), necessary for the PRBS (*Pseudo Random Binary Sequence*) function configuration.

The function non-effective parameters can accept any value, the value 0 is recommended for simplicity.

NOTE:
If during the excitation signal generation the stage position exceeds the user minimum or maximum target positions, the motor excitation command is stopped and an error is returned.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_IN_INITIALIZATION (-21)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_UNCOMPATIBLE (-24)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
ERR_WRONG_TYPE_INT (-15)
ERR_EXCITATION_SIGNAL_INITIALIZATION (-112)
SUCCESS (0) : no error

TCL

Prototype

PositionerExcitationSignalSet SocketID $PositionerName $SignalType $Frequency $Amplitude $Time

Input parameters

SocketID	integer	Socket identifier gets by the “TCP_ConnectToServer” function
PositionerName	string	Positioner name (maximum size = 250)
SignalType	integer	Type of signal
Frequency	floating point	Frequency (Hz)
Amplitude	floating point	Amplitude (acceleration, velocity or voltage unit)
Time	floating point	During time (seconds)

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **PositionerExcitationSignalSet** (int SocketID, char *PositionerName, int SignalType, double Frequency, double Amplitude, double Time)

Input parameters

SocketID	int	Socket identifier gets by the “TCP_ConnectToServer”function
PositionerName	char *	Positioner name
SignalType	int	Type of signal
Frequency	double	Frequency (Hz)
Amplitude	double	Amplitude (acceleration, velocity or voltage unit)
Time	double	During time (seconds)

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **PositionerExcitationSignalSet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal SignalType As Long, ByVal Frequency As Double, ByVal Amplitude As Double, ByVal Time As Double)

Input parameters

SocketID	Long	Socket identifier gets by the “TCP_ConnectToServer” function
PositionerName	String	Positioner name
SignalType	Long	Type of signal
Frequency	Double	Frequency (Hz)
Amplitude	Double	Amplitude (acceleration, velocity or voltage unit)
Time	Double	During time (seconds)

Output parameters

None

Return

Function error code

MATLAB

Prototype

[Error] **PositionerExcitationSignalSet** (int32 SocketID, cstring PositionerName, SignalType, Frequency, Amplitude, Time)

Input parameters

SocketID	int32	Socket identifier gets by the “TCP_ConnectToServer”function
PositionerName	cstring	Positioner name
SignalType	int32	Type of signal
Frequency	double	Frequency (Hz)
Amplitude	double	Amplitude (acceleration, velocity or voltage unit)
Time	double	During time (seconds)

Return

Error	int32	Function error code

PYTHON

Prototype

[Error] **PositionerExcitationSignalSet** (integer SocketID, string PositionerName, SignalType, Frequency, Amplitude, Time)

Input parameters

SocketID	integer	Socket identifier gets by the “TCP_ConnectToServer”function
PositionerName	string	Positioner name
SignalType	integer	Type of signal
Frequency	double	Frequency (Hz)
Amplitude	double	Amplitude (acceleration, velocity or voltage unit)
Time	double	During time (seconds)

Return

Error	integer	Function error code

2.2.4.36. PositionerHardInterpolatorFactorGet

NAME

PositionerHardInterpolatorFactorGet – Gets the interpolation factor from position compare mode.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner (must be not a secondary positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Check the encoder type (must be "AnalogInterpolated"): ERR_WRONG_OBJECT_TYPE (-8)
- Check output parameter type: ERR_WRONG_TYPE_INT (-15)

DESCRIPTION

This function returns the interpolation factor of the hardware interpolator used in the "Position Compare" mode. The interpolation factor value is defined as like:

InterpolationFactor = round (EncoderScalePitch / HardInterpolatorResolution)

NOTE:
The encoder type must be "**AnalogInterpolated**" in the stages.ini file ("EncoderType" parameter).

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

PositionerHardInterpolatorFactorGet $SocketID $FullPositionerName InterpolationFactor

Input parameters

SocketID ..... integer ..... Socket identifier got from "TCP_ConnectToServer" function
FullPositionerName ..... string ..... Positioner name

Output parameters

InterpolationFactor..... interger ..... Interpolation factor

Return

Error..... integer ..... TCL error code (0=success or 1=syntax error) or function error code

C / C++

Prototype

int **PositionerHardInterpolatorFactorGet** (int SocketID, char FullPositionerName[250] , int * InterpolationFactor)

Input parameters

SocketID ..... int ..... Socket identifier got from "TCP_ConnectToServer" function
FullPositionerName ..... char * ..... Positioner name

Output parameters

InterpolationFactor..... int * ..... Interpolation factor

Return

Error..... int ..... Function error code

VISUAL BASIC

Prototype

Long **PositionerHardInterpolatorFactorGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, InterpolationFactor As Integer)

Input parameters

SocketID .....Long..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....String..... Positioner name

Output parameters

InterpolationFactor.....Integer ..... Interpolation factor

Return

Error.....Long..... Function error code

MATLAB

Prototype

[Error, SinusOffset, CosinusOffset, DifferentialGain, PhaseCompensation]
PositionerHardInterpolatorFactorGet (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID .....int32 ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....cstring ..... Positioner name

Return

Error.....int32 ..... Function error code
InterpolationFactor.....int32 ..... Interpolation factor

PYTHON

Prototype

[Error, SinusOffset, CosinusOffset, DifferentialGain, PhaseCompensation]
PositionerHardInterpolatorFactorGet (integer SocketID, string FullPositionerName)

Input parameters

SocketID .....integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....string ..... Positioner name

Return

Error.....integer ..... Function error code
InterpolationFactor.....integer ..... Interpolation factor

2.2.4.37. PositionerHardInterpolatorFactorSet

NAME

PositionerHardInterpolatorFactorGet – Sets the interpolation factor from position compare mode.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type (must be not a secondary positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Check the positioner name: ERR_POSITIONER_NAME (-18)
- Check the group state (must be NOT INITIALIZED): ERR_NOT_ALLOWED_ACTION (-22)
- Check the encoder type (must be “AnalogInterpolated”): ERR_WRONG_OBJECT_TYPE (-8)
- Check input parameter type: ERR_WRONG_TYPE_INT (-15)
- Check input parameter value:ERR_PARAMETER_OUT_OF_RANGE (-17)

DESCRIPTION

This function allows to set the interpolation factor of the hardware interpolator used in the “PositionCompare” mode. The IP200 is updated and the position compare resolution is setting as like:

PositionCompareResolution = EncoderScalePitch / InterpolationFactor

The “InterpolationFactor“ value must be define with one of these values:

20	25	40	50	80	100	160	200

If the input interpolator factor value is different of these values then the ERR_PARAMETER_OUT_OF_RANGE error is returned.

NOTE:
The group must be NOT INITIALIZED to use this function else the ERR_NOT_ALLOWED_ACTION error is returned.
The encoder type must be “**AnalogInterpolated**” in the stages.ini file (“EncoderType” parameter) else the error is returned

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

PositionerHardInterpolatorFactorGet $SocketID $FullPositionerName InterpolationFactor

Input parameters

SocketID .....integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....string ..... Positioner name

Output parameters

InterpolationFactor.....interger..... Interpolation factor

Return

Error.....integer ..... TCL error code (0=success or 1=syntax error) or function error code

C / C++

Prototype

int **PositionerHardInterpolatorFactorGet** (int SocketID, char FullPositionerName[250] , int * InterpolationFactor)

Input parameters

SocketID .....int ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....char * ..... Positioner name

Output parameters

InterpolationFactor.....int * ..... Interpolation factor

Return

Error.....int ..... Function error code

VISUAL BASIC

Prototype

Long **PositionerHardInterpolatorFactorGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, InterpolationFactor As Integer)

Input parameters

SocketID .....Long..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....String..... Positioner name

Output parameters

InterpolationFactor.....Integer ..... Interpolation factor

Return

Error.....Long..... Function error code

MATLAB

Prototype

[Error, InterpolationFactor] **PositionerHardInterpolatorFactorGet** (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID .....int32 ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....cstring ..... Positioner name

Return

Error.....int32 ..... Function error code
InterpolationFactor.....int32 ..... Interpolation factor

PYTHON

Prototype

[Error, InterpolationFactor] **PositionerHardInterpolatorFactorGet** (integer SocketID, string FullPositionerName)

Input parameters

SocketID .....integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....string ..... Positioner name

Return

Error.....integer ..... Function error code
InterpolationFactor.....integer ..... Interpolation factor

2.2.4.38. PositionerHardwareStatusGet

NAME

PositionerHardwareStatusGet – Gets the positioner hardware status code.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner (must be not a secondary positioner): ERR_WRONG_OBJECT_TYPE (-8), ERR_UNCOMPATIBLE (-24), ERR_POSITIONER_NAME (-18)
- Check output parameter type: ERR_WRONG_TYPE_INT (-15)

DESCRIPTION

This function allows to return the hardware status of the selected positioner. The positioner hardware status is composed of the “corrector” hardware status and the “servitudes” hardware status:

- The “Corrector” returns the motor interface and the position encoder hardware status.
- The “Servitudes” returns the general inhibit and the end of runs hardware status.

NOTE:
See the positioner hardware status list describes in § 2.17

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

PositionerHardwareStatusGet $SocketID $FullPositionerName PositionerHardwareStatus

Input parameters

SocketID .....integer .... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....string ..... Positioner name

Output parameters

PositionerHardwareStatus.....integer .... Hardware status code

Return

Error.....integer .... TCL error code (0 = success or 1 = syntax error) or function error

C / C++

Prototype

int **PositionerHardwareStatusGet** (int SocketID, char FullPositionerName[250] , int * PositionerHardwareStatus)

Input parameters

SocketID ..... int .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... char * .....Positioner name

Output parameters

PositionerHardwareStatus..... int *.....Hardware status code

Return

Error..... int.....Function error code

VISUAL BASIC

Prototype

Long **PositionerHardwareStatusGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, PositionerHardwareStatus As Integer)

Input parameters

SocketID ..... Long .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String .....Positioner name

Output parameters

PositionerHardwareStatus..... Integer.....Hardware status code

Return

Error..... Long .....Function error code

MATLAB

Prototype

[Error, PositionerHardwareStatus] **PositionerHardwareStatusGet** (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID ..... int32.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring.....Positioner name

Return

Error..... int32.....Function error code
PositionerHardwareStatus..... int32.....Hardware status code

PYTHON

Prototype

[Error, PositionerHardwareStatus] **PositionerHardwareStatusGet** (integer SocketID, string FullPositionerName)

Input parameters

SocketID ..... integer.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string.....Positioner name

Return

Error..... integer.....Function error code
PositionerHardwareStatus..... integer.....Hardware status code

2.2.4.39. PositionerHardwareStatusStringGet

NAME

PositionerHarwareStatusStringGet – Gets the positioner error description.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check input parameter type: ERR_WRONG_TYPE_INT (-15)
- Check output parameter type: ERR_WRONG_TYPE_CHAR (-13)

DESCRIPTION

This function returns the hardware status description from a positioner hardware status code.

NOTE:
See the positioner hardware status list describes in § 2.17

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

PositionerHarwareStatusStringGet $SocketID $FullPositionerName $PositionerHardwareStatusCode PositionerHardwareStatusString

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name
PositionerHardwareStatusCode	integer	Positioner hardware status code

Output parameters

PositionerHardwareStatusString	integer	Positioner hardware status description

Return

Error	integer	TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerHarwareStatusStringGet** (int SocketID, char FullPositionerName[250] , int PositionerHardwareStatusCode, char * PositionerHardwareStatusString)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	char *	Positioner name
PositionerHardwareStatusCode	int *	Positioner hardware status code

Output parameters

PositionerHardwareStatusString	int *	Positioner hardware status description

Return

Error	int	Function error code

VISUAL BASIC

Prototype

Long **PositionerHarwareStatusStringGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal PositionerHardwareStatusCode As Integer, ByVal PositionerHardwareStatusString As String)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	String	Positioner name
PositionerHardwareStatusCode	Integer	Positioner hardware status code

Output parameters

PositionerHardwareStatusString	Integer	Positioner hardware status description

Return

Error	Long	Function error code

MATLAB

Prototype

[Error, PositionerHardwareStatusString] **PositionerHarwareStatusStringGet** (int32 SocketID, cstring FullPositionerName, int32 PositionerHardwareStatusCode)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	cstring	Positioner name
PositionerHardwareStatusCode	int32	Positioner hardware status code

Return

Error	int32	Function error code
PositionerHardwareStatusString	cstring	Positioner hardware status description

PYTHON

Prototype

[Error, PositionerHardwareStatusString] **PositionerHarwareStatusStringGet** (integer SocketID, string FullPositionerName, integer PositionerHardwareStatusCode)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name
PositionerHardwareStatusCode	integer	Positioner hardware status code

Return

Error	integer	Function error code
PositionerHardwareStatusString	string	Positioner hardware status description

2.2.4.40. PositionerMaximumVelocityAndAccelerationGet

NAME

PositionerMaximumVelocityAndAccelerationGet – Gets the maximum of the velocity and the acceleration from profiler generators.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner (must not be a secondary positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Check output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function returns the maximum velocity and maximum acceleration of the profile generators. These parameters represents the limits in the profiler and are defined in the stages.ini file:

```
MaximumVelocity =          ; unit / second
MaximumAcceleration =      ; unit / second2
```

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerMaximumVelocityAndAccelerationGet $SocketID $FullPositionerName MaximumVelocity MaximumAcceleration

Input parameters

SocketID .....integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....string ..... Positioner name

Output parameters

MaximumVelocity .....double..... Maximum velocity (units / seconds)
MaximumAcceleration .....double..... Maximum acceleration (units / seconds²)

Return

Error.....integer ..... TCL error code (0=success or 1=syntax error) or function error code

C / C++

Prototype

int **PositionerMaximumVelocityAndAccelerationGet** (int SocketID, char FullPositionerName[250] , double * MaximumVelocity, double * MaximumAcceleration)

Input parameters

SocketID .....int ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....char *..... Positioner name

Output parameters

MaximumVelocity .....double *..... Maximum velocity (units / seconds)
MaximumAcceleration .....double *..... Maximum acceleration (units / seconds²)

Return

Error.....int ..... Function error code

VISUAL BASIC

Prototype

Long **PositionerMaximumVelocityAndAccelerationGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, MaximumVelocity As Double, MaximumAcceleration As Double)

Input parameters

SocketID ..... Long .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String .....Positioner name

Output parameters

MaximumVelocity ..... Double .....Maximum velocity (units / seconds)
MaximumAcceleration ..... Double .....Maximum acceleration (units / seconds²)

Return

Error..... Long .....Function error code

MATLAB

Prototype

[Error, MaximumVelocity, MaximumAcceleration] **PositionerMaximumVelocityAndAccelerationGet** (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID ..... int32.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring.....Positioner name

Return

Error..... int32.....Function error code
MaximumVelocity ..... double.....Maximum velocity (units / seconds)
MaximumAcceleration ..... double.....Maximum acceleration (units / seconds²)

PYTHON

Prototype

[Error, MaximumVelocity, MaximumAcceleration] **PositionerMaximumVelocityAndAccelerationGet** (integer SocketID, string FullPositionerName)

Input parameters

SocketID ..... integer.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string.....Positioner name

Return

Error..... integer.....Function error code
MaximumVelocity ..... double.....Maximum velocity (units / seconds)
MaximumAcceleration ..... double.....Maximum acceleration (units / seconds²)

2.2.4.41. PositionerMotionDoneGet

NAME

PositionerMotionDoneGet – Gets the motion done parameters.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner (must not be a secondary positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Check the motion done mode (must be “VelocityAndPositionWindow”): ERR_WRONG_OBJECT_TYPE (-8)
- Check output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function returns the motion done parameters only for the “VelocityAndPositionWindow” MotionDone mode. If the MotionDone mode is defined as “Theoretical” then the ERR_WRONG_OBJECT_TYPE (-8) error is returned.
The “MotionDoneMode” parameter from the stages.ini file allows to define a motion done mode. The motion done can be defined “Theoretical” (the motion done mode is not used) or “VelocityAndPositionWindow”. For a more thorough description of the motion done mode, please refer to the XPS Motion Tutorial, section named Motion / Motion Done.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerMotionDoneGet $SocketID $FullPositionerName PositionWindow VelocityWindow CheckingTime MeanPeriod Timeout

Input parameters

SocketID .....integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....string ..... Positioner name

Output parameters

PositionWindow.....double..... Position window (units)
VelocityWindow.....double..... Velocity window (units / seconds)
CheckingTime.....double..... Checking time (seconds)
MeanPeriod.....double..... Mean period (seconds)
Timeout.....double..... Motion done time out (seconds)

Return

Error.....integer ..... TCL error code (0=success or 1=syntax error) or function error code

C / C++

Prototype

int **PositionerMotionDoneGet** (int SocketID, char FullPositionerName[250] , double * PositionWindow, double * VelocityWindow, double * CheckingTime, double * MeanPeriod, double * Timeout)

Input parameters

SocketID ..... int .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... char * .....Positioner name

Output parameters

PositionWindow..... double * .....Position window (units)
VelocityWindow..... double * .....Velocity window (units / seconds)
CheckingTime..... double * .....Checking time (seconds)
MeanPeriod..... double * .....Mean period (seconds)
Timeout..... double * .....Motion done time out (seconds)

Return

Error..... int.....Function error code

VISUAL BASIC

Prototype

Long **PositionerMotionDoneGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, PositionWindow As Double, VelocityWindow As Double, CheckingTime As Double, MeanPeriod As Double, Timeout As Double)

Input parameters

SocketID ..... Long .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String .....Positioner name

Output parameters

PositionWindow..... Double .....Position window (units)
VelocityWindow..... Double .....Velocity window (units / seconds)
CheckingTime..... Double .....Checking time (seconds)
MeanPeriod..... Double .....Mean period (seconds)
Timeout..... Double .....Motion done time out (seconds)

Return

Error..... Long .....Function error code

MATLAB

Prototype

[Error, PositionWindow, VelocityWindow, CheckingTime, MeanPeriod, Timeout] **PositionerMotionDoneGet** (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID ..... int32.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring.....Positioner name

Return

Error..... int32.....Function error code
PositionWindow..... double.....Position window (units)
VelocityWindow..... double.....Velocity window (units / seconds)
CheckingTime..... double.....Checking time (seconds)
MeanPeriod..... double.....Mean period (seconds)
Timeout..... double.....Motion done time out (seconds)

PYTHON

Prototype

[Error, PositionWindow, VelocityWindow, CheckingTime, MeanPeriod, Timeout] **PositionerMotionDoneGet** (integer SocketID, string FullPositionerName)

Input parameters

SocketID ..... integer.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string.....Positioner name

Return

Error..... integer.....Function error code
PositionWindow..... double.....Position window (units)
VelocityWindow..... double.....Velocity window (units / seconds)
CheckingTime..... double.....Checking time (seconds)
MeanPeriod..... double.....Mean period (seconds)
Timeout..... double.....Motion done time out (seconds)

2.2.4.42. PositionerMotionDoneSet

NAME

PositionerMotionDoneSet – Sets the motion done parameters.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type: ERR_WRONG_OBJECT_TYPE (-8)
- Check output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function allows to update the motion done parameters only for the “VelocityAndPositionWindow” MotionDone mode. The “MotionDoneMode” parameter from the stages.ini file must be defined as “VelocityAndPositionWindow” else the ERR_WRONG_OBJECT_TYPE (-8) error is returned.

For a more thorough description of the Motion Done mode, please refer to the XPS Motion Tutorial, section named Motion / Motion Done.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerMotionDoneSet $SocketID $FullPositionerName $PositionWindow $VelocityWindow $CheckingTime $MeanPeriod $Timeout

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name
PositionWindow	double	Position window (units)
VelocityWindow	double	Velocity window (units / seconds)
CheckingTime	double	Checking time (seconds)
MeanPeriod	double	Mean period (seconds)
Timeout	double	Motion done time out (seconds)

Output parameters (None)

Return

Error	integer	TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerMotionDoneSet** (int SocketID, char FullPositionerName[250] , double * PositionWindow, double * VelocityWindow, double * CheckingTime, double * MeanPeriod, double * Timeout)

Input parameters

SocketID ..... int .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... char * .....Positioner name
PositionWindow..... double .....Position window (units)
VelocityWindow..... double .....Velocity window (units / seconds)
CheckingTime..... double .....Checking time (seconds)
MeanPeriod..... double .....Mean period (seconds)
Timeout..... double .....Motion done time out (seconds)

Output parameters (None)

Return

Error..... int.....Function error code

VISUAL BASIC

Prototype

Long **PositionerMotionDoneSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal PositionWindow As Double, ByVal VelocityWindow As Double, ByVal CheckingTime As Double, ByVal MeanPeriod As Double, ByVal Timeout As Double)

Input parameters

SocketID ..... Long.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String .....Positioner name
PositionWindow..... Double .....Position window (units)
VelocityWindow..... Double .....Velocity window (units / seconds)
CheckingTime..... Double .....Checking time (seconds)
MeanPeriod..... Double .....Mean period (seconds)
Timeout..... Double .....Motion done time out (seconds)

Output parameters (None)

Return

Error..... Long.....Function error code

MATLAB

Prototype

[Error] **PositionerMotionDoneSet** (int32 SocketID, cstring FullPositionerName, double PositionWindow, double VelocityWindow, double CheckingTime, double MeanPeriod, double Timeout)

Input parameters

SocketID ..... int32.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring .....Positioner name
PositionWindow..... double .....Position window (units)
VelocityWindow..... double .....Velocity window (units / seconds)
CheckingTime..... double .....Checking time (seconds)
MeanPeriod..... double .....Mean period (seconds)
Timeout..... double .....Motion done time out (seconds)

Return

Error..... int32.....Function error code

PYTHON

Prototype

[Error] **PositionerMotionDoneSet** (integer SocketID, string FullPositionerName, double PositionWindow, double VelocityWindow, double CheckingTime, double MeanPeriod, double Timeout)

Input parameters

SocketID ..... integer .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string .....Positioner name
PositionWindow..... double .....Position window (units)
VelocityWindow..... double .....Velocity window (units / seconds)
CheckingTime..... double .....Checking time (seconds)
MeanPeriod..... double .....Mean period (seconds)
Timeout..... double .....Motion done time out (seconds)

Return

Error..... integer .....Function error code

2.2.4.43. PositionerPositionCompareDisable

NAME

PositionerPositionCompareDisable – Disables the position compare mode.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner (must not be a secondary positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Check the encoder (“AquadB” or “AnalogInterpolated”): ERR_WRONG_OBJECT_TYPE (-8)

DESCRIPTION

This function allows to disable the position compare mode.

For a more thorough description of the position compare output, please refer to the XPS Motion Tutorial, section named Triggers / Position Compare Output.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype

PositionerPositionCompareDisable $SocketID $FullPositionerName

Input parameters

SocketID ..... integer .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string .....Positioner name

Output parameters

None

Return

Error ..... integer .....TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerPositionCompareDisable** (int SocketID, char FullPositionerName[250])

Input parameters

SocketID ..... int .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... char * .....Positioner name

Output parameters

None

Return

Error ..... int .....Function error code

VISUAL BASIC

Prototype

Long **PositionerPositionCompareDisable** (ByVal SocketID As Long, ByVal FullPositionerName As String)

Input parameters

SocketID ..... Long.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String .....Positioner name

Output parameters

None

Return

Error..... Long.....Function error code

MATLAB

Prototype

[Error] **PositionerPositionCompareDisable** (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID ..... int32.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring .....Positioner name

Return

Error..... int32.....Function error code

PYTHON

Prototype

[Error] **PositionerPositionCompareDisable** (integer SocketID, string FullPositionerName)

Input parameters

SocketID ..... integer.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string.....Positioner name

Return

Error..... integer.....Function error code

2.2.4.44. PositionerPositionCompareEnable

NAME

PositionerPositionCompareEnable – Enables the position compare mode.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner (must not be a secondary positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Check the positioner name: ERR_POSITIONER_NAME (-18)
- Check the encoder (“AquadB” or “AnalogInterpolated”): ERR_WRONG_OBJECT_TYPE (-8)
- Check the group state (must be READY): ERR_NOT_ALLOWED_ACTION (-22)
- Check the position compare parameters (must be configured): ERR_NOT_ALLOWED_ACTION (-22)

DESCRIPTION

This function allows to enable the position compare mode. The group must be in READY state to use this function else the ERR_NOT_ALLOWED_ACTION (-22) error is returned.
If the position compare parameters are not configured (by the “PositionerPositionCompareSet” function) then the ERR_NOT_ALLOWED_ACTION (-22) error is returned.

For a more thorough description of the position compare output, please refer to the XPS Motion Tutorial, section named Triggers / Position Compare Output.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype

PositionerPositionCompareEnable $SocketID $FullPositionerName

Input parameters

SocketID ..... integer .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string .....Positioner name

Output parameters

None

Return

Error..... integer .....TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerPositionCompareEnable** (int SocketID, char FullPositionerName[250])

Input parameters

SocketID ..... int .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... char * .....Positioner name

Output parameters (None)

Return

Error..... int.....Function error code

VISUAL BASIC

Prototype

Long **PositionerPositionCompareEnable** (ByVal SocketID As Long, ByVal FullPositionerName As String)

Input parameters

SocketID ..... Long.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String .....Positioner name

Output parameters

None

Return

Error..... Long.....Function error code

MATLAB

Prototype

[Error] **PositionerPositionCompareEnable** (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID ..... int32.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring .....Positioner name

Return

Error..... int32.....Function error code

PYTHON

Prototype

[Error] **PositionerPositionCompareEnable** (integer SocketID, string FullPositionerName)

Input parameters

SocketID ..... integer.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string.....Positioner name

Return

Error..... integer.....Function error code

2.2.4.45. PositionerPositionCompareGet

NAME

PositionerPositionCompareGet – Gets the position compare parameters.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Check the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner (must not be a secondary positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Check the position compare parameters (must be configured): ERR_POSITION_COMPARE_NOT_SET (-23)
- Check the position encoder (“AquadB” or “AnalogInterpolated”): ERR_WRONG_OBJECT_TYPE (-8)
- Check output parameter type: ERR_WRONG_TYPE_DOUBLE (-14), ERR_WRONG_TYPE_BOOL (-12)

DESCRIPTION

This function returns the configured parameters of the position compare output trigger and gives its state (enabled or disabled).

For a more thorough description of the position compare output, please refer to the XPS Motion Tutorial, section named Triggers / Position Compare Output.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_POSITION_COMPARE_NOT_SET (-23)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_BOOL (-12)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerPositionCompareGet $SocketID $FullPositionerName MinimumPosition MaximumPosition PositionStep EnableState

Input parameters

SocketID .....integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....string ..... Positioner name

Output parameters

MinimumPosition .....double..... Minimum position (units)
MaximumPosition.....double..... Maximum position (units)
PositionStep .....double..... Position step (units)
EnableState .....bool ..... Position compare state (true=enabled or false=disabled)

Return

Error.....integer ..... TCL error code (0=success or 1=syntax error) or function error code

C / C++

Prototype

int **PositionerPositionCompareGet** (int SocketID, char FullPositionerName[250] , double* MinimumPosition, double* MaximumPosition, double* PositionStep, bool * EnableState)

Input parameters

SocketID ..... int .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... char * .....Positioner name

Output parameters

MinimumPosition ..... double * .....Minimum position (units)
MaximumPosition..... double * .....Maximum position (units)
PositionStep ..... double * .....Position step (units)
EnableState ..... bool * .....Position compare state (true=enabled or false=disabled)

Return

Error..... int.....Function error code

VISUAL BASIC

Prototype

Long **PositionerPositionCompareGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, MinimumPosition As Double, MaximumPosition As Double, PositionStep As Double, EnableState As Boolean)

Input parameters

SocketID ..... Long .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String .....Positioner name

Output parameters

MinimumPosition ..... Double .....Minimum position (units)
MaximumPosition..... Double .....Maximum position (units)
PositionStep ..... Double .....Position step (units)
EnableState ..... Boolean.....Position compare state (true=enabled or false=disabled)

Return

Error..... Long .....Function error code

MATLAB

Prototype

[Error, MinimumPosition, MaximumPosition, PositionStep, EnableState] **PositionerPositionCompareGet** (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID ..... int32.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring.....Positioner name

Return

Error..... int32.....Function error code
MinimumPosition ..... double.....Minimum position (units)
MaximumPosition..... double.....Maximum position (units)
PositionStep ..... double.....Position step (units)
EnableState ..... bool.....Position compare state (true=enabled or false=disabled)

PYTHON

Prototype

[Error, MinimumPosition, MaximumPosition, PositionStep, EnableState] **PositionerPositionCompareGet** (integer SocketID, string FullPositionerName)

Input parameters

SocketID ..... integer.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string.....Positioner name

Return

Error..... integer.....Function error code
MinimumPosition ..... double.....Minimum position (units)
MaximumPosition..... double.....Maximum position (units)
PositionStep ..... double.....Position step (units)
EnableState ..... bool.....Position compare state (true=enabled or false=disabled)

2.2.4.46. PositionerPositionCompareSet

NAME

PositionerPositionCompareSet – Sets the position compare parameters.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Check the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type: ERR_WRONG_OBJECT_TYPE (-8)
- Check the position encoder (“AquadB” or “AnalogInterpolated”): ERR_WRONG_OBJECT_TYPE (-8)
- Check output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)
- Check output parameter values: ERR_PARAMETER_OUT_OF_RANGE (-17)
  MinimumPosition < MaximumPosition
  MinimumPosition >= MinimumTargetPosition
  MaximumPosition <= MaximumTargetPosition
  0 <= PositionStep <= (MaximumPosition – MinimumPosition)
- Check position compare state (must be disabled): ERR_NOT_ALLOWED_ACTION (-22)

DESCRIPTION

This function allows to set the parameters for the position compare output trigger of the PCO connector on the XPS controller cards.
These parameters are used only when the position compare mode is enabled. For a more thorough description of the position compare output, please refer to the XPS Motion Tutorial, section named Triggers / Position Compare Output.

NOTE:
This function can be used only with a position encoder. If no position encoder then the ERR_UNCOMPATIBLE (-24) error is returned.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_IN_INITIALIZATION (-21)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_UNCOMPATIBLE (-24)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerPositionCompareSet $SocketID $FullPositionerName $MinimumPosition $MaximumPosition $PositionStep

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name
MinimumPosition	double	Minimum position (units)
MaximumPosition	double	Maximum position (units)
PositionStep	double	Position step (units)

Output parameters (None)

Return

Error	integer	TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerPositionCompareSet** (int SocketID, char FullPositionerName[250] , double MinimumPosition, double MinimumPosition, double PositionStep)

Input parameters

SocketID ..... int .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... char * .....Positioner name
MinimumPosition ..... double .....Minimum position (units)
MaximumPosition..... double .....Maximum position (units)
PositionStep ..... double .....Position step (units)

Output parameters (None)

Return

Error..... int.....Function error code

VISUAL BASIC

Prototype

Long **PositionerPositionCompareSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal MinimumPosition As Double, ByVal MaximumPosition As Double, ByVal PositionStep As Double)

Input parameters

SocketID ..... Long.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String .....Positioner name
MinimumPosition ..... Double .....Minimum position (units)
MaximumPosition..... Double .....Maximum position (units)
PositionStep ..... Double .....Position step (units)

Output parameters (None)

Return

Error..... Long.....Function error code

MATLAB

Prototype

[Error] **PositionerPositionCompareSet** (int32 SocketID, cstring FullPositionerName, double MinimumPosition, double MaximumPosition, double PositionStep)

Input parameters

SocketID ..... int32.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring .....Positioner name
MinimumPosition ..... double .....Minimum position (units)
MaximumPosition..... double .....Maximum position (units)
PositionStep ..... double .....Position step (units)

Return

Error..... int32.....Function error code

PYTHON

Prototype

[Error] **PositionerPositionCompareSet** (integer SocketID, string FullPositionerName, double MinimumPosition, double MaximumPosition, double PositionStep)

Input parameters

SocketID ..... integer.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string.....Positioner name
MinimumPosition ..... double .....Minimum position (units)
MaximumPosition..... double .....Maximum position (units)
PositionStep ..... double .....Position step (units)

Return

Error..... integer.....Function error code

2.2.4.47. PositionerPositionComparePulseParametersGet

NAME

PositionerPositionComparePulseParametersGet – Gets the position compare PCO pulse parameters.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Check the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)
- Check the positioner (must not be a secondary positioner): ERR_WRONG_OBJECT_TYPE (-8)

DESCRIPTION

This function returns the configured parameters of the position compare PCO pulse parameters.

For a more thorough description of the position compare output, please refer to the XPS Motion Tutorial, section named Triggers / Position Compare Output.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerPositionComparePulseParametersGet $SocketID $FullPositionerName PCOPulseWidth EncoderSettlingTime

Input parameters

SocketID .....integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....string ..... Positioner name

Output parameters

PCOPulseWidth.....double..... Width of PCO pulses (µs)
EncoderSettlingTime .....double..... Encoder signal settling time (µs)

Return

Error.....integer ..... TCL error code (0=success or 1=syntax error) or function error code

C / C++

Prototype

int **PositionerPositionComparePulseParametersGet** (int SocketID, char FullPositionerName[250] , double* PCOPulseWidth, double* EncoderSettlingTime)

Input parameters

SocketID ..... int .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... char * .....Positioner name

Output parameters

PCOPulseWidth..... double * .....Width of PCO pulses (µs)
EncoderSettlingTime ..... double * .....Encoder signal settling time (µs)

Return

Error..... int.....Function error code

VISUAL BASIC

Prototype

Long **PositionerPositionComparePulseParametersGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, PCOPulseWidth As Double, EncoderSettlingTime As Double)

Input parameters

SocketID ..... Long .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String .....Positioner name

Output parameters

PCOPulseWidth ..... Double .....Width of PCO pulses (µs)
EncoderSettlingTime ..... Double .....Encoder signal settling time (µs)

Return

Error ..... Long .....Function error code

MATLAB

Prototype

[Error, PCOPulseWidth, EncoderSettlingTime] **PositionerPositionComparePulseParametersGet** (int32 SocketID, cstring FullPositionerName)

Input parameter

SocketID ..... int32.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring .....Positioner name

Return

Error ..... int32.....Function error code
PCOPulseWidth ..... double .....Width of PCO pulses (µs)
EncoderSettlingTime ..... double .....Encoder signal settling time (µs)

PYTHON

Prototype

[Error, PCOPulseWidth, EncoderSettlingTime] **PositionerPositionComparePulseParametersGet** (integer SocketID, string FullPositionerName)

Input parameters

SocketID ..... integer.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string.....Positioner name

Return

Error ..... integer.....Function error code
PCOPulseWidth ..... double .....Width of PCO pulses (µs)
EncoderSettlingTime ..... double .....Encoder signal settling time (µs)

2.2.4.48. PositionerPositionComparePulseParametersSet

NAME

PositionerPositionComparePulseParametersSet – Sets the position compare PCO pulse parameters.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Check the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type (must not be a secondary positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Check the position encoder ("AquadB" or "AnalogInterpolated"): ERR_UNCOMPATIBLE (-24)
- Check input parameter values: ERR_PARAMETER_OUT_OF_RANGE (-17)

  PCOPulseWidth *value must equal to* ***0.2 (default), 1, 2.5 or 10*** *(µs)*
  EncoderSettlingTime *value must equal to* ***0.075 (default), 1, 4 or 12*** *(µs)*

- Check position compare state (must be disabled): ERR_NOT_ALLOWED_ACTION (-22)
- Check if the CIE board supports this function: ERR_HARDWARE_FUNCTION_NOT_SUPPORTED (-115)

DESCRIPTION

This function allows to set two additional parameters for the position compare output trigger of the PCO connector on the XPS controller cards. The first additional parameter is the pulse width. The second parameter is the encoder settling time value which is the time the encoder inputs have to be stable after a change of state to be detected. These parameters are used only when the position compare mode is enabled. For a more thorough description of the position compare output, please refer to the XPS Motion Tutorial, section named Triggers / Position Compare Output.

NOTE:

- This function can be used only with a position encoder. If no position encoder then the ERR_UNCOMPATIBLE (-24) error is returned.
- This function is called automatically at controller reboot and at GroupInitialize() execution to set the position compare pulse parameters to default values (PCOPulseWidth to 0.2 µs, EncoderSettlingTime to 0.075 µs)

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_IN_INITIALIZATION (-21)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_UNCOMPATIBLE (-24)
ERR_HARDWARE_FUNCTION_NOT_SUPPORTED (-115)
SUCCESS (0) : no error

TCL

Prototype

PositionerPositionComparePulseParametersSet $SocketID $FullPositionerName $PCOPulseWidth $EncoderSettlingTime

Input parameters

SocketID ..... integer .....Socket identifier got from "TCP_ConnectToServer" function
FullPositionerName ..... string .....Positioner name
PCOPulseWidth ..... double .....Width of PCO pulses (µs)
EncoderSettlingTime .... double .....Encoder signal settling time (µs)

Output parameters (None)

Return

Error ..... integer ..... TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerPositionComparePulseParametersSet** (int SocketID, char FullPositionerName[250] , double PCOPulseWidth, double EncoderSettlingTime)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... char * ..... Positioner name
PCOPulseWidth ..... double ..... Width of PCO pulses (µs)
EncoderSettlingTime .... double ..... Encoder signal settling time (µs)

Output parameters (None)

Return

Error ..... int ..... Function error code

VISUAL BASIC

Prototype

Long **PositionerPositionComparePulseParametersSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal PCOPulseWidth As Double, ByVal EncoderSettlingTime As Double)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String ..... Positioner name
PCOPulseWidth ..... Double ..... Width of PCO pulses (µs)
EncoderSettlingTime .... Double ..... Encoder signal settling time (µs)

Output parameters (None)

Return

Error ..... Long ..... Function error code

MATLAB

Prototype

[Error] **PositionerPositionComparePulseParametersSet** (int32 SocketID, cstring FullPositionerName, double PCOPulseWidth, EncoderSettlingTime)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring ..... Positioner name
PCOPulseWidth ..... double ..... Width of PCO pulses (µs)
EncoderSettlingTime .... double ..... Encoder signal settling time (µs)

Return

Error ..... int32 ..... Function error code

PYTHON

Prototype

[Error] **PositionerPositionComparePulseParametersSet** (integer SocketID, string FullPositionerName, double PCOPulseWidth, double EncoderSettlingTime)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string ..... Positioner name
PCOPulseWidth ..... double ..... Width of PCO pulses (µs)
EncoderSettlingTime .... double ..... Encoder signal settling time (µs)

Return

Error ..... integer ..... Function error code

2.2.4.49. PositionerPositionCompareAquadBAlwaysEnable

NAME

PositionerPositionCompareAquadBAlwaysEnable – Enable the AquadB signal in the always mode.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Check the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type: ERR_WRONG_OBJECT_TYPE (-8)
- Check the position encoder (“AquadB” or “AnalogInterpolated”): ERR_UNCOMPATIBLE (-24)
- Check if the CIE board supports this function: ERR_HARDWARE_FUNCTION_NOT_SUPPORTED (-115)

DESCRIPTION

This function enables the generation of AquadB output signal on the PCO connector (the 2&3 or 4&5 pins) of the XPS controller cards. The always mode means that the AquadB signal is generated all the time (not position windowed).

NOTE:
This function can be used only with a position encoder. If there is no position encoder then the ERR_UNCOMPATIBLE (-24) error is returned.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_UNCOMPATIBLE (-24)
ERR_HARDWARE_FUNCTION_NOT_SUPPORTED (-115)
SUCCESS (0) : no error

TCL

Prototype
PositionerPositionCompareAquadBAlwaysEnable $SocketID $FullPositionerName

Input parameters
SocketID ..... integer .....Socket identifier gets by the “TCP_ConnectToServer” function
FullPositionerName ..... string .....Positioner name

Output parameters (None)

Return
Error ..... integer .....TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype
int **PositionerPositionCompareAquadBAlwaysEnable** (int SocketID, char FullPositionerName[250])

Input parameters
SocketID ..... int .....Socket identifier gets by the “TCP_ConnectToServer” function
FullPositionerName ..... char * .....Positioner name

Output parameters (None)

Return
Error ..... int .....Function error code

VISUAL BASIC

Prototype

Long **PositionerPositionCompareAquadBAlwaysEnable** (ByVal SocketID As Long, ByVal FullPositionerName As String)

Input parameters

SocketID ..... Long.....Socket identifier gets by the “TCP_ConnectToServer” function
FullPositionerName ..... String .....Positioner name

Output parameters (None)

Return

Error..... Long.....Function error code

MATLAB

Prototype

[Error] **PositionerPositionCompareAquadBAlwaysEnable** (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID ..... int32.....Socket identifier gets by the “TCP_ConnectToServer” function
FullPositionerName ..... cstring .....Positioner name

Return

Error..... int32.....Function error code

PYTHON

Prototype

[Error] **PositionerPositionCompareAquadBAlwaysEnable** (integer SocketID, string FullPositionerName)

Input parameters

SocketID ..... integer.....Socket identifier gets by the “TCP_ConnectToServer” function
FullPositionerName ..... string.....Positioner name

Return

Error..... integer.....Function error code

2.2.4.50. PositionerPositionCompareAquadBWindowedGet

NAME

PositionerPositionCompareAquadBWindowedGet – Gets the windowed AquadB mode parameters and state..

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Check the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner (must not be a secondary positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Check the position compare parameters (must be configured): ERR_POSITION_COMPARE_NOT_SET (-23)
- Check the configured mode type (must be WindowedAquadB): ERR_UNCOMPATIBLE (-24)
- Check output parameter type: ERR_WRONG_TYPE_DOUBLE (-14), ERR_WRONG_TYPE_BOOL (-12)

DESCRIPTION

This function returns the configured parameters of the position windowed AquadB output signal and gives its state (enabled or disabled).

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_POSITION_COMPARE_NOT_SET (-23)
ERR_UNCOMPATIBLE (-24)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_BOOL (-12)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerPositionCompareAquadBWindowedGet $SocketID $FullPositionerName MinimumPosition MaximumPosition EnableState

Input parameters

SocketID .....integer ..... Socket identifier gets by the “TCP_ConnectToServer” function
FullPositionerName .....string ..... Positioner name

Output parameters

MinimumPosition .....double..... Minimum position (units)
MaximumPosition.....double..... Maximum position (units)
EnableState .....bool ..... Windowed AquadB state (true=enabled or false=disabled)

Return

Error.....integer ..... TCL error code (0=success or 1=syntax error) or function error code

C / C++

Prototype

int **PositionerPositionCompareAquadBWindowedGet** (int SocketID, char FullPositionerName[250] , double* MinimumPosition, double* MaximumPosition, bool * EnableState)

Input parameters

SocketID ..... int .....Socket identifier gets by the “TCP_ConnectToServer” function
FullPositionerName ..... char *.....Positioner name

Output parameters

MinimumPosition ..... double *.....Minimum position (units)
MaximumPosition..... double *.....Maximum position (units)

EnableState ..... bool *.....Windowed AquadB state (true=enabled or false=disabled)

Return

Error..... int.....Function error code

VISUAL BASIC

Prototype

Long **PositionerPositionCompareAquadBWindowedGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, MinimumPosition As Double, MaximumPosition As Double, EnableState As Boolean)

Input parameters

SocketID ..... Long .....Socket identifier gets by the “TCP_ConnectToServer” function
FullPositionerName ..... String .....Positioner name

Output parameters

MinimumPosition ..... Double .....Minimum position (units)
MaximumPosition..... Double .....Maximum position (units)
EnableState ..... Boolean.....Windowed AquadB state (true=enabled or false=disabled)

Return

Error..... Long .....Function error code

MATLAB

Prototype

[Error, MinimumPosition, MaximumPosition, EnableState]
PositionerPositionCompareAquadBWindowedGet (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID ..... int32.....Socket identifier gets by the “TCP_ConnectToServer” function
FullPositionerName ..... cstring.....Positioner name

Return

Error..... int32.....Function error code
MinimumPosition ..... double.....Minimum position (units)
MaximumPosition..... double.....Maximum position (units)
EnableState ..... bool.....Windowed AquadB state (true=enabled or false=disabled)

PYTHON

Prototype

[Error, MinimumPosition, MaximumPosition, EnableState]
PositionerPositionCompareAquadBWindowedGet (integer SocketID, string FullPositionerName)

Input parameters

SocketID ..... integer.....Socket identifier gets by the “TCP_ConnectToServer” function
FullPositionerName ..... string.....Positioner name

Return

Error..... integer.....Function error code
MinimumPosition ..... double.....Minimum position (units)
MaximumPosition..... double.....Maximum position (units)
EnableState ..... bool.....Windowed AquadB state (true=enabled or false=disabled)

2.2.4.51. PositionerPositionCompareAquadBWindowedSet

NAME

PositionerPositionCompareAquadBWindowedSet – Sets the windowed AquadB signal parameters.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Check the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type: ERR_WRONG_OBJECT_TYPE (-8)
- Check the position encoder (“AquadB” or “AnalogInterpolated”): ERR_UNCOMPATIBLE (-24)
- Check input parameter values: ERR_PARAMETER_OUT_OF_RANGE (-17)
  MinimumPosition < MaximumPosition
  MinimumPosition >= MinimumTargetPosition
  MaximumPosition <= MaximumTargetPosition
- Check if the CIE board supports this function: ERR_HARDWARE_FUNCTION_NOT_SUPPORTED (-115)
- Check position compare state (must be disabled): ERR_NOT_ALLOWED_ACTION (-22)

DESCRIPTION

This function allows to set the parameters for the position windowed AquadB output signal on the PCO connector (the 2&3 or 4&5 pins) of the XPS controller cards.
These parameters are effectives only when the position compare mode is enabled by the *PositionerPositionCompareEnable()* function.

NOTE:
This function can be used only with a position encoder (“AquadB” or “AnalogInterpolated”). If there is no position encoder then the ERR_UNCOMPATIBLE (-24) error is returned.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_IN_INITIALIZATION (-21)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_UNCOMPATIBLE (-24)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_HARDWARE_FUNCTION_NOT_SUPPORTED (-115)
SUCCESS (0) : no error

TCL

Prototype
PositionerPositionCompareAquadBWindowedSet $SocketID $FullPositionerName $MinimumPosition $MaximumPosition

Input parameters
SocketID ..... integer .....Socket identifier gets by the “TCP_ConnectToServer” function
FullPositionerName ..... string .....Positioner name
MinimumPosition ..... double .....Minimum position (units)
MaximumPosition..... double .....Maximum position (units)

Output parameters (None)

Return
Error..... integer .....TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **PositionerPositionCompareAquadBWindowedSet** (int SocketID, char FullPositionerName[250] , double * MinimumPosition, double * MaximumPosition)

Input parameters

SocketID	int	Socket identifier gets by the “TCP_ConnectToServer” function
FullPositionerName	char *	Positioner name
MinimumPosition	double	Minimum position (units)
MaximumPosition	double	Maximum position (units)

Output parameters (None)

Return

Error	int	Function error code

VISUAL BASIC

Prototype

Long **PositionerPositionCompareAquadBWindowedSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal MinimumPosition As Double, ByVal MaximumPosition As Double)

Input parameters

SocketID	Long	Socket identifier gets by the “TCP_ConnectToServer” function
FullPositionerName	String	Positioner name
MinimumPosition	double	Minimum position (units)
MaximumPosition	double	Maximum position (units)

Output parameters (None)

Return

Error	Long	Function error code

MATLAB

Prototype

[Error] **PositionerPositionCompareAquadBWindowedSet** (int32 SocketID, cstring FullPositionerName, double MinimumPosition, double MaximumPosition)

Input parameters

SocketID	int32	Socket identifier gets by the “TCP_ConnectToServer” function
FullPositionerName	cstring	Positioner name
MinimumPosition	double	Minimum position (units)
MaximumPosition	double	Maximum position (units)

Return

Error	int32	Function error code

PYTHON

Prototype

[Error] **PositionerPositionCompareAquadBWindowedSet** (integer SocketID, string FullPositionerName, double MinimumPosition, double MaximumPosition)

Input parameters

SocketID	integer	Socket identifier gets by the “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name
MinimumPosition	double	Minimum position (units)
MaximumPosition	double	Maximum position (units)

Return

Error	integer	Function error code

2.2.4.52. PositionerRawEncoderPositionGet

NAME

PositionerRawEncoderPositionGet – Returns the raw encoder position for a positioner.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid object type: ERR_WRONG_OBJECT_TYPE (-8)
- Valid positioner name: ERR_GROUP_NAME (-19)
- Valid output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function allows get the raw encoder position from a user corrected position for a positioner.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0): no error

TCL

Prototype

PositionerRawEncoderPositionGet SocketID PositionerName UserEncoderPosition RawEncoderPosition

Input parameters

SocketID ..... integer ..... Socket identifier gets by the “TCP_ConnectToServer” function
PositionerName..... string ..... Positioner name (maximum size = 250)
UserEncoderPosition..... floating point..... User corrected encoder position

Output parameters

RawEncoderPosition..... floating point..... Raw encoder position

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **PositionerRawEncoderPositionGet** (int SocketID, char * PositionerName, double UserEncoderPosition, double * RawEncoderPosition)

Input parameters

SocketID ..... int ..... Socket identifier gets by the “TCP_ConnectToServer”function
PositionerName..... char * ..... Positioner name
UserEncoderPosition..... double ..... User corrected encoder position

Output parameters

RawEncoderPosition..... double * ..... Raw encoder position

Return

Function error code

VISUAL BASIC

Prototype

Long **PositionerRawEncoderPositionGet** (ByVal SocketID As Long, ByVal PositionerName As String, ByVal UserEncoderPosition As Double, RawEncoderPosition As Double)

Input parameters

SocketID ..... Long ..... Socket identifier gets by the “TCP_ConnectToServer” function
PositionerName ..... String ..... Positioner name
UserEncoderPosition ..... Double ..... User corrected encoder position

Output parameters

RawEncoderPosition ..... Double ..... Raw encoder position

Return

Function error code

MATLAB

Prototype

[Error, RawEncoderPosition] **PositionerRawEncoderPositionGet** (int32 SocketID, cstring PositionerName, double UserEncoderPosition)

Input parameters

SocketID ..... int32 ..... Socket identifier gets by the “TCP_ConnectToServer”function
PositionerName ..... cstring ..... Positioner name
UserEncoderPosition ..... double ..... User corrected encoder position

Return

Error ..... int32 ..... Function error code
RawEncoderPosition ..... doublePtr ..... Raw encoder position

PYTHON

Prototype

[Error, RawEncoderPosition] **PositionerRawEncoderPositionGet** (integer SocketID, string PositionerName, double UserEncoderPosition)

Input parameters

SocketID ..... integer ..... Socket identifier gets by the “TCP_ConnectToServer”function
PositionerName ..... string ..... Positioner name
UserEncoderPosition ..... double ..... User corrected encoder position

Return

Error ..... integer ..... Function error code
RawEncoderPosition ..... doublePtr ..... Raw encoder position

2.2.4.53. PositionersEncoderIndexDifferenceGet

NAME

PositionersEncoderIndexDifferenceGet – Gets the distance between the two index encoders (gantry).

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Check the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the group type (must be a SingleAxis or an XY): ERR_WRONG_OBJECT_TYPE (-8)
- Check the positioner name : ERR_POSITIONER_NAME (-18)
- Check the positioner type (must not be a secondary positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Check the positioner type (must be “gantry”): ERR_UNCOMPATIBLE (-24)
- Check the positioner was at least once homed: ERR_NEED_TO_BE_HOMED_AT_LEAST_ONCE (-109)
- Check output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function returns the distance between the two index encoders of a “primary positioner – secondary positioner” couple. To use this function, the positioner must be configured in “grantry” mode else the ERR_UNCOMPATIBLE (-24) error is returned.

For further information about the gantry mode, refer to the “SYSTEM – Manual Configuration – Gantries (Secondary Positioners)” section in the user’s manual.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_NEED_TO_BE_HOMED_AT_LEAST_ONCE (-109)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionersEncoderIndexDifferenceGet $SocketID $FullPositionerName Distance

Input parameters

SocketID .....integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....string ..... Positioner name

Output parameters

Distance .....double..... Distance between the two index encoders (units)

Return

Error.....integer ..... TCL error code (0=success or 1=syntax error) or function error code

C / C++

Prototype
int **PositionersEncoderIndexDifferenceGet** (int SocketID, char FullPositionerName[250] , double* Distance)

Input parameters
SocketID ..... int .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... char * .....Positioner name

Output parameters
Distance ..... double * .....Distance between the two index encoders (units)

Return
Error..... int.....Function error code

VISUAL BASIC

Prototype
Long **PositionersEncoderIndexDifferenceGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, Distance As Double)

Input parameters
SocketID ..... Long .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String .....Positioner name

Output parameters
Distance ..... Double .....Distance between the two index encoders (units)

Return
Error..... Long .....Function error code

MATLAB

Prototype
[Error, Distance] **PositionersEncoderIndexDifferenceGet** (int32 SocketID, cstring FullPositionerName)

Input parameters
SocketID ..... int32.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring.....Positioner name

Return
Error..... int32.....Function error code
Distance ..... double.....Distance between the two index encoders (units)

PYTHON

Prototype
[Error, Distance] **PositionersEncoderIndexDifferenceGet** (integer SocketID, string FullPositionerName)

Input parameters
SocketID ..... integer.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string.....Positioner name

Return
Error..... integer.....Function error code
Distance ..... double.....Distance between the two index encoders (units)

2.2.4.54. PositionerSGammaExactVelocityAjustedDisplacementGet

NAME

PositionerSGammaExactVelocityAjustedDisplacementGet – Gets adjusted displacement to get exact velocity.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Check command format: ERR_WRONG_FORMAT (-7)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the profiler type (must be “SGamma”): ERR_WRONG_OBJECT_TYPE (-8)
- Check the positioner type (must not be a secondary positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Check output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function returns the closest optimum displacement to obtain a most precise velocity during the displacement.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerSGammaExactVelocityAjustedDisplacementGet $SocketID $FullPositionerName $DesiredDisplacement AdjustedDisplacement

Input parameters

SocketID .....integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....string ..... Positioner name
DesiredDisplacement .....double..... Desired displacement (units)

Output parameters

AdjustedDisplacement .....double..... Ajusted displacement (units)

Return

Error.....integer ..... TCL error code (0=success or 1=syntax error) or function error code

C / C++

Prototype

int **PositionerSGammaExactVelocityAjustedDisplacementGet** (int SocketID, char FullPositionerName[250] , double DesiredDisplacement, double * AdjustedDisplacement)

Input parameters

SocketID ..... int .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... char *.....Positioner name
DesiredDisplacement ..... double .....Desired displacement (units)

Output parameters

AdjustedDisplacement ..... double * .....Ajusted displacement (units)

Return

Error..... int.....Function error code

VISUAL BASIC

Prototype

Long **PositionerSGammaExactVelocityAjustedDisplacementGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal DesiredDisplacement As Double, AdjustedDisplacement As Double)

Input parameters

SocketID ..... Long .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String .....Positioner name
DesiredDisplacement ..... double .....Desired displacement (units)

Output parameters

AdjustedDisplacement ..... double .....Ajusted displacement (units)

Return

Error..... Long .....Function error code

MATLAB

Prototype

[Error, AdjustedDisplacement] **PositionerSGammaExactVelocityAjustedDisplacementGet** (int32 SocketID, cstring FullPositionerName, double DesiredDisplacement)

Input parameters

SocketID ..... int32.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring.....Positioner name
DesiredDisplacement ..... double .....Desired displacement (units)

Return

Error..... int32.....Function error code
AdjustedDisplacement ..... double .....Ajusted displacement (units)

PYTHON

Prototype

[Error, AdjustedDisplacement] **PositionerSGammaExactVelocityAjustedDisplacementGet** (integer SocketID, string FullPositionerName, double DesiredDisplacement)

Input parameters

SocketID ..... integer.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string.....Positioner name
DesiredDisplacement ..... double .....Desired displacement (units)

Return

Error..... integer.....Function error code
AdjustedDisplacement ..... double .....Ajusted displacement (units)

2.2.4.55. PositionerSGammaParametersSet

NAME

PositionerSGammaParametersSet – Sets new motion values for the SGamma profiler.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Check the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the profiler type (must be “SGamma”): ERR_WRONG_OBJECT_TYPE (-8)
- Check the positioner type (must not be a secondary positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Check input parameter types: ERR_WRONG_TYPE_DOUBLE (-14)
- Check input parameter values: ERR_PARAMETER_OUT_OF_RANGE (-17)

  $0 < \mathbf{NewVelocity} \leq MaximumVelocity$

  $0 < \mathbf{NewAcceleration} \leq MaximumAcceleration$

  $2 * ISRProfileGeneratorPeriod \leq \mathbf{NewMinimumJerkTime} \leq \mathbf{NewMaximumJerkTime}$

  (with ISRProfileGeneratorPeriod = 0.0004 ms)

DESCRIPTION

This function allows to define the new SGamma profiler values that will be use in the future displacements.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerSGammaParametersSet $SocketID $FullPositionerName $Velocity $Acceleration $MinimumJerkTime $MaximumJerkTime

Input parameters

SocketID .....integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....string ..... Positioner name
Velocity.....double.....motion velocity (units / seconds)
Acceleration.....double.....motion acceleration (units / seconds²)
MinimumJerkTime .....double.....minimum jerk time (seconds)
MaximumJerkTime.....double.....maximum jerk time (seconds)

Output parameters (None)

Return

Error.....integer ..... TCL error code (0=success or 1=syntax error) or function error code

C / C++

Prototype

int **PositionerSGammaParametersSet** (int SocketID, char FullPositionerName[250] , double Velocity, double Acceleration, double MinimumJerkTime, double MaximumJerkTime)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	char *	Positioner name
Velocity	double	motion velocity (units / seconds)
Acceleration	double	motion acceleration (units / seconds²)
MinimumJerkTime	double	minimum jerk time (seconds)
MaximumJerkTime	double	maximum jerk time (seconds)

Output parameters (None)

Return

Error	int	Function error code

VISUAL BASIC

Prototype

Long **PositionerSGammaParametersSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal Velocity As Double, ByVal Acceleration As Double, ByVal MinimumJerkTime As Double, ByVal MaximumJerkTime As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	String	Positioner name
Velocity	Double	motion velocity (units / seconds)
Acceleration	Double	motion acceleration (units / seconds²)
MinimumJerkTime	Double	minimum jerk time (seconds)
MaximumJerkTime	Double	maximum jerk time (seconds)

Output parameters (None)

Return

Error	Long	Function error code

MATLAB

Prototype

[Error] **PositionerSGammaParametersSet** (int32 SocketID, cstring FullPositionerName, double Velocity, double Acceleration, double MinimumJerkTime, double MaximumJerkTime)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	cstring	Positioner name
Velocity	double	motion velocity (units / seconds)
Acceleration	double	motion acceleration (units / seconds²)
MinimumJerkTime	double	minimum jerk time (seconds)
MaximumJerkTime	double	maximum jerk time (seconds)

Return

Error	int32	Function error code

PYTHON

Prototype

[Error] **PositionerSGammaParametersSet** (integer SocketID, string FullPositionerName, double Velocity, double Acceleration, double MinimumJerkTime, double MaximumJerkTime)

Input parameters

SocketID ..... integer.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string.....Positioner name
Velocity..... double .....motion velocity (units / seconds)
Acceleration..... double .....motion acceleration (units / seconds²)
MinimumJerkTime ..... double .....minimum jerk time (seconds)
MaximumJerkTime..... double .....maximum jerk time (seconds)

Return

Error..... integer.....Function error code

2.2.4.56. PositionerSGammaParametersGet

NAME

PositionerSGammaParametersGet – Gets current motion values from the SGamma profiler.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Check the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the profiler type (must be “SGamma”): ERR_WRONG_OBJECT_TYPE (-8)
- Check the positioner type (must not be a secondary positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Check input parameter types: ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function allow to get the current SGamma profiler values that are used in the displacements.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerSGammaParametersGet $SocketID $FullPositionerName Velocity Acceleration MinimumJerkTime MaximumJerkTime

Input parameters

SocketID .....integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....string ..... Positioner name

Output parameters

Velocity.....double.....motion velocity (units / seconds)
Acceleration.....double.....motion acceleration (units / seconds²)
MinimumJerkTime .....double.....minimum jerk time (seconds)
MaximumJerkTime.....double.....maximum jerk time (seconds)

Return

Error.....integer ..... TCL error code (0=success or 1=syntax error) or function error code

C / C++

Prototype

int **PositionerSGammaParametersGet** (int SocketID, char FullPositionerName[250] , double *Velocity, double *Acceleration, double *MinimumJerkTime, double *MaximumJerkTime)

Input parameters

SocketID ..... int .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... char * .....Positioner name

Output parameters

Velocity..... double * .....motion velocity (units / seconds)
Acceleration..... double * .....motion acceleration (units / seconds²)
MinimumJerkTime ..... double * .....minimum jerk time (seconds)
MaximumJerkTime..... double * .....maximum jerk time (seconds)

Return

Error..... int.....Function error code

VISUAL BASIC

Prototype

Long **PositionerSGammaParametersGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, Velocity As Double, Acceleration As Double, MinimumJerkTime As Double, MaximumJerkTime As Double)

Input parameters

SocketID ..... Long .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String .....Positioner name

Output parameters

Velocity..... Double .....motion velocity (units / seconds)
Acceleration..... Double .....motion acceleration (units / seconds²)
MinimumJerkTime ..... Double .....minimum jerk time (seconds)
MaximumJerkTime..... Double .....maximum jerk time (seconds)

Return

Error..... Long .....Function error code

MATLAB

Prototype

[Error, Velocity, Acceleration, MinimumJerkTime, MaximumJerkTime] **PositionerSGammaParametersGet** (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID ..... int32.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring.....Positioner name

Return

Error..... int32.....Function error code
Velocity..... double.....motion velocity (units / seconds)
Acceleration..... double.....motion acceleration (units / seconds²)
MinimumJerkTime ..... double.....minimum jerk time (seconds)
MaximumJerkTime..... double.....maximum jerk time (seconds)

PYTHON

Prototype

[Error, Velocity, Acceleration, MinimumJerkTime, MaximumJerkTime] **PositionerSGammaParametersGet** (integer SocketID, string FullPositionerName)

Input parameters

SocketID ..... integer.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string.....Positioner name

Return

Error..... integer.....Function error code
Velocity..... double.....motion velocity (units / seconds)
Acceleration..... double.....motion acceleration (units / seconds²)
MinimumJerkTime ..... double.....minimum jerk time (seconds)
MaximumJerkTime..... double.....maximum jerk time (seconds)

2.2.4.57. PositionerSGammaPreviousMotionTimesGet

NAME

PositionerSGammaPreviousMotionTimesGet – Gets the setting time and the settling time.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the profiler type (must be “SGamma”): ERR_WRONG_OBJECT_TYPE (-8)
- Check the positioner type (must not be a secondary positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Check input parameter types: ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function returns the motion (setting) and settling times from the previous motion.
The setting time represents the defined time to do the previous displacement.
The settling time represents the effective settling time for a motion done.

ERROR CODES

ERR_FATAL_INIT (-20
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerSGammaPreviousMotionTimesGet $SocketID $FullPositionerName SettingTime SettlingTime

Input parameters

SocketID .....integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....string ..... Positioner name

Output parameters

SettingTime.....double.....setting time (seconds)
SettlingTime.....double.....settling time (seconds)

Return

Error.....integer ..... TCL error code (0=success or 1=syntax error) or function error code

C / C++

Prototype

int **PositionerSGammaPreviousMotionTimesGet** (int SocketID, char FullPositionerName[250] , double* SettingTime, double* SettlingTime)

Input parameters

SocketID ..... int .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... char *.....Positioner name

Output parameters

SettingTime..... double *.....setting time (seconds)
SettlingTime..... double *.....settling time (seconds)

Return

Error..... int.....Function error code

VISUAL BASIC

Prototype

Long **PositionerSGammaPreviousMotionTimesGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, SettingTime As Double, SettlingTime As Double)

Input parameters

SocketID ..... Long .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String .....Positioner name

Output parameters

SettingTime..... double .....setting time (seconds)
SettlingTime..... double .....settling time (seconds)

Return

Error..... Long .....Function error code

MATLAB

Prototype

[Error, SettingTime, SettlingTime] **PositionerSGammaPreviousMotionTimesGet** (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID ..... int32.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring.....Positioner name

Return

Error..... int32.....Function error code
SettingTime..... double .....setting time (seconds)
SettlingTime..... double .....settling time (seconds)

PYTHON

Prototype

[Error, SettingTime, SettlingTime] **PositionerSGammaPreviousMotionTimesGet** (integer SocketID, string FullPositionerName)

Input parameters

SocketID ..... integer.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string.....Positioner name

Return

Error..... integer.....Function error code
SettingTime..... double .....setting time (seconds)
SettlingTime..... double .....settling time (seconds)

2.2.4.58. PositionerStageParameterGet

NAME

PositionerStageParameterGet – Gets a stage parameter value from the stages.ini file.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check input/output parameter types: ERR_WRONG_TYPE_CHAR (-13)
- Check the positioner name and the parameter name: ERR_UNCOMPATIBLE (-24)

DESCRIPTION

This function allows return the stage parameter value from the stages.ini file for a selected positioner.
The positioner name allows get the stage name. And next, the parameter name is reading in the section of this stage name.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_UNCOMPATIBLE (-24)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
SUCCESS (0) : no error

TCL

Prototype

PositionerStageParameterGet $SocketID $FullPositionerName $ParameterName ParameterValue

Input parameters

SocketID .....integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....string ..... Positioner name
ParameterName.....string ..... Parameter name

Output parameters

ParameterValue.....string ..... Parameter value

Return

Error.....integer ..... TCL error code (0=success or 1=syntax error) or function error code

C / C++

Prototype

int **PositionerStageParameterGet** (int SocketID, char FullPositionerName[250] , char * ParameterName, char * ParameterValue)

Input parameters

SocketID ..... int .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... char *.....Positioner name
ParameterName..... char *.....Parameter name

Output parameters

ParameterValue..... char *.....Parameter value

Return

Error..... int.....Function error code

VISUAL BASIC

Prototype

Long **PositionerStageParameterGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal ParameterName As String, ByVal ParameterValue As String)

Input parameters

SocketID ..... Long .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String .....Positioner name
ParameterName..... String .....Parameter name

Output parameters

ParameterValue..... String .....Parameter value

Return

Error..... Long .....Function error code

MATLAB

Prototype

[Error, ParameterValue] **PositionerStageParameterGet** (int32 SocketID, cstring FullPositionerName, cstring ParameterName)

Input parameters

SocketID ..... int32.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring.....Positioner name
ParameterName..... cstring.....Parameter name

Return

Error..... int32.....Function error code
ParameterValue..... cstring.....Parameter value

PYTHON

Prototype

[Error, ParameterValue] **PositionerStageParameterGet** (integer SocketID, string FullPositionerName, string ParameterName)

Input parameters

SocketID ..... integer.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string.....Positioner name
ParameterName..... string.....Parameter name

Return

Error..... integer.....Function error code
ParameterValue..... string.....Parameter value

2.2.4.59. PositionerStageParameterSet

NAME

PositionerStageParameterSet – Saves a new stage parameter value into the stages.ini file.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check input/output parameter types: ERR_WRONG_TYPE_CHAR (-13)
- Check the positioner name and the parameter name: ERR_UNCOMPATIBLE (-24)
- Check the user rights (must be identified as administrator): ERR_NEED_ADMINISTRATOR_RIGHTS (-107)

DESCRIPTION

This function allows save a new stage parameter value in the “stages.ini” file.
The positioner name allows get the stage name and next, the parameter name is scanned in the section of this stage name. Once the parameter is found, the parameter value is modified by the new value.

If the file reading failed then the ERR_READ_FILE (-61) error is returned
If the file writing failed then the ERR_WRITE_FILE (-60) error is returned

NOTE:
To use this function, the user must be identified with administrator rights (“Login” function)

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_NEED_ADMINISTRATOR_RIGHTS (-107)
ERR_READ_FILE (-61)
ERR_UNCOMPATIBLE (-24)
ERR_WRITE_FILE (-60)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
SUCCESS (0) : no error

TCL

Prototype

PositionerStageParameterSet $SocketID $FullPositionerName $ParameterName ParameterValue

Input parameters

SocketID .....integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....string ..... Positioner name
ParameterName.....string ..... Parameter name

Output parameters

ParameterValue.....string ..... Parameter value

Return

Error.....integer ..... TCL error code (0=success or 1=syntax error) or function error code

C / C++

Prototype

int **PositionerStageParameterSet** (int SocketID, char FullPositionerName[250] , char * ParameterName, char * ParameterValue)

Input parameters

SocketID ..... int .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... char * .....Positioner name
ParameterName..... char * .....Parameter name

Output parameters

ParameterValue..... char * .....Parameter value

Return

Error..... int.....Function error code

VISUAL BASIC

Prototype

Long **PositionerStageParameterSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal ParameterName As String, ByVal ParameterValue As String)

Input parameters

SocketID ..... Long .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String .....Positioner name
ParameterName..... String .....Parameter name

Output parameters

ParameterValue..... String .....Parameter value

Return

Error..... Long .....Function error code

MATLAB

Prototype

[Error, ParameterValue] **PositionerStageParameterSet** (int32 SocketID, cstring FullPositionerName, cstring ParameterName)

Input parameters

SocketID ..... int32.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring.....Positioner name
ParameterName..... cstring.....Parameter name

Return

Error..... int32.....Function error code
ParameterValue..... cstring.....Parameter value

PYTHON

Prototype

[Error, ParameterValue] **PositionerStageParameterSet** (integer SocketID, string FullPositionerName, string ParameterName)

Input parameters

SocketID ..... integer.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string.....Positioner name
ParameterName..... string.....Parameter name

Return

Error..... integer.....Function error code
ParameterValue..... string.....Parameter value

2.2.4.60. PositionerTimeFlasherDisable

NAME

PositionerTimeFlasherDisable – Disables the time flasher mode.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type (must not be a secondary positioner): ERR_WRONG_OBJECT_TYPE (-8)

DESCRIPTION

This function disables the time flasher mode. The time flasher mode is a trigger output per axis that can be either configured to output distance spaced pulses or time spaced pulses. The output pulses are accessible from the PCO connector at the back of the XPS controller.

For a more thorough description of the position compare output, please refer to the XPS User's manual, section named Triggers / Position Compare Output.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_OBJECT_TYPE (-8)
SUCCESS (0) : no error

TCL

Prototype

PositionerTimeFlasherDisable $SocketID $FullPositionerName

Input parameters

SocketID .....integer ..... Socket identifier got from "TCP_ConnectToServer" function
FullPositionerName .....string ..... Positioner name

Output parameters

None

Return

Error.....integer ..... TCL error code (0=success or 1=syntax error) or function error code

C / C++

Prototype

int **PositionerTimeFlasherDisable** (int SocketID, char FullPositionerName[250])

Input parameters

SocketID ..... int .....Socket identifier got from "TCP_ConnectToServer" function
FullPositionerName ..... char *.....Positioner name

Output parameters

None

Return

Error..... int.....Function error code

VISUAL BASIC

Prototype

Long **PositionerTimeFlasherDisable** (ByVal SocketID As Long, ByVal FullPositionerName As String)

Input parameters

SocketID ..... Long .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String .....Positioner name

Output parameters

None

Return

Error..... Long .....Function error code

MATLAB

Prototype

[Error] **PositionerTimeFlasherDisable** (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID ..... int32.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring.....Positioner name

Return

Error..... int32.....Function error code

PYTHON

Prototype

[Error] **PositionerTimeFlasherDisable** (integer SocketID, string FullPositionerName)

Input parameters

SocketID ..... integer.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string.....Positioner name

Return

Error..... integer.....Function error code

2.2.4.61. PositionerTimeFlasherEnable

NAME

PositionerTimeFlasherEnable – Enables the time flasher mode.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type (must not be a secondary positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Check the time flasher parameters (must be configured) : ERR_NOT_ALLOWED_ACTION (-22)

DESCRIPTION

This function enables the time flasher mode. The time flasher mode is a trigger output per axis that can be either configured to output distance spaced pulses or time spaced pulses. The output pulses are accessible from the PCO connector at the back of the XPS controller.

To use this function, the group must be in READY state else the ERR_NOT_ALLOWED_ACTION (-22) error is returned.

For a more thorough description of the position compare output, please refer to the XPS User's manual, section named Triggers / Position Compare Output.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_OBJECT_TYPE (-8)
SUCCESS (0) : no error

TCL

Prototype

PositionerTimeFlasherEnable $SocketID $FullPositionerName

Input parameters

SocketID .....integer ..... Socket identifier got from "TCP_ConnectToServer" function
FullPositionerName .....string ..... Positioner name

Output parameters

None

Return

Error.....integer ..... TCL error code (0=success or 1=syntax error) or function error code

C / C++

Prototype

int **PositionerTimeFlasherEnable** (int SocketID, char FullPositionerName[250])

Input parameters

SocketID ..... int .....Socket identifier got from "TCP_ConnectToServer" function
FullPositionerName ..... char * .....Positioner name

Output parameters

None

Return

Error..... int.....Function error code

VISUAL BASIC

Prototype

Long **PositionerTimeFlasherEnable** (ByVal SocketID As Long, ByVal FullPositionerName As String)

Input parameters

SocketID ..... Long .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... String .....Positioner name

Output parameters

None

Return

Error..... Long .....Function error code

MATLAB

Prototype

[Error] **PositionerTimeFlasherEnable** (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID ..... int32.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... cstring.....Positioner name

Return

Error..... int32.....Function error code

PYTHON

Prototype

[Error] **PositionerTimeFlasherEnable** (integer SocketID, string FullPositionerName)

Input parameters

SocketID ..... integer.....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName ..... string.....Positioner name

Return

Error..... integer.....Function error code

2.2.4.62. PositionerTimeFlasherGet

NAME

PositionerTimeFlasherGet – Gets the time flasher parameters.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type (must not be a secondary positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Check output parameter types: ERR_WRONG_TYPE_DOUBLE (-14), ERR_WRONG_TYPE_BOOL (-12)
- Check the time flasher parameters (must be configured) : ERR_POSITION_COMPARE_NOT_SET (-23)

DESCRIPTION

This function returns the parameters of the time flasher trigger. The time flasher mode is defined by:

- a position window defined by a minimum position and a maximum position
- a time period to set the time spaced pulses.

For a more thorough description of the position compare output, please refer to the XPS Motion Tutorial, section named Triggers / Position Compare Output.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_POSITION_COMPARE_NOT_SET (-23)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_BOOL (-12)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

PositionerTimeFlasherGet $SocketID $FullPositionerName MinimumPosition MaximumPosition TimePeriod EnableState

Input parameters

SocketID .....integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....string ..... Positioner name

Output parameters

MinimumPosition .....double..... Minimum position (units)
MaximumPosition.....double..... Maximum position (units)
TimePeriod.....double..... Time period (seconds)
EnableState .....bool ..... Enable time flasher state (true=enabled and false=disabled)

Return

Error.....integer ..... TCL error code (0=success or 1=syntax error) or function error code

C / C++

Prototype

int **PositionerTimeFlasherGet** (int SocketID, char FullPositionerName[250] , double * MinimumPosition, double * MaximumPosition, double * TimePeriod, bool * EnableState)

Input parameters

SocketID .....int .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....char *.....Positioner name

Output parameters

MinimumPosition .....double *.....Minimum position (units)
MaximumPosition.....double *.....Maximum position (units)
TimePeriod.....double *.....Time period (seconds)
EnableState .....bool * .....Enable time flasher state (true=enabled and false=disabled)

Return

Error.....int .....Function error code

VISUAL BASIC

Prototype

Long **PositionerTimeFlasherGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, MinimumPosition As Double, MaximumPosition As Double, TimePeriod As Double, EnableState As Boolean)

Input parameters

SocketID .....Long .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....String.....Positioner name

Output parameters

MinimumPosition .....Double.....Minimum position (units)
MaximumPosition.....Double.....Maximum position (units)
TimePeriod.....Double.....Time period (seconds)
EnableState .....Boolean .....Enable time flasher state (true=enabled and false=disabled)

Return

Error.....Long .....Function error code

MATLAB

Prototype

[Error, MinimumPosition, MaximumPosition, TimePeriod, EnableState] **PositionerTimeFlasherGet** (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID .....int32 .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....cstring.....Positioner name

Return

Error.....int32 .....Function error code
MinimumPosition .....double.....Minimum position (units)
MaximumPosition.....double.....Maximum position (units)
TimePeriod.....double.....Time period (seconds)
EnableState .....bool .....Enable time flasher state (true=enabled and false=disabled)

PYTHON

Prototype

[Error, MinimumPosition, MaximumPosition, TimePeriod, EnableState] **PositionerTimeFlasherGet** (integer SocketID, string FullPositionerName)

Input parameters

SocketID .....integer .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....string .....Positioner name

Return

Error.....integer .....Function error code
MinimumPosition .....double.....Minimum position (units)
MaximumPosition.....double.....Maximum position (units)
TimePeriod.....double.....Time period (seconds)
EnableState .....bool .....Enable time flasher state (true=enabled and false=disabled)

2.2.4.63. PositionerTimeFlasherSet

NAME

PositionerTimeFlasherSet – Sets the time flasher parameters.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Check the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type (must not be a secondary positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Check the position encoder (must be used): ERR_UNCOMPATIBLE (-24)
- Check input parameter types: ERR_WRONG_TYPE_DOUBLE (-14)
- Check if the CIE board supports this function: ERR_HARDWARE_FUNCTION_NOT_SUPPORTED (-115)
- Check the time flasher state (must be disabled): ERR_NOT_ALLOWED_ACTION (-22)
- Check input parameter values: ERR_PARAMETER_OUT_OF_RANGE (-17)
  MinimumPosition < MaximumPosition
  MinimumPosition >= MinimumTravelLimit
  MaximumPosition <= MaximumTravelLimit
  0.0000004 <= TimePeriod <= 50.0 (Max 2.5 MHz and Min 0.02 Hz)

DESCRIPTION

This function configured the time flasher parameters. The time flasher output trigger uses the PCO connector on the XPS controller cards. The time flasher mode is defined by:

- a position window defined by a minimum position and a maximum position
- a time period to set the time spaced pulses.

NOTES:
This function is not available without a position encoder.
These parameters are used only when the time flasher mode is enabled. To enable the time flasher mode, use the "PositionerPositionCompareEnable" function.

For a more thorough description of the position compare output, please refer to the XPS Motion Tutorial, section named Triggers / Position Compare Output.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_UNCOMPATIBLE (-24)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
ERR_HARDWARE_FUNCTION_NOT_SUPPORTED (-115)
SUCCESS (0) : no error

TCL

Prototype
PositionerTimeFlasherSet $SocketID $FullPositionerName $MinimumPosition $MaximumPosition $TimePeriod

Input parameters
SocketID .....integer ..... Socket identifier got from "TCP_ConnectToServer" function
FullPositionerName .....string ..... Positioner name
MinimumPosition .....double..... Minimum position (units)
MaximumPosition.....double..... Maximum position (units)

TimePeriod.....double..... Time period (seconds)

Output parameters (None)

Return

Error.....integer ..... TCL error code (0=success or 1=syntax error) or function error code

C / C++

Prototype

int **PositionerTimeFlasherSet** (int SocketID, char FullPositionerName[250] , double * MinimumPosition, double * MaximumPosition, double * TimePeriod, bool * EnableState)

Input parameters

SocketID .....int .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....char *.....Positioner name
MinimumPosition .....double *.....Minimum position (units)
MaximumPosition.....double *.....Maximum position (units)
TimePeriod.....double *.....Time period (seconds)

Output parameters (None)

Return

Error.....int .....Function error code

VISUAL BASIC

Prototype

Long **PositionerTimeFlasherSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, MinimumPosition As Double, MaximumPosition As Double, TimePeriod As Double, EnableState As Boolean)

Input parameters

SocketID .....Long .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....String.....Positioner name
MinimumPosition .....Double.....Minimum position (units)
MaximumPosition.....Double.....Maximum position (units)
TimePeriod.....Double.....Time period (seconds)

Output parameters (None)

Return

Error.....Long .....Function error code

MATLAB

Prototype

[Error] **PositionerTimeFlasherSet** (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID .....int32 .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....cstring.....Positioner name
MinimumPosition .....double.....Minimum position (units)
MaximumPosition.....double.....Maximum position (units)
TimePeriod.....double.....Time period (seconds)

Return

Error.....int32 .....Function error code

PYTHON

Prototype

[Error] **PositionerTimeFlasherSet** (integer SocketID, string FullPositionerName)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name
MinimumPosition	double	Minimum position (units)
MaximumPosition	double	Maximum position (units)
TimePeriod	double	Time period (seconds)

Return

Error	integer	Function error code

2.2.4.64. PositionerUserTravelLimitsGet

NAME

PositionerUserTravelLimitsGet – Gets the user travel limits.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type (must not be a secondary positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Check output parameter types: ERR_WRONG_TYPE_DOUBLE (-14)
- If piezo driver, check if driver is not initialized: ERR_NOT_ALLOWED_DRIVER_NOT_INITIALIZED (-118)

DESCRIPTION

This function returns the user travel limits defined for the selected positioner. These limits are used to check each displacement.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
ERR_NOT_ALLOWED_DRIVER_NOT_INITIALIZED (-118)
SUCCESS (0) : no error

TCL

Prototype

PositionerUserTravelLimitsGet $SocketID $FullPositionerName UserMinimumTarget UserMaximumTarget

Input parameters

SocketID .....integer ..... Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....string ..... Positioner Name

Output parameters

UserMinimumTarget.....double..... User minimum travel limit (units)
UserMaximumTarget.....double..... User maximum travel limit (units)

Return

Error.....integer ..... TCL error code (0=success or 1=syntax error) or function error code

C / C++

Prototype

int **PositionerUserTravelLimitsGet** (int SocketID, char FullPositionerName[250] , double * UserMinimumTarget, double * UserMaximumTarget)

Input parameters

SocketID .....int .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....char *.....Positioner name

Output parameters

UserMinimumTarget.....double *.....User minimum travel limit (units)
UserMaximumTarget.....double *.....User maximum travel limit (units)

Return

Error.....int .....Function error code

VISUAL BASIC

Prototype

Long **PositionerUserTravelLimitsGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, UserMinimumTarget As Double, UserMaximumTarget As Double)

Input parameters

SocketID .....Long .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....String.....Positioner name

Output parameters

UserMinimumTarget.....Double.....User minimum travel limit (units)
UserMaximumTarget.....Double.....User maximum travel limit (units)

Return

Error.....Long .....Function error code

MATLAB

Prototype

[Error, UserMinimumTarget, UserMaximumTarget] **PositionerUserTravelLimitsGet** (int32 SocketID, cstring FullPositionerName)

Input parameters

SocketID .....int32 .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....cstring.....Positioner name

Return

Error.....int32 .....Function error code
UserMinimumTarget.....double.....User minimum travel limit (units)
UserMaximumTarget.....double.....User maximum travel limit (units)

PYTHON

Prototype

[Error, UserMinimumTarget, UserMaximumTarget] **PositionerUserTravelLimitsGet** (integer SocketID, string FullPositionerName)

Input parameters

SocketID .....integer .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....string .....Positioner name

Return

Error.....integer .....Function error code
UserMinimumTarget.....double.....User minimum travel limit (units)
UserMaximumTarget.....double.....User maximum travel limit (units)

2.2.4.65. PositionerUserTravelLimitsSet

NAME

PositionerUserTravelLimitsSet – Sets the user travel limits.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Check the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner type (must not be a secondary positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Check output parameter types: ERR_WRONG_TYPE_DOUBLE (-14)
- Check input parameter values: ERR_PARAMETER_OUT_OF_RANGE (-17)
  - *UserMinimumTargetPosition < UserMaximumTargetPosition*
  - *MinimumTargetPosition <= UserMinimumTargetPosition <= MaximumTargetPosition*
  - *MinimumTargetPosition <= UserMaximumTargetPosition <= MaximumTargetPosition*
  - *UserMinimumTargetPosition <= ProfilerPosition*
  - *UserMaximumTargetPosition >= ProfilerPosition*
- If piezo driver, check if driver is not initialized: ERR_NOT_ALLOWED_DRIVER_NOT_INITIALIZED (-118)

DESCRIPTION

This function sets the new user travel limits of the selected positioner. These limits are used to check each displacement.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
ERR_NOT_ALLOWED_DRIVER_NOT_INITIALIZED (-118)
SUCCESS (0) : no error

TCL

Prototype

PositionerUserTravelLimitsSet $SocketID $FullPositionerName $UserMinimumTarget $UserMaximumTarget

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	string	Positioner name
UserMinimumTarget	double	User minimum travel limit (units)
UserMaximumTarget	double	User maximum travel limit (units)

Output parameters (None)

Return

Error	integer	TCL error code (0=success or 1=syntax error) or function error code

C / C++

Prototype

int **PositionerUserTravelLimitsSet** (int SocketID, char FullPositionerName[250] , double UserMinimumTarget, double UserMaximumTarget)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	char *	Positioner name
UserMinimumTarget	double	User minimum travel limit (units)
UserMaximumTarget	double	User maximum travel limit (units)

Output parameters (None)

Return

Error	int	Function error code

VISUAL BASIC

Prototype

Long **PositionerUserTravelLimitsSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal UserMinimumTarget As Double, ByVal UserMaximumTarget As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	String	Positioner name
UserMinimumTarget	Double	User minimum travel limit (units)
UserMaximumTarget	Double	User maximum travel limit (units)

Output parameters

None

Return

Error	Long	Function error code

MATLAB

Prototype

[Error] **PositionerUserTravelLimitsSet** (int32 SocketID, cstring FullPositionerName, double UserMinimumTarget, double UserMaximumTarget)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName	cstring	Positioner name
UserMinimumTarget	double	User minimum travel limit (units)
UserMaximumTarget	double	User maximum travel limit (units)

Return

Error	int32	Function error code

PYTHON

Prototype

[Error] **PositionerUserTravelLimitsSet** (integer SocketID, string FullPositionerName, double UserMinimumTarget, double UserMaximumTarget)

Input parameters

SocketID .....integer .....Socket identifier got from “TCP_ConnectToServer” function
FullPositionerName .....string .....Positioner name
UserMinimumTarget.....double.....User minimum travel limit (units)
UserMaximumTarget.....double.....User maximum travel limit (units)

Return

Error.....integer .....Function error code

2.2.5. Configuration files

Two configuration files are used by the controller: “System.ini” and “Stages.ini”. These configuration files are read during the booting of the controller.

1. The **system.ini** file specifies the system configuration and defines the used motion groups.

2. The **stages.ini** file defines the stage parameters for all positioners (see “ConfigurationWizard” document to know the stage parameters). The stages.ini file must at least include those positioners referenced to in the system.ini file, but it might also include positioners that are currently not used by the system.

System.ini file:

```
[GENERAL]
BootScriptFileName =
BootScriptArguments =                    ; the separator is the comma


[GROUPS]
SingleAxisInUse = MyGROUP                ; the separator is the comma
SpindleInUse =
XYInUse =
XYZInUse =
MultipleAxesInUse =


[MyGROUP]
PositionerInUse =  MyPOSITIONER          ; the separator is the comma


[MyGROUP. MyPOSITIONER]
PlugNumber =
PositionerName = MySTAGE                 ; see “ stages.ini” file
```

Stages.ini file:

```
[MySTAGE]

;--- STAGE
SmartStageName =                    ; Smart stage name

;--- DRIVER
DriverName =      ; XPS-DRV00 (pass through board)
                  ; XPS-DRV00P (driver for external driver)
                  ; XPS-DRV01 (driver for DC servo and stepper motors)
                  ; XPS-DRV02 (driver for 3-phase DC brushless motors)
                  ; XPS-DRV03
                  ; XPS-DRVMx (driver for dedicated motors : x = 1 to 5)
                  ; XPS-DRVP1 (driver for piezo motors)

; If DriverName = XPS-DRV01 driver
    ;  If  MotorDriverInterface = AnalogVelocity
    DriverPWMFrequency =
    DriverErrorAmplifierGain =
    DriverTachometerGain=

    ;  If  MotorDriverInterface = AnalogVoltage
    PWMFrequency =

    ;  If  MotorDriverInterface = AnalogStepper
    PWMFrequency =
    DriverStepperWinding=

; If DriverName = XPS-DRVMx driver   (x = 1 to 5)
DriverBridgeFreeWheel =
DriverBrake =

; If DriverName = XPS-DRV02 driver
DriverMotorResistance =
DriverMotorInductance =
DriverCutOffFrequency =
DriverMaximumPeakCurrent =
DriverMaximumRMSCurrent =
DriverRMSIntegrationTime =
DriverThermistanceThreshold =

; If DriverName = XPS-DRV03 driver
    ;  If  MotorDriverInterface = AnalogAcceleration
    DriverMotorResistance =
    DriverMotorInductance =
    DriverCurrentCutOffFrequency =
    DriverMaximumPeakCurrent =
    DriverMaximumRMSCurrent =
    DriverRMSIntegrationTime =
    DriverMaximumMotorVoltage =
    ;  If  MotorDriverInterface = AnalogVoltage
    DriverMaximumRMSCurrent =
    DriverRMSIntegrationTime =
    ;  If  MotorDriverInterface = AnalogVelocity
    DriverMotorResistance =
    DriverMotorInductance =
    DriverCurrentCutOffFrequency =
    DriverMaximumRMSCurrent =
    DriverRMSIntegrationTime =
```

```
    DriverMaximumMotorVoltage =
    DriverVelocityCutOffFrequency =
    DriverMotorVoltageConstant =
    DriverTachoGeneratorVoltage =
    DriverStageInertia =
    DriverGearRatio =

; If DriverName = XPS-DRVP1  driver
    DriverNotchFrequency =
    DriverNotchBandwidth =
    DriverNotchGain =
    DriverLowpassFrequency =
    DriverKI =
    DriverFatalFollowingError =
    DriverStagePositionOffset =
    DriverTravelCorrection =


;--- MOTOR DRIVER INTERFACE
MotorDriverInterface =      ; AnalogStepperPosition
                            ; AnalogVelocity
                            ; AnalogVoltage
                            ; AnalogAcceleration
                            ; AnalogSin60Acceleration
                            ; AnalogSin90Acceleration
                            ; AnalogSin120Acceleration
                            ; AnalogDualSin60Acceleration
                            ; AnalogDualSin90Acceleration
                            ; AnalogDualSin120Acceleration
                            ; AnalogPosition
                            ; AnalogPositionPiezo
                            ; PulseDir
                            ; PulsePulse
                            ; AnalogSin60AccelerationLMI
                            ; AnalogSin90AccelerationLMI
                            ; AnalogSin120AccelerationLMI

; If MotorDriverInterface = AnalogVelocity
ScalingVelocity =                   ; unit / s
VelocityLimit =                     ; unit / s

; If MotorDriverInterface = AnalogAcceleration
ScalingAcceleration =               ; unit / s²
AccelerationLimit =                 ; unit / s²

; If MotorDriverInterface = AnalogVoltage
MaximumCurrent =                    ; amps
VoltageLimit =                      ; volts

; If MotorDriverInterface = AnalogPosition
MinimumTargetPositionVoltage =      ; volts
MaximumTargetPositionVoltage =      ; volts

; If MotorDriverInterface = AnalogPositionPiezo
                                    ; No parameter

; If MotorDriverInterface = AnalogStepperPosition
DisplacementPerFullStep =           ; units
ScalingCurrent =                    ; amps for 10 volts
PeakCurrentPerPhase =               ; amps
```

```
StandbyPeakCurrentPerPhase =                ; amps
BaseVelocity =                              ; units / s

; If MotorDriverInterface = PulseDir or PulsePulse
DigitalStepperPulseLogic =                  ; Positive or Negative
DigitalStepperDirectionLogic =              ; Positive or Negative
DisplacementPerFullStep =                   ; units
BaseVelocity =                              ; units / s

; If MotorDriverInterface = AnalogSin …
ScalingAcceleration =                       ; unit / s²
AccelerationLimit =                         ; unit / s²
MagneticTrackPeriod =                       ; units
InitializationAccelerationLevel =           ; percent (LMI)
InitializationCycleDuration =               ; seconds (LMI)

; If MotorDriverInterface = AnalogDualSin …
ScalingAcceleration =                       ; unit / s²
AccelerationLimit =                         ; unit / s²
MagneticTrackPeriod =                       ; units
InitializationAccelerationLevel =           ; percent
InitializationCycleDuration =               ; seconds
FirstMotorForceBalance =
SecondMotorForceBalance =

;--- ENCODER
EncoderType =      ; AquadB
                   ; AnalogInterpolated

; If EncoderType = AquadB
EncoderResolution =                         ; units

; If EncoderType = AnalogInterpolated
EncoderZMPlug =      ; Driver
                     ; Encoder
EncoderResolution =                         ; units
EncoderInterpolationFactor =
EncoderScalePitch =                         ; units
EncoderADC1Offset =                         ; volts
EncoderADC2Offset =                         ; volts
EncoderPhaseCompensation =                  ; deg
EncoderDifferentialGain =

;--- BACKLASH
Backlash =                                  ; unit  (0 = not activated)

;--- POSITIONER MAPPING
LinearEncoderCorrection =                   ; ppm
PositionerMappingFileName =

; If PositionerMappingFileName is defined then the mapping is enabled and must be configured :
PositionerMappingLineNumber =
PositionerMappingMaxPositionError =

;--- TRAVELS
; If ServitudesType = Spindle
    HomePreset =                            ; units
    SpindlePeriod =                         ; units
; If ServitudesType = Piezo
                                            ; No parameter
```

```
; else
    MinimumTargetPosition =                 ; units
    HomePreset =                            ; units
    MaximumTargetPosition =                 ; units

;--- PROFILE GENERATOR
MaximumVelocity =                           ; units / second
MaximumAcceleration =                       ; units / second²
EmergencyDecelerationMultiplier =
MinimumJerkTime =                           ; seconds
MaximumJerkTime =                           ; seconds
TrackingCutOffFrequency =                   ; Hz

;--- HOME
HomeSearchSequenceType =    ; MechanicalZeroAndIndexHomeSearch
                            ; MechanicalZeroHomeSearch
                            ; MinusEndOfRunAndIndexHomeSearch
                            ; MinusEndOfRunHomeSearch
                            ; PlusEndOfRunHomeSearch
                            ; IndexHomeSearch
                            ; CurrentPositionAsHome
HomeSearchMaximumVelocity =                 ; units / second
HomeSearchMaximumAcceleration =             ; units / second²
HomeSearchTimeout =                         ; seconds

;--- CORRECTOR
CorrectorType =     ; PIDFFAcceleration     => MotorDriverInterface « Acceleration »
                    ; PIDFFVelocity         => MotorDriverInterface « Velocity »
                    ; PIDDualFFVoltage      => MotorDriverInterface « Voltage »
                    ; PIPosition            => MotorDriverInterface « Position »
                    ; NoEncoderPosition     => MotorDriverInterface « Position »

; If   CorrectorType is PIDFFAcceleration
KP =                                        ; 1 / seconds²
KI =                                        ; 1 / seconds²
KD =                                        ; 1 / seconds²
KS =
IntegrationTime =                           ; seconds
DerivativeFilterCutOffFrequency =           ; Hertz
GKP =
GKD =
GKI =
KForm =                                     ; units
KFeedforwardAcceleration =
ClosedLoopStatus =  ; Opened or Closed
FatalFollowingError =                       ; units
DeadBandThreshold =                         ; units

; If CorrectorType is PIDFFVelocity
KP =                                        ; 1 / seconds
KI =                                        ; 1 / seconds²
KD =
KS =
IntegrationTime =                           ; seconds
DerivativeFilterCutOffFrequency =           ; Hertz
GKP =
GKD =
GKI =
KForm =                                     ; units
KFeedforwardVelocity =
```

```
ClosedLoopStatus =  ; Opened or Closed
FatalFollowingError =                   ; units
DeadBandThreshold =                     ; units

; If CorrectorType is PIDDualFFVoltage
KP =                                    ; volts / units
KI =                                    ; volts / units / seconds
KD =                                    ; volts * seconds / units
KS =
IntegrationTime =                       ; seconds
DerivativeFilterCutOffFrequency =       ; Hertz
GKP =
GKD =
GKI =
KForm =                                 ; units
KFeedforwardAcceleration =              ; volts / (units / seconds²)
KFeedforwardVelocity =                  ; volts / (units / seconds)
KFeedforwardVelocityOpenLoop =          ;
Friction =                              ; volts
ClosedLoopStatus =  ; Opened or Closed
FatalFollowingError =                   ; units
DeadBandThreshold =                     ; units

; If CorrectorType is PIPosition
KP =
KI =                                    ; 1 / seconds
KD =                                    ; seconds
KS =
IntegrationTime =                       ; seconds
DerivativeFilterCutOffFrequency =       ; Hertz
GKP =
GKD =
GKI =
KForm =                                 ; units
ClosedLoopStatus =  ; Opened or Closed
FatalFollowingError =                   ; units
DeadBandThreshold =                     ; units

;--- NOTCH FILTER
NotchFrequency1 =                       ; Hertz  (0 = not activated)
NotchBandwith1 =                        ; Hertz
NotchGain1 =
NotchFrequency2 =                       ; Hertz (0 = not activated)
NotchBandwith2 =                        ; Hertz
NotchGain2 =

;--- GATHERING FILTERS
CurrentVelocityCutOffFrequency =        ; Hertz
CurrentAccelerationCutOffFrequency =    ; Hertz

;--- MOTION DONE
MotionDoneMode =      ; Theoretical
                      ; VelocityAndPositionWindow

; If MotionDoneMode = VelocityAndPositionWindow
MotionDonePositionThreshold =       ; units
MotionDoneVelocityThreshold =       ; units / second
MotionDoneCheckingTime =            ; seconds
MotionDoneMeanPeriod =              ; seconds
MotionDoneTimeout =                 ; seconds
```

```
;--- SERVITUDES
ServitudesType =    ; StandardEORDriverPlug
                    ; StandardEOREncoderPlug
                    ; Spindle
                    ; Piezo
```

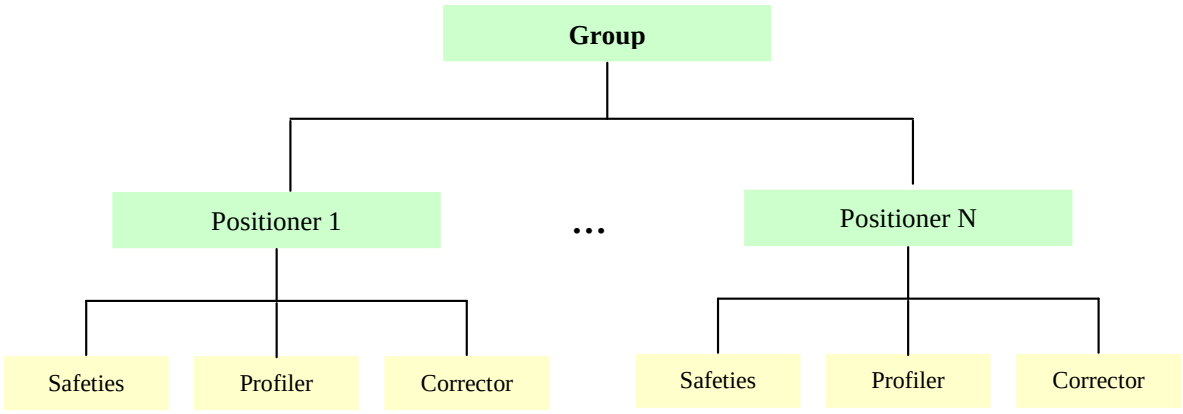
2.3. Group

2.3.1. Description

The “Group” objects are used to define one or several “positioners” in a same motion group. The availabled motion groups are defined is the section [GROUPS] in the system.ini file and the group types are:

1. **SingleAxis** (1 positioner) / “**Gantry” SingleAxis** (2 positioners)
2. **Spindle** (1 positioner)
3. **XY** (2 positioners) / “**Gantry” XY** (3 or 4 positioners)
4. **XYZ** (3 positioners)
5. **MultipleAxes** (1 to 8 positioners)

2.3.2. Object structure



A motion “**Group**” is built in relation to a **group type** (SingleAxis, Spindle, XY, XYZ or MultipleAxes).
A group is defined by a **group name**.

To defined a new group see §2.2.5 (configuration file).

NOTE:
The maximum number of positioners in a same group is limited to **8**.

2.3.3. Function description

2.3.3.1. GroupAccelerationSetpointGet

NAME

GroupAccelerationSetpointGet – Returns the setpoint acceleration for one or all positioners of the selected group.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid object type (group or positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Valid group name: ERR_GROUP_NAME (-19)
- Valid positioner name: ERR_POSITIONER_NAME (-18)
- Valid output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

Returns the setpoint acceleration for one or all positioners of the selected group.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

GroupAccelerationSetpointGet SocketID GroupName SetpointAcceleration …

Input parameters

SocketID ..... integer ..... Socket identifier gets by the “TCP_ConnectToServer” function
GroupName..... string ..... Group name (maximum size = 250)

Output parameters

SetpointAcceleration..... floating point..... Setpoint acceleration (must be repeated for each positioner of group)

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupAccelerationSetpointGet** (int SocketID, char *GroupName, int NbPositioners, double * SetpointAcceleration)

Input parameters

SocketID ..... int ..... Socket identifier gets by the “TCP_ConnectToServer”function
GroupName..... char * ..... Group name
NbPositioners..... int..... Number of positioners in the selected group

Output parameters

SetpointAcceleration..... double * ..... Setpoint Acceleration array

Return

Function error code

VISUAL BASIC

Prototype

Long **GroupAccelerationSetpointGet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal NbPositioners As Long, SetpointAcceleration As Double)

Input parameters

SocketID	Long	Socket identifier gets by the “TCP_ConnectToServer” function
GroupName	String	Group name
NbPositioners	Long	Number of positioners in the selected group

Output parameters

SetpointAcceleration	Double	Setpoint Acceleration array

Return

Function error code

MATLAB

Prototype

[Error, SetpointAcceleration] **GroupAccelerationSetpointGet** (int32 SocketID, cstring GroupName)

Input parameters

SocketID	int32	Socket identifier gets by the “TCP_ConnectToServer”function
GroupName	cstring	Group name

Return

Error	int32	Function error code
SetpointAcceleration	doublePtr	Setpoint Acceleration array

PYTHON

Prototype

[Error, SetpointAcceleration] **GroupAccelerationSetpointGet** (integer SocketID, string GroupName)

Input parameters

SocketID	integer	Socket identifier gets by the “TCP_ConnectToServer”function
GroupName	string	Group name

Return

Error	integer	Function error code
SetpointAcceleration	doublePtr	Setpoint Acceleration array

2.3.3.2. GroupAnalogTrackingModeDisable

NAME

GroupAnalogTrackingModeDisable - Exits the analog tracking mode.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid group name: ERR_GROUP_NAME (-19)
- Group status must be “ANALOG TRACKING”: ERR_NOT_ALLOWED_ACTION (-22)

DESCRIPTION

Disables the analog tracking mode. The group exits the “ANALOG TRACKING” state to come back to the “READY” state. If the group state is not “ANALOG TRACKING”, the ERR_NOT_ALLOWED_ACTION (-22) error is returned.

NOTE:
The tracking mode allows to interpret ADC value as a position command or as a velocity command.
To enable the analog tracking mode used the “GroupAnalogTrackingModeEnable” function.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype

GroupAnalogTrackingModeDisable $SocketID $GroupName

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... SingleAxis group name (maximum size = 250)

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupAnalogTrackingModeDisable** (int SocketID, char *GroupName)

Input parameters

SocketID ... int ..... Socket identifier got from “TCP_ConnectToServer” function
GroupNamechar * ..... SingleAxis group name

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **GroupAnalogTrackingModeDisable** (ByVal SocketID As Long, ByVal GroupName As String)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... String ..... SingleAxis group name

Output parameters

None

Return

Function error code

MATLAB

Prototype

int32 **GroupAnalogTrackingModeDisable** (int32 SocketID, cstring GroupName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... cstring ..... SingleAxis group name

Return

Function error code

PYTHON

Prototype

integer **GroupAnalogTrackingModeDisable** (integer SocketID, string GroupName)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... string ..... SingleAxis group name

Return

Function error code

2.3.3.3. GroupAnalogTrackingModeEnable

NAME

GroupAnalogTrackingModeEnable - Exit the analog tracking mode.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Group status must be “READY”: ERR_NOT_ALLOWED_ACTION (-22)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid group name: ERR_GROUP_NAME (-19)
- Valid tracking type (“Position” or “Velocity”): ERR_WRONG_OBJECT_TYPE (-8)
- Configured tracking: ERR_NOT_ALLOWED_ACTION (-22)

DESCRIPTION

Enables the analog tracking mode. To use this function, the group must be in READY state and the tracking must be configured before, else an error ERR_NOT_ALLOWED_ACTION (-22) is returned.

Once the tracking mode is enabled, the group status must be “ANALOG TRACKING” (48 : Analog tracking state due to a TrackingEnable command).

“Position” analog tracking
In case of “Position” tracking type, the analog input is interpreted as a position command. The parameters must be settled by the “AnalogTrackingPositionParametersSet” function and can be read by the “AnalogTrackingPositionParametersGet” function.

“Velocity” analog tracking
In case of “Velocity” tracking type, the analog input is interpreted as a velocity command. The parameters must be settled by the “AnalogTrackingVelocityParametersSet” function and can be read by the “AnalogTrackingVelocityParametersGet” function.

NOTE:
To disable the analog tracking mode used the “GroupAnalogTrackingModeDisable” function.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_OBJECT_TYPE (-8)
SUCCESS (0) : no error

TCL

Prototype
GroupAnalogTrackingModeEnable $SocketID $GroupName $Type

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	SingleAxis group name (maximum size = 250)
Type	string	Tracking type (“Position” or “Velocity”)

Output parameters
None

Return
TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupAnalogTrackingModeEnable** (int SocketID, char *GroupName, char *Type)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
GroupName	char *	SingleAxis group name
Type	char *	Tracking type (“Position” or “Velocity”)

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **GroupAnalogTrackingModeEnable** (ByVal SocketID As Long, ByVal GroupName As String, ByVal Type As String)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GroupName	String	SingleAxis group name
Type	String	Tracking type (“Position” or “Velocity”)

Output parameters

None

Return

Function error code

MATLAB

Prototype

int32 **GroupAnalogTrackingModeEnable** (int32 SocketID, cstring GroupName, cstring Type)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupName	cstring	SingleAxis group name
Type	cstring	Tracking type (“Position” or “Velocity”)

Return

Function error code

PYTHON

Prototype

integer **GroupAnalogTrackingModeEnable** (integer SocketID, string GroupName, string Type)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	SingleAxis group name
Type	string	Tracking type (“Position” or “Velocity”)

Return

Function error code

2.3.3.4. GroupCorrectorOutputGet

NAME

GroupCorrectorOutputGet – Returns corrector output for one or for all positioners of the selected group.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Valid output parameter: ERR_WRONG_TYPE_DOUBLE (-14)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)

DESCRIPTION

Returns corrector output for one or for all positioners of the selected group.

The input parameter “group name” can be a positioner name.
For a group, this function returns the corrector output for each positioner from the selected group.
For a positioner, this function returns only the corrector output associated to the selected positioner.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

GroupCorrectorOutputGet $SocketID $GroupName CorrectorOutput …

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Group name or Positioner name (maximum size = 250)

Output parameters

CorrectorOutput ..... floating point..... Corrector output, must be repeated for each positioner of group

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupCorrectorOutputGet** (int SocketID, char *GroupName, int NbPositioners, double * CorrectorOutputs)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... char * ..... Group name or Positioner name
NbPositioners..... int..... Number of positioners in the selected group (1 if positioner)

Output parameters

CorrectorOutputs ..... double * ..... Corrector output array

Return

Function error code

Note : *CorrectorOutputs* must have array size in accordance to *NbPositioners* to avoid array overrun.

VISUAL BASIC

Prototype

Long **GroupCorrectorOutputGet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal NbPositioners As Long, CorrectorOutputs As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GroupName	String	Group name or Positioner name
NbPositioners	Long	Number of positioners in the selected group (1 if positioner)

Output parameters

CorrectorOutputs	Double	Corrector output array

Return

Function error code

Note : *CorrectorOutputs* must have array size in accordance to *NbPositioners* to avoid array overrun.

MATLAB

Prototype

[Error, CorrectorOutput] **GroupCorrectorOutputGet** (int32 SocketID, cstring GroupName, int32 NbPositioners)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupName	cstring	Group name or Positioner name
NbPositioners	int32	Number of positioners in the selected group (1 if positioner)

Return

Error	int32	Function error code
CorrectorOutput	doublePtr	Corrector output array

Note : *CorrectorOutputs* must have array size in accordance to *NbPositioners* to avoid array overrun.

PYTHON

Prototype

[Error, CorrectorOutput] **GroupCorrectorOutputGet** (integer SocketID, string GroupName, integer NbPositioners)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	Group name or Positioner name
NbPositioners	integer	Number of positioners in the selected group (1 if positioner)

Return

Error	integer	Function error code
CorrectorOutputs	double list	List of corrector outputs

Note : *CorrectorOutputs* has the elements number equel to *NbPositioners*.

2.3.3.5. GroupCurrentFollowingErrorGet

NAME

GroupCurrentFollowingErrorGet – Returns the current following error for one or all positioners of the selected group.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid object type (group or positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Valid group name: ERR_GROUP_NAME (-19)
- Valid positioner name: ERR_POSITIONER_NAME (-18)
- Valid output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

Returns the current following error for one or all positioners of the selected group.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

GroupCurrentFollowingErrorGet SocketID GroupName CurrentFollowing error …

Input parameters

SocketID	integer	Socket identifier gets by the “TCP_ConnectToServer” function
GroupName	string	Group name (maximum size = 250)

Output parameters

CurrentFollowingError	floating point	Current following error (must be repeated for each positioner of group)

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupCurrentFollowingErrorGet** (int SocketID, char *GroupName, int NbPositioners, double * CurrentFollowingError)

Input parameters

SocketID	int	Socket identifier gets by the “TCP_ConnectToServer”function
GroupName	char *	Group name
NbPositioners	int	Number of positioners in the selected group

Output parameters

CurrentFollowingError	double *	Current following error array

Return

Function error code

VISUAL BASIC

Prototype

Long **GroupCurrentFollowingErrorGet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal NbPositioners As Long, CurrentFollowingError As Double)

Input parameters

SocketID	Long	Socket identifier gets by the “TCP_ConnectToServer” function
GroupName	String	Group name
NbPositioners	Long	Number of positioners in the selected group

Output parameters

CurrentFollowingError	Double	Current following error array

Return

Function error code

MATLAB

Prototype

[Error, CurrentFollowingError] **GroupCurrentFollowingErrorGet** (int32 SocketID, cstring GroupName)

Input parameters

SocketID	int32	Socket identifier gets by the “TCP_ConnectToServer”function
GroupName	cstring	Group name

Return

Error	int32	Function error code
CurrentFollowingError	doublePtr	Current following error array

PYTHON

Prototype

[Error, CurrentFollowingError] **GroupCurrentFollowingErrorGet** (integer SocketID, string GroupName)

Input parameters

SocketID	integer	Socket identifier gets by the “TCP_ConnectToServer”function
GroupName	string	Group name

Return

Error	integer	Function error code
CurrentFollowingError	doublePtr	Current following error array

2.3.3.6. GroupInitialize

NAME

GroupInitialize - Initializes the motor and activates the servo loop of the selected group.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Group status must be "Not initialized": ERR_NOT_ALLOWED_ACTION (-22)
- Actor must be a group: ERR_WRONG_OBJECT_TYPE (-8), ERR_POSITIONER_NAME (-18)
- Valid group name: ERR_GROUP_NAME (-19)

DESCRIPTION

The selected group must be in "NOT INITIALIZED" state, else the ERR_NOT_ALLOWED_ACTION (-22) error is returned.

This function begins to check the positioner error. If an error is detected, the hardware status register is reset (motor on) and the positioner error is cleared before to check it again. If a positioner error is always present, the motor is turn off, the ERR_POSITIONER_ERROR (-5) error is returned and the group is "NOT INITIALIZED".

If no positioner error then the group status becomes "MOTOR_INIT". The master-slave error is cleared, the encoder is preset (update encoder position) and the user travel limits are checked. If a travel limit error is detected then the motor is turn off, the error ERR_TRAVEL_LIMITS (-35) is returned and the group is "NOT INITIALIZED".

If no error, the motor is initialized in case of "AnalogSinAcc" or "AnalogDualSinAcc". The error ERR_MOTOR_INITIALIZATION_ERROR (-50) is returned if the initialization failed and the group is "NOT INITIALIZED".

If successful, the positions are preset, the servo loop is activated and the motor is on. The group is now "NOT REFERENCED".

NOTE:
In Master-Slave mode, after an emergency stop, the master group and the slave group are in "Not Initialized" status. To restart a master-slave relation the slave group(s) must be reinitialised **before** the master group.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_POSITIONER_ERROR (-5)
ERR_POSITIONER_NAME (-18)
ERR_MOTOR_INITIALIZATION_ERROR (-50)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_TRAVEL_LIMITS (-35)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype

GroupInitialize $SocketID $GroupName

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Group name (maximum size = 250)

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupInitialize** (int SocketID, char *GroupName)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... char * ..... Goup name

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **GroupInitialize** (ByVal SocketID As Long, ByVal GroupName As String)

Input parameters

SocketID ..... Long..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... String ..... Group name

Output parameters

None

Return

Function error code

MATLAB

Prototype

int32 **GroupInitialize** (int32 SocketID, cstring GroupName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... cstring ..... Group name

Return

Function error code

PYTHON

Prototype

integer **GroupInitialize** (integer SocketID, string GroupName)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Group name

Return

Function error code

2.3.3.7. GroupInitializeWithEncoderCalibration

NAME

GroupInitializeWithEncoderCalibration - Initializes motor, calibrates encoder and activates servo loop.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Group status must be "Not initialized": ERR_NOT_ALLOWED_ACTION (-22)
- Actor must be a group: ERR_WRONG_OBJECT_TYPE (-8), ERR_POSITIONER_NAME (-18)
- Valid group name: ERR_GROUP_NAME (-19)

DESCRIPTION

If the selected group is not in “NOT INITIALIZED” state, then the “ERR_NOT_ALLOWED_ACTION (-22)” error is returned by this function.

Initializes the motor, calibrates the encoder and activates the servo loop of each positioner of the selected group. To get the calibration results for each positioner, use the “PositionerEncoderCalibrationParametersGet” function.

This function begins to check the positioner error. If an error is detected, the hardware status register is reset (motor on) and the positioner error is cleared before to check it again. If a positioner error is always present, the motor is turn off, the ERR_POSITIONER_ERROR (-5) error is returned and the group is “NOT INITIALIZED”.

If no positioner error then the group status becomes “MOTOR_INIT”. The master-slave error is cleared, the encoder is preset (update encoder position) and the user travel limits are checked. If a travel limit error is detected then the motor is turn off, the error ERR_TRAVEL_LIMITS (-35) error is returned and the group is “NOT INITIALIZED”.

If no error, the motor is initialized in case of “AnalogSinAcc” or “AnalogDualSinAcc”. The ERR_MOTOR_INITIALIZATION_ERROR (-50) error is returned if the initialization failed and the group is “NOT INITIALIZED”.

After the group initialization, the encoder is calibrating and the group status becomes “ENCODER_CALIBRATING”. If a following error is occurred during the calibration, the ERR_FOLLOWING_ERROR (-25) error is returned and the group is “NOT INITIALIZED”.

If successful, the motor is initialized, the encoder is calibrated and the servo loop is activated. The group is now “NOT REFERENCED”.

NOTE:
In Master-Slave mode, after an emergency stop, the master group and the slave group are in “Not Initialized” status. To restart a master-slave relation the slave group(s) must be reinitialised **before** the master group.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_FOLLOWING_ERROR (-25)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_MOTOR_INITIALIZATION_ERROR (-50)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_POSITIONER_ERROR (-5)
ERR_POSITIONER_NAME (-18)
ERR_TRAVEL_LIMITS (-35)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype

GroupInitializeWithEncoderCalibration $SocketID $GroupName

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Group name (maximum size = 250)

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupInitializeWithEncoderCalibration** (int SocketID, char *GroupName)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... char * ..... Goup name

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **GroupInitializeWithEncoderCalibration** (ByVal SocketID As Long, ByVal GroupName As String)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... String ..... Group name

Output parameters

None

Return

Function error code

MATLAB

Prototype

int32 **GroupInitializeWithEncoderCalibration** (int32 SocketID, cstring GroupName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... cstring ..... Group name

Return

Function error code

PYTHON

Prototype

integer **GroupInitializeWithEncoderCalibration** (integer SocketID, string GroupName)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Group name

Return

Function error code

2.3.3.8. GroupHomeSearch

NAME

GroupHomeSearch - Initiates a home search.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Group status must be "Not referenced": ERR_NOT_ALLOWED_ACTION (-22)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid group name: ERR_GROUP_NAME (-19)

DESCRIPTION

This function initiates a home search for each positioner of the selected group.

The group must be initialized and the group must be in "NOT REFERENCED" state else this function returns the ERR_NOT_ALLOWED_ACTION (-22) error. If no error then the group status becomes "HOMING".

The home search can be failed due to:
- a following error: ERR_FOLLOWING_ERROR (-25)
- a ZM detection error: ERR_GROUP_HOME_SEARCH_ZM_ERROR (-49)
- a home search time out: ERR_GROUP_MOTION_DONE_TIMEOUT (-33)

For all these error cases, the group comes back to the "NOT INITIALIZED" state.

After the home search sequence, each positioner error is checked. If an error is detected, the hardware status register is reset (motor on) and the positioner error is cleared before to check it again. If a positioner error is always present, the ERR_TRAVEL_LIMITS (-35) error is returned and the group becomes "NOT INITIALIZED".

Once the home search is finished with success, the group must be in "READY" state.

NOTE:

- The home search routine for each positioner is defined in the *"stages.ini"* file by the "HomeSearchSequenceType" key.
- The home search time out is defined in the *"stages.ini"* file by the "HomeSearchTimeOut" key.
- The home search sequence is defined in the *"system.ini"* file by the "InitializationAndHomeSearchSequence" key for each group with several positioners.

  XY group
  The home search sequence can be "Together", "XthenY" or "YthenX" in a standard XY configuration.
  If the XY group is "Gantry" (dual positioner on X or on Y axis) only "XthenY" or "YthenX" are allowed.

  XYZ group
  The home search sequence can be "Together" or "XthenYthenZ".

  MultipleAxes group
  The home search sequence can be "Together" or "OneAfterAnother".

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_FOLLOWING_ERROR (-25)
ERR_GROUP_HOME_SEARCH_TIMEOUT (-28)
ERR_GROUP_HOME_SEARCH_ZM_ERROR (-49)
ERR_GROUP_MOTION_DONE_TIMEOUT (-33)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)

ERR_TRAVEL_LIMITS (-35)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype

GroupHomeSearch $SocketID $GroupName

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Group name (maximum size = 250)

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupHomeSearch** (int SocketID, char *GroupName)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... char * ..... Group name

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **GroupHomeSearch** (ByVal SocketID As Long, ByVal GroupName As String)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... String ..... Group name

Output parameters

None

Return

Function error code

MATLAB

Prototype

int32 **GroupHomeSearch** (int32 SocketID, cstring GroupName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... cstring ..... Group name

Return

Function error code

PYTHON

Prototype

integer **GroupHomeSearch** (integer SocketID, string GroupName)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Group name

Return

Function error code

2.3.3.9. GroupHomeSearchAndRelativeMove

NAME

GroupHomeSearchAndRelativeMove - Initiates a home search followed by a relative move.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Group status must be "Not referenced": ERR_NOT_ALLOWED_ACTION (-22)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid group name: ERR_GROUP_NAME (-19)
- Valid "Displacement" parameter: ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function initiates a home search followed by a relative move at the end of the home search.

The group must be initialized and the group must be in "NOT REFERENCED" state else this function returns the ERR_NOT_ALLOWED_ACTION (-22) error. If no error then the group status becomes "HOMING".

The home search sequence can be failed due to:

- a following error: ERR_FOLLOWING_ERROR (-25)
- a ZM detection error: ERR_GROUP_HOME_SEARCH_ZM_ERROR (-49)
- a home search time out: ERR_GROUP_MOTION_DONE_TIMEOUT (-33)

For all these error cases, the group comes back to the "NOT INITIALIZED" state.

Once the home search is realized, a relative move is executed. After this sequence without error, each positioner error is checked. If an error is detected, the hardware status register is reset (motor on) and the positioner error is cleared before to check it again. If a positioner error is always present, the ERR_TRAVEL_LIMITS (-35) error is returned and the group is "NOT INITIALIZED".

If the home search is successful, the group must be in "READY" state.

NOTE:

The home search routine for each positioner is defined in the *stages.ini* file by the "HomeSearchSequenceType" key.

The home search time out is defined in the *stages.ini* file by the "HomeSearchTimeOut" key.

The home search sequence is defined in the *system.ini* file by the "InitializationAndHomeSearchSequence" key for each group with several positioners:

XY group
The home search sequence can be "Together", "XthenY" or "YthenX" if the XY group is standard configuration. If the XY group is Gantry (dual positioner on X or on Y axis) only the "XthenY" or "YthenX" are allowed.

XYZ group
The home search sequence can be "Together" or "XthenYthenZ".

MultipleAxes group
The home search sequence can be "Together" or "OneAfterAnother".

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_FOLLOWING_ERROR (-25)
ERR_GROUP_NAME (-19)
ERR_GROUP_HOME_SEARCH_TIMEOUT (-28)
ERR_GROUP_HOME_SEARCH_ZM_ERROR (-49)
ERR_GROUP_MOTION_DONE_TIMEOUT (-33)

ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_TRAVEL_LIMITS (-35)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

GroupHomeSearchAndRelativeMove $SocketID $GroupName $Displacement …

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	Group name (maximum size = 250)
Displacement	floating point	Relative displacement (must be repeated for each positioner of group)

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupHomeSearchAndRelativeMove** (int SocketID, char *GroupName, int NbPositioners, double *Displacement)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
GroupName	char *	Group name
NbPositioners	int	Number of positioners in the selected group
Displacement	double*	Relative displacement array

Output parameters

None

Return

Function error code

Note : *Displacement* must have array size in accordance to *NbPositioners* to avoid array overrun.

VISUAL BASIC

Prototype

Long **GroupHomeSearchAndRelativeMove** (ByVal SocketID As Long, ByVal GroupName As String, ByVal NbPositioners As Long, Displacement As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GroupName	String	Group name
NbPositioners	Long	Number of positioners in the selected group
Displacement	Double	Relative displacement array

Output parameters

None

Return

Function error code

Note : *Displacement* must have array size in accordance to *NbPositioners* to avoid array overrun.

MATLAB

Prototype

int32 **GroupHomeSearchAndRelativeMove** (int32 SocketID, cstring GroupName, doublePtr Displacement)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupName	cstring	Group name
Displacement	doublePtr	Relative displacement array

Return

Function error code

Note : *Displacement* must have the same elements number as the positioners number of the group.

PYTHON

Prototype

integer **GroupHomeSearchAndRelativeMove** (integer SocketID, string GroupName, double Displacement)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	Group name
Displacement	double list	List of relative displacements

Return

Function error code

Note : *Displacement* must have the same elements number as the positioners number of the group.

2.3.3.10. GroupJogModeDisable

NAME

GroupJogModeDisable – Disables the jog mode *
- Not allowed for a spindle group -

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters [1]: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid object type (group): ERR_WRONG_OBJECT_TYPE (-8)
- Valid group name: ERR_GROUP_NAME (-19)
- Group status must be “JOGGING”: ERR_NOT_ALLOWED_ACTION (-22)

DESCRIPTION

Disables the Jog mode. To use this function, the group must be in the “JOGGING” state and all positioners must be idle (means velocity must be 0).

This function allows to exit the “JOGGING” state and to come back to the “READY” state. If the group state is not “JOGGING” or if the profiler velocity is not null then the error ERR_NOT_ALLOWED_ACTION (-22) is returned.

NOTE:
To enable the jog mode used the “GroupJogModeEnable” function.

CAUTION:
The jog mode can not be used with a spindle group.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype
GroupJogModeDisable $SocketID $GroupName

Input parameters
SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Group name (maximum size = 250)

Output parameters
None

Return
TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupJogModeDisable** (int SocketID, char *GroupName)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... char * ..... Group name

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **GroupJogModeDisable** (ByVal SocketID As Long, ByVal GroupName As String)

Input parameters

SocketID ..... Long..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... String ..... Group name

Output parameters

None

Return

Function error code

MATLAB

Prototype

int32 **GroupJogModeDisable** (int32 SocketID, cstring GroupName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... cstring ..... Group name

Return

Function error code

PYTHON

Prototype

integer **GroupJogModeDisable** (integer SocketID, string GroupName)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Group name

Return

Function error code

2.3.3.11. GroupJogModeEnable

NAME

GroupJogModeEnable – Enables the jog mode *- Not allowed for a spindle group -*

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Backlash must not be activated: ERR_NOT_ALLOWED_BACKLASH (-46)
- Number of command parameters [1]: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid object type (group): ERR_WRONG_OBJECT_TYPE (-8), ERR_POSITIONER_NAME (-18)
- Valid group name: ERR_GROUP_NAME (-19)
- Group status must be “READY”: ERR_NOT_ALLOWED_ACTION (-22)

DESCRIPTION

Enables the Jog mode. To use this function, the group must be in the “READY” state and all positioners must be idle (means velocity must be 0).

This function allows to go to the “JOGGING” state. If the group state is not “READY”, the ERR_NOT_ALLOWED_ACTION (-22) error is returned.

NOTE:
To disable the jog mode used the “GroupJogModeDisable” function.

CAUTION:
The jog mode can not be used with a spindle group.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_NOT_ALLOWED_BACKLASH (-46)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype
GroupJogModeEnable $SocketID $GroupName

Input parameters
SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Group name (maximum size = 250)

Output parameters
None

Return
TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupJogModeEnable** (int SocketID, char *GroupName)

Input parameters

SocketID ... int .....Socket identifier got from “TCP_ConnectToServer” function
GroupNamechar *.....Group name

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **GroupJogModeEnable** (ByVal SocketID As Long, ByVal GroupName As String)

Input parameters

SocketID .....Long.....Socket identifier got from “TCP_ConnectToServer” function
GroupName.....String .....Group name

Output parameters

None

Return

Function error code

MATLAB

Prototype

int32 **GroupJogModeEnable** (int32 SocketID, cstring GroupName)

Input parameters

SocketID .....int32.....Socket identifier got from “TCP_ConnectToServer” function
GroupName.....cstring .....Group name

Return

Function error code

PYTHON

Prototype

integer **GroupJogModeEnable** (integer SocketID, string GroupName)

Input parameters

SocketID .....integer.....Socket identifier got from “TCP_ConnectToServer” function
GroupName.....string.....Group name

Return

Function error code

2.3.3.12. GroupJogCurrentGet

NAME

GroupJogCurrentGet – Returns the current velocity and acceleration from the jog profiler.
- Not allowed for a spindle group -

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Valid object type (group or positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Valid group name: ERR_GROUP_NAME (-19)
- Valid positioner name: ERR_POSITIONER_NAME (-18)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function returns the current velocity and acceleration from the jog profiler for one positioner or for all positioners of the selected group.

So, this function must be called when the group is in “JOGGING” mode else the current velocity and the current acceleration will be null.

CAUTION:
The jog mode can not be used with a spindle group.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

GroupJogCurrentGet $SocketID $GroupName Velocity Acceleration …

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	Group name or Positioner name (maximum size = 250)

Output parameters

Velocity	floating point	Current velocity	must be repeated for each positioner of group
Acceleration	floating point	Current Acceleration	

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupJogCurrentGet** (int SocketID, char *GroupName, int NbPositioners, double *Velocity, double *Acceleration)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
GroupName	char *	Group name or Positioner name
NbPositioners	int	Number of positioners in the selected group (1 if a positioner)

Output parameters

Velocity	double *	Current velocity array
Acceleration	double *	Current Acceleration array

Return

Function error code

Note : *Velocity* and *Acceleration* must have array size in accordance to *NbPositioners* to avoid array overrun.

VISUAL BASIC

Prototype

Long **GroupJogCurrentGet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal NbPositioners As Long, Velocity As Double, Acceleration As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GroupName	String	Group name or Positioner name
NbPositioners	Long	Number of positioners in the selected group (1 if a positioner)

Output parameters

Velocity	Double	Current velocity array
Acceleration	Double	Current Acceleration array

Return

Function error code

Note : *Velocity* and *Acceleration* must have array size in accordance to *NbPositioners* to avoid array overrun.

MATLAB

Prototype

[Error, Velocity, Acceleration] **GroupJogCurrentGet** (int32 SocketID, cstring GroupName, int32 NbPositioners)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupName	cstring	Group name or Positioner name
NbPositioners	int32	Number of positioners in the selected group (1 if a positioner)

Return

Error	int32	Function error code
Velocity	doublePtr	Current velocity array
Acceleration	doublePtr	Current Acceleration array

Note : *Velocity* and *Acceleration* must have array size in accordance to *NbPositioners* to avoid array overrun.

PYTHON

Prototype

[Error, Velocity, Acceleration] **GroupJogCurrentGet** (integer SocketID, string GroupName, integer NbPositioners)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	Group name or Positioner name
NbPositioners	integer	Number of positioners in the selected group (1 if a positioner)

Return

Error	integer	Function error code	
Velocity	double	Current velocity	} repeated for each positioner of group
Acceleration	double	Current acceleration	

2.3.3.13. GroupJogParametersGet

NAME

GroupJogParametersGet – Returns the velocity and acceleration set by “GroupJogParametersSet”.
- Not allowed for a spindle group -

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid object type (group or positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Valid group name: ERR_GROUP_NAME (-19)
- Valid positioner name: ERR_POSITIONER_NAME (-18)
- Output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function returns the velocity and the acceleration set by the user to use the jog mode for one positioner or for all positioners of the selected group.

So, this function must be called when the group is in “JOGGING mode” else the velocity and the acceleration will be null.

To change on fly the velocity and the acceleration in the jog mode, call the “GroupJogParametersSet” function.

CAUTION:
The jog mode can not be used with a spindle group.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype
GroupJogParametersGet $SocketID $GroupName Velocity Acceleration …

Input parameters
SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Group name (maximum size = 250)

Output parameters
Velocity..... floating point..... User jog velocity
Acceleration ..... floating point..... User jog Acceleration
} must be repeated for each positioner

Return
TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupJogParametersGet** (int SocketID, char *GroupName, int NbPositioners, double *Velocity, double *Acceleration)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
GroupName	char *	Group name
NbPositioners	int	Number of positioners in the selected group

Output parameters

Velocity	double *	User jog velocity array
Acceleration	double *	User jog Acceleration array

Return

Function error code

Note : *Velocity* and *Acceleration* must have array size in accordance to *NbPositioners* to avoid array overrun.

VISUAL BASIC

Prototype

Long **GroupJogParametersGet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal NbPositioners As Long, Velocity As Double, Acceleration As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GroupName	String	Group name
NbPositioners	Long	Number of positioners in the selected group

Output parameters

Velocity	Double	User jog velocity array
Acceleration	Double	User jog Acceleration array

Return

Function error code

Note : *Velocity* and *Acceleration* must have array size in accordance to *NbPositioners* to avoid array overrun.

MATLAB

Prototype

[Error, Velocity, Acceleration] **GroupJogParametersGet** (int32 SocketID, cstring GroupName, int32 NbPositioners)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupName	cstring	Group name
NbPositioners	int32	Number of positioners in the selected group

Return

Error	int32	Function error code
Velocity	double array	User jog velocity array
Acceleration	double array	User jog acceleration array

Note : *Velocity* and *Acceleration* must have array size in accordance to *NbPositioners* to avoid array overrun.

PYTHON

Prototype

[Error, Velocity, Acceleration] **GroupJogParametersGet** (integer SocketID, string GroupName, integer NbPositioners)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	Group name
NbPositioners	integer	Number of positioners in the selected group

Return

Error	integer	Function error code	
Velocity	double	User jog velocity	repeated for each positioner of group
Acceleration	double	User jog Acceleration	

2.3.3.14. GroupJogParametersSet

NAME

GroupJogParametersSet – Changes “on the fly” the velocity and the acceleration in the jog mode.
- Not allowed for a spindle group -

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Group status must be “JOGGING”: ERR_NOT_ALLOWED_ACTION (-22)
- Valid object type (group or positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Valid group name: ERR_GROUP_NAME (-19)
- Valid positioner name: ERR_POSITIONER_NAME (-18)
- Input parameters for each positioner :
    1) Velocity > MaximumVelocity => Velocity = MaximumVelocity
    2) Velocity < -MaximumVelocity => Velocity = -MaximumVelocity
    3) Acceleration ≤ 0 => ERR_JOG_OUT_OF_RANGE (-42)
    4) Acceleration > MaximumAcceleration => Acceleration = MaximumAcceleration

DESCRIPTION

This function allows to change “on the fly” the velocity and the acceleration used by the jog mode. If an error occurs, each positioner stops and the velocity value is setting to zero.

To use this function, the jog mode must be enabled (requires call of the ”GroupJogModeEnable” function). If the group status is not “JOGGING” then an “ERR_NOT_ALLOWED_ACTION (-22)” error is returned.

If a slave error or a following error is detected during the jog setting then an “ERR_FOLLOWING_ERROR (-25)” or “ERR_SLAVE (-44)” error is returned. In this case, the motion is stopped, the jog mode is disabled and the group status becomes “DISABLE”.

CAUTION:
The jog mode can not be used with a spindle group.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_FOLLOWING_ERROR (-25)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_JOG_OUT_OF_RANGE (-42)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_POSITIONER_NAME (-18)
ERR_SLAVE (-44)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_OBJECT_TYPE (-8)
SUCCESS (0) : no error

TCL

Prototype

GroupJogParametersSet $SocketID $GroupName $Velocity $Acceleration

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Group name (maximum size = 250)
Velocity..... floating point..... user jog velocity } must be repeated for each positioner of group
Acceleration ..... floating point..... user jog Acceleration

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupJogParametersSet** (int SocketID, char *GroupName, int NbPositioners, double *Velocity, double *Acceleration)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... char * ..... Group name
NbPositioners..... int..... Number of positioners in the selected group
Velocity..... double * ..... user jog velocity array
Acceleration ..... double * ..... user jog Acceleration array

Output parameters

None

Return

Function error code

Note : *Velocity* and *Acceleration* must have array size in accordance to *NbPositioners* to avoid array overrun.

VISUAL BASIC

Prototype

Long **GroupJogParametersSet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal NbPositioners As Long, Velocity As Double, Acceleration As Double)

Input parameters

SocketID ..... Long..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... String ..... Group name
NbPositioners..... Long..... Number of positioners in the selected group
Velocity..... Double ..... user jog velocity array
Acceleration ..... Double ..... user jog Acceleration array

Output parameters

None

Return

Function error code

Note : *Velocity* and *Acceleration* must have array size in accordance to *NbPositioners* to avoid array overrun.

MATLAB

Prototype

[Error] **GroupJogParametersSet** (int32 SocketID, cstring GroupName, double array Velocity, double array Acceleration)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupName	cstring	Group name
Velocity	double array	user jog velocity array
Acceleration	double array	user jog acceleration array

Return

Error	int32	Function error code

Note : *Velocity* and *Acceleration* must have array size in accordance to the positioners number of the group.

PYTHON

Prototype

[Error] **GroupJogParametersSet** (integer SocketID, string GroupName, double list Velocity, double list Acceleration)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	Group name
Velocity	double list	user jog velocity
Acceleration	double list	user jog Acceleration

Return

Error	integer	Function error code

Note : *Velocity* and *Acceleration* must have elements number in accordance to the positioners number of the group.

2.3.3.15. GroupKill

NAME

GroupKill - Kills the selected group to go in the “not initialized” status.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid object type (group): ERR_WRONG_OBJECT_TYPE (-8), ERR_POSITIONER_NAME (-18)
- Valid group name: ERR_GROUP_NAME (-19)

DESCRIPTION

Kills the selected group to stop its action. The group comes back to “NOT INITIALIZED” state.

NOTE:
If an initialization, an encoder calibrating, a homing, a referencing, a moving or a trajectory is in progress, an “emergency stop” will be done. So, for each of these functions, an “ ERR_EMERGENCY_SIGNAL (-26)” error will be generated.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype
GroupKill $SocketID $GroupName

Input parameters
SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Group name (maximum size = 250)

Output parameters
None

Return
TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype
int **GroupKill** (int SocketID, char *GroupName)

Input parameters
SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... char * ..... Goup name

Output parameters
None

Return
Function error code

VISUAL BASIC

Prototype

Long **GroupKill** (ByVal SocketID As Long, ByVal GroupName As String)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... String ..... Group name

Output parameters

None

Return

Function error code

MATLAB

Prototype

int32 **GroupKill** (int32 SocketID, cstring GroupName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... cstring ..... Group name

Return

Function error code

PYTHON

Prototype

integer **GroupKill** (integer SocketID, string GroupName)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... string ..... Group name

Return

Function error code

2.3.3.16. GroupMotionDisable

NAME

GroupMotionDisable – Disables a “ready” group.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Group status must be "READY”: ERR_NOT_ALLOWED_ACTION (-22)
- Actor must be a group: ERR_WRONG_OBJECT_TYPE (-8), ERR_POSITIONER_NAME (-18)
- Valid group name: ERR_GROUP_NAME (-19)

DESCRIPTION

Turns OFF the motors, stops the corrector servo loop and disables the position compare mode if it’s active. The group status becomes “DISABLE”.

If the group is not in the “READY” status then an ERR_NOT_ALLOWED_ACTION (-22) error is returned.

NOTE:
In “DISABLED” status the encoder is still read.
To come back in “READY” status, call the “GroupMotionEnable” function.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype

GroupMotionDisable $SocketID $GroupName

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Group name (maximum size = 250)

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupMotionDisable** (int SocketID, char *GroupName)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
GroupName	char *	Goup name

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **GroupMotionDisable** (ByVal SocketID As Long, ByVal GroupName As String)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GroupName	String	Group name

Output parameters

None

Return

Function error code

MATLAB

Prototype

int32 **GroupMotionDisable** (int32 SocketID, cstring GroupName)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupName	cstring	Group name

Return

Function error code

PYTHON

Prototype

integer **GroupMotionDisable** (integer SocketID, string GroupName)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	Group name

Return

Function error code

2.3.3.17. GroupMotionEnable

NAME

GroupMotionEnable – Enables a “disabled” group to turn motor on and to restart corrector loops.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Group status must be "DISABLE”: ERR_NOT_ALLOWED_ACTION (-22)
- Actor must be a group: ERR_WRONG_OBJECT_TYPE (-8), ERR_POSITIONER_NAME (-18)
- Valid group name: ERR_GROUP_NAME (-19)

DESCRIPTION

Turns ON the motors and restarts the corrector servo loop. The group status becomes “READY”.
If the group is not in the “DISABLE” status then the “ERR_NOT_ALLOWED_ACTION (-22)” error is returned.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype

GroupMotionEnable $SocketID $GroupName

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Group name (maximum size = 250)

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupMotionEnable** (int SocketID, char *GroupName)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... char * ..... Goup name

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **GroupMotionEnable** (ByVal SocketID As Long, ByVal GroupName As String)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... String ..... Group name

Output parameters

None

Return

Function error code

MATLAB

Prototype

int32 **GroupMotionEnable** (int32 SocketID, cstring GroupName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... cstring ..... Group name

Return

Function error code

PYTHON

Prototype

integer **GroupMotionEnable** (integer SocketID, string GroupName)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... string ..... Group name

Return

Function error code

2.3.3.18. GroupMoveAbort

NAME

GroupMoveAbort – abort the motion or the jog in progress for a group or a positioner.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters [1]: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Group status must be "Not initialized": ERR_NOT_ALLOWED_ACTION (-22)
- Valid object type (group or positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Valid group name: ERR_GROUP_NAME (-19)
- Valid positioner name: ERR_POSITIONER_NAME (-18)

DESCRIPTION

This function allows to aborts a motion or a jog in progress. The group status must be “MOVING” or “JOGGING” else the “ERR_NOT_ALLOWED_ACTION (-22)” error is returned.

For a group:
If the group status is “MOVING”, this function stops all motion in progress.
If the group status is “JOGGING”, this function stops all “jog” motion in progress and disables the jog mode. After this “group move abort” action, the group status becomes “READY”.

For a positioner :
If the group status is “MOVING”, this function stops the motion in progress of the selected positioner.
If the group status is “JOGGING”, this function stops the “jog” motion in progress of the selected positioner.
If the positioner is idle, an ERR_NOT_ALLOWED_ACTION (-22) error is returned.
After this “positioner move abort” action, if all positioners are idle then the group status becomes “READY”, else the group stays in the same state.

NOTE:
If the “move abort” action failed, an ERR_GROUP_ABORT_MOTION (-27) error is returned.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_ABORT_MOTION (-27)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype

GroupMoveAbort $SocketID $GroupName

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Group name (maximum size = 250)

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype
int **GroupMoveAbort** (int SocketID, char *GroupName)

Input parameters
SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... char * ..... Goup name

Output parameters
None

Return
Function error code

VISUAL BASIC

Prototype
Long **GroupMoveAbort** (ByVal SocketID As Long, ByVal GroupName As String)

Input parameters
SocketID ..... Long..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... String ..... Group name

Output parameters
None

Return
Function error code

MATLAB

Prototype
int32 **GroupMoveAbort** (int32 SocketID, cstring GroupName)

Input parameters
SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... cstring ..... Group name

Return
Function error code

PYTHON

Prototype
integer **GroupMoveAbort** (integer SocketID, string GroupName)

Input parameters
SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Group name

Return
Function error code

2.3.3.19. GroupMoveAbsolute

NAME

GroupMoveAbsolute - Initiates an absolute move for a positioner or a group.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Group status must be "READY": ERR_NOT_ALLOWED_ACTION (-22)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid object type (group or positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Valid group name: ERR_GROUP_NAME (-19)
- Valid positioner name: ERR_POSITIONER_NAME (-18)
- Valid input parameter type: ERR_WRONG_TYPE_DOUBLE (-14)
- Verify target position in relation with the travel limits: ERR_PARAMETER_OUT_OF_RANGE (-17)
  TargetPosition $\geq$ MinimumTargetPosition
  TargetPosition $\leq$ MaximumTargetPosition

DESCRIPTION

Initiates an absolute move to one or all positioners of the selected group. The group state must be “READY” or “MOVING” else the ERR_NOT_ALLOWED_ACTION (-22) error is returned. If the group is “ready” then the group status becomes “MOVING”.

An absolute motion is defined by the distance between to the current position and the target position. If the current position is the same as the target position then no move will be done.

Each “positioner” move refers to the acceleration, velocity, minimumTjerkTime and maximumTjerkTime as defined in the “Stages.ini” file or as redefined by the “PositionerSGammaParametersSet” function.

If a slave error or a following error is detected during the moving then ERR_FOLLOWING_ERROR (-25) or ERR_SLAVE (-44) error is returned. In this case, the motion in progress is stopped and the group status becomes “DISABLE”.

If a “MotionDoneMode” is defined as “VelocityAndPositionWindowMotionDone” then an ERR_GROUP_MOTION_DONE_TIMEOUT (-33) error can be returned if the time out (defined by “MotionDoneTimeout” in the stages.ini file) is reached before the motion done.

If a “GroupMoveAbort” is done, an ERR_GROUP_ABORT_MOTION (-27) error is returned. In this case, the motion in progress is stopped and the group status becomes “READY”.

If a “GroupKill” command, an emergency brake or an emergency stop is occurred, an “ERR_EMERGENCY_SIGNAL (-26)” error is returned. In this case, the motion in progress is stopped and the group status becomes “NOT INITIALIZED”.

NOTE:
The asynchronous moves for positioners of a same group are possible through the use of different sockets to send the functions.

ERROR CODES

ERR_EMERGENCY_SIGNAL (-26)
ERR_FATAL_INIT (-20)
ERR_FOLLOWING_ERROR (-25)
ERR_GROUP_ABORT_MOTION (-27)
ERR_GROUP_MOTION_DONE_TIMEOUT (-33)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)

ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_POSITIONER_NAME (-18)
ERR_SLAVE (-44)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

GroupMoveAbsolute $SocketID $GroupName $TargetPosition …

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	Group name (maximum size = 250)
TargetPosition	floating point	Target position (must be repeated for each positioner of group)

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupMoveAbsolute** (int SocketID, char *GroupName, int NbPositioners, double *TargetPosition)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
GroupName	char *	Group name
NbPositioners	int	Number of positioners in the selected group
TargetPosition	double*	Target position array

Output parameters

None

Return

Function error code

Note : *TargetPosition* must have array size in accordance to *NbPositioners* to avoid array overrun.

VISUAL BASIC

Prototype

Long **GroupMoveAbsolute** (ByVal SocketID As Long, ByVal GroupName As String, ByVal NbPositioners As Long, TargetPosition As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GroupName	String	SingleAxis group name
NbPositioners	Long	Number of positioners in the selected group
TargetPosition	Double	Target position array

Output parameters

None

Return

Function error code

Note : *TargetPosition* must have array size in accordance to *NbPositioners* to avoid array overrun.

MATLAB

Prototype

int32 **GroupMoveAbsolute** (int32 SocketID, cstring GroupName, double TargetPosition)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupName	cstring	SingleAxis group name
TargetPosition	double array	Target position array

Return

Function error code

Note : *TargetPosition* must have the same elements number as the positioners number of the group.

PYTHON

Prototype

integer **GroupMoveAbsolute** (integer SocketID, string GroupName, double list TargetPosition)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	SingleAxis group name
TargetPosition	double list	Target position

Return

Function error code

Note : *TargetPosition* must have the same elements number as the positioners number of the group.

2.3.3.20. GroupMoveRelative

NAME

GroupMoveRelative - Initiates a relative move for a positioner or a group.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Group status must be "READY": ERR_NOT_ALLOWED_ACTION (-22)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid object type (group or positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Valid group name: ERR_GROUP_NAME (-19)
- Valid positioner name: ERR_POSITIONER_NAME (-18)
- Valid input parameter type: ERR_WRONG_TYPE_DOUBLE (-14)
- Verify target displacment in relation with the travel limits: ERR_PARAMETER_OUT_OF_RANGE (-17)
  TargetPosition $\geq$ MinimumTargetPosition
  TargetPosition $\leq$ MaximumTargetPosition

DESCRIPTION

Initiates a relative move defined by the target displacement to one or all positioners of the selected group. The group state must be “READY” or “MOVING” else the ERR_NOT_ALLOWED_ACTION (-22) error is returned. If the group is “ready” then the group status becomes “MOVING”.

The target displacement and the current position allows to define the new target position to reach:
NewTargetPosition = CurrentTargetPosition + TargetDisplacement

Each “positioner” move refers to the acceleration, velocity, minimumTjerkTime and maximumTjerkTime as defined in the “Stages.ini” file or as redefined by the “PositionerSGammaParametersSet” function.

If a slave error or a following error is detected during the moving then an “ERR_FOLLOWING_ERROR (-25)” or “ERR_SLAVE (-44)” error is returned. In this case, the motion in progress is stopped and the group status becomes “DISABLE”.

If a “MotionDoneMode” is defined as “VelocityAndPositionWindowMotionDone” then an “ERR_GROUP_MOTION_DONE_TIMEOUT (-33)” error can be returned if the time out (defined by “MotionDoneTimeout” in the stages.ini file) is reached before the motion done.

If a “GroupMoveAbort” is done, an “ERR_GROUP_ABORT_MOTION (-27)” error is returned. In this case, the motion in progress is stopped and the group status becomes “READY”.

If a “GroupKill” command, an emergency brake or an emergency stop is occurred, an “ERR_EMERGENCY_SIGNAL (-26)” error is returned. In this case, the motion in progress is stopped and the group status becomes “NOT INITIALIZED”.

NOTE:
The asynchronous moves for positioners of a same group are possible through the use of different sockets to send the functions.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_EMERGENCY_SIGNAL (-26)
ERR_FOLLOWING_ERROR (-25)
ERR_GROUP_ABORT_MOTION (-27)
ERR_GROUP_MOTION_DONE_TIMEOUT (-33)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)

ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_POSITIONER_NAME (-18)
ERR_SLAVE (-44)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

GroupMoveRelative $SocketID $GroupName $Displacement …

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	Group name (maximum size = 250)
Displacement	floating point	Relative displacement (must be repeated for each positioner of group)

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupMoveRelative** (int SocketID, char *GroupName, int NbPositioners, double *Displacement)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
GroupName	char *	Group name
NbPositioners	int	Number of positioners in the selected group
Displacement	double*	Relative displacement array

Output parameters

None

Return

Function error code

Note : *Displacement* must have array size in accordance to *NbPositioners* to avoid array overrun.

VISUAL BASIC

Prototype

Long **GroupMoveRelative** (ByVal SocketID As Long, ByVal GroupName As String, ByVal NbPositioners As Long, Displacement As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GroupName	String	SingleAxis group name
NbPositioners	Long	Number of positioners in the selected group
Displacement	Double	Relative displacement array

Output parameters

None

Return

Function error code

Note : *Displacement* must have array size in accordance to *NbPositioners* to avoid array overrun.

MATLAB

Prototype

int32 **GroupMoveRelative** (int32 SocketID, cstring GroupName, double Displacement)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupName	cstring	SingleAxis group name
Displacement	double	Relative displacement array

Return

Function error code

Note : *Displacement* must have the same elements number as the positioners number of the group.

PYTHON

Prototype

integer **GroupMoveRelative** (integer SocketID, string GroupName, double list Displacement)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	SingleAxis group name
Displacement	double list	Relative displacement

Return

Function error code

Note : *Displacement* must have the same elements number as the positioners number of the group.

2.3.3.21. GroupPositionCurrentGet

NAME

GroupPositionCurrentGet – Returns the current position for one or all positioners of the selected group.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid object type (group or positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Valid group name: ERR_GROUP_NAME (-19)
- Valid positioner name: ERR_POSITIONER_NAME (-18)
- Valid output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

Returns the current position for one or all positioners of the selected group.
The current position is defined as like:
CurrentPosition = SetpointPosition - FollowingError

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

GroupPositionCurrentGet $SocketID $GroupName CurrentPosition …

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	Group name (maximum size = 250)

Output parameters

CurrentPosition	floating point	Current Position (must be repeated for each positioner of group)

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupPositionCurrentGet** (int SocketID, char *GroupName, int NbPositioners, double * CurrentPosition)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
GroupName	char *	Group name
NbPositioners	int	Number of positioners in the selected group

Output parameters

CurrentPosition	double *	Current position array

Return

Function error code

Note : *CurrentPosition* must have array size in accordance to *NbPositioners* to avoid array overrun.

VISUAL BASIC

Prototype

Long **GroupPositionCurrentGet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal NbPositioners As Long, CurrentPosition As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GroupName	String	Group name
NbPositioners	Long	Number of positioners in the selected group

Output parameters

CurrentPosition	Double	Current position array

Return

Function error code

Note : *CurrentPosition* must have array size in accordance to *NbPositioners* to avoid array overrun.

MATLAB

Prototype

[Error, CurrentPosition] **GroupPositionCurrentGet** (int32 SocketID, cstring GroupName, int32 NbPositioners)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupName	cstring	Group name
NbPositioners	int32	Number of positioners in the selected group

Return

Error	int32	Function error code
CurrentPosition	double	Current position array

Note : *CurrentPosition* must have array size in accordance to *NbPositioners* to avoid array overrun.

PYTHON

Prototype

[Error, CurrentPosition] **GroupPositionCurrentGet** (integer SocketID, string GroupName, integer NbPositioners)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	Group name
NbPositioners	integer	Number of positioners in the selected group

Return

Error	integer	Function error code
CurrentPosition	double list	List of current positions

Note : *CurrentPosition* has the elements number equel to *NbPositioners*.

2.3.3.22. GroupPositionPCORawEncoderGet

NAME

GroupPositionPCORawEncoderGet – Returns the PCO raw encoders position for an XY group.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid object type: ERR_WRONG_OBJECT_TYPE (-8)
- Valid group type (must be a XY group): ERR_POSITIONER_NAME (-18)
- Valid group name: ERR_GROUP_NAME (-19)
- Valid output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function allows get the PCO raw encoder positions X and Y from the user corrected positions X and Y.

NOTE:
This function is only allowed with a XY group.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0): no error

TCL

Prototype

GroupPositionPCORawEncoderGet SocketID GroupName PositionX PositionY PCORawPositionX CorrectedPositionY

Input parameters

SocketID	integer	Socket identifier gets by the “TCP_ConnectToServer” function
GroupName	string	XY group name (maximum size = 250)
PositionX	floating point	User corrected position X
PositionY	floating point	User corrected position Y

Output parameters

PCORawPositionX	floating point	PCO Raw position X
PCORawPositionY	floating point	PCO Raw position Y

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupPositionPCORawEncoderGet** (int SocketID, char *GroupName, double PositionX, double PositionY, double * PCORawPositionX, double * PCORawPositionY)

Input parameters

SocketID	int	Socket identifier gets by the “TCP_ConnectToServer”function
GroupName	char *	XY group name
PositionX	double	User corrected position X
PositionY	double	User corrected position Y

Output parameters

PCORawPositionX	double *	PCO Raw position X
PCORawPositionY	double *	PCO Raw position Y

Return

Function error code

VISUAL BASIC

Prototype

Long **GroupPositionPCORawEncoderGet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal PositionX As Double, ByVal PositionY As Double, PCORawPositionX As Double, PCORawPositionY As Double)

Input parameters

SocketID	Long	Socket identifier gets by the “TCP_ConnectToServer” function
GroupName	String	XY group name
PositionX	Double	User corrected position X
PositionY	Double	User corrected position Y

Output parameters

PCORawPositionX	Double	PCO Raw position X
PCORawPositionY	Double	PCO Raw position Y

Return

Function error code

MATLAB

Prototype

[Error, PCORawPositionX, PCORawPositionY] **GroupPositionPCORawEncoderGet** (int32 SocketID, cstring GroupName, double PositionX, double PositionY)

Input parameters

SocketID	int32	Socket identifier gets by the “TCP_ConnectToServer”function
GroupName	cstring	XY group name
PositionX	double	User corrected position X
PositionY	double	User corrected position Y

Return

Error	int32	Function error code
PCORawPositionX	doublePtr	PCO Raw position X
PCORawPositionY	doublePtr	PCO Raw position Y

PYTHON

Prototype

[Error, PCORawPositionX, PCORawPositionY] **GroupPositionPCORawEncoderGet** (integer SocketID, string GroupName, double PositionX, double PositionY)

Input parameters

SocketID	integer	Socket identifier gets by the “TCP_ConnectToServer”function
GroupName	string	XY group name
PositionX	double	User corrected position X
PositionY	double	User corrected position Y

Return

Error	integer	Function error code
PCORawPositionX	doublePtr	PCO Raw position X
PCORawPositionY	doublePtr	PCO Raw position Y

2.3.3.23. GroupPositionSetpointGet

NAME

GroupPositionSetpointGet – Returns the setpoint position for one or all positioners of the selected group.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid object type (group or positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Valid group name: ERR_GROUP_NAME (-19)
- Valid positioner name: ERR_POSITIONER_NAME (-18)
- Valid output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

Returns the setpoint position for one or all positioners of the selected group.
The “setpoint” position is calculated by the profiler and represents the “theoretical” position to reach.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

GroupPositionSetpointGet $SocketID $GroupName SetpointPosition …

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	Group name (maximum size = 250)

Output parameters

SetpointPosition	floating point	Setpoint position (must be repeated for each positioner of group)

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupPositionSetpointGet** (int SocketID, char *GroupName, int NbPositioners, double * SetpointPosition)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
GroupName	char *	Group name
NbPositioners	int	Number of positioners in the selected group

Output parameters

SetpointPosition	double *	Setpoint position array

Return

Function error code

Note : *SetpointPosition* must have array size in accordance to *NbPositioners* to avoid array overrun.

VISUAL BASIC

Prototype

Long **GroupPositionSetpointGet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal NbPositioners As Long, SetpointPosition As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GroupName	String	Group name
NbPositioners	Long	Number of positioners in the selected group

Output parameters

SetpointPosition	Double	Setpoint position array

Return

Function error code

Note : *SetpointPosition* must have array size in accordance to *NbPositioners* to avoid array overrun.

MATLAB

Prototype

[Error, SetpointPosition] **GroupPositionSetpointGet** (int32 SocketID, cstring GroupName, int32 NbPositioners)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupName	cstring	Group name
NbPositioners	int32	Number of positioners in the selected group

Return

Error	int32	Function error code
SetpointPosition	double	Setpoint position array

Note : *SetpointPosition* must have array size in accordance to *NbPositioners* to avoid array overrun.

PYTHON

Prototype

[Error, SetpointPosition] **GroupPositionSetpointGet** (integer SocketID, string GroupName, integer NbPositioners)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	Group name
NbPositioners	integer	Number of positioners in the selected group

Return

Error	integer	Function error code
SetpointPosition	double list	List of setpoint positions

Note : *SetpointPosition* has the elements number equel to *NbPositioners*.

2.3.3.24. GroupPositionTargetGet

NAME

GroupPositionTargetGet – Returns the target position for one or all positioners of the selected group.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid object type (group or positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Valid group name: ERR_GROUP_NAME (-19)
- Valid positioner name: ERR_POSITIONER_NAME (-18)
- Valid output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

Returns the target position for one or all positioners of the selected group.

The target position represents the “end” position after the move.

For instance, during a move from 0 to 10 units, the position values are:
GroupPositionTargetGet => **10**
GroupPositionCurrentGet => **4.9995**
GroupPositionSetpointGet => **5**

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
ERR_WRONG_OBJECT_TYPE (-8)
SUCCESS (0) : no error

TCL

Prototype
GroupPositionTargetGet $SocketID $GroupName TargetPosition …

Input parameters
SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Group name (maximum size = 250)

Output parameters
TargetPosition..... floating point..... Target position (must be repeated for each positioner of group)

Return
TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupPositionTargetGet** (int SocketID, char *GroupName, int NbPositioners, double * TargetPosition)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... char * ..... Group name
NbPositioners..... int..... Number of positioners in the selected group

Output parameters

TargetPosition..... double * ..... Target position array

Return

Function error code

Note : *TargetPosition* must have array size in accordance to *NbPositioners* to avoid array overrun.

VISUAL BASIC

Prototype

Long **GroupPositionTargetGet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal NbPositioners As Long, TargetPosition As Double)

Input parameters

SocketID ..... Long..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... String ..... Group name
NbPositioners..... Long..... Number of positioners in the selected group

Output parameters

TargetPosition..... Double ..... Target position array

Return

Function error code

Note : *TargetPosition* must have array size in accordance to *NbPositioners* to avoid array overrun.

MATLAB

Prototype

[Error, TargetPosition] **GroupPositionTargetGet** (int32 SocketID, cstring GroupName, int32 NbPositioners)

Input parameters

SocketID ..... int32..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... cstring ..... Group name
NbPositioners..... int32..... Number of positioners in the selected group

Return

Error..... int32..... Function error code
TargetPosition..... double ..... Target position array

Note : *TargetPosition* must have array size in accordance to *NbPositioners* to avoid array overrun.

PYTHON

Prototype

[Error, TargetPosition] **GroupPositionTargetGet** (integer SocketID, string GroupName, integer NbPositioners)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Group name
NbPositioners..... integer ..... Number of positioners in the selected group

Return

Error	integer	Function error code
TargetPosition	double list	List of target positions

Note : *TargetPosition* has the elements number equel to *NbPositioners*.

2.3.3.25. GroupStatusGet

NAME

GroupStatusGet – Returns the group status code.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid output parameter type: ERR_WRONG_TYPE_INT (-15)
- Valid group name: ERR_GROUP_NAME (-19)

DESCRIPTION

Returns the group status code. The group status codes are listed in the “Group status list” § 2.19.
The description of the group status code can be get with the “GroupStatusStringGet” function.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

GroupStatusGet $SocketID $GroupName GroupStatus

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Group name (maximum size = 250)

Output parameters

GroupStatus ..... interger ..... State of the group.

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupStatusGet** (int SocketID, char *GroupName, int *GroupStatus)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... char * ..... Group name

Output parameters

GroupStatus ..... int * ..... Status of the group

Return

Function error code

VISUAL BASIC

Prototype

Long **GroupStatusGet** (ByVal SocketID As Long, ByVal GroupName As String, GroupStatus As Long)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... String ..... Group name

Output parameters

GroupStatus ..... Long ..... Status of the group

Return

Function error code

MATLAB

Prototype

[Error, GroupStatus] **GroupStatusGet** (int32 SocketID, cstring GroupName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... cstring ..... Group name

Return

Error..... int32 ..... Function error code
GroupStatus ..... int32 ..... Status of the group

PYTHON

Prototype

[Error, GroupStatus] **GroupStatusGet** (integer SocketID, string GroupName)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Group name

Return

Error..... integer ..... Function error code
GroupStatus ..... integer ..... Status of the group

2.3.3.26. GroupReferencingActionExecute

NAME

GroupReferencingActionExecute – Initiates the given action, with the given sensor and parameters.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid positioner name: ERR_POSITIONER_NAME (-18)
- Group status must be “NOT REFERENCED”: ERR_NOT_ALLOWED_ACTION (-22)
- Valid action name and sensor name: ERR_WRONG_OBJECT_TYPE (-8)
- Valid input parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

- Input parameter coherence: ERR_PARAMETER_OUT_OF_RANGE (-17)
  For a “LatchOnHighToLowTransition” or “LatchOnLowToHighTransition” or “LatchOnIndex” or “LatchOnIndexAfterSensorHighToLowTransition” or “MoveToPreviouslyLatchedPosition” action.
  Parameter $\leq$ MaximumVelocity
  Parameter $\neq 0$

- Referencing state: ERR_NOT_ALLOWED_ACTION (-22)
  Latch must be done since referencing start for a “MoveToPreviouslyLatchedPosition” action.

DESCRIPTION

Initiates a referencing action for a positioner. A referencing action is defined by a given action name (see action list), with a given sensor name (see sensor list) and parameters. For more detail, see XPS User’s manual referencing section.

Action list	**Sensor to define**
LatchOnHighToLowTransition	Yes
LatchOnIndex	None
LatchOnIndexAfterSensorHighToLowTransition	Yes
LatchOnLowToHighTransition	Yes
MoveRelative	None
MoveToPreviouslyLatchedPosition	None
SetPosition	None
SetPositionToHomePreset	None

Sensor list
MechanicalZero
MinusEndOfRun
PlusEndOfRun
None

If a following error is occurred during the referencing, an emergency brake is done if a motion is in progress and the ERR_FOLLOWING_ERROR (-25) error is returned. The group status becomes “NOT INITIALIZED”.

If the home search time out is reached, the ERR_GROUP_HOME_SEARCH_TIMEOUT (-28) error is returned. The group status becomes “NOT INITIALIZED”.

When the referencing is done, you can exit the “REFERENCING” state to go in “READY” state with the “GroupReferencingStop” function.

CAUTION:
This function must be only used with a **positioner**.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_FOLLOWING_ERROR (-25)
ERR_GROUP_HOME_SEARCH_TIMEOUT (-28)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

GroupReferencingActionExecute $SocketID $PositionerName $Action $Sensor $Parameter

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
PositionerName	string	Positioner name (maximum size = 250)
Action	string	Referencing action name
Sensor	string	Referencing sensor name
Parameter	floating point	Referencing parameter (related to the referencing action)

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupReferencingActionExecute** (int SocketID, char * PositionerName, char * Action, char * Sensor, double Parameter)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
PositionerName	char *	Positioner name
Action	char *	Referencing action name
Sensor	char *	Referencing sensor name
Parameter	double	Referencing parameter (related to the referencing action)

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **GroupReferencingActionExecute** (ByVal SocketID As Long, String PositionerName, ByVal Action As String, ByVal Sensor As String, ByVal Parameter As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
PositionerName	String	Positioner name
Action	String	Referencing action name
Sensor	String	Referencing sensor name
Parameter	Double	Referencing parameter (related to the referencing action)

Output parameters

None

Return

Function error code

MATLAB

Prototype

[Error] **GroupReferencingActionExecute** (int32 SocketID, cstring PositionerName, cstring Action, cstring Sensor, double Parameter)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
PositionerName	cstring	Positioner name
Action	cstring	Referencing action name
Sensor	cstring	Referencing sensor name
Parameter	double	Referencing parameter (related to the referencing action)

Return

Error	int32	Function error code

PYTHON

Prototype

[Error] **GroupReferencingActionExecute** (integer SocketID, string PositionerName, string Action, string Sensor, double Parameter)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
PositionerName	string	Positioner name
Action	string	Referencing action name
Sensor	string	Referencing sensor name
Parameter	double	Referencing parameter (related to the referencing action)

Return

Error	integer	Function error code

2.3.3.27. GroupReferencingStart

NAME

GroupReferencingStart – Starts the referencing mode.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid group name: ERR_GROUP_NAME (-19)
- Group status must be “NOT REFERENCED”: ERR_NOT_ALLOWED_ACTION (-22)

DESCRIPTION

Starts the referencing mode and sets the group status as “REFERENCING”.
To use this function, the selected group must be in “NOT REFERENCED” state, else the ERR_NOT_ALLOWED_ACTION (-22) error is returned.

To stop the referencing mode and to go in “READY” state, call the “GroupReferencingStop” function.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype

GroupReferencingStart $SocketID $GroupName

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Group name

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupReferencingStart** (int SocketID, char * GroupName)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... char * ..... Group name

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **GroupReferencingStart** (ByVal SocketID As Long, ByVal GroupName As String)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... String ..... Group name

Output parameters

None

Return

Function error code

MATLAB

Prototype

[Error] **GroupReferencingStart** (int32 SocketID, cstring GroupName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... cstring ..... Group name

Return

Error ..... int32 ..... Function error code

PYTHON

Prototype

[Error] **GroupReferencingStart** (integer SocketID, string GroupName)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... string ..... Group name

Return

Error ..... integer ..... Function error code

2.3.3.28. GroupReferencingStop

NAME

GroupReferencingStop – Stops the referencing mode.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid group name: ERR_GROUP_NAME (-19)
- Group status must be “REFERENCING”: ERR_NOT_ALLOWED_ACTION (-22)

DESCRIPTION

Stops the referencing mode and sets the group status as “READY”.
To use this function, the selected group must be in “REFERENCING” state, else the ERR_NOT_ALLOWED_ACTION (-22) error is returned.

The travel limits are checked before to stop referencing mode. The ERR_TRAVEL_LIMITS (-35) error is returned if the profiler position is out of range of the software travel limits and so the group stays in the “REFERENCING” state.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_TRAVEL_LIMITS (-35)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype
GroupReferencingStop $SocketID $GroupName

Input parameters
SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Group name

Output parameters
None

Return
TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype
int **GroupReferencingStop** (int SocketID, char * GroupName)

Input parameters
SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... char * ..... Group name

Output parameters
None

Return
Function error code

VISUAL BASIC

Prototype
Long **GroupReferencingStop** (ByVal SocketID As Long, ByVal GroupName As String)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GroupName	String	Group name

Output parameters
None

Return
Function error code

MATLAB

Prototype
[Error] **GroupReferencingStop** (int32 SocketID, cstring GroupName)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupName	cstring	Group name

Return

Error	int32	Function error code

PYTHON

Prototype
[Error] **GroupReferencingStop** (integer SocketID, string GroupName)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	Group name

Return

Error	integer	Function error code

2.3.3.29. GroupVelocityCurrentGet

NAME

GroupVelocityCurrentGet – Returns the current velocity for one or all positioners of the selected group.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid object type (group or positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Valid group name: ERR_GROUP_NAME (-19)
- Valid positioner name: ERR_POSITIONER_NAME (-18)
- Valid output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

Returns the current velocity for one or all positioners of the selected group.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

GroupVelocityCurrentGet $SocketID $GroupName CurrentVelocity …

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Group name (maximum size = 250)

Output parameters

CurrentPosition ..... floating point..... Current velocity (must be repeated for each positioner of group)

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupVelocityCurrentGet** (int SocketID, char *GroupName, int NbPositioners, double * CurrentVelocity)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... char * ..... Group name
NbPositioners..... int..... Number of positioners in the selected group

Output parameters

CurrentVelocity ..... double * ..... Current velocity array

Return

Function error code

Note : *CurrentVelocity* must have array size in accordance to *NbPositioners* to avoid array overrun.

VISUAL BASIC

Prototype

Long **GroupVelocityCurrentGet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal NbPositioners As Long, CurrentVelocity As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GroupName	String	Group name
NbPositioners	Long	Number of positioners in the selected group

Output parameters

CurrentVelocity	Double	Current velocity array

Return

Function error code

Note : *CurrentVelocity* must have array size in accordance to *NbPositioners* to avoid array overrun.

MATLAB

Prototype

[Error, CurrentVelocity] **GroupVelocityCurrentGet** (int32 SocketID, cstring GroupName, int32 NbPositioners)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupName	cstring	Group name
NbPositioners	int32	Number of positioners in the selected group

Return

Error	int32	Function error code
CurrentVelocity	double array	Current velocity array

Note : *CurrentVelocity* must have array size in accordance to *NbPositioners* to avoid array overrun.

PYTHON

Prototype

[Error, CurrentVelocity] **GroupVelocityCurrentGet** (integer SocketID, string GroupName, integer NbPositioners)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	Group name
NbPositioners	integer	Number of positioners in the selected group

Return

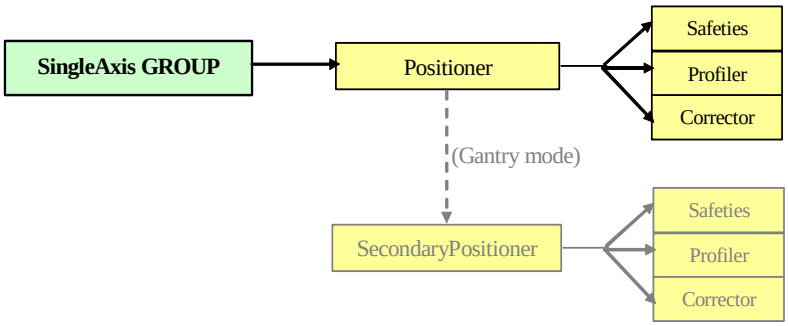
Error	integer	Function error code
CurrentVelocity	double list	List of current velocities

Note : *CurrentVelocity* has the elements number equel to *NbPositioners*.

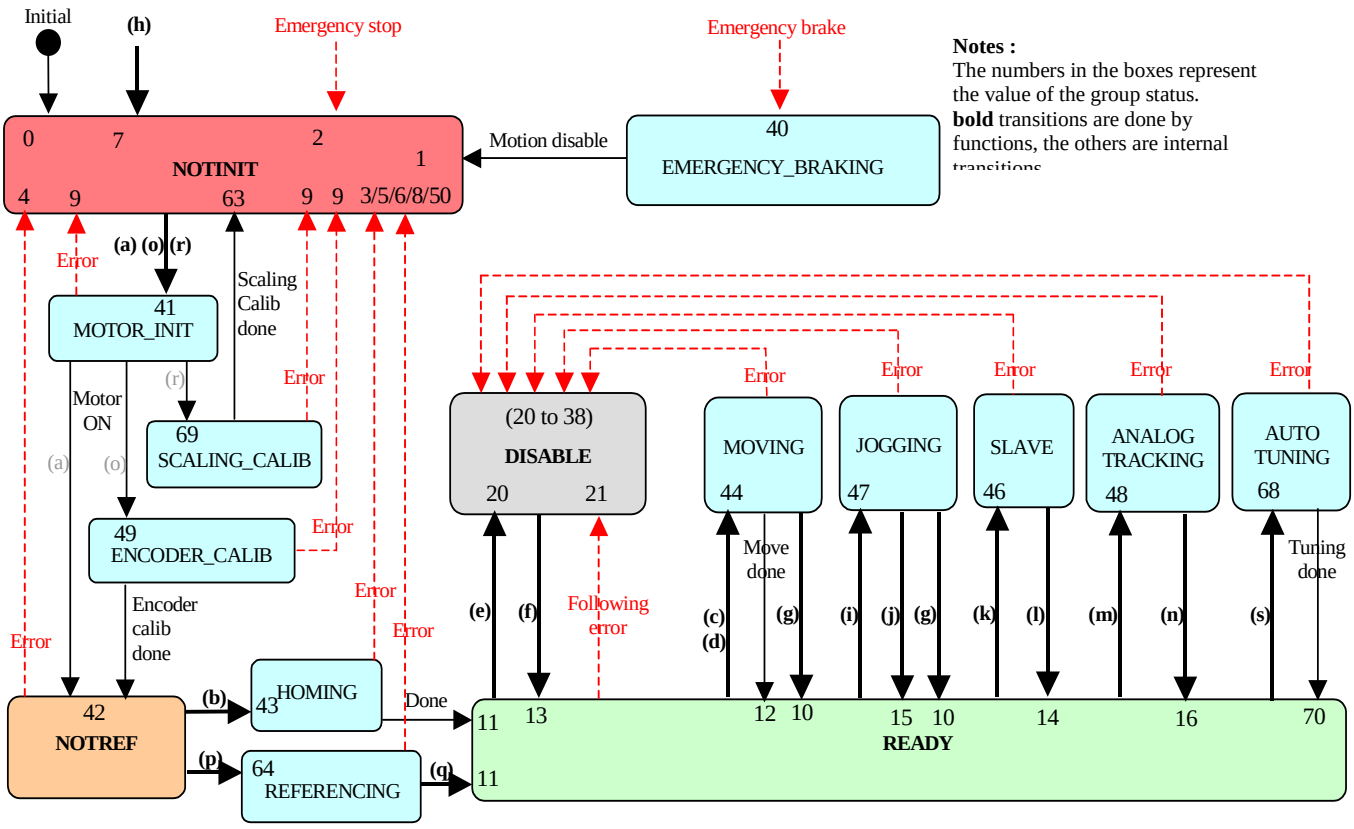
2.4. SingleAxis group

2.4.1. Description

The SingleAxis is composed of one single positioner object that allows execution of motion commands.
A SingleAxis group can be use in GANTRY mode (dual positioner).
The controller can handle several SingleAxis objects (1 to 8).
There is no relation between SingleAxis objects and other objects handled by the controller.



2.4.2. State diagram



Called function:

(a) GroupInitialize
(b) GroupHomeSearch
(c) GroupMoveAbsolute
(d) GroupMoveRelative
(e) GroupMotionDisable
(f) GroupMotionEnable
(g) GroupMoveAbort
(h) GroupKill or KillAll
(i) GroupJogModeEnable
(j) GroupJogModeDisable
(k) GroupSlaveModeEnable
(k) GroupSlaveModeDisable
(m) GroupAnalogTrackingModeEnable
(n) GroupAnalogTrackingModeDisable
(o) GroupInitializeWithEncoderCalibration
(p) GroupReferencingStart
(q) GroupReferencingStop
(r) PositionerAccelerationAutoScaling
(s) PositionerCorrectorAutoTuning

2.4.3. Specific function description

2.4.3.1. SingleAxisSlaveModeDisable

NAME

SingleAxisSlaveModeDisable – Disables the slave-master mode.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Group state must be "SLAVE": ERR_NOT_ALLOWED_ACTION (-22)
- Check the group name: ERR_GROUP_NAME (-19)
- Check the positioner name: ERR_POSITIONER_NAME (-18)
- Check the group type (must be a SingleAxis group): ERR_WRONG_OBJECT_TYPE (-8)

DESCRIPTION

This function allows disable the master-slave mode. If a motion is in progress then it is aborted.
To use this function, the group state must be SLAVE (46). If it's not the case then the ERR_NOT_ALLOWED_ACTION (-22) error is returned.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype

SingleAxisSlaveModeDisable $SocketID $GroupName

Input parameters

SocketID	integer	Socket identifier got from "TCP_ConnectToServer" function
GroupName	string	SingleAxis group name (maximum size = 250)

Output parameters (None)

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **SingleAxisSlaveModeDisable** (int SocketID, char *GroupName)

Input parameters

SocketID	int	Socket identifier got from "TCP_ConnectToServer" function
GroupName	char *	SingleAxis group name

Output parameters (None)

Return

Function error code

VISUAL BASIC

Prototype

Long **SingleAxisSlaveModeDisable** (ByVal SocketID As Long, ByVal GroupName As String)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... String ..... SingleAxis group name

Output parameters

None

Return

Function error code

MATLAB

Prototype

int32 **SingleAxisSlaveModeDisable** (int32 SocketID, cstring GroupName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... cstring ..... SingleAxis group name

Return

Function error code

PYTHON

Prototype

integer **SingleAxisSlaveModeDisable** (integer SocketID, string GroupName)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... string ..... SingleAxis group name

Return

Function error code

2.4.3.2. SingleAxisSlaveModeEnable

NAME

SingleAxisSlaveModeEnable – Enables the slave-master mode.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Group state must be "READY": ERR_NOT_ALLOWED_ACTION (-22)
- Check the group name: ERR_GROUP_NAME (-19)
- Check the positioner name: ERR_POSITIONER_NAME (-18)
- Check the group type (must be a SingleAxis group): ERR_WRONG_OBJECT_TYPE (-8)
- Check the slave parameters (must be configured): ERR_SLAVE_CONFIGURATION (-41)

DESCRIPTION

This function enables the master-slave mode only if the slave group is in ready mode. In this mode the slave must be defined as a SingleAxis group and the master can be a positioner from any group.

To use this function, the SingleAxis group must be in the READY state. If it's not the case then the ERR_NOT_ALLOWED_ACTION (-22) error is returned.

To use this function, the master positioner and the slave ratio must be configured by the "SingleAxisSlaveParametersSet" function. If it's not the case then the ERR_SLAVE_CONFIGURATION (-41) error is returned.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_POSITIONER_NAME (-18)
ERR_SLAVE_CONFIGURATION (-41)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype

SingleAxisSlaveModeEnable $SocketID $GroupName

Input parameters

SocketID ..... integer ..... Socket identifier got from "TCP_ConnectToServer" function
GroupName..... string ..... SingleAxis group name (maximum size = 250)

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **SingleAxisSlaveModeEnable** (int SocketID, char *GroupName)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... char * ..... SingleAxis group name

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **SingleAxisSlaveModeEnable** (ByVal SocketID As Long, ByVal GroupName As String)

Input parameters

SocketID ..... Long..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... String ..... SingleAxis group name

Output parameters

None

Return

Function error code

MATLAB

Prototype

int32 **SingleAxisSlaveModeEnable** (int32 SocketID, cstring GroupName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... cstring ..... SingleAxis group name

Return

Function error code

PYTHON

Prototype

integer **SingleAxisSlaveModeEnable** (integer SocketID, string GroupName)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... SingleAxis group name

Return

Function error code

2.4.3.3. SingleAxisSlaveParametersGet

NAME

SingleAxisSlaveParametersGet – Returns the slave parameters.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner name: ERR_POSITIONER_NAME (-18)
- Check the group type (must be a SingleAxis group): ERR_WRONG_OBJECT_TYPE (-8)
- Valid output parameter type: ERR_WRONG_TYPE_DOUBLE (-14), ERR_WRONG_TYPE_CHAR (-13)
- Check the slave parameters (must be configured): ERR_NOT_ALLOWED_ACTION (-22)

DESCRIPTION

This function returns the slave parameters: the master positioner name and the master-slave ratio.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

SingleAxisSlaveParametersGet $SocketID $GroupName MasterPositionerName Ratio

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Group name (maximum size = 250)

Output parameters

MasterPositionerName..... string ..... Master positioner name from any group
Ratio..... double ..... Gear ratio between the master and the slave

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **SingleAxisSlaveParametersGet** (int SocketID, char *GroupName, char * MasterPositionerName , double * Ratio)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
GroupName	char *	Group name

Output parameters

MasterPositionerName	char *	Master positioner name from any group
Ratio	double *	Gear ratio between the master and the slave

Return

Function error code

VISUAL BASIC

Prototype

Long **SingleAxisSlaveParametersGet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal MasterPositionerName As String, Ratio As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GroupName	String	Group name

Output parameters

MasterPositionerName	String	Master positioner name from any group
Ratio	Double	Gear ratio between the master and the slave

Return

Function error code

MATLAB

Prototype

[Error, MasterPositionerName, Ratio] **SingleAxisSlaveParametersGet** (int32 SocketID, cstring GroupName)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupName	cstring	Group name

Return

Error	int32	Function error code
MasterPositionerName	cstring	Master positioner name from any group
Ratio	double	Gear ratio between the master and the slave

PYTHON

Prototype

[Error, MasterPositionerName, Ratio] **SingleAxisSlaveParametersGet** (integer SocketID, string GroupName)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	Group name

Return

Error	integer	Function error code
MasterPositionerName	string	Master positioner name from any group
Ratio	double	Gear ratio between the master and the slave

2.4.3.4. SingleAxisSlaveParametersSet

NAME

SingleAxisSlaveParametersSet – Sets the slave parameters.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the master positioner name: ERR_POSITIONER_NAME (-18)
- Check the master group type: ERR_WRONG_OBJECT_TYPE (-8)
- Check input parameter type: ERR_WRONG_TYPE_DOUBLE (-14), ERR_WRONG_TYPE_CHAR (-13)
- Check the slave parameters (must be configured): ERR_NOT_ALLOWED_ACTION (-22)
- Check the base velocity value (must be null): ERR_BASE_VELOCITY (-48)
- Check the ratio value (Ratio > 0): ERR_PARAMETER_OUT_OF_RANGE (-17)

DESCRIPTION

This function configures the slave parameters only for a SingleAxis group.
The slave is a master copy and a ratio can be applied : Slave = Ratio * Master.
The slave-master mode is activated only after the call of “SingleAxisSlaveModeEnable” function.

The master can be a positioner from any group, except from the spindle group. If the master group is a spindle then the ERR_NOT_ALLOWED_ACTION (-22) error is returned. The master positioner must be different to the slave positioner else the ERR_WRONG_OBJECT_TYPE (-8) is returned.

NOTE :
After an emergency stop, the master group and the slave group are in “Not Initialized” status. To restart a master-slave relation, the slave group(s) must be reinitialised **before** the master group.

ERROR CODES

ERR_BASE_VELOCITY (-48)
ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype
SingleAxisSlaveParametersSet $SocketID $GroupName $MasterPositionerName $Ratio

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	Group name (maximum size = 250)
MasterPositionerName	string	Master positioner name from any group
Ratio	double	Gear ratio between the master and the slave

Output parameters (None)

Return
TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **SingleAxisSlaveParametersSet** (int SocketID, char *GroupName, char * MasterPositionerName , double Ratio)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
GroupName	char *	Group name
MasterPositionerName	char *	Master positioner name from any group
Ratio	double	Gear ratio between the master and the slave

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **SingleAxisSlaveParametersSet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal MasterPositionerName As String, ByVal Ratio As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GroupName	String	Group name
MasterPositionerName	String	Master positioner name from any group
Ratio	Double	Gear ratio between the master and the slave

Output parameters

None

Return

Function error code

MATLAB

Prototype

[Error] **SingleAxisSlaveParametersSet** (int32 SocketID, cstring GroupName, cstring MasterPositionerName, double Ratio)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupName	cstring	Group name
MasterPositionerName	cstring	Master positioner name from any group
Ratio	double	Gear ratio between the master and the slave

Return

Error	int32	Function error code

PYTHON

Prototype

[Error] **SingleAxisSlaveParametersSet** (integer SocketID, string GroupName, string MasterPositionerName, double Ratio)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	Group name
MasterPositionerName	string	Master positioner name from any group
Ratio	double	Gear ratio between the master and the slave

Return

Error	integer	Function error code

2.4.4. Configuration files

Example of the definition of a SingleAxis group (named “MySingleAxis”) in the system.ini file. The “MySingleAxis” group is build by a positioner named “MyPositioner”.

The positioner “MyPositioner” uses the parameters of “MYSTAGE” from the stages.ini file and is connected to the plug 1 of the XPS controller. The Positioner “MyPositioner” has a secondary positioner (option to build a “gantry” position). This secondary positioner uses the parameters of “MYSTAGE” from the stages.ini file and is connected to the plug 2 of the XPS controller.

System.ini file :

```
[GROUPS]
SingleAxisInUse = MySingleAxis

[MySingleAxis]  ; Configuration of “MySingleAxis” SingleAxis group
PositionerInUse = MyPositioner

[MySingleAxis. MyPositioner]
PlugNumber = 1
StageName = MYSTAGE

;---- Secondary positioner (optional)
SecondaryPlugNumber = 2
SecondaryStageName = MYSTAGE
SecondaryPositionerGantryEndReferencingPosition = 0
SecondaryPositionerGantryEndReferencingTolerance = 0
SecondaryPositionerGantryOffsetAfterInitialization = 0
```

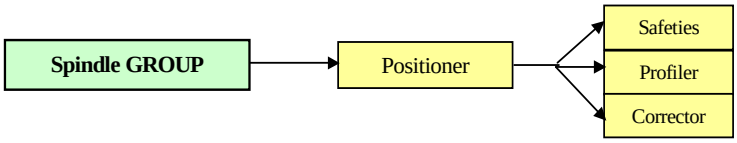
Stages.ini file :

```
[MYSTAGE]
MYSTAGE configuration => See § “Positioner : Configuration filess”
```

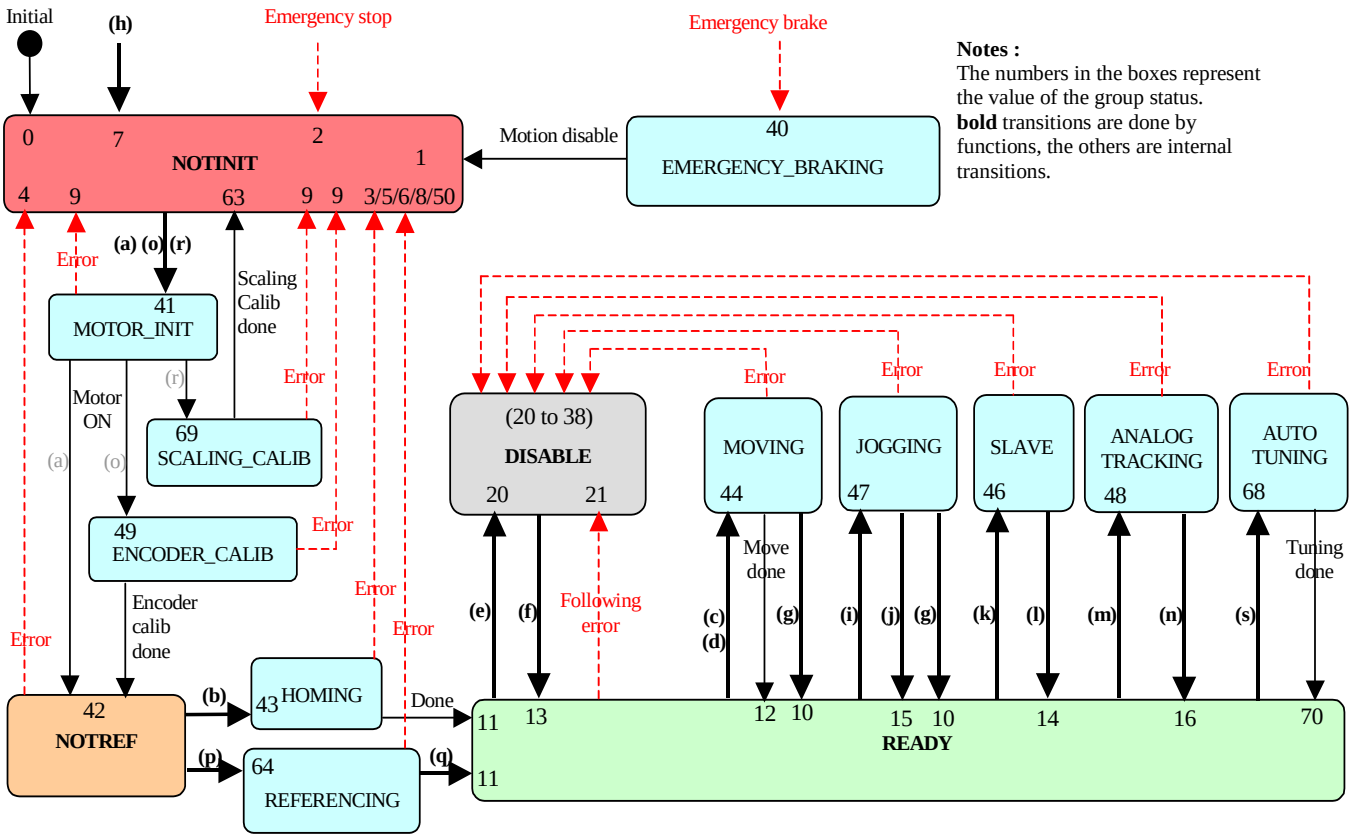
2.5. Spindle group

2.5.1. Description

A Spindle group is very similar to the SingleAxis group. It's composed of only one positioner. It has one main difference, it does not handle software or hardware end of runs. Therefore, it is allowed to spin indefinitely in any direction.
SingleAxis group motion commands are still allowed (beside the jog, that is replaced by spin).
The controller can handle several Spindle objects.
There is no relation between Spindle objects and other objects handled by the controller.



2.5.2. State diagram



Called function:

(a) GroupInitialize
(b) GroupHomeSearch
(c) GroupMoveAbsolute
(d) GroupMoveRelative
(e) GroupMotionDisable
(f) GroupMotionEnable
(g) GroupMoveAbort
(h) GroupKill or KillAll
(i) GroupJogModeEnable
(j) GroupJogModeDisable
(k) GroupSlaveModeEnable
(k) GroupSlaveModeDisable
(m) GroupAnalogTrackingModeEnable
(n) GroupAnalogTrackingModeDisable
(o) GroupInitializeWithEncoderCalibration
(p) GroupReferencingStart
(q) GroupReferencingStop
(r) PositionerAccelerationAutoScaling
(s) PositionerCorrectorAutoTuning

2.5.3. Specific function description

2.5.3.1. GroupSpinCurrentGet

NAME

GroupSpinCurrentGet – Returns the spin mode parameters.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner name: ERR_POSITIONER_NAME (-18)
- Check the group type (must be a Spindle group): ERR_WRONG_OBJECT_TYPE (-8)
- Valid output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function returns the current (or actual) velocity and acceleration used by the SPIN mode.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

GroupSpinCurrentGet $SocketID $GroupName Velocity Acceleration

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Spindle group name (maximum size = 250)

Output parameters

Velocity..... double ..... Velocity (units / s)
Acceleration..... double ..... Acceleration (units / s²)

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupSpinCurrentGet** (int SocketID, char *GroupName, double * Velocity, double * Acceleration)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... char * ..... Spindle group name

Output parameters

Velocity..... double * ..... Velocity (units / s)
Acceleration..... double * ..... Acceleration (units / s²)

Return

Function error code

VISUAL BASIC

Prototype

Long **GroupSpinCurrentGet** (ByVal SocketID As Long, ByVal GroupName As String, Velocity As Double, Acceleration As Double)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... String ..... Spindle group name

Output parameters

Velocity ..... Double ..... Velocity (units / s)
Acceleration ..... Double ..... Acceleration (units / s^2)

Return

Function error code

MATLAB

Prototype

[Error, Velocity, Acceleration] **GroupSpinCurrentGet** (int32 SocketID, cstring GroupName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... cstring ..... Spindle group name

Return

Error ..... int32 ..... Function error code
Velocity ..... double ..... Velocity (units / s)
Acceleration ..... double ..... Acceleration (units / s^2)

PYTHON

Prototype

[Error, Velocity, Acceleration] **GroupSpinCurrentGet** (integer SocketID, string GroupName)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... string ..... Spindle group name

Return

Error ..... integer ..... Function error code
Velocity ..... double ..... Velocity (units / s)
Acceleration ..... double ..... Acceleration (units / s^2)

2.5.3.2. GroupSpinModeStop

NAME

GroupSpinModeStop – Stops motion of the spindle group.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner name (must be a group name): ERR_POSITIONER_NAME (-18)
- Check the group type (must be a Spindle group): ERR_WRONG_OBJECT_TYPE (-8)

DESCRIPTION

This function stops motion of a spindle group and sets the group state to READY.
To use this function, the group must be in SPINNING status else the ERR_NOT_ALLOWED_ACTION (-22) is returned.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype

GroupSpinModeStop $SocketID $GroupName

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Spindle group name (maximum size = 250)

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupSpinModeStop** (int SocketID, char *GroupName)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... char * ..... Spindle group name

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **GroupSpinModeStop** (ByVal SocketID As Long, ByVal GroupName As String)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... String ..... Spindle group name

Output parameters

None

Return

Function error code

MATLAB

Prototype

[Error] **GroupSpinModeStop** (int32 SocketID, cstring GroupName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... cstring ..... Spindle group name

Return

Error ..... int32 ..... Function error code

PYTHON

Prototype

[Error] **GroupSpinModeStop** (integer SocketID, string GroupName)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... string ..... Spindle group name

Return

Error ..... integer ..... Function error code

2.5.3.3. GroupSpinParametersGet

NAME

GroupSpinParametersGet – Returns the spin profiler parameters.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the group type (must be a spindle group): ERR_WRONG_OBJECT_TYPE (-8)
- Check the function (must be a spindle function): ERR_POSITIONER_NAME (-18)
- Valid output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function returns the “Setpoint” (theoretical) velocity and acceleration used by the SPIN mode.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

GroupSpinParametersGet $SocketID $GroupName Velocity Acceleration

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Spindle group name (maximum size = 250)

Output parameters

Velocity..... double ..... Setpoint Velocity (units / s)
Acceleration..... double ..... Setpoint Acceleration (units / s^2)

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupSpinParametersGet** (int SocketID, char *GroupName, double * Velocity, double * Acceleration)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... char * ..... Spindle group name

Output parameters

Velocity..... double * ..... Setpoint Velocity (units / s)
Acceleration..... double * ..... Setpoint Acceleration (units / s^2)

Return

Function error code

VISUAL BASIC

Prototype

Long **GroupSpinParametersGet** (ByVal SocketID As Long, ByVal GroupName As String, Velocity As Double, Acceleration As Double)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... String ..... Spindle group name

Output parameters

Velocity ..... Double ..... Setpoint Velocity (units / s)
Acceleration ..... Double ..... Setpoint Acceleration (units / s²)

Return

Function error code

MATLAB

Prototype

[Error, Velocity, Acceleration] **GroupSpinParametersGet** (int32 SocketID, cstring GroupName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... cstring ..... Spindle group name

Return

Error ..... int32 ..... Function error code
Velocity ..... double ..... Setpoint Velocity (units / s)
Acceleration ..... double ..... Setpoint Acceleration (units / s²)

PYTHON

Prototype

[Error, Velocity, Acceleration] **GroupSpinParametersGet** (integer SocketID, string GroupName)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... string ..... Spindle group name

Return

Error ..... integer ..... Function error code
Velocity ..... double ..... Setpoint Velocity (units / s)
Acceleration ..... double ..... Setpoint Acceleration (units / s²)

2.5.3.4. GroupSpinParametersSet

NAME

GroupSpinParametersSet – Sets the spin profiler parameters.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the group type (must be a spindle group): ERR_WRONG_OBJECT_TYPE (-8)
- Check the function (must be a spindle function): ERR_POSITIONER_NAME (-18)
- Valid output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)
- Check input parameter value: ERR_PARAMETER_OUT_OF_RANGE (-17)
  - *Velocity <= MaximumVelocity*
  - *Velocity >= - MaximumVelocity*
  - *Acceleration > 0*
  - *Acceleration ≤ MaximumAcceleration*

DESCRIPTION

This function starts the SPIN mode and also allows change on-the-fly the velocity and the acceleration used by the SPIN mode. If an error occurs, the positioner stops and the velocity value is setting to zero.

After the tests on input values:

If Velocity > MaximumVelocity	=> Velocity = MaximumVelocity
If Velocity < - MaximumVelocity	=> Velocity = - MaximumVelocity
If Acceleration ≤ 0	=> ERROR and stop motion
If Acceleration > MaximumAcceleration	=> Acceleration = MaximumAcc.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

GroupSpinParametersSet $SocketID $GroupName $Velocity $Acceleration

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Spindle group name (maximum size = 250)
Velocity..... double ..... Setpoint Velocity (units / s)
Acceleration..... double ..... Setpoint Acceleration (units / s²)

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupSpinParametersSet** (int SocketID, char *GroupName, double Velocity, double Acceleration)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
GroupName	char *	Spindle group name
Velocity	double	Setpoint Velocity (units / s)
Acceleration	double	Setpoint Acceleration (units / s²)

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **GroupSpinParametersSet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal Velocity As Double, ByVal Acceleration As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GroupName	String	Spindle group name
Velocity	Double	Setpoint Velocity (units / s)
Acceleration	Double	Setpoint Acceleration (units / s²)

Output parameters

None

Return

Function error code

MATLAB

Prototype

[Error] **GroupSpinParametersSet** (int32 SocketID, cstring GroupName, double Velocity, double Acceleration)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupName	cstring	Spindle group name
Velocity	double	Setpoint Velocity (units / s)
Acceleration	double	Setpoint Acceleration (units / s²)

Return

Error	int32	Function error code

PYTHON

Prototype

[Error, Velocity, Acceleration] **GroupSpinParametersSet** (integer SocketID, string GroupName, double Velocity, double Acceleration)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	Spindle group name
Velocity	double	Setpoint Velocity (units / s)
Acceleration	double	Setpoint Acceleration (units / s²)

Return

Error	integer	Function error code

2.5.3.5. SpindleSlaveModeDisable

NAME

SpindleSlaveModeDisable – Disables the slave-master mode.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Group state must be "SLAVE": ERR_NOT_ALLOWED_ACTION (-22)
- Check the group name: ERR_GROUP_NAME (-19)
- Check the positioner name: ERR_POSITIONER_NAME (-18)
- Check the group type (must be a Spindle group): ERR_WRONG_OBJECT_TYPE (-8)

DESCRIPTION

This function allows disable the master-slave mode for a spindle group. If a motion is in progress then it is aborted.

To use this function, the group state must be SLAVE (46). If it's not the case then the ERR_NOT_ALLOWED_ACTION (-22) error is returned.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype

SpindleSlaveModeDisable $SocketID $GroupName

Input parameters

SocketID ..... integer ..... Socket identifier got from "TCP_ConnectToServer" function
GroupName..... string ..... Spindle group name (maximum size = 250)

Output parameters (None)

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **SpindleSlaveModeDisable** (int SocketID, char *GroupName)

Input parameters

SocketID ..... int ..... Socket identifier got from "TCP_ConnectToServer" function
GroupName..... char * ..... Spindle group name

Output parameters (None)

Return

Function error code

VISUAL BASIC

Prototype

Long **SpindleSlaveModeDisable** (ByVal SocketID As Long, ByVal GroupName As String)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... String ..... Spindle group name

Output parameters

None

Return

Function error code

MATLAB

Prototype

int32 **SpindleSlaveModeDisable** (int32 SocketID, cstring GroupName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... cstring ..... Spindle group name

Return

Function error code

PYTHON

Prototype

integer **SpindleSlaveModeDisable** (integer SocketID, string GroupName)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... string ..... Spindle group name

Return

Function error code

2.5.3.6. SpindleSlaveModeEnable

NAME

SpindleSlaveModeEnable – Enables the slave-master mode.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Group state must be "READY": ERR_NOT_ALLOWED_ACTION (-22)
- Check the group name: ERR_GROUP_NAME (-19)
- Check the positioner name: ERR_POSITIONER_NAME (-18)
- Check the group type (must be a Spindle group): ERR_WRONG_OBJECT_TYPE (-8)
- Check the slave parameters (must be configured): ERR_SLAVE_CONFIGURATION (-41)

DESCRIPTION

This function enables the master-slave mode only if the slave group is in ready mode. In this mode the slave must be defined as a Spindle group and the master can be a positioner from any group.

To use this function, the Spindle group must be in the READY state. If it's not the case then the ERR_NOT_ALLOWED_ACTION (-22) error is returned.

To use this function, the master positioner and the slave ratio must be configured by the "SpindleSlaveParametersSet" function. If it's not the case then the ERR_SLAVE_CONFIGURATION (-41) error is returned.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_POSITIONER_NAME (-18)
ERR_SLAVE_CONFIGURATION (-41)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype

SpindleSlaveModeEnable $SocketID $GroupName

Input parameters

SocketID ..... integer ..... Socket identifier got from "TCP_ConnectToServer" function
GroupName..... string ..... Spindle group name (maximum size = 250)

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **SpindleSlaveModeEnable** (int SocketID, char *GroupName)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... char * ..... Spindle group name

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **SpindleSlaveModeEnable** (ByVal SocketID As Long, ByVal GroupName As String)

Input parameters

SocketID ..... Long..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... String ..... Spindle group name

Output parameters

None

Return

Function error code

MATLAB

Prototype

int32 **SpindleSlaveModeEnable** (int32 SocketID, cstring GroupName)

Input parameters

SocketID ..... int32..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... cstring ..... Spindle group name

Return

Function error code

PYTHON

Prototype

integer **SpindleSlaveModeEnable** (integer SocketID, string GroupName)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Spindle group name

Return

Function error code

2.5.3.7. SpindleSlaveParametersGet

NAME

SpindleSlaveParametersGet – Returns the slave parameters.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the positioner name: ERR_POSITIONER_NAME (-18)
- Check the group type (must be a Spindle group): ERR_WRONG_OBJECT_TYPE (-8)
- Valid output parameter type: ERR_WRONG_TYPE_DOUBLE (-14), ERR_WRONG_TYPE_CHAR (-13)
- Check the slave parameters (must be configured): ERR_NOT_ALLOWED_ACTION (-22)

DESCRIPTION

This function returns the slave parameters: the master positioner name and the master-slave ratio.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

SpindleSlaveParametersGet $SocketID $GroupName MasterPositionerName Ratio

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... Spindle Group name (maximum size = 250)

Output parameters

MasterPositionerName..... string ..... Master positioner name from any group
Ratio..... double ..... Gear ratio between the master and the slave

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **SpindleSlaveParametersGet** (int SocketID, char *GroupName, char * MasterPositionerName , double * Ratio)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... char * ..... Spindle Group name

Output parameters

MasterPositionerName..... char * ..... Master positioner name from any group
Ratio..... double * ..... Gear ratio between the master and the slave

Return

Function error code

VISUAL BASIC

Prototype

Long **SpindleSlaveParametersGet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal MasterPositionerName As String, Ratio As Double)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... String ..... Spindle Group name

Output parameters

MasterPositionerName ..... String ..... Master positioner name from any group
Ratio ..... Double ..... Gear ratio between the master and the slave

Return

Function error code

MATLAB

Prototype

[Error, MasterPositionerName, Ratio] **SpindleSlaveParametersGet** (int32 SocketID, cstring GroupName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... cstring ..... Spindle Group name

Return

Error ..... int32 ..... Function error code
MasterPositionerName ..... cstring ..... Master positioner name from any group
Ratio ..... double ..... Gear ratio between the master and the slave

PYTHON

Prototype

[Error, MasterPositionerName, Ratio] **SpindleSlaveParametersGet** (integer SocketID, string GroupName)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName ..... string ..... Spindle Group name

Return

Error ..... integer ..... Function error code
MasterPositionerName ..... string ..... Master positioner name from any group
Ratio ..... double ..... Gear ratio between the master and the slave

2.5.3.8. SpindleSlaveParametersSet

NAME

SpindleSlaveParametersSet – Sets the slave parameters.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the master positioner name: ERR_POSITIONER_NAME (-18)
- Check the master group type: ERR_WRONG_OBJECT_TYPE (-8)
- Check input parameter type: ERR_WRONG_TYPE_DOUBLE (-14), ERR_WRONG_TYPE_CHAR (-13)
- Check the slave parameters (must be configured): ERR_NOT_ALLOWED_ACTION (-22)
- Check the base velocity value (must be null): ERR_BASE_VELOCITY (-48)
- Check the ratio value (Ratio > 0): ERR_PARAMETER_OUT_OF_RANGE (-17)

DESCRIPTION

This function configures the slave parameters only for a Spindle group.
The slave is a master copy and a ratio can be applied : Slave = Ratio * Master.
The slave-master mode is activated only after the call of “SingleAxisSlaveModeEnable” function.

The master can be a positioner from a spindle group only. If the master group is another group type then the ERR_NOT_ALLOWED_ACTION (-22) error is returned. The master positioner must be different to the slave positioner else the ERR_WRONG_OBJECT_TYPE (-8) is returned.

NOTE :
After an emergency stop, the master group and the slave group are in “Not Initialized” status. To restart a master-slave relation, the slave group(s) must be reinitialised **before** the master group.

ERROR CODES

ERR_BASE_VELOCITY (-48)
ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype
SpindleSlaveParametersSet $SocketID $GroupName $MasterPositionerName $Ratio

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	Spindle Group name (maximum size = 250)
MasterPositionerName	string	Master positioner name from any group
Ratio	double	Gear ratio between the master and the slave

Output parameters (None)

Return
TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **SpindleSlaveParametersSet** (int SocketID, char *GroupName, char * MasterPositionerName , double Ratio)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
GroupName	char *	Spindle Group name
MasterPositionerName	char *	Master positioner name from any group
Ratio	double	Gear ratio between the master and the slave

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **SpindleSlaveParametersSet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal MasterPositionerName As String, ByVal Ratio As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GroupName	String	Spindle Group name
MasterPositionerName	String	Master positioner name from any group
Ratio	Double	Gear ratio between the master and the slave

Output parameters

None

Return

Function error code

MATLAB

Prototype

[Error] **SpindleSlaveParametersSet** (int32 SocketID, cstring GroupName, cstring MasterPositionerName, double Ratio)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupName	cstring	Spindle Group name
MasterPositionerName	cstring	Master positioner name from any group
Ratio	double	Gear ratio between the master and the slave

Return

Error	int32	Function error code

PYTHON

Prototype

[Error] **SpindleSlaveParametersSet** (integer SocketID, string GroupName, string MasterPositionerName, double Ratio)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	Spindle Group name
MasterPositionerName	string	Master positioner name from any group
Ratio	double	Gear ratio between the master and the slave

Return

Error	integer	Function error code

2.5.4. Configuration files

Example of the definition of a Spindle group (named “AXIS”) in the system.ini file. The group POSITIONER is build by a positioner named “STAGE”. The positioner “STAGE” uses the parameters of “MYSTAGE” from the stages.ini file and is connected to the plug 1 of the XPS controller.

System.ini file :

```
[GROUPS]
SpindleInUse = POSITIONER

[POSITIONER]                    ; AXIS Spindle group configuration
PositionerInUse = STAGE


[POSITIONER.STAGE]
PlugNumber = 1
StageName = MYSTAGE
```

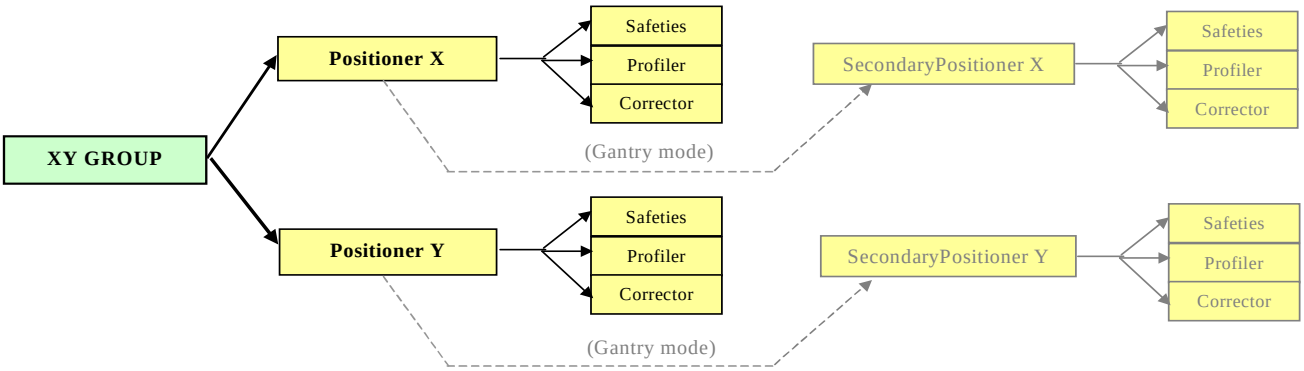
Stages.ini file :

```
[MYSTAGE]
MYSTAGE configuration => See § “Positioner : Configuration filess”
```

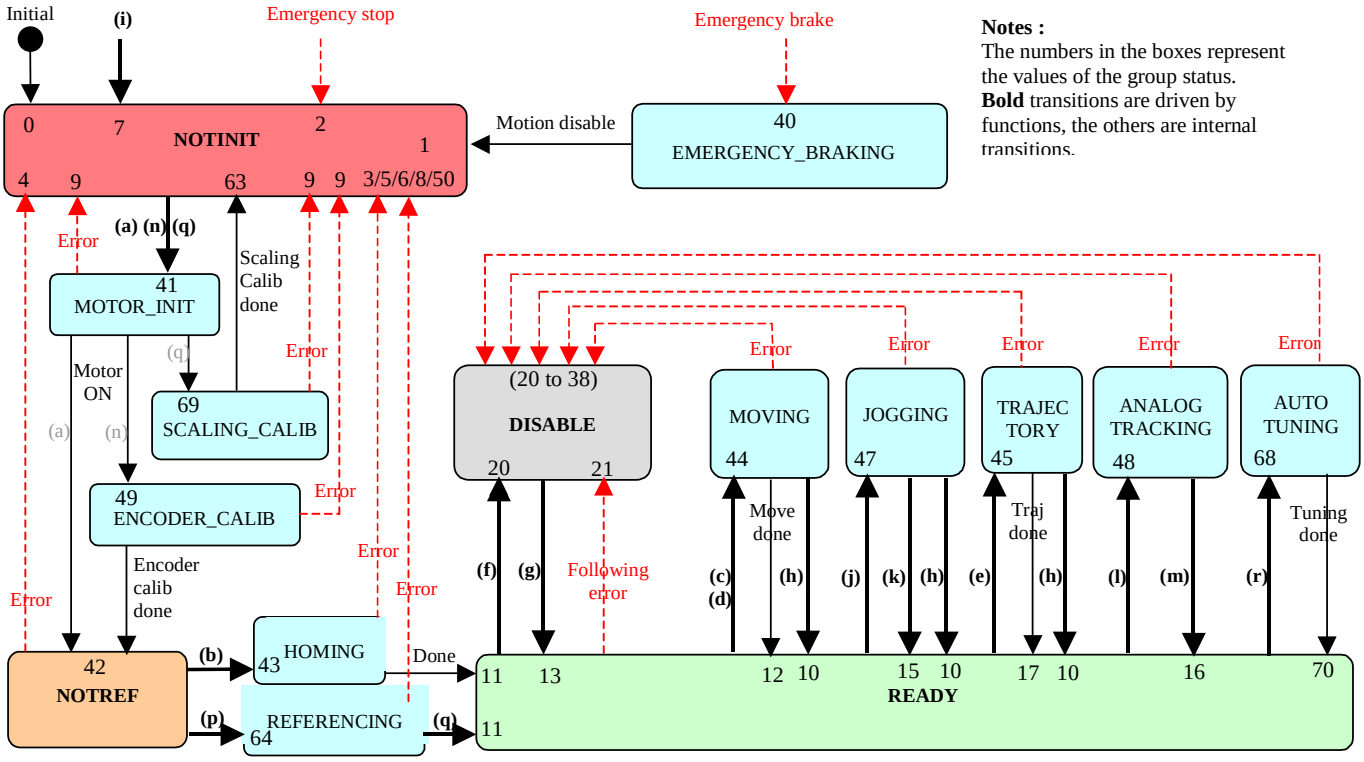
2.6. XY group

2.6.1. Description

An XY group is composed of two positioner objects, typically in an orthogonal XY configuration.
An XY group can be use in GANTRY mode (dual positioner for X or for Y).
It includes an XY mapping feature : XY = f (XY)
It includes an XY line-arc trajectory (2D).



2.6.2. State diagram



Called function :

(a) GroupInitialize
(b) GroupHomeSearch
(c) GroupMoveAbsolute
(d) GroupMoveRelative
(e) XYLineArcExecution
(f) GroupMotionDisable
(g) GroupMotionEnable
(h) GroupMoveAbort
(i) GroupKill or KillAll
(j) GroupJogModeEnable
(k) GroupJogModeDisable
(l) GroupAnalogTrackingModeEnable
(m) GroupAnalogTrackingModeDisable
(n) GroupInitializeWithEncoderCalibration
(o) GroupReferencingStart
(p) GroupReferencingStop
(q) PositionerAccelerationAutoScaling
(r) PositionerCorrectorAutoTuning

2.6.3. Specific function description

2.6.3.1. GroupPositionCorrectedProfilerGet

NAME

GroupPositionCorrectedProfilerGet – Returns the corrected profiler position corresponding to a profiler position.

INPUT TESTS

- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)
- Check parameter range: ERR_PARAMETER_OUT_OF_RANGE (-17)
- Valid object type (group or positioner): ERR_WRONG_OBJECT_TYPE (-8)
- Valid group name: ERR_GROUP_NAME (-19)
- Valid positioner name: ERR_POSITIONER_NAME (-18)

DESCRIPTION

Returns the corrected profiler position for one or all positioners of the selected group.
This function is to be used with Position Compare Output or Time Flasher hardware functions. The positions returned by this function will allow to place accurately the start and stop positions taking in account the encoder and XY mappings.

ERROR CODES

ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_GROUP_NAME (-19)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_OBJECT_TYPE (-8)
SUCCESS (0) : no error

TCL

Prototype

GroupPositionCorrectedProfilerGet $SocketID $GroupName $ProfilerPosition CorrectedProfilerPosition

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	Group name (maximum size = 250)
ProfilerPosition	floating point	Profiler position

Output parameters

CorrectedProfilerPosition	floating point	Current Position (must be repeated for each positioner of group)

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GroupPositionCorrectedProfilerGet** (int SocketID, char *GroupName, double PositionX, double PositionY, double * CorrectedProfilerPositionX, double * CorrectedProfilerPositionY)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
GroupName	char *	Group name
PositionX	double	X profiler position
PositionY	double	Y profiler position

Output parameters

CorrectedProfilerPositionX	double *	X Corrected Profiler position
CorrectedProfilerPositionY	double *	Y Corrected Profiler position

Return

Function error code

VISUAL BASIC

Prototype

Long **GroupPositionCorrectedProfilerGet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal PositionX As Double, ByVal PositionY As Double, CorrectedProfilerPositionX As Double, CorrectedProfilerPositionY As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GroupName	String	Group name
PositionX	Double	X profiler position
PositionY	Double	Y profiler position

Output parameters

CorrectedProfilerPositionX	Double	X Corrected Profiler position
CorrectedProfilerPositionY	Double	Y Corrected Profiler position

Return

Function error code

MATLAB

Prototype

[Error, CorrectedProfilerPositionX, CorrectedProfilerPositionY] **GroupPositionCorrectedProfilerGet** (int32 SocketID, cstring GroupName, double PositionX, double PositionY)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupName	cstring	Group name
PositionX	double	Profiler Position
PositionY	double	Profiler Position

Return

Error	int32	Function error code
CorrectedProfilerPositionX	double	X corrected position
CorrectedProfilerPositionY	double	Y corrected position

PYTHON

Prototype

[Error, CorrectedProfilerPositionX, CorrectedProfilerPositionY] **GroupPositionCorrectedProfilerGet** (integer SocketID, string GroupName)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	Group name
PositionX	double	X Profiler position
PositionY	double	Y Profiler position

Return

Error	integer	Function error code
CorrectedProfilerPositionX	double	X Profiler position
CorrectedProfilerPositionY	double	Y Profiler position

2.6.3.2. XYLineArcExecution

NAME

XYLineArcExecution – Executes an XY LineArc trajectory.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Group state must be "READY": ERR_NOT_ALLOWED_ACTION (-22)
- Check the group (must not be a gantry) : ERR_POSITIONER_NAME (-18)
- Check the group type (must be an XY group): ERR_WRONG_OBJECT_TYPE (-8)
- Check backlash (must not be enabled): ERR_NOT_ALLOWED_BACKLASH (-46)
- Check the trajectory file name length: ERR_STRING_TOO_LONG (-3)
- Check the trajectory file existence: ERR_READ_FILE (-61)
- Check the input value (> 0): ERR_PARAMETER_OUT_OF_RANGE (-17)
- Check the velocity (0 < Velocity <= TrajectoryMaximumVelocity): ERR_TRAJ_VEL_LIMIT (-68)
- Check the acceleration (0 < Acceleration <= TrajectoryMaximumAcceleration): ERR_TRAJ_ACC_LIMIT (-69)
- Check input parameter types: ERR_WRONG_TYPE_DOUBLE (-14), ERR_WRONG_TYPE_INT (-15)

DESCRIPTION

This function executes an XY LineArc trajectory. The trajectory file must be stored in the folder "\ADMIN\Public\Trajectory" of the XPS controller. If the trajectory can not be initialized (message queue or task error) then the ERR_TRAJ_INITIALIZATION (-72) is returned.
Before a trajectory execution, it is recommended to check its possible execution using the "XYLineArcVerification" and "XYLineArcVerificationResultGet" functions.

For a more thorough description of the line-arc trajectory capability, please refer to the XPS Motion Tutorial, section named Trajectories / Line-arc Trajectories.

NOTE:
In case of a ERR_GROUP_MOTION_DONE_TIMEOUT (-33) error, a ERR_FOLLOWING_ERROR (-25) error or ERR_SLAVE (-44) error, the group state becomes DISABLE. To help you to know what is the error source, check the positioner errors, the hardware status and the driver status.

ERROR CODES

ERR_BASE_VELOCITY (-48)
ERR_FATAL_INIT (-20)
ERR_FOLLOWING_ERROR (-25)
ERR_GROUP_MOTION_DONE_TIMEOUT (-33)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_NOT_ALLOWED_BACKLASH (-46)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_POSITIONER_NAME (-18)
ERR_READ_FILE (-61)
ERR_SLAVE (-44)
ERR_STRING_TOO_LONG (-3)
ERR_TRAJ_ACC_LIMIT (-69)
ERR_TRAJ_VEL_LIMIT (-68)
ERR_TRAJ_INITIALIZATION (-72)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14),
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

XYLineArcExecution $SocketID $GroupName $FileName $Velocity $Acceleration $ExecutionNumber

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	XY group name (maximum size = 250)
FileName	string	Trajectory file name (maximum size = 250)
Velocity	double	Trajectory velocity (units / seconds)
Acceleration	double	Trajectory acceleration (units / seconds²)
ExecutionNumber	integer	Number of trajectory executions

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **XYLineArcExecution** (int SocketID, char *GroupName, char *FileName , double Velocity, double Acceleration, int ExecutionNumber)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
GroupName	char *	XY group name
FileName	char *	Trajectory file name (maximum size = 250)
Velocity	double	Trajectory velocity (units / seconds)
Acceleration	double	Trajectory acceleration (units / seconds²)
ExecutionNumber	int	Number of trajectory executions

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **XYLineArcExecution** (ByVal SocketID As Long, ByVal GroupName As String, ByVal FileName As String, ByVal Velocity As Double, ByVal Acceleration As Double, ByVal ExecutionNumber As Integer)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GroupName	String	XY group name
FileName	String	Trajectory file name (maximum size = 250)
Velocity	Double	Trajectory velocity (units / seconds)
Acceleration	Double	Trajectory acceleration (units / seconds²)
ExecutionNumber	Integer	Number of trajectory executions

Output parameters

None

Return

Function error code

MATLAB

Prototype

int32 **XYLineArcExecution** (int32 SocketID, cstring GroupName, cstring FileName, double Velocity, double Acceleration, int ExecutionNumber)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupName	cstring	XY group name
FileName	cstring	Trajectory file name (maximum size = 250)
Velocity	double	Trajectory velocity (units / seconds)
Acceleration	double	Trajectory acceleration (units / seconds²)
ExecutionNumber	int32	Number of trajectory executions

Return

Function error code

PYTHON

Prototype

integer **XYLineArcExecution** (integer SocketID, string GroupName, string FileName, double Velocity, double Acceleration, int ExecutionNumber)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	XY group name
FileName	string	Trajectory file name (maximum size = 250)
Velocity	double	Trajectory velocity (units / seconds)
Acceleration	double	Trajectory acceleration (units / seconds²)
ExecutionNumber	integer	Number of trajectory executions

Return

Function error code

2.6.3.3. XYLineArcParametersGet

NAME

XYLineArcParametersGet – Returns the LineArc trajectory parameters.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the group (must not be a gantry) : ERR_POSITIONER_NAME (-18)
- Check the group type (must be a XY group): ERR_WRONG_OBJECT_TYPE (-8)
- Check the LineArc trajectory type: ERR_NOT_ALLOWED_ACTION (-22)
- Valid output parameter type: ERR_WRONG_TYPE_CHAR (-13), ERR_WRONG_TYPE_DOUBLE (-14), ERR_WRONG_TYPE_INT (-15)

DESCRIPTION

This function returns the XY LineArc trajectory parameters (trajectory name, trajectory velocity, trajectory acceleration, current executing element number) of the currently executed LineArc trajectory.

For a more thorough description of the line-arc trajectory capability, please refer to the XPS Motion Tutorial, section named Trajectories / Line-arc Trajectories.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
ERR_WRONG_TYPE_DOUBLE (-14)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

XYLineArcParametersGet $SocketID $GroupName FileName Velocity Acceleration ElementNumber

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	XY Group name (maximum size = 250)

Output parameters

FileName	string	Executing trajectory file name (maximum size = 250)
Velocity	double	Trajectory velocity (units / seconds)
Acceleration	double	Trajectory acceleration (units / seconds²)
ElementNumber	integer	Current executing element number

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **XYLineArcParametersGet** (int SocketID, char *GroupName, char * FileName, double * Velocity, double * Acceleration, int * ElementNumber)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
GroupName	char *	XY Group name

Output parameters

FileName	char *	Executing trajectory file name (maximum size = 250)
Velocity	double *	Trajectory velocity (units / seconds)
Acceleration	double *	Trajectory acceleration (units / seconds²)
ElementNumber	int *	Current executing element number

Return

Function error code

VISUAL BASIC

Prototype

Long **XYLineArcParametersGet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal FileName As String, Velocity As Double, Acceleration As Double, ElementNumber As Integer)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GroupName	String	XY Group name

Output parameters

FileName	String	Executing trajectory file name (maximum size = 250)
Velocity	Double	Trajectory velocity (units / seconds)
Acceleration	Double	Trajectory acceleration (units / seconds²)
ElementNumber	Integer	Current executing element number

Return

Function error code

MATLAB

Prototype

[Error, FileName, Velocity, Acceleration, ElementNumber] **XYLineArcParametersGet** (int32 SocketID, cstring GroupName)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupName	cstring	XY Group name

Return

Error	int32	Function error code
FileName	cstring	Executing trajectory file name (maximum size = 250)
Velocity	double	Trajectory velocity (units / seconds)
Acceleration	double	Trajectory acceleration (units / seconds²)
ElementNumber	int32	Current executing element number

PYTHON

Prototype

[Error, FileName, Velocity, Acceleration, ElementNumber] **XYLineArcParametersGet** (integer SocketID, string GroupName)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	XY Group name

Return

Error	integer	Function error code
FileName	string	Executing trajectory file name (maximum size = 250)
Velocity	double	Trajectory velocity (units / seconds)
Acceleration	double	Trajectory acceleration (units / seconds²)
ElementNumber	integer	Current executing element number

2.6.3.4. XYLineArcPulseOutputGet

NAME

XYLineArcPulseOutputGet – Returns the configuration of pulse generation on LineArc trajectory.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the group (must not be a positioner) : ERR_POSITIONER_NAME (-18)
- Check the group type (must be a XY group): ERR_WRONG_OBJECT_TYPE (-8)
- Valid output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function returns the last configuration of pulse generation on XY LineArc trajectory.
The pulse output configuration is defined by a start length, an end length, and a path length interval.

Example :
One pulse is generated every 10 µm on the Line-Arc trajectory between 10 mm and 30 mm.
Start length = 10 mm
End length = 30 mm
path length interval = 0.01 mm

For a more thorough description of the line-arc trajectory capability, please refer to the XPS Motion Tutorial, section named Trajectories / Line-arc Trajectories.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

XYLineArcPulseOutputGet $SocketID $GroupName StartLength EndLength PathLengthInterval

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	XY Group name (maximum size = 250)

Output parameters

StartLength	double	Start length (units)
EndLength	double	End length (units)
PathLengthInterval	double	Path length interval (units)

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **XYLineArcPulseOutputGet** (int SocketID, char *GroupName, double * StartLength, double * EndLength, double * PathLengthInterval)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
GroupName	char *	XY Group name

Output parameters

StartLength	double *	Start length (units)
EndLength	double *	End length (units)
PathLengthInterval	double *	Path length interval (units)

Return

Function error code

VISUAL BASIC

Prototype

Long **XYLineArcPulseOutputGet** (ByVal SocketID As Long, ByVal GroupName As String, StartLength As Double, EndLength As Double, PathLengthInterval As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GroupName	String	XY Group name

Output parameters

StartLength	Double	Start length (units)
EndLength	Double	End length (units)
PathLengthInterval	Double	Path length interval (units)

Return

Function error code

MATLAB

Prototype

[Error, StartLength, EndLength, PathLengthInterval] **XYLineArcPulseOutputGet** (int32 SocketID, cstring GroupName)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupName	cstring	XY Group name

Return

Error	int32	Function error code
StartLength	double	Start length (units)
EndLength	double	End length (units)
PathLengthInterval	double	Path length interval (units)

PYTHON

Prototype

[Error, StartLength, EndLength, PathLengthInterval] **XYLineArcPulseOutputGet** (integer SocketID, string GroupName)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	XY Group name

Return

Error	integer	Function error code
StartLength	double	Start length (units)
EndLength	double	End length (units)
PathLengthInterval	double	Path length interval (units)

2.6.3.5. XYLineArcPulseOutputSet

NAME

XYLineArcPulseOutputSet – Sets the configuration of pulse generation on LineArc trajectory.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the group (must not be a positioner) : ERR_POSITIONER_NAME (-18)
- Check the group type (must be a XY group): ERR_WRONG_OBJECT_TYPE (-8)
- Check the pulse generation must not be in progress : ERR_NOT_ALLOWED_ACTION (-22)
- Valid input parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function configures and activates the pulse generation on XY LineArc trajectory. The pulse generation is defined by a start length, an end length, and a path length interval. If a pulse generation is already activated on the selected XY Line-Arc trajectory then this function returns the ERR_NOT_ALLOWED_ACTION error.
Please note, that the pulse output settings are automatically removed when the trajectory is over. Hence, with the execution of every new trajectory, it is also required to define the pulse output settings again.

This capability allows output of pulses at constant trajectory length intervals on Line-Arc trajectories. The pulses are generated between a start length and an end length. All lengths are calculated in an orthogonal XY plane. The StartLength, EndLength, and PathLengthInterval refer to the Setpoint positions.
Example:
XYLineArcPulseOutputSet(XY, 10, 30, 0.01)
One pulse will be generated every 10 µm on the next Line-Arc trajectory between 10 mm and 30 mm

For a more thorough description of the line-arc trajectory capability, please refer to the XPS Motion Tutorial, section named Trajectories / Line-arc Trajectories.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

XYLineArcPulseOutputSet $SocketID $GroupName $StartLength $EndLength $PathLengthInterval

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	XY Group name (maximum size = 250)
StartLength	double	Start length (units)
EndLength	double	End length (units)
PathLengthInterval	double	Path length interval (units)

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **XYLineArcPulseOutputSet** (int SocketID, char *GroupName, double StartLength, double EndLength, double PathLengthInterval)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
GroupName	char *	XY Group name
StartLength	double	Start length (units)
EndLength	double	End length (units)
PathLengthInterval	double	Path length interval (units)

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **XYLineArcPulseOutputSet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal StartLength As Double, ByVal EndLength As Double, ByVal PathLengthInterval As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GroupName	String	XY Group name
StartLength	Double	Start length (units)
EndLength	Double	End length (units)
PathLengthInterval	Double	Path length interval (units)

Output parameters

None

Return

Function error code

MATLAB

Prototype

[Error] **XYLineArcPulseOutputSet** (int32 SocketID, cstring GroupName, double StartLength, double EndLength, double PathLengthInterval)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupName	cstring	XY Group name
StartLength	double	Start length (units)
EndLength	double	End length (units)
PathLengthInterval	double	Path length interval (units)

Return

Error	int32	Function error code

PYTHON

Prototype

[Error] **XYLineArcPulseOutputSet** (integer SocketID, string GroupName, double StartLength, double EndLength, double PathLengthInterval)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	XY Group name
StartLength	double	Start length (units)
EndLength	double	End length (units)
PathLengthInterval	double	Path length interval (units)

Return

Error	integer	Function error code

2.6.3.6. XYLineArcVerification

NAME

XYLineArcVerification – Verifies a line-arc trajectory data file.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the group (must not be a gantry) : ERR_POSITIONER_NAME (-18)
- Check the group type (must be an XY group): ERR_WRONG_OBJECT_TYPE (-8)
- Check the trajectory file name length: ERR_STRING_TOO_LONG (-3)
- Check the trajectory file existence: ERR_READ_FILE (-61)
- Check the trajectory (number of elements > 0) : ERR_TRAJ_EMPTY (-66)
- Check trajectory element types in file (FirstTangent, …): ERR_WRONG_TYPE_DOUBLE (-14)
- Check keys in file ("FirstTangent" and "DiscontinuityAngle"): ERR_READ_FILE_PARAMETER_KEY (-74)
- Check trajectory element (distance and tangent) : ERR_TRAJ_ELEM_LINE (-65)
  | XElementDistance | ≥1e-14
  | YElementDistance | ≥1e-14
  TangentOut ≠ 1.797e308
- Check trajectory element (radius) : ERR_TRAJ_ELEM_RADIUS (-63)
  Radius ≥ 1e-14
- Check trajectory element (sweep angle) : ERR_TRAJ_ELEM_ SWEEP (-64)
  SweepAngle ≥1e-14

DESCRIPTION

This function verifies the possible execution of an XY LineArc trajectory. The results of the verification can be gathered with the "XYLineArcVerificationResultGet" function. The trajectory file must be stored in the folder "\ADMIN\Public\Trajectory" of the XPS controller. If the trajectory can not be initialized (message queue or task error) then the ERR_TRAJ_INITIALIZATION (-72) is returned.

This function can be executed at any time and is independent from the trajectory execution. This function performs the following:

- ✓ Checks the trajectory file for data coherence.
- ✓ Calculates the trajectory limits, which are: the required travel per positioner, the maximum possible trajectory velocity and the maximum possible trajectory acceleration. This function helps define the parameters for the trajectory execution.
- ✓ If all is OK, it returns "SUCCESS" (0). Otherwise, it returns a corresponding error.

NOTE :
The "XYLineArcVerification" function is independent from the "XYLineArcExecution" function. So users don't need to execute this function before executing a LineArc trajectory. However, it is recommended to do that.

For a more thorough description of the line-arc trajectory capability, please refer to the XPS Motion Tutorial, section named Trajectories / Line-arc Trajectories.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_POSITIONER_NAME (-18)
ERR_READ_FILE (-61)
ERR_READ_FILE_PARAMETER_KEY (-74)
ERR_STRING_TOO_LONG (-3)
ERR_TRAJ_ELEM_LINE (-65)
ERR_TRAJ_ELEM_RADIUS (-63)
ERR_TRAJ_ELEM_SWEEP (-64)
ERR_TRAJ_EMPTY (-66)

ERR_TRAJ_INITIALIZATION (-72)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

XYLineArcVerification $SocketID $GroupName $FileName

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	XY group name (maximum size = 250)
FileName	string	Trajectory file name (maximum size = 250)

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **XYLineArcVerification** (int SocketID, char *GroupName, char *FileName)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
GroupName	char *	XY group name
FileName	char *	Trajectory file name (maximum size = 250)

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **XYLineArcVerification** (ByVal SocketID As Long, ByVal GroupName As String, ByVal FileName As String)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GroupName	String	XY group name
FileName	String	Trajectory file name (maximum size = 250)

Output parameters

None

Return

Function error code

MATLAB

Prototype

int32 **XYLineArcVerification** (int32 SocketID, cstring GroupName, cstring FileName)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupName	cstring	XY group name
FileName	cstring	Trajectory file name (maximum size = 250)

Return

Function error code

PYTHON

Prototype

integer **XYLineArcVerification** (integer SocketID, string GroupName, string FileName)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	XY group name
FileName	string	Trajectory file name (maximum size = 250)

Return

Function error code

2.6.3.7. XYLineArcVerificationResultGet

NAME

XYLineArcVerificationResultGet – Returns the results of the previous “XYLineArcVerification” function.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the group type (must be a XY group): ERR_WRONG_OBJECT_TYPE (-8)
- Check the last XY LineArc verification (must be done): ERR_NOT_ALLOWED_ACTION (-22)
- Check the positioner name length: ERR_STRING_TOO_LONG (-3)
- Check the positioner (must not be a gantry): ERR_POSITIONER_NAME (-18)
- Valid output parameter type: ERR_WRONG_TYPE_CHAR (-13), ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function returns the results of the previous “XYLineArcVerification” function, positioner by positioner. The results are the travel requirements (min and max values), the possible maximum velocity and the possible maximum acceleration.

If no verification has be done then the ERR_NOT_ALLOWED_ACTION (-22) error is returned.

For a more thorough description of the line-arc trajectory capability, please refer to the XPS Motion Tutorial, section named Trajectories / Line-arc Trajectories.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_POSITIONER_NAME (-18)
ERR_STRING_TOO_LONG (-3)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13),
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

XYLineArcVerificationResultGet $SocketID $ PositionerName FileName MinimumPosition MaximumPosition MaximumVelocity MaximumAcceleration

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
PositionerName	string	XY positioner name (maximum size = 250)

Output parameters

FileName	string	Examined trajectory file name (maximum size = 250)
MinimumPosition	double	Minimum position (units)
MaximumPosition	double	Maximum position (units)
MaximumVelocity	double	Maximum trajectory velocity (units / seconds)
MaximumAcceleration	double	Maximum trajectory acceleration (units / seconds²)

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **XYLineArcVerificationResultGet** (int SocketID, char * PositionerName, char * FileName, double * MinimumPosition, double * MaximumPosition, double * MaximumVelocity, double * MaximumAcceleration)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
PositionerName	char *	XY positioner name

Output parameters

FileName	char *	Examined trajectory file name (maximum size = 250)
MinimumPosition	double *	Minimum position (units)
MaximumPosition	double *	Maximum position (units)
MaximumVelocity	double *	Maximum trajectory velocity (units / seconds)
MaximumAcceleration	double *	Maximum trajectory acceleration (units / seconds²)

Return

Function error code

VISUAL BASIC

Prototype

Long **XYLineArcVerificationResultGet** (ByVal SocketID As Long, ByVal PositionerName As String, ByVal FileName As String, MinimumPosition As Double, MaximumPosition As Double, MaximumVelocity As Double, MaximumAcceleration As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
PositionerName	String	XY positioner name

Output parameters

FileName	String	Examined trajectory file name (maximum size = 250)
MinimumPosition	Double	Minimum position (units)
MaximumPosition	Double	Maximum position (units)
MaximumVelocity	Double	Maximum trajectory velocity (units / seconds)
MaximumAcceleration	Double	Maximum trajectory acceleration (units / seconds²)

Return

Function error code

MATLAB

Prototype

[Error, FileName, MinimumPosition, MaximumPosition, MaximumVelocity, MaximumAcceleration] **XYLineArcVerificationResultGet** (int32 SocketID, cstring PositionerName)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
PositionerName	cstring	XY positioner name

Return

Error	int32	Function error code
FileName	cstring	Examined trajectory file name (maximum size = 250)
MinimumPosition	double	Minimum position (units)
MaximumPosition	double	Maximum position (units)
MaximumVelocity	double	Trajectory trajectory velocity (units / seconds)
MaximumAcceleration	double	Trajectory trajectory acceleration (units / seconds²)

PYTHON

Prototype

[Error, FileName, MinimumPosition, MaximumPosition, MaximumVelocity, MaximumAcceleration] **XYLineArcVerificationResultGet** (integer SocketID, string PositionerName)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
PositionerName	string	XY positioner name

Return

Error	integer	Function error code
FileName	string	Examined trajectory file name (maximum size = 250)
MinimumPosition	double	Minimum position (units)
MaximumPosition	double	Maximum position (units)
MaximumVelocity	double	Trajectory velocity (units / seconds)
MaximumAcceleration	double	Trajectory acceleration (units / seconds²)

2.6.4. Configuration files

Example of the definition of an XY group (named “XY”) in the system.ini file. The group XY is build by two positioners named “XGantry” and “Y”. The XY home search sequence is “XThenY “ and no XY mapping is used.

The positioner “XGantry” uses the parameters of “MYSTAGE1” from the stages.ini file and is connected to the plug 1 of the XPS controller. The positioner “XGantry” has a secondary positioner that uses the parameters of “MYSTAGE1” from the stages.ini file and is connected to the plug 3 of the XPS controller.

The positioner “Y” uses the parameters of “MYSTAGE2” from the stages.ini file and is connected to the plug 2 of the XPS controller.

System.ini file :

```
[GROUPS]
XYInUse = XY

[XY] ; XY group configuration
PositionerInUse = XGantry, Y
InitializationAndHomeSearchSequence = XThenY          ; Together,   XThenY

;--- Mapping XY
XMappingFileName =
XMappingLineNumber =
XMappingColumnNumber =          } If  “XMappingFileName” is defined
XMappingMaxPositionError=             ; must be same unit as positioner
YMappingFileName =
YMappingLineNumber =
YMappingColumnNumber =          } If  “YMappingFileName” is defined
YMappingMaxPositionError=             ; must be same unit as positioner

;--- XY Gantry Force Ratio parameters
YOffsetForForceRatio = 0
PrimaryYForceRatio = 0
SecondaryYForceRatio = 0

[XY.XGantry]
PlugNumber = 1
StageName = MYSTAGE1

;---- Secondary positioner (X2)
SecondaryPlugNumber = 3
SecondaryStageName = MYSTAGE1
SecondaryPositionerGantryEndReferencingPosition = 0
SecondaryPositionerGantryEndReferencingTolerance = 0
SecondaryPositionerGantryOffsetAfterInitialization = 0

[XY.Y]
PlugNumber = 2
StageName = MYSTAGE2

;---- Secondary positioner (X2)
; None
```

Stages.ini file :

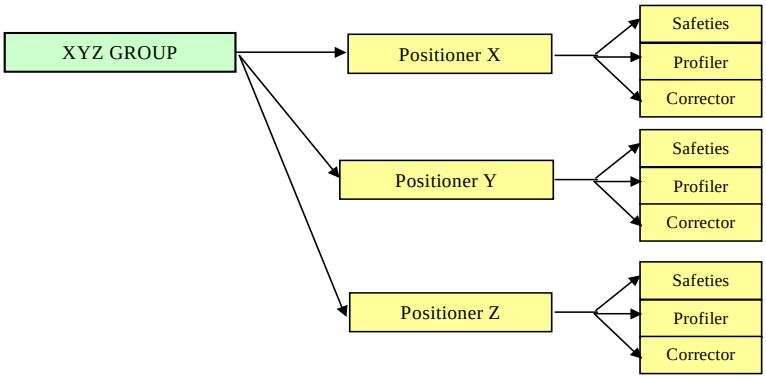
```
[MYSTAGE1]
MYSTAGE positioner configuration => See § “Positioner : Configuration files”

[MYSTAGE2]
MYSTAGE positioner configuration => See § “Positioner : Configurationfiles”
```

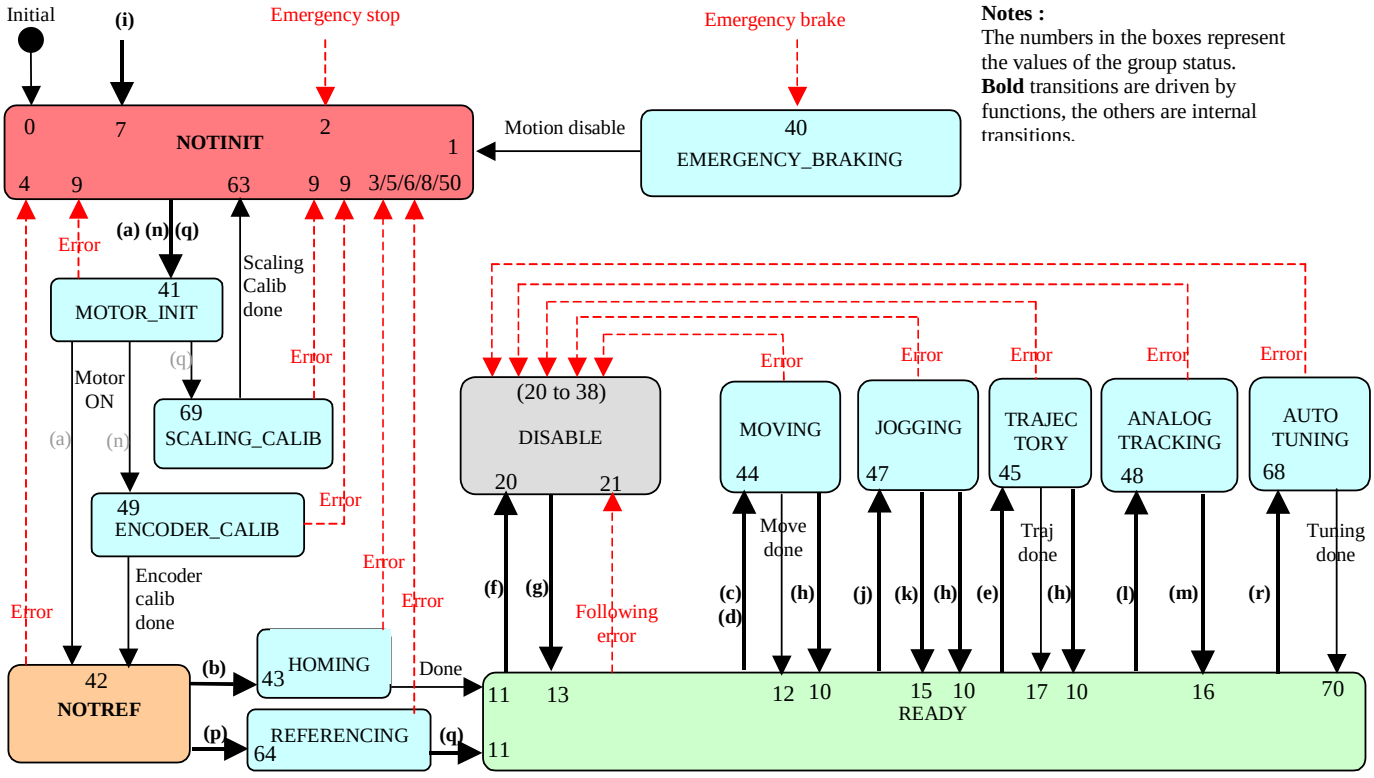
2.7. XYZ group

2.7.1. Description

An XYZ group is a three positioner object, typically in an orthogonal XYZ configuration.
It includes an XZ mapping feature: XYZ = f(XYZ)
It also includes 3D spline trajectories.



2.7.2. State diagram



Notes :
The numbers in the boxes represent the values of the group status.
Bold transitions are driven by functions, the others are internal transitions.

Called function :

(a) GroupInitialize
(b) GroupHomeSearch
(c) GroupMoveAbsolute
(d) GroupMoveRelative
(e) XYLineArcExecution
(f) GroupMotionDisable
(g) GroupMotionEnable
(h) GroupMoveAbort
(i) GroupKill or KillAll
(j) GroupJogModeEnable
(k) GroupJogModeDisable
(l) GroupAnalogTrackingModeEnable
(m) GroupAnalogTrackingModeDisable
(n) GroupInitializeWithEncoderCalibration
(o) GroupReferencingStart
(p) GroupReferencingStop
(q) PositionerAccelerationAutoScaling
(r) PositionerCorrectorAutoTuning

2.7.3. Specific function description

2.7.3.1. XYZGroupPositionCorrectedProfilerGet

NAME

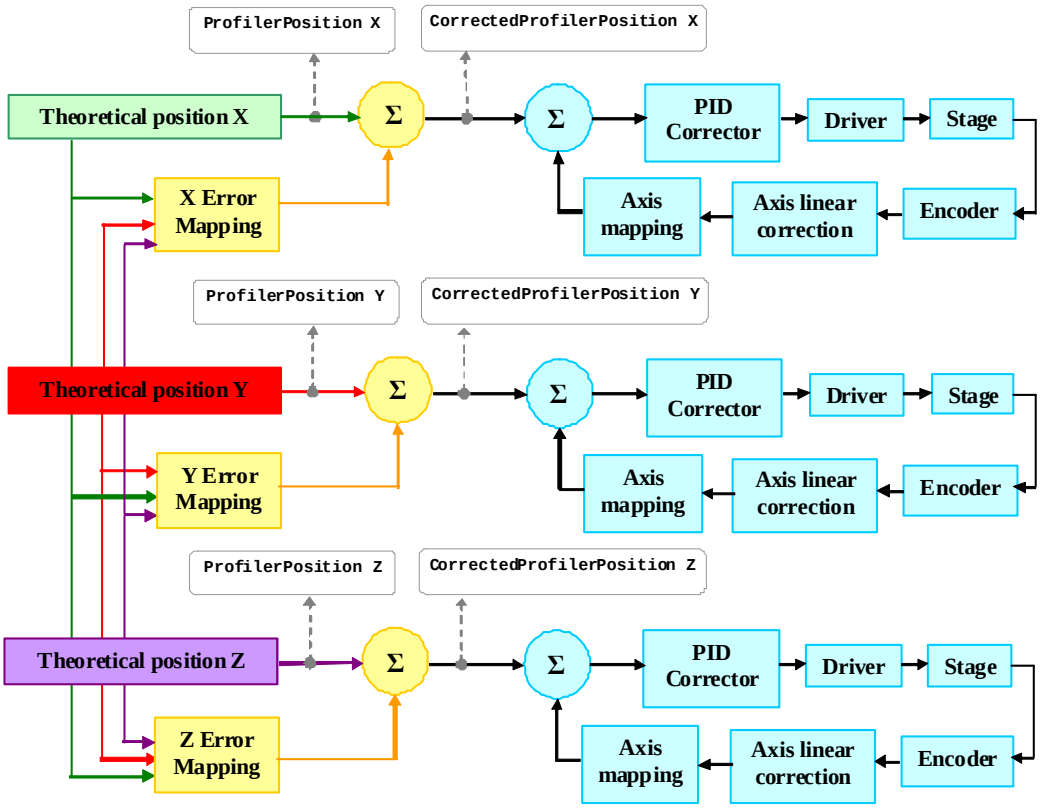
XYSGroupPositionCorrectedProfilerGet – Returns the corrected profiler position for all positioners of an XYZ group.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Valid object type: ERR_WRONG_OBJECT_TYPE (-8)
- Valid group type (must be a XYZ group): ERR_POSITIONER_NAME (-18)
- Valid group name: ERR_GROUP_NAME (-19)
- Valid output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function allows to correct a theoretical position. This corrected position is the theoretical position recalculated with the XYZ mapping correction.



This function applies the XYZ mapping on the theoretical user positions and returns the corrected positions. These corrected profiler positions (X, Y and Z) take the XYZ mapping correction into account.

NOTE:
This function is only allowed with a XYZ group.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GROUP_NAME (-19)
ERR_IN_INITIALIZATION (-21)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0): no error

Prototype

XYZGroupPositionCorrectedProfilerGet SocketID GroupName PositionX PositionY PositionZ CorrectedPositionX CorrectedPositionY CorrectedPositionZ

Input parameters

SocketID	integer	Socket identifier gets by the “TCP_ConnectToServer” function
GroupName	string	XYZ group name (maximum size = 250)
PositionX	floating point	Theoretical position X
PositionY	floating point	Theoretical position Y
PositionZ	floating point	Theoretical position Z

Output parameters

CorrectedPositionX	floating point	Corrected theoretical position X
CorrectedPositionY	floating point	Corrected theoretical position Y
CorrectedPositionZ	floating point	Corrected theoretical position Z

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **XYZGroupPositionCorrectedProfilerGet** (int SocketID, char *GroupName, double PositionX, double PositionY, double PositionZ, double * CorrectedPositionX, double * CorrectedPositionY, double * CorrectedPositionZ)

Input parameters

SocketID	int	Socket identifier gets by the “TCP_ConnectToServer”function
GroupName	char *	XYZ group name
PositionX	double	Theoretical position X
PositionY	double	Theoretical position Y
PositionZ	double	Theoretical position Z

Output parameters

CorrectedPositionX	double *	Corrected theoretical position X
CorrectedPositionY	double *	Corrected theoretical position Y
CorrectedPositionZ	double *	Corrected theoretical position Z

Return

Function error code

VISUAL BASIC

Prototype

Long **XYZGroupPositionCorrectedProfilerGet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal PositionX As Double, ByVal PositionY As Double, ByVal PositionZ As Double, CorrectedPositionX As Double, CorrectedPositionY As Double, CorrectedPositionZ As Double)

Input parameters

SocketID	Long	Socket identifier gets by the “TCP_ConnectToServer” function
GroupName	String	XYZ group name
PositionX	Double	Theoretical position X
PositionY	Double	Theoretical position Y
PositionZ	Double	Theoretical position Z

Output parameters

CorrectedPositionX	Double	Corrected theoretical position X
CorrectedPositionY	Double	Corrected theoretical position Y
CorrectedPositionZ	Double	Corrected theoretical position Z

Return

Function error code

MATLAB

Prototype

[Error, CorrectedPositionX, CorrectedPositionY, CorrectedPositionZ]
XYZGroupPositionCorrectedProfilerGet (int32 SocketID, cstring GroupName, double PositionX, double PositionY, double PositionZ)

Input parameters

SocketID	int32	Socket identifier gets by the “TCP_ConnectToServer”function
GroupName	cstring	XYZ group name
PositionX	double	Theoretical position X
PositionY	double	Theoretical position Y
PositionZ	double	Theoretical position Z

Return

Error	int32	Function error code
CorrectedPositionX	doublePtr	Corrected theoretical position X
CorrectedPositionY	doublePtr	Corrected theoretical position Y
CorrectedPositionZ	doublePtr	Corrected theoretical position Z

PYTHON

Prototype

[Error, CorrectedPositionX, CorrectedPositionY, CorrectedPositionZ]
XYZGroupPositionCorrectedProfilerGet (integer SocketID, string GroupName, double PositionX, double PositionY, double PositionZ)

Input parameters

SocketID	integer	Socket identifier gets by the “TCP_ConnectToServer”function
GroupName	string	XYZ group name
PositionX	double	Theoretical position X
PositionY	double	Theoretical position Y
PositionZ	double	Theoretical position Z

Return

Error	integer	Function error code
CorrectedPositionX	doublePtr	Corrected theoretical position X
CorrectedPositionY	doublePtr	Corrected theoretical position Y
CorrectedPositionZ	doublePtr	Corrected theoretical position Z

2.7.3.2. XYZSplineExecution

NAME

XYZSplineExecution – Executes an XYZ spline trajectory.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Group state must be "READY": ERR_NOT_ALLOWED_ACTION (-22)
- Check the base velocity value (must be null) : ERR_BASE_VELOCITY (-48)
- Check the group: ERR_POSITIONER_NAME (-18)
- Check the group type (must be an XYZ group): ERR_WRONG_OBJECT_TYPE (-8)
- Check backlash (must not be enabled): ERR_NOT_ALLOWED_BACKLASH (-46)
- Check the trajectory file name length: ERR_STRING_TOO_LONG (-3)
- Check the trajectory file existence or file reading: ERR_READ_FILE (-61)
- Check the message queue filling: ERR_MSG_QUEUE (-71)
- Check the input value (velocity and acceleration > 0): ERR_PARAMETER_OUT_OF_RANGE (-17)
- Check the velocity (0 < Velocity <= TrajectoryMaximumVelocity): ERR_TRAJ_VEL_LIMIT (-68)
- Check the acceleration (0 < Acceleration <= TrajectoryMaximumAcceleration): ERR_TRAJ_ACC_LIMIT (-69)
- Check input parameter types: ERR_WRONG_TYPE_DOUBLE (-14), ERR_WRONG_TYPE_INT (-15)

DESCRIPTION

This function executes an XYZ Spline trajectory. The trajectory file must be stored in the folder "\ADMIN\Public\Trajectory" of the XPS controller. If the trajectory can not be initialized (message queue or task error) then the ERR_TRAJ_INITIALIZATION (-72) is returned.
Before a trajectory execution, it is recommended to check its possible execution using the "XYZSplineVerification" and "XYZSplineVerificationResultGet" functions.

For a more thorough description of the spline trajectory capability, please refer to the XPS Motion Tutorial, section named Trajectories / Spline Trajectories.

NOTE:
In case of a ERR_GROUP_MOTION_DONE_TIMEOUT (-33) error, a ERR_FOLLOWING_ERROR (-25) error or ERR_SLAVE (-44) error, the group state becomes DISABLE. To help you to know what is the error source, check the positioner errors, the hardware status and the driver status.

ERROR CODES

ERR_BASE_VELOCITY (-48)
ERR_FATAL_INIT (-20)
ERR_FOLLOWING_ERROR (-25)
ERR_GROUP_MOTION_DONE_TIMEOUT (-33)
ERR_IN_INITIALIZATION (-21)
ERR_MSG_QUEUE (-71)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_NOT_ALLOWED_BACKLASH (-46)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_POSITIONER_NAME (-18)
ERR_READ_FILE (-61)
ERR_SLAVE (-44)
ERR_STRING_TOO_LONG (-3)
ERR_TRAJ_ACC_LIMIT (-69)
ERR_TRAJ_VEL_LIMIT (-68)
ERR_TRAJ_INITIALIZATION (-72)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)

ERR_WRONG_TYPE_DOUBLE (-14),
SUCCESS (0) : no error

TCL

Prototype

XYZSplineExecution $SocketID $GroupName $FileName $Velocity $Acceleration

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	XYZ group name (maximum size = 250)
FileName	string	Trajectory file name (maximum size = 250)
Velocity	double	Trajectory velocity (units / seconds)
Acceleration	double	Trajectory acceleration (units / seconds²)

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **XYZSplineExecution** (int SocketID, char *GroupName, char *FileName , double Velocity, double Acceleration)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
GroupName	char *	XYZ group name
FileName	char *	Trajectory file name (maximum size = 250)
Velocity	double	Trajectory velocity (units / seconds)
Acceleration	double	Trajectory acceleration (units / seconds²)

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **XYZSplineExecution** (ByVal SocketID As Long, ByVal GroupName As String, ByVal FileName As String, ByVal Velocity As Double, ByVal Acceleration As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GroupName	String	XYZ group name
FileName	String	Trajectory file name (maximum size = 250)
Velocity	Double	Trajectory velocity (units / seconds)
Acceleration	Double	Trajectory acceleration (units / seconds²)

Output parameters

None

Return

Function error code

MATLAB

Prototype

int32 **XYZSplineExecution** (int32 SocketID, cstring GroupName, cstring FileName, double Velocity, double Acceleration)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupName	cstring	XYZ group name
FileName	cstring	Trajectory file name (maximum size = 250)
Velocity	double	Trajectory velocity (units / seconds)
Acceleration	double	Trajectory acceleration (units / seconds²)

Return

Function error code

PYTHON

Prototype

integer **XYZSplineExecution** (integer SocketID, string GroupName, string FileName, double Velocity, double Acceleration)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	XYZ group name
FileName	string	Trajectory file name (maximum size = 250)
Velocity	double	Trajectory velocity (units / seconds)
Acceleration	double	Trajectory acceleration (units / seconds²)

Return

Function error code

2.7.3.3. XYZSplineParametersGet

NAME

XYZSplineParametersGet – Returns the Spline trajectory parameters.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the group type (must be a XYZ group): ERR_WRONG_OBJECT_TYPE (-8)
- Check the trajectory type (Spline): ERR_NOT_ALLOWED_ACTION (-22)
- Valid output parameter type: ERR_WRONG_TYPE_CHAR (-13), ERR_WRONG_TYPE_DOUBLE (-14), ERR_WRONG_TYPE_INT (-15)

DESCRIPTION

This function returns the XYZ Spline trajectory parameters (trajectory name, trajectory velocity, trajectory acceleration, current executing element number) of the current executed XYZ Spline trajectory.

For a more thorough description of the Spline trajectory capability, please refer to the XPS Motion Tutorial, section named Trajectories / Spline Trajectories.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
ERR_WRONG_TYPE_DOUBLE (-14)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

XYZSplineParametersGet $SocketID $GroupName FileName Velocity Acceleration ElementNumber

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... XYZ Group name (maximum size = 250)

Output parameters

FileName..... string ..... Executing trajectory file name (maximum size = 250)
Velocity..... double ..... Trajectory velocity (units / seconds)
Acceleration ..... double ..... Trajectory acceleration (units / seconds²)
ElementNumber ..... integer ..... Current executing element number

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **XYZSplineParametersGet** (int SocketID, char *GroupName, char * FileName, double * Velocity, double * Acceleration, int * ElementNumber)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
GroupName	char *	XYZ Group name

Output parameters

FileName	char *	Executing trajectory file name (maximum size = 250)
Velocity	double *	Trajectory velocity (units / seconds)
Acceleration	double *	Trajectory acceleration (units / seconds²)
ElementNumber	int *	Current executing element number

Return

Function error code

VISUAL BASIC

Prototype

Long **XYZSplineParametersGet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal FileName As String, Velocity As Double, Acceleration As Double, ElementNumber As Integer)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GroupName	String	XYZ Group name

Output parameters

FileName	String	Executing trajectory file name (maximum size = 250)
Velocity	Double	Trajectory velocity (units / seconds)
Acceleration	Double	Trajectory acceleration (units / seconds²)
ElementNumber	Integer	Current executing element number

Return

Function error code

MATLAB

Prototype

[Error, FileName, Velocity, Acceleration, ElementNumber] **XYZSplineParametersGet** (int32 SocketID, cstring GroupName)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupName	cstring	XYZ Group name

Return

Error	int32	Function error code
FileName	cstring	Executing trajectory file name (maximum size = 250)
Velocity	double	Trajectory velocity (units / seconds)
Acceleration	double	Trajectory acceleration (units / seconds²)
ElementNumber	int32	Current executing element number

PYTHON

Prototype

[Error, FileName, Velocity, Acceleration, ElementNumber] **XYZSplineParametersGet** (integer SocketID, string GroupName)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	XYZ Group name

Return

Error	integer	Function error code
FileName	string	Executing trajectory file name (maximum size = 250)
Velocity	double	Trajectory velocity (units / seconds)
Acceleration	double	Trajectory acceleration (units / seconds^2)
ElementNumber	integer	Current executing element number

2.7.3.4. XYZSplineVerification

NAME

XYZSplineVerification – Verifies a XYZ Spline trajectory data file.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the group type (must be an XYZ group): ERR_WRONG_OBJECT_TYPE (-8)
- Check the base velocity value (must be null) : ERR_BASE_VELOCITY (-48)
- Check the trajectory file name length: ERR_STRING_TOO_LONG (-3)
- Check the trajectory file existence and the file format: ERR_READ_FILE (-61)
- Check the trajectory (number of elements > 0) : ERR_TRAJ_EMPTY (-66)
- Check trajectory element types in file: ERR_WRONG_TYPE_DOUBLE (-14)
-

DESCRIPTION

This function verifies the possible execution of an XYZ Spline trajectory. The results of the verification can be gathered with the "XYZSplineVerificationResultGet" function. The trajectory file must be stored in the folder "\ADMIN\Public\Trajectory" of the XPS controller. If the trajectory can not be initialized (message queue or task error) then the ERR_TRAJ_INITIALIZATION (-72) is returned.

This function can be executed at any time and is independent from the trajectory execution. This function performs the following:

- ✓ Checks the trajectory file for data coherence.
- ✓ Calculates the trajectory limits, which are: the required travel per positioner, the maximum possible trajectory velocity and the maximum possible trajectory acceleration. This function helps define the parameters for the trajectory execution.
- ✓ If all is OK, it returns "SUCCESS" (0). Otherwise, it returns a corresponding error.

NOTE :
The "XYZSplineVerification" function is independent from the "XYZSplineExecution" function. So users don't need to execute this function before executing a Spline trajectory. However, it is recommended to do that.

For a more thorough description of the Spline trajectory capability, please refer to the XPS Motion Tutorial, section named Trajectories / Spline Trajectories.

ERROR CODES

ERR_BASE_VELOCITY (-48)
ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_POSITIONER_NAME (-18)
ERR_READ_FILE (-61)
ERR_STRING_TOO_LONG (-3)
ERR_TRAJ_EMPTY (-66)
ERR_TRAJ_INITIALIZATION (-72)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

XYZSplineVerification $SocketID $GroupName $FileName

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... XYZ group name (maximum size = 250)
FileName..... string ..... Trajectory file name (maximum size = 250)

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **XYZSplineVerification** (int SocketID, char *GroupName, char *FileName)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... char * ..... XYZ group name
FileName..... char * ..... Trajectory file name (maximum size = 250)

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **XYZSplineVerification** (ByVal SocketID As Long, ByVal GroupName As String, ByVal FileName As String)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... String ..... XYZ group name
FileName..... String ..... Trajectory file name (maximum size = 250)

Output parameters

None

Return

Function error code

MATLAB

Prototype

int32 **XYZSplineVerification** (int32 SocketID, cstring GroupName, cstring FileName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... cstring ..... XYZ group name
FileName..... cstring ..... Trajectory file name (maximum size = 250)

Return

Function error code

PYTHON

Prototype

integer **XYZSplineVerification** (integer SocketID, string GroupName, string FileName)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	XYZ group name
FileName	string	Trajectory file name (maximum size = 250)

Return

Function error code

2.7.3.5. XYZSplineVerificationResultGet

NAME

XYZSplineVerificationResultGet – Returns the results of the previous "**XYZSplineVerification**" function.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the group type (must be a XYZ group): ERR_WRONG_OBJECT_TYPE (-8)
- Check the positioner name length: ERR_STRING_TOO_LONG (-3)
- Check the positioner name: ERR_POSITIONER_NAME (-18)
- Check the last XYZ Spline verification (must be done): ERR_NOT_ALLOWED_ACTION (-22)
- Valid output parameter type: ERR_WRONG_TYPE_CHAR (-13), ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function returns the results of the previous "XYZSplineVerification" function, positioner by positioner. The results are the travel requirements (min and max values), the possible maximum velocity and the possible maximum acceleration.

If no verification has be done then the ERR_NOT_ALLOWED_ACTION (-22) error is returned.

For a more thorough description of the XYZ Spline trajectory capability, please refer to the XPS Motion Tutorial, section named Trajectories / Spline Trajectories.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_POSITIONER_NAME (-18)
ERR_STRING_TOO_LONG (-3)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13),
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

XYZSplineVerificationResultGet $SocketID $PositionerName FileName MinimumPosition MaximumPosition MaximumVelocity MaximumAcceleration

Input parameters

SocketID	integer	Socket identifier got from "TCP_ConnectToServer" function
PositionerName	string	XYZ positioner name (maximum size = 250)

Output parameters

FileName	string	Examined trajectory file name (maximum size = 250)
MinimumPosition	double	Minimum position (units)
MaximumPosition	double	Maximum position (units)
MaximumVelocity	double	Maximum trajectory velocity (units / seconds)
MaximumAcceleration	double	Maximum trajectory acceleration (units / seconds²)

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **XYZSplineVerificationResultGet** (int SocketID, char * PositionerName, char * FileName, double * MinimumPosition, double * MaximumPosition, double * MaximumVelocity, double * MaximumAcceleration)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
PositionerName..... char * ..... XYZ positioner name

Output parameters

FileName..... char * ..... Examined trajectory file name (maximum size = 250)
MinimumPosition ..... double * ..... Minimum position (units)
MaximumPosition..... double * ..... Maximum position (units)
MaximumVelocity..... double * ..... Maximum trajectory velocity (units / seconds)
MaximumAcceleration .... double * ..... Maximum trajectory acceleration (units / seconds²)

Return

Function error code

VISUAL BASIC

Prototype

Long **XYZSplineVerificationResultGet** (ByVal SocketID As Long, ByVal PositionerName As String, ByVal FileName As String, MinimumPosition As Double, MaximumPosition As Double, MaximumVelocity As Double, MaximumAcceleration As Double)

Input parameters

SocketID ..... Long..... Socket identifier got from “TCP_ConnectToServer” function
PositionerName..... String ..... XYZ positioner name

Output parameters

FileName..... String ..... Examined trajectory file name (maximum size = 250)
MinimumPosition ..... Double ..... Minimum position (units)
MaximumPosition..... Double ..... Maximum position (units)
MaximumVelocity..... Double ..... Maximum trajectory velocity (units / seconds)
MaximumAcceleration .... Double ..... Maximum trajectory acceleration (units / seconds²)

Return

Function error code

MATLAB

Prototype

[Error, FileName, MinimumPosition, MaximumPosition, MaximumVelocity, MaximumAcceleration] **XYZSplineVerificationResultGet** (int32 SocketID, cstring PositionerName)

Input parameters

SocketID ..... int32..... Socket identifier got from “TCP_ConnectToServer” function
PositionerName..... cstring ..... XYZ positioner name

Return

Error..... int32..... Function error code
FileName..... cstring ..... Examined trajectory file name (maximum size = 250)
MinimumPosition ..... double ..... Minimum position (units)
MaximumPosition..... double ..... Maximum position (units)
MaximumVelocity..... double ..... Trajectory trajectory velocity (units / seconds)
MaximumAcceleration .... double ..... Trajectory trajectory acceleration (units / seconds²)

PYTHON

Prototype

[Error, FileName, MinimumPosition, MaximumPosition, MaximumVelocity, MaximumAcceleration]
XYZSplineVerificationResultGet (integer SocketID, string PositionerName)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
PositionerName	string	XYZ positioner name

Return

Error	integer	Function error code
FileName	string	Examined trajectory file name (maximum size = 250)
MinimumPosition	double	Minimum position (units)
MaximumPosition	double	Maximum position (units)
MaximumVelocity	double	Trajectory velocity (units / seconds)
MaximumAcceleration	double	Trajectory acceleration (units / seconds²)

2.7.4. Configuration files

Example of the definition of an XYZ group (named “XYZ”) in the system.ini file. The XYZ gour is buildby three positioners named “X” , “Y” and “Z”. The positioner “X” uses the parameters of “MYSTAGE1” from the stages.ini file and is connected to the plug 1 of the XPS controller. The HomeSearchSequence is “Together”, and an XYZ mapping is used.

System.ini file :

```
[GROUPS]
XYZInUse = XYZ

[XYZ]   ; XYZ group configuration
PositionerInUse = X, Y, Z
InitializationAndHomeSearchSequence = Together          ; Together or  XThenYThenZ

;--- Mapping XYZ
XMappingFileName = MatriceX.txt
XMappingXLineNumber = 5
XMappingYColumnNumber = 3
XMappingZDimNumber = 3
XMappingMaxPositionError = 2                            ; must be same unit as positioner

YMappingFileName = MatriceY.txt
YMappingXLineNumber = 5
YMappingYColumnNumber = 3
YMappingZDimNumber = 3
YMappingMaxPositionError = 2                            ; must be same unit as positioner

ZMappingFileName = MatriceZ.txt
ZMappingXLineNumber = 5
ZMappingYColumnNumber = 3
ZMappingZDimNumber = 3
ZMappingMaxPositionError = 2                            ; must be same unit as positioner

[XYZ.X]
PlugNumber = 1
StageName = MYSTAGE1
X positioner configuration => See § “Positioner : Configuration files”

[XYZ.Y]
PlugNumbere = 2
StageName = MYSTAGE2
X positioner configuration => See § “Positioner : Configuration files”

[XYZ.Z]
PlugNumber = 3
StageName = MYSTAGE3
X positioner configuration => See § “Positioner : Configuration files”
```

Stages.ini file :

[MYSTAGE1]
MYSTAGE1 configuration => See § "Positioner : Configuration files"

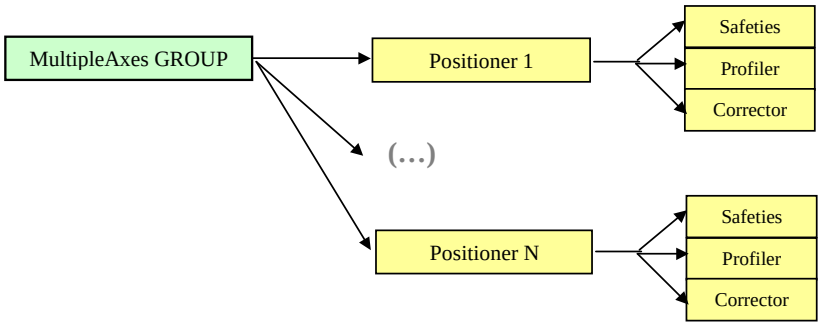
[MYSTAGE2]
MYSTAGE2 configuration => See § "Positioner : Configuration files"

[MYSTAGE3]
MYSTAGE3 configuration => See § "Positioner : Configuration files"

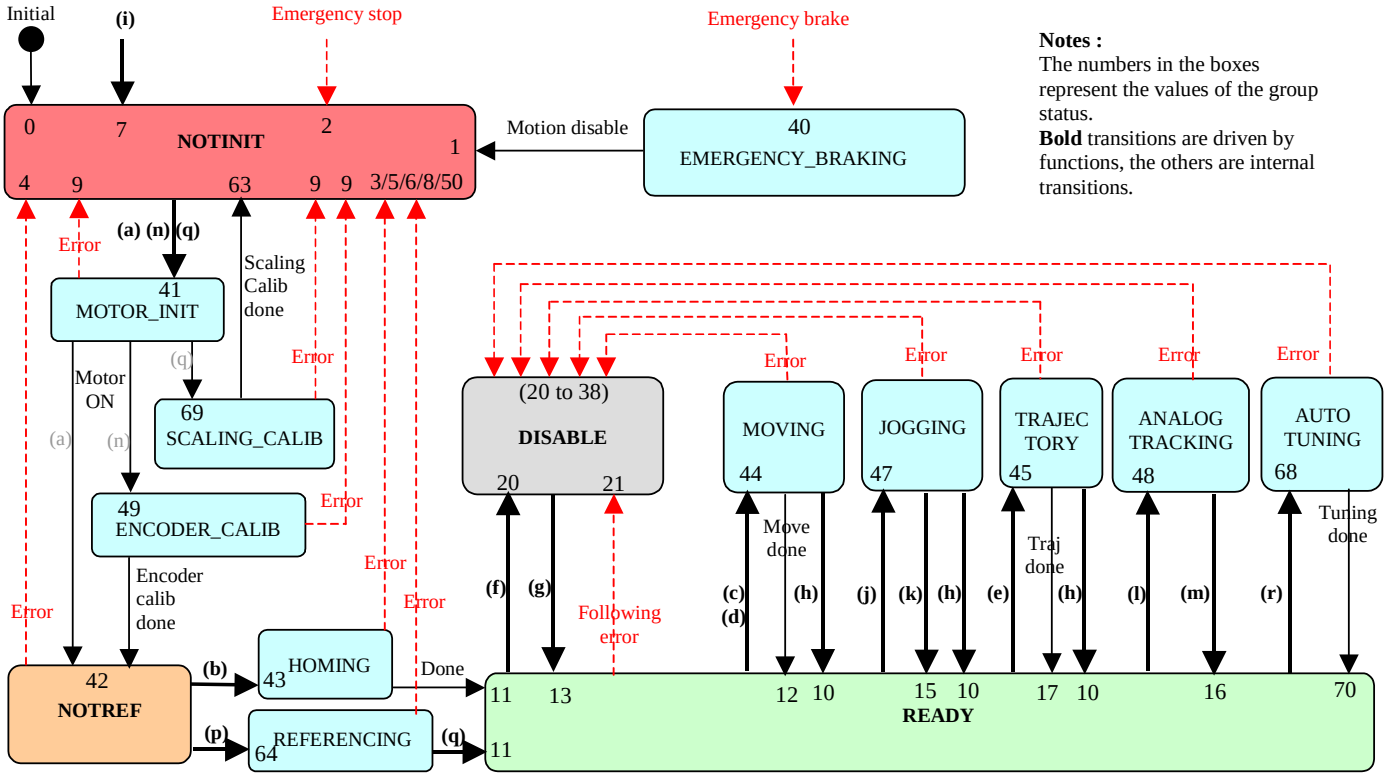
2.8. MultipleAxes group

2.8.1. Description

A MultipleAxes group is a n-positioner(s) object, where n can be any number from one to eight (N max = 8).
It includes the PVT trajectory.



2.8.2. State diagram



Notes :
The numbers in the boxes represent the values of the group status.
Bold transitions are driven by functions, the others are internal transitions.

Called function :

(a) GroupInitialize
(b) GroupHomeSearch
(c) GroupMoveAbsolute
(d) GroupMoveRelative
(e) XYLineArcExecution
(f) GroupMotionDisable
(g) GroupMotionEnable
(h) GroupMoveAbort
(i) GroupKill or KillAll
(j) GroupJogModeEnable
(k) GroupJogModeDisable
(l) GroupAnalogTrackingModeEnable
(m) GroupAnalogTrackingModeDisable
(n) GroupInitializeWithEncoderCalibration
(o) GroupReferencingStart
(p) GroupReferencingStop
(q) PositionerAccelerationAutoScaling
(r) PositionerCorrectorAutoTuning

2.8.3. Specific function description

2.8.3.1. MultipleAxesPVTExecution

NAME

MultipleAxesPVTExecution – Executes a PVT trajectory.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Group state must be "READY": ERR_NOT_ALLOWED_ACTION (-22)
- Check backlash (must not be enabled): ERR_NOT_ALLOWED_BACKLASH (-46)
- Check the base velocity value (must be null) : ERR_BASE_VELOCITY (-48)
- Check the group: ERR_POSITIONER_NAME (-18)
- Check the group type (must be an XYZ group): ERR_WRONG_OBJECT_TYPE (-8)
- Check the input parameter type: ERR_WRONG_TYPE_INT (-15)
- Check the input value (Number of executions > 0): ERR_PARAMETER_OUT_OF_RANGE (-17)
- Check the trajectory file name length: ERR_STRING_TOO_LONG (-3)
- Check the trajectory file existence or file reading: ERR_READ_FILE (-61)
- Check the message queue filling: ERR_MSG_QUEUE (-71)

DESCRIPTION

This function executes a PVT (Position Velocity Time) trajectory. The trajectory file must be stored in the folder "\ADMIN\Public\Trajectory" of the XPS controller. If the trajectory can not be initialized (message queue or task error) then the ERR_TRAJ_INITIALIZATION (-72) is returned.
Before a trajectory execution, it is recommended to check its possible execution using the "MultipleAxesPVTVerification" and "MultipleAxesPVTVerificationResultGet" functions.

For a more thorough description of the PVT trajectory capability, please refer to the XPS Motion Tutorial, section named Trajectories / PVT Trajectories.

NOTE:
In case of a ERR_GROUP_MOTION_DONE_TIMEOUT (-33) error, a ERR_FOLLOWING_ERROR (-25) error or ERR_SLAVE (-44) error, the group state becomes DISABLE. To help you to know what is the error source, check the positioner errors, the hardware status and the driver status.

ERROR CODES

ERR_BASE_VELOCITY (-48)
ERR_FATAL_INIT (-20)
ERR_FOLLOWING_ERROR (-25)
ERR_GROUP_MOTION_DONE_TIMEOUT (-33)
ERR_IN_INITIALIZATION (-21)
ERR_MSG_QUEUE (-71)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_NOT_ALLOWED_BACKLASH (-46)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_POSITIONER_NAME (-18)
ERR_READ_FILE (-61)
ERR_SLAVE (-44)
ERR_STRING_TOO_LONG (-3)
ERR_TRAJ_INITIALIZATION (-72)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_INT (-15)

SUCCESS (0) : no error

TCL

Prototype

MultipleAxesPVTExecution $SocketID $GroupName $TrajectoryFileName $ExecutionNumber

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... MultipleAxes group name (maximum size = 250)
TrajectoryFileName ..... string ..... Trajectory file name (maximum size = 250)
ExecutionNumber ..... integer ..... Number of trajectory executions

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **MultipleAxesPVTExecution** (int SocketID, char *GroupName , char *TrajectoryFileName, int ExecutionNumber)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... char * ..... MultipleAxes group name
TrajectoryFileName ..... char * ..... Trajectory file name (maximum size = 250)
ExecutionNumber ..... int ..... Number of trajectory executions

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **MultipleAxesPVTExecution** (ByVal SocketID As Long, ByVal GroupName As String, ByVal TrajectoryFileName As String, ByVal ExecutionNumber As Long)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... String ..... MultipleAxes group name
TrajectoryFileName ..... String ..... Trajectory file name (maximum size = 250)
ExecutionNumber ..... Long ..... Number of trajectory executions

Output parameters

None

Return

Function error code

MATLAB

Prototype

int32 **MultipleAxesPVTExecution** (int32 SocketID, cstring GroupName, cstring TrajectoryFileName, int32 ExecutionNumber)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupName	cstring	MultipleAxes group name
TrajectoryFileName	cstring	Trajectory file name (maximum size = 250)
ExecutionNumber	int32	Number of trajectory executions

Return

Function error code

PYTHON

Prototype

integer **MultipleAxesPVTExecution** (integer SocketID, string GroupName, string TrajectoryFileName, integer ExecutionNumber)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	MultipleAxes group name
TrajectoryFileName	string	Trajectory file name (maximum size = 250)
ExecutionNumber	integer	Number of trajectory executions

Return

Function error code

2.8.3.2. MultipleAxesPVTParametersGet

NAME

MultipleAxesPVTParametersGet – Returns the PVT trajectory parameters.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the group type (must be a MultipleAxes group): ERR_WRONG_OBJECT_TYPE (-8)
- Check the trajectory type (PVT): ERR_NOT_ALLOWED_ACTION (-22)
- Valid output parameter type: ERR_WRONG_TYPE_CHAR (-13), ERR_WRONG_TYPE_INT (-15)

DESCRIPTION

This function returns the PVT trajectory parameters (trajectory name and current executing element number) of the current executed PVT trajectory.

For a more thorough description of the PVT trajectory capability, please refer to the XPS Motion Tutorial, section named Trajectories / PVT Trajectories.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

MultipleAxesPVTParametersGet $SocketID $GroupName FileName ElementNumber

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... MultipleAxes Group name (maximum size = 250)

Output parameters

FileName..... string ..... Executing trajectory file name (maximum size = 250)
ElementNumber ..... integer ..... Current executing element number

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **MultipleAxesPVTParametersGet** (int SocketID, char *GroupName, char * FileName, int * ElementNumber)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... char * ..... MultipleAxes Group name

Output parameters

FileName..... char * ..... Executing trajectory file name (maximum size = 250)
ElementNumber ..... int * ..... Current executing element number

Return

Function error code

VISUAL BASIC

Prototype

Long **MultipleAxesPVTParametersGet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal FileName As String, ElementNumber As Integer)

Input parameters

SocketID ..... Long..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... String ..... MultipleAxes Group name

Output parameters

FileName..... String ..... Executing trajectory file name (maximum size = 250)
ElementNumber ..... Integer..... Current executing element number

Return

Function error code

MATLAB

Prototype

[Error, FileName, ElementNumber] **MultipleAxesPVTParametersGet** (int32 SocketID, cstring GroupName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... cstring ..... MultipleAxes Group name

Return

Error..... int32 ..... Function error code
FileName..... cstring ..... Executing trajectory file name (maximum size = 250)
ElementNumber ..... int32 ..... Current executing element number

PYTHON

Prototype

[Error, FileName, ElementNumber] **MultipleAxesPVTParametersGet** (integer SocketID, string GroupName)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... MultipleAxes Group name

Return

Error..... integer ..... Function error code
FileName..... string ..... Executing trajectory file name (maximum size = 250)
ElementNumber ..... integer ..... Current executing element number

2.8.3.3. MultipleAxesPVTPulseOutputGet

NAME

MultipleAxesPVTPulseOutputGet – Returns the configuration of pulse generation on PVT trajectory.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the group (must not be a positioner) : ERR_POSITIONER_NAME (-18)
- Check the group type (must be a MultipleAxes group): ERR_WRONG_OBJECT_TYPE (-8)
- Valid output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function returns the last configuration of pulse generation on PVT trajectory.
The pulse output configuration is defined by a start element, an end element, and a time interval in seconds.

Example :
One pulse will be generated every 10 ms between the start of the 3rd element and the end of the 5th element.
Start element= 3
End element = 5
Time interval = 0.01 seconds

For a more thorough description of the PVT trajectory capability, please refer to the XPS Motion Tutorial, sections named Trajectories / PVT Trajectories and Ouput triggers.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

MultipleAxesPVTPulseOutputGet $SocketID $GroupName StartElement EndElement TimeInterval

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... MultipleAxes Group name (maximum size = 250)

Output parameters

StartElement..... integer ..... Start Element number
EndElement..... integer ..... End Element number
TimeInterval..... double ..... Time interval (seconds)

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **MultipleAxesPVTPulseOutputGet** (int SocketID, char *GroupName, int * StartElement, int * EndElement, double * TimeInterval)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
GroupName	char *	MultipleAxes Group name

Output parameters

StartElement	int *	Start Element number
EndElement	int *	End Element number
TimeInterval	double *	Time interval (seconds)

Return

Function error code

VISUAL BASIC

Prototype

Long **MultipleAxesPVTPulseOutputGet** (ByVal SocketID As Long, ByVal GroupName As String, StartElement As Long, EndElement As Long, TimeInterval As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GroupName	String	MultipleAxes Group name

Output parameters

StartElement	Long	Start Element number
EndElement	Long	End Element number
TimeInterval	Double	Time interval (seconds)

Return

Function error code

MATLAB

Prototype

[Error, StartElement, EndElement, TimeInterval] **MultipleAxesPVTPulseOutputGet** (int32 SocketID, cstring GroupName)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupName	cstring	MultipleAxes Group name

Return

Error	int32	Function error code
StartElement	int32	Start Element number
EndElement	int32	End Element number
TimeInterval	double	Time interval (seconds)

PYTHON

Prototype

[Error, StartElement, EndElement, TimeInterval] **MultipleAxesPVTPulseOutputGet** (integer SocketID, string GroupName)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	MultipleAxes Group name

Return

Error	integer	Function error code
StartElement	integer	Start Element number
EndElement	integer	End Element number
TimeInterval	double	Time interval (seconds)

2.8.3.4. MultipleAxesPVTPulseOutputSet

NAME

MultipleAxesPVTPulseOutputSet – Sets the configuration of pulse generation on PVT trajectory.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the group (must not be a positioner) : ERR_POSITIONER_NAME (-18)
- Check the group type (must be a MultipleAxes group): ERR_WRONG_OBJECT_TYPE (-8)
- Check the pulse generation must not be in progress : ERR_NOT_ALLOWED_ACTION (-22)
- Valid input parameter type: ERR_WRONG_TYPE_DOUBLE (-14), ERR_WRONG_TYPE_INT (-15)

DESCRIPTION

This function configures and activates the pulse generation on PVT trajectory. The pulse generation is defined by a start element, an end element, and a time interval in seconds. If a pulse generation is already activated on the selected PVT trajectory then this function returns the ERR_NOT_ALLOWED_ACTION error.
Please note, that the pulse output settings are automatically removed when the trajectory is over. Hence, with the execution of every new trajectory, it is also required to define the pulse output settings again.

This capability allows output of pulses at constant time intervals on a PVT trajectory. The pulses are generated between a first and a last trajectory element. The minimum possible time interval is 100 µs.
Example:
MultipleAxesGroupPVTPulseOutputSet (Group1, 3, 5, 0.01)
One pulse will be generated every 10 ms between the start of the 3rd element and the end of the 5th element.

For a more thorough description of the PVT trajectory capability, please refer to the XPS Motion Tutorial, sections named Trajectories / PVT Trajectories and Ouput triggers.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_POSITIONER_NAME (-18)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype
MultipleAxesPVTPulseOutputSet $SocketID $GroupName $StartElement $EndElement $TimeInterval

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	XY Group name (maximum size = 250)
StartElement	integer	Start Element number
EndElement	integer	End Element number
TimeInterval	double	Time interval (seconds)

Output parameters
None

Return
TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **MultipleAxesPVTPulseOutputSet** (int SocketID, char *GroupName, int StartElement, int EndElement, double TimeInterval)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
GroupName	char *	XY Group name
StartElement	int	Start Element number
EndElement	int	End Element number
TimeInterval	double	Time interval (seconds)

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **MultipleAxesPVTPulseOutputSet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal StartElement As Long, ByVal EndElement As Long, ByVal TimeInterval As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GroupName	String	XY Group name
StartElement	Long	Start Element number
EndElement	Long	End Element number
TimeInterval	Double	Time interval (seconds)

Output parameters

None

Return

Function error code

MATLAB

Prototype

[Error] **MultipleAxesPVTPulseOutputSet** (int32 SocketID, cstring GroupName, int32 StartElement, int32 EndElement, double TimeInterval)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GroupName	cstring	XY Group name
StartElement	int32	Start Element number
EndElement	int32	End Element number
TimeInterval	double	Time interval (seconds)

Return

Error	int32	Function error code

PYTHON

Prototype

[Error] **MultipleAxesPVTPulseOutputSet** (integer SocketID, string GroupName, integer StartElement, integer EndElement, double TimeInterval)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	XY Group name
StartElement	integer	Start Element number
EndElement	integer	End Element number
TimeInterval	double	Time interval (seconds)

Return

Error	integer	Function error code

2.8.3.5. MultipleAxesPVTVerification

NAME

MultipleAxesPVTVerification – Verifies a PVT trajectory data file.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the group type (must be a MultipleAxes group): ERR_WRONG_OBJECT_TYPE (-8)
- Check the base velocity value (must be null) : ERR_BASE_VELOCITY (-48)
- Check the trajectory file name length: ERR_STRING_TOO_LONG (-3)
- Check the trajectory file existence and the file format: ERR_READ_FILE (-61)
- Check the trajectory (number of elements > 0) : ERR_TRAJ_EMPTY (-66)
- Check trajectory element types in file: ERR_WRONG_TYPE_DOUBLE (-14)
- Check the end output velocity (must be null): ERR_TRAJ_FINAL_VELOCITY (-70)
- Check the velocity (Minimum Velocity <= Velocity <= Maximum Velocity): ERR_TRAJ_VEL_LIMIT (-68)
- Check the acceleration (Minimum acc. <= acceleration <= Maximum acc.): ERR_TRAJ_ACC_LIMIT (-69)
- Check the delta time (Delta Time > 0): ERR_TRAJ_TIME (-75)

DESCRIPTION

This function verifies the possible execution of a PVT trajectory. The results of the verification can be gathered with the "MultipleAxesPVTVerificationResultGet" function. The trajectory file must be stored in the folder "\ADMIN\Public\Trajectory" of the XPS controller. If the trajectory can not be initialized (message queue or task error) then the ERR_TRAJ_INITIALIZATION (-72) is returned.

This function can be executed at any time and is independent from the trajectory execution. This function performs the following:

- ✓ Checks the trajectory file for data coherence.
- ✓ Calculates the trajectory limits, which are: the required travel per positioner, the maximum possible trajectory velocity and the maximum possible trajectory acceleration. This function helps define the parameters for the trajectory execution.
- ✓ If all is OK, it returns "SUCCESS" (0). Otherwise, it returns a corresponding error.

NOTE :
The "MultipleAxesPVTVerification" function is independent from the "MultipleAxesPVTExecution" function. So users don't need to execute this function before executing a PVT trajectory. However, it is recommended to do that.

For a more thorough description of the PVT trajectory capability, please refer to the XPS Motion Tutorial, section named Trajectories / PVT Trajectories.

ERROR CODES

ERR_BASE_VELOCITY (-48)
ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_POSITIONER_NAME (-18)
ERR_READ_FILE (-61)
ERR_STRING_TOO_LONG (-3)
ERR_TRAJ_EMPTY (-66)
ERR_TRAJ_ACC_LIMIT (-69)
ERR_TRAJ_FINAL_VELOCITY (-70)
ERR_TRAJ_INITIALIZATION (-72)
ERR_TRAJ_TIME (-75)
ERR_TRAJ_VEL_LIMIT (-68)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)

ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

MultipleAxesPVTVerification $SocketID $GroupName $FileName

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... string ..... MultipleAxes group name (maximum size = 250)
FileName..... string ..... Trajectory file name (maximum size = 250)

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **MultipleAxesPVTVerification** (int SocketID, char *GroupName, char *FileName)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... char * ..... MultipleAxes group name
FileName..... char * ..... Trajectory file name (maximum size = 250)

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **MultipleAxesPVTVerification** (ByVal SocketID As Long, ByVal GroupName As String, ByVal FileName As String)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... String ..... MultipleAxes group name
FileName..... String ..... Trajectory file name (maximum size = 250)

Output parameters

None

Return

Function error code

MATLAB

Prototype

int32 **MultipleAxesPVTVerification** (int32 SocketID, cstring GroupName, cstring FileName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
GroupName..... cstring ..... MultipleAxes group name
FileName..... cstring ..... Trajectory file name (maximum size = 250)

Return

Function error code

PYTHON

Prototype

integer **MultipleAxesPVTVerification** (integer SocketID, string GroupName, string FileName)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GroupName	string	MultipleAxes group name
FileName	string	Trajectory file name (maximum size = 250)

Return

Function error code

2.8.3.6. MultipleAxesPVTVerificationResultGet

NAME

MultipleAxesPVTVerificationResultGet – Returns the results of "**MultipleAxesPVTVerification**" function.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of command parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the group type (must be a MultipleAxes group): ERR_WRONG_OBJECT_TYPE (-8)
- Check the positioner name length: ERR_STRING_TOO_LONG (-3)
- Check the positioner name: ERR_POSITIONER_NAME (-18)
- Check the last MultipleAxes PVT verification (must be done): ERR_NOT_ALLOWED_ACTION (-22)
- Valid output parameter type: ERR_WRONG_TYPE_CHAR (-13), ERR_WRONG_TYPE_DOUBLE (-14)

DESCRIPTION

This function returns the results of the previous "MultipleAxesPVTVerification" function, positioner by positioner. The results are the travel requirements (min and max values), the possible maximum velocity and the possible maximum acceleration.

If no verification has be done then the ERR_NOT_ALLOWED_ACTION (-22) error is returned.

For a more thorough description of the PVT trajectory capability, please refer to the XPS Motion Tutorial, section named Trajectories / PVT Trajectories.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_POSITIONER_NAME (-18)
ERR_STRING_TOO_LONG (-3)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13),
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

MultipleAxesPVTVerificationResultGet $SocketID $PositionerName FileName MinimumPosition MaximumPosition MaximumVelocity MaximumAcceleration

Input parameters

SocketID	integer	Socket identifier got from "TCP_ConnectToServer" function
PositionerName	string	MultipleAxes positioner name (maximum size = 250)

Output parameters

FileName	string	Examined trajectory file name (maximum size = 250)
MinimumPosition	double	Minimum position (units)
MaximumPosition	double	Maximum position (units)
MaximumVelocity	double	Maximum trajectory velocity (units / seconds)
MaximumAcceleration	double	Maximum trajectory acceleration (units / seconds²)

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **MultipleAxesPVTVerificationResultGet** (int SocketID, char *PositionerName, char * FileName, double * MinimumPosition, double * MaximumPosition, double * MaximumVelocity, double * MaximumAcceleration)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
PositionerName	char *	MultipleAxes positioner name

Output parameters

FileName	char *	Examined trajectory file name (maximum size = 250)
MinimumPosition	double *	Minimum position (units)
MaximumPosition	double *	Maximum position (units)
MaximumVelocity	double *	Maximum trajectory velocity (units / seconds)
MaximumAcceleration	double *	Maximum trajectory acceleration (units / seconds²)

Return

Function error code

VISUAL BASIC

Prototype

Long **MultipleAxesPVTVerificationResultGet** (ByVal SocketID As Long, ByVal PositionerName As String, ByVal FileName As String, MinimumPosition As Double, MaximumPosition As Double, MaximumVelocity As Double, MaximumAcceleration As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
PositionerName	String	MultipleAxes positioner name

Output parameters

FileName	String	Examined trajectory file name (maximum size = 250)
MinimumPosition	Double	Minimum position (units)
MaximumPosition	Double	Maximum position (units)
MaximumVelocity	Double	Maximum trajectory velocity (units / seconds)
MaximumAcceleration	Double	Maximum trajectory acceleration (units / seconds²)

Return

Function error code

MATLAB

Prototype

[Error, FileName, MinimumPosition, MaximumPosition, MaximumVelocity, MaximumAcceleration] **MultipleAxesPVTVerificationResultGet** (int32 SocketID, cstring PositionerName)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
PositionerName	cstring	MultipleAxes positioner name

Return

Error	int32	Function error code
FileName	cstring	Examined trajectory file name (maximum size = 250)
MinimumPosition	double	Minimum position (units)
MaximumPosition	double	Maximum position (units)
MaximumVelocity	double	Trajectory trajectory velocity (units / seconds)
MaximumAcceleration	double	Trajectory trajectory acceleration (units / seconds²)

PYTHON

Prototype

[Error, FileName, MinimumPosition, MaximumPosition, MaximumVelocity, MaximumAcceleration]
MultipleAxesPVTVerificationResultGet (integer SocketID, string PositionerName)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
PositionerName	string	MultipleAxes positioner name

Return

Error	integer	Function error code
FileName	string	Examined trajectory file name (maximum size = 250)
MinimumPosition	double	Minimum position (units)
MaximumPosition	double	Maximum position (units)
MaximumVelocity	double	Trajectory velocity (units / seconds)
MaximumAcceleration	double	Trajectory acceleration (units / seconds²)

2.8.4. Configuration files

Example of the definition of a MultipleAxes group (named “MULTI”) in the system.ini file. The group MULTI is build by two positioners named “ONE” and “TWO”. The positioner “ONE” uses the parameters of “MYSTAGE1” from the stages.ini file and is connected to the plug 1 of the XPS controller. The HomeSearchSequence is “Together”.

System.ini file :

```
[GROUPS]
MultipleAxesInUse = MULTI

[MULTI]                                          ; AXIS MultipleAxes group configuration
PositionerInUse = ONE, TWO
PositionerNumber = 2
InitializationAndHomeSearchSequence = Together   ; Together or OneAfterAnother

[MULTI.ONE]
PLugNumber = 1
StageName = MYSTAGE1
STAGE configuration => See § “Positioner : Configurationfiles”

[MULTI.TWO]
PLugNumber = 2
StageName = MYSTAGE2
STAGE configuration => See § “Positioner : Configuration files”
```

Stages.ini file :

```
[MYSTAGE1]
MYSTAGE1 configuration => See § “Positioner : Configuration files”

[MYSTAGE2]
MYSTAGE2 configuration => See § “Positioner : Configuration files”
```

2.9. Analog and digital I/O

2.9.1. GPIO name list

2.9.1.1. Digital inputs

GPIO1.DI	Digital Input of the I/O board connector # 1 (8 bits)
GPIO2.DI	Digital Input of the I/O board connector # 2 (6 bits)
GPIO3.DI	Digital Input of the I/O board connector # 3 (6 bits)
GPIO4.DI	Digital Input of the I/O board connector # 4 (16 bits)

2.9.1.2. Digital outputs

GPIO1.DO	Digital Output of the I/O board connector # 1 (8 bits)
GPIO3.DO	Digital Output of the I/O board connector # 3 (6 bits)
GPIO4.DO	Digital Output of the I/O board connector # 4 (16 bits)

2.9.1.3. Analog inputs

GPIO2.ADC1	Analog Input # 1 of the I/O board connector # 2
GPIO2.ADC2	Analog Input # 2 of the I/O board connector # 2
GPIO2.ADC3	Analog Input # 3 of the I/O board connector # 2
GPIO2.ADC4	Analog Input # 4 of the I/O board connector # 2

2.9.1.4. Analog outputs

GPIO2.DAC1	Analog Output # 1 of the I/O board connector # 2
GPIO2.DAC2	Analog Output # 2 of the I/O board connector # 2
GPIO2.DAC3	Analog Output # 3 of the I/O board connector # 2
GPIO2.DAC4	Analog Output # 4 of the I/O board connector # 2

2.9.2. Function description

2.9.2.1. GPIOAnalogGainGet

NAME

GPIOAnalogGainGet – Gets the gain for one or several analog inputs (ADC)

INPUT TESTS

- XPS configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Command format: ERR_WRONG_FORMAT (-7)
- Number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Output parameter type: ERR_WRONG_TYPE_INT (-15)
- Check board: ERR_WRONG_OBJECT_TYPE (-8)
- GPIO name (ADC): ERR_WRONG_OBJECT_TYPE (-8)
- Hardware compatibility or XPS initialization in progress: ERR_NOT_ALLOWED_ACTION (-22)

DESCRIPTION

Gets the gain value for one or several analog inputs. Please refer to the appendix *B.5 Analog I/O* of the XPS Motion Tutorial for further information about the ADC gain.

The gain value must be 1, 2, 4 or 8.

The maximum number of INT boards, that can be plugged inside the XPS controller, is setting to 2. So, you can increase the number of analog outputs: 4 to 8 ADC.

CAUTION:
The programmable GPIO's are not available anymore since hardware **E4224**.
This version number can be checked in the XPS web site. Choose the "Administrator" connection and select the "CONTROLLER CONFIGURATION" menu. Next, click on the "General" sub-menu … the version numbers are displayed in the "Internal hardware" informations.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

GPIOAnalogGainGet $SocketID $GPIOName AnalogGainValue…

Input parameters

SocketID ..... integer ..... Socket identifier got from "TCP_ConnectToServer" function
GPIOName..... string ..... Analog input name (maximum size = 250)

Output parameters

AnalogGainValue ..... integer ..... value of analog input gain

Return

TCL error (0 = success or 1 = syntax error) or Function error code

Note : *$GPIOName* and *AnalogGainValue* pair can be repeted to get several channels.

C / C++

Prototype

int **GPIOAnalogGainGet** (int SocketID, int NbElements, char* GPIONames, double* AnalogGainValues)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
NbElements..... int ..... number of analog GPIO to read.
GPIONames ..... char * ..... List of analog input names – separator is semicolon

Output parameters

AnalogGainValues ..... int * ..... array of analog input gain

Return

Function error code

Note : *GPIONames* and *AnalogGainValues* must have array size in accordance to *NbElements* to avoid array overrun.

VISUAL BASIC

Prototype

Long **GPIOAnalogGainGet** (ByVal SocketID As Long, ByVal NbElements As Long, GPIONameList As String, AnalogGainValueArray As Long)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
NbElements..... Long ..... number of analog GPIO to read.
GPIONameList ..... String..... List of analog input names – separator is semicolon

Output parameters

AnalogGainValueArray ..... Long ..... array of analog input gain

Return

Function error code

Note : *GPIONameList* and *AnalogGainValueArray* must have array size in accordance to *NbElements* to avoid array overrun.

MATLAB

Prototype

[Error, AnalogGainValues] **GPIOAnalogGainGet** (int32 SocketID, cstring GPIONames)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
GPIONames ..... cstring ..... Analog input names (separated by space (blank) character)

Return

Error..... int32 ..... Function error code
AnalogGainValues..... int32Ptr ..... array of analog input gain

Note : *AnalogGainValues* must have the same elements number as that of *GPIONames*.

PYTHON

Prototype

[Error, AnalogGainValues] **GPIOAnalogGainGet** (integer SocketID, string list GPIONames)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GPIONames ..... string list ..... Analog input names list

Return

Error	integer	Function error code
AnalogGainValues	integer list	list of analog input gain values

Note : *AnalogGainValues* have the same elements number as that of *GPIONames.*

2.9.2.2. GPIOAnalogGainSet

NAME

GPIOAnalogGainSet – Sets a gain for one or several analog inputs (ADC)

INPUT TESTS

- XPS configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Command format: ERR_WRONG_FORMAT (-7)
- Number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Input parameter type: ERR_WRONG_TYPE_INT (-15)
- Check board: ERR_WRONG_OBJECT_TYPE (-8)
- GPIO name (ADC): ERR_WRONG_OBJECT_TYPE (-8)
- Hardware compatibility or XPS initialization in progress: ERR_NOT_ALLOWED_ACTION (-22)
- Check outpout value (1, 2, 4 or 8): ERR_PARAMETER_OUT_OF_RANGE (-17)

DESCRIPTION

Sets a gain value for one or several analog inputs.
The gain value can be: 1, 2, 4 or 8

If the conversion of the gain value to bits failed then the ERR_NOT_ALLOWED_ACTION is returned.

The maximum number of INT boards, that can be plugged inside the XPS controller, is setting to 2. So, you can increase the number of analog outputs: 4 to 8 ADC.

CAUTION:
The programmable GPIO's are not available anymore since hardware **E4224**.
This version number can be checked in the XPS web site. Choose the "Administrator" connection and select the "CONTROLLER CONFIGURATION" menu. Next, click on the "General" sub-menu … the version numbers are displayed in the "Internal hardware" informations.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

GPIOAnalogGainSet $SocketID $GPIOName $AnalogGainValue…

Input parameters

SocketID	integer	Socket identifier got from "TCP_ConnectToServer" function
GPIOName	string	Analog input name (maximum size = 250)
AnalogGainValue	integer	value of analog input gain

Output parameters

None

Return

TCL error (0 = success or 1 = syntax error) or Function error

Note : *$GPIOName* and $*AnalogGainValue* pair can be repeted to set several channels.

C / C++

Prototype

int **GPIOAnalogGainSet** (int SocketID, int NbElements, char* GPIONames, double* AnalogGainValues)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
NbElements	int	number of analog GPIO to set.
GPIONames	char *	List of analog input names – separator is semicolon
AnalogGainValues	int *	arrays of analog input gain

Output parameters

None

Return

Function error code

Note : *GPIONames* and *AnalogGainValues* must have elements number in accordance to *NbElements* to avoid array overrun.

VISUAL BASIC

Prototype

Long **GPIOAnalogGainSet** (ByVal SocketID As Long, ByVal NbElements As Long, GPIONames As String, AnalogGainValues As Long)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
NbElements	Long	number of analog GPIO to set.
GPIONames	String	List of analog input names – separator is semicolon
AnalogGainValues	Long array	array of analog input gain

Output parameters

None

Return

Function error code

Note : *GPIONames* and *AnalogGainValues* must have elements number in accordance to *NbElements* to avoid array overrun.

MATLAB

Prototype

[Error] **GPIOAnalogGainSet** (int32 SocketID, cstring GPIONames, int32 array AnalogGainValues)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GPIONames	cstring	Analog input names (separated by space (blank) character)
AnalogGainValues	int32 array	array of analog input gain

Return

Error	int32	Function error code

Return

Function error

Note : *AnalogGainValues* must have the same elements number as that of *GPIONames*.

PYTHON

Prototype

[Error] **GPIOAnalogGainSet** (integer SocketID, string list GPIONames, integer list AnalogGainValues)

Input parameters

SocketID	integer	Socket identifier got from "TCP_ConnectToServer" function
GPIONames	string list	Analog input name list
AnalogGainValues	integer list	list of analog input gain value

Return

Error	integer	Function error code

Note : *AnalogGainValues* must have the same elements number as that of *GPIONameList*.

2.9.2.3. GPIOAnalogGet

NAME

GPIOAnalogGet – Read one or several analog GPIO (DAC or ADC)

INPUT TESTS

- XPS configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Command format: ERR_WRONG_FORMAT (-7)
- Number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)
- GPIO name (ADC or DAC): ERR_WRONG_OBJECT_TYPE (-8)
- Hardware compatibility or XPS initialization in progress: ERR_NOT_ALLOWED_ACTION (-22)

DESCRIPTION

Read one or several analog IO and returns the value(s) in an array.
The GPIO must be one or several analog inputs (ADC) and/or analog outputs (DAC) of GPIO2 connector.
See analog input list §2.9.1.3 and analog output list §2.9.1.4

NOTE:
The GPIO2 connector is present on the INT board inside the controller. The maximum number of INT boards, that can be plugged inside the XPS controller, is setting to 2. So, you can increase the number of analog IOs: 4 to 8 ADC and 4 to 8 DAC.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

GPIOAnalogGet $SocketID $GPIOName AnalogValue…

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GPIOName..... string ..... Analog input name (maximum size = 250)

Output parameters

AnalogValue ..... floating number..... value of analog input

Return

TCL error (0 = success or 1 = syntax error) or Function error code

Note : *$GPIOName* and *AnalogValue* pair can be repeted to get several channels.

C / C++

Prototype

int **GPIOAnalogGet** (int SocketID, int NbElements, char* GPIONames, double* AnalogValues)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
NbElements..... int ..... number of analog GPIO to read.

GPIONames.....char * .....List of analog input names – separator is semicolon

Output parameters

AnalogValues.....double * .....array of analog inputs

Return

Function error code

Note : *GPIONames* and *AnalogValues* must have the same elements number as that of *NbElements.*

VISUAL BASIC

Prototype

Long **GPIOAnalogGet** (ByVal SocketID As Long, ByVal NbElements As Long, GPIONames As String, AnalogValues As Long)

Input parameters

SocketID .....Long.....Socket identifier got from "TCP_ConnectToServer" function
NbElements.....Long.....number of analog GPIO to read.
GPIONames .....String.....List of analog input names – separator is semicolon

Output parameters

AnalogValues.....Double.....array of analog inputs

Return

Function error code

Note : *GPIONames* and *AnalogValues* must have the same elements number as that of *NbElements.*

MATLAB

Prototype

[Error, AnalogValues] **GPIOAnalogGet** (int32 SocketID, cstring GPIONames)

Input parameters

SocketID .....int32.....Socket identifier got from "TCP_ConnectToServer" function
GPIONames.....cstring .....Analog input names (separated by space (blank) characracter)

Return

Error.....int32.....Function error code
AnalogValues.....doublePtr.....array of analog inputs

Note : *AnalogValues* must have the same elements number as that of *GPIONames*.

PYTHON

Prototype

[Error, AnalogValues] **GPIOAnalogGet** (integer SocketID, string list GPIONames)

Input parameters

SocketID .....integer.....Socket identifier got from "TCP_ConnectToServer" function
GPIONames.....string list .....Analog input name list

Return

Error.....integer.....Function error code
AnalogValues.....double list .....list of analog inputs

Note : *AnalogValues* have the same elements number as that of the *GPIONames.*

2.9.2.4. GPIOAnalogSet

NAME

GPIOAnalogSet – Sets one or several analog output (DAC)

INPUT TESTS

- XPS configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Command format: ERR_WRONG_FORMAT (-7)
- Number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)
- Check board: ERR_WRONG_OBJECT_TYPE (-8)
- GPIO name (DAC): ERR_WRONG_OBJECT_TYPE (-8)
- Hardware compatibility or XPS initialization in progress: ERR_NOT_ALLOWED_ACTION (-22)
- Check outpout value [-10V..10V]: ERR_PARAMETER_OUT_OF_RANGE (-17)

DESCRIPTION

Sets the analog value for one or several analog outputs (DAC) of GPIO2 connector.
See analog output list §2.9.1.4

NOTE:
The GPIO2 connector is present on the INT board inside the controller. The maximum number of INT boards, that can be plugged inside the XPS controller, is setting to 2. So, you can increase the number of analog outputs: 4 to 8 DAC.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_DOUBLE (-14)
SUCCESS (0) : no error

TCL

Prototype

GPIOAnalogSet $SocketID $GPIOName $AnalogValue …

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GPIOName..... string ..... Analog GPIO name (maximum size = 250)
AnalogValue ..... floating point..... value of analog GPIO (DAC)

Output parameters

None

Return

TCL error (0 = success or 1 = syntax error) or Function error

Note : *$GPIOName* and $*AnalogValue* pair can be repeted to set several channels.

C / C++

Prototype

int **GPIOAnalogSet** (int SocketID, int NbElements, char* GPIONames, double *AnalogValues)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
NbElements	int	number of analog GPIO to set.
GPIONames	char *	List of analog GPIO names – separator is semicolon
AnalogValues	double *	Analog GPIO value array (DAC)

Output parameters

None

Return

Function error

Note : *GPIONames* and *AnalogValues* must have array size in accordance to *NbElements* to avoid array overrun.

VISUAL BASIC

Prototype

Long **GPIOAnalogSet** (ByVal SocketID As Long, ByVal NbElements As Long, GPIONames As String, AnalogValues As Double)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
NbElements	Long	number of analog GPIO to set.
GPIONames	String	List of analog GPIO names – separator is semicolon
AnalogValues	Double array	Analog GPIO value array (DAC)

Output parameters

None

Return

Function error

Note : *GPIONames* and *AnalogValues* must have array size in accordance to *NbElements* to avoid array overrun.

MATLAB

Prototype

[Error] **GPIOAnalogSet** (int32 SocketID, cstring GPIONames, double array AnalogValues)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GPIONames	cstring	Analog GPIO names (separated by space (blank) character)
AnalogValues	double array	Analog GPIO value array (DAC)

Return

Function error

Note : *AnalogValues* must have the same elements number as that of *GPIONames*.

PYTHON

Prototype

[Error] **GPIOAnalogSet** (integer SocketID, string list GPIONames, integer list AnalogValues)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GPIONames	string list	Analog output name list
AnalogValues	double list	list of analog output value

Return

Error.....integer.....Function error code

Note : *AnalogValues* must have the same elements number as that of *GPIONames*.

2.9.2.5. GPIODigitalGet

NAME

GPIODigitalGet – Read one digital input or output.

INPUT TESTS

- XPS configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Command format: ERR_WRONG_FORMAT (-7)
- Number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Output parameter type: ERR_WRONG_TYPE_UNSIGNEDINT (-16)
- GPIO name (DI or DO): ERR_WRONG_OBJECT_TYPE (-8)
- Hardware compatibility or XPS initialization in progress: ERR_NOT_ALLOWED_ACTION (-22)

DESCRIPTION

Returns the value of the digital input (DI) or of the digital output (DO).
See digital output list §2.9.1.2 and digital input list §2.9.1.1

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_UNSIGNEDINT (-16)
SUCCESS (0) : no error

TCL

Prototype

GPIODigitalGet $SocketID $GPIOName DigitalValue

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GPIOName	string	Digital GPIO name (maximum size = 250)

Output parameters

DigitalValue	interger	Digital value (DI or DO)

Return

TCL error (0 = success or 1 = syntax error) or Function error

C / C++

Prototype

int **GPIODigitalGet** (int SocketID, char* GPIOName, unsigned int* DigitalValue)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
GPIOName	char *	Digital GPIO name (maximum size = 250)

Output parameters

DigitalValue	unsigned int *	Digital value (DI or DO)

Return

Function error

VISUAL BASIC

Prototype

Long **GPIODigitalGet** (ByVal SocketID As Long, GPIOName As String, DigitalValue As Integer)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GPIOName	String	Digital GPIO name (maximum size = 250)

Output parameters

DigitalValue	Integer	Digital value (DI or DO)

Return

Function error

MATLAB

Prototype

[Error, DigitalValue] **GPIODigitalGet** (int32 SocketID, cstring GPIOName)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GPIOName	cstring	Digital GPIO name (maximum size = 250)

Return

Function error

DigitalValue	uint16Ptr	Digital value (DI or DO)

PYTHON

Prototype

[Error, DigitalValue] **GPIODigitalGet** (integer SocketID, string GPIOName)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GPIOName	string	Digital GPIO name (maximum size = 250)

Return

Error	integer	Function error
DigitalValue	unsigned short *	Digital value (DI or DO)

2.9.2.6. GPIODigitalSet

NAME

GPIODigitalSet – Sets one digital output.

INPUT TESTS

- XPS configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Command format: ERR_WRONG_FORMAT (-7)
- Number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Output parameter type: ERR_WRONG_TYPE_UNSIGNEDINT (-16)
- GPIO name (DO): ERR_WRONG_OBJECT_TYPE (-8)
- Hardware compatibility or XPS initialization in progress: ERR_NOT_ALLOWED_ACTION (-22)

DESCRIPTION

Sets the value of the selected digital output (DO).
See digital output list §2.9.1.2

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_NOT_ALLOWED_ACTION (-22)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_UNSIGNEDINT (-16)
SUCCESS (0) : no error

TCL

Prototype

GPIODigitalSet $SocketID $GPIOName $Mask $DigitalOutputValue

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GPIOName	string	Digital output name (maximum size = 250)
Mask	integer	Mask
DigitalOutputValue	integer	Digital output value

Output parameters

None

Return

TCL error (0 = success or 1 = syntax error) or Function error

C / C++

Prototype

int **GPIODigitalSet** (int SocketID, char* GPIOName, unsigned short Mask, unsigned short DigitalOutputValue)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
GPIOName	char *	Digital output name (maximum size = 250)
Mask	unsigned short	Mask
DigitalOutputValue	unsigned short	Digital output value

Output parameters

None

Return

Function error

VISUAL BASIC

Prototype

Long **GPIODigitalSet** (ByVal SocketID As Long, GPIOName As String, ByVal Mask As Integer, ByVal DigitalOutputValue As Integer)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
GPIOName	String	Digital output name (maximum size = 250)
Mask	Integer	Mask
DigitalOutputValue	Integer	Digital output value

Output parameters

None

Return

Function error

MATLAB

Prototype

[Error] **GPIODigitalSet** (int32 SocketID, cstring GPIOName, uint16 Mask, uint16 DigitalOutputValue)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GPIOName	cstring	Digital output name (maximum size = 250)
Mask	uint16	Mask
DigitalOutputValue	uint16	Digital output value

Return

Error	int32	Function error code

PYTHON

Prototype

[Error] **GPIODigitalSet** (integer SocketID, string GPIOName, unsigned short Mask, unsigned short DigitalOutputValue)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GPIOName	string	Digital output name (maximum size = 250)
Mask	unsigned short	Mask
DigitalOutputValue	unsigned short	Digital output value

Return

Error	integer	Function error code

2.10. Gathering

2.10.1. Function description

2.10.1.1. GatheringConfigurationGet

NAME

GatheringConfigurationGet – Returns the current configuration of the internal triggered data gathering.

INPUT TESTS

- XPS configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Command format: ERR_WRONG_FORMAT (-7)
- Number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Output parameter type: ERR_WRONG_TYPE_ CHAR (-13)
- Gathering must be configured: ERR_GATHERING_NOT_CONFIGURED (-32)

DESCRIPTION

This function returns the current configuration of the internal triggered data gathering.
Use the “GatheringListGet” function to retrieve a complete list of allowed gathering types.

For a more thorough description of the internal data gathering capability, please refer to the XPS Motion Tutorial, section named Data Gathering / Internal Data Gathering.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GATHERING_NOT_CONFIGURED (-32)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_TYPE_CHAR (-13)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype

GatheringConfigurationGet $SocketID TypeList

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

TypeList ..... string ..... List of configured gathering types (separator is semicolon)

Return

TCL error (0 = success or 1 = syntax error) or Function error

C / C++

Prototype

int **GatheringConfigurationGet** (int SocketID, char * TypeList)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

TypeList ..... char * ..... List of configured gathering types (separator is semicolon)

Return

Function error

VISUAL BASIC

Prototype

Long **GatheringConfigurationGet** (ByVal SocketID As Long, ByVal TypeList As String)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

TypeList ..... String ..... List of configured gathering types (separator is semicolon)

Return

Function error

MATLAB

Prototype

[Error, TypeList] **GatheringConfigurationGet** (int32 SocketID)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function

Return

Error ..... int32 ..... Function error

TypeList ..... cstring ..... List of configured gathering types (separator is semicolon)

PYTHON

Prototype

[Error, TypeList] **GatheringConfigurationGet** (integer SocketID)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function

Return

Error ..... integer ..... Function error

TypeList ..... string ..... List of configured gathering types (separator is semicolon)

2.10.1.2. GatheringConfigurationSet

NAME

GatheringConfigurationSet – Configures a gathering.

INPUT TESTS

- XPS configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Command format: ERR_WRONG_FORMAT (-7)
- Number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check input gathering mnemonic: ERR_MNEMOTYPEGATHERING (-29)
- Gathering must not be in progress: ERR_GATHERING_RUNNING (-43)

DESCRIPTION

Defines one or several type of data gathered during the internal triggered data gathering.

Maximum of 1000000 points can be acquired.
Maximum of 25 data types can be configured in a gathering.

Gathering data types:

PositionerName.CorrectorOutput
PositionerName.CurrentAcceleration
PositionerName.CurrentPosition
PositionerName.CurrentVelocity
PositionerName.FollowingError
PositionerName.SetpointAcceleration
PositionerName.SetpointPosition
PositionerName.SetpointVelocity
PositionerName.ExcitationSignalInput
GPIO1.DI
GPIO1.DO
GPIO2.DI
GPIO3.DI
GPIO3.DO
GPIO4.DI
GPIO4.DO
GPIO2.ADC1
GPIO2.ADC2
GPIO2.ADC3
GPIO2.ADC4
GPIO2.DAC1
GPIO2.DAC2
GPIO2.DAC3
GPIO2.DAC4
CPUTotalLoadRatio

The “GatheringListGet” function can be used to retrieve a complete list of gathering types.
For a more thorough description of the internal data gathering capability, please refer to the XPS Motion Tutorial, section named Data Gathering / Internal Data Gathering.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GATHERING_RUNNING (-43)
ERR_IN_INITIALIZATION (-21)
ERR_MNEMOTYPEGATHERING (-29)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_TYPE_CHAR (-13)
ERR_WRONG_PARAMETERS_NUMBER (-9)

SUCCESS (0) : no error

TCL

Prototype

GatheringConfigurationSet $SocketID $GatheringType …

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GatheringType ..... string ..... Gathering type

Output parameters

None

Return

TCL error (0 = success or 1 = syntax error) or Function error

Note : *$GatheringType* can be repeted to set several gathering types.

C / C++

Prototype

int **GatheringConfigurationSet** (int SocketID, int NbElements, char * TypeArray)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
NbElements..... int ..... Number of types
GatheringTypes..... char * ..... Gathering types (seperated by semicolon)

Output parameters

None

Return

Function error

Note : *GatheringTypes* must have elements number in accordance to *NbElements* to avoid array overrun.

VISUAL BASIC

Prototype

Long **GatheringConfigurationSet** (ByVal SocketID As Long, ByVal NbElements As Long, ByVal GatheringTypes As String)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
NbElements..... Long ..... Number of types
GatheringTypes..... String ..... Gathering types (seperated by semicolon)

Output parameters

None

Return

Function error

Note : *GatheringTypes* must have elements number in accordance to *NbElements* to avoid array overrun.

MATLAB

Prototype

[Error] **GatheringConfigurationSet** (int32 SocketID, cstring GatheringTypes)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
GatheringTypes..... cstring ..... Gathering types (types separated by space (blank) character)

Return

Error ..... int32 ..... Function error

Note : *GatheringTypes* can have some elements to set in one time several gathering types.

PYTHON

Prototype

[Error] **GatheringConfigurationSet** (integer SocketID, string list GatheringTypes)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GatheringTypes ..... string list ..... List of gathering types

Return

Error ..... integer ..... Function error

Note : *GatheringTypes* can have some elements to set in one time several gathering types.

2.10.1.3. GatheringCurrentNumberGet

NAME

GatheringCurrentNumberGet – Returns the current and maximum number of gathered data.

INPUT TESTS

- XPS configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Command format: ERR_WRONG_FORMAT (-7)
- Number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Output parameter type: ERR_WRONG_TYPE_INT (-15)
- Gathering must be configured: ERR_GATHERING_NOT_CONFIGURED (-32)

DESCRIPTION

This function returns the current and maximum number of data gathered during the internal triggered data gathering.

For a more thorough description of the internal data gathering capability, please refer to the XPS Motion Tutorial, section named Data Gathering / Internal Data Gathering.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GATHERING_NOT_CONFIGURED (-32)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_TYPE_INT (-15)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype

GatheringCurrentNumberGet $SocketID CurrentNumber MaxSamplesNumber

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

CurrentNumber ..... integer ..... Current number during acquisition
MaxSamplesNumber..... integer ..... Maximum number of samples

Return

TCL error (0 = success or 1 = syntax error) or Function error

C / C++

Prototype

int **GatheringCurrentNumberGet** (int SocketID, int * CurrentNumber, int * MaxSamplesNumber)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

CurrentNumber ..... int * ..... Current number during acquisition
MaxSamplesNumber..... int * ..... Maximum number of samples

Return

Function error

VISUAL BASIC

Prototype

Long **GatheringCurrentNumberGet** (ByVal SocketID As Long, CurrentNumber As Long, MaxSamplesNumber As Long)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

CurrentNumber ..... Long ..... Current number during acquisition
MaxSamplesNumber..... Long ..... Maximum number of samples

Return

Function error

MATLAB

Prototype

[Error, CurrentNumber, MaxSamplesNumber] **GatheringCurrentNumberGet** (int32 SocketID)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function

Return

Error..... int32 ..... Function error
CurrentNumber ..... int32 ..... Current number during acquisition
MaxSamplesNumber..... int32 ..... Maximum number of samples

PYTHON

Prototype

[Error, CurrentNumber, MaxSamplesNumber] **GatheringCurrentNumberGet** (integer SocketID)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function

Return

Error..... integer ..... Function error
CurrentNumber ..... integer ..... Current number during acquisition
MaxSamplesNumber..... integer ..... Maximum number of samples

2.10.1.4. GatheringDataAcquire

NAME

GatheringDataAcquire – Acquires manually only one data.

INPUT TESTS

- XPS configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Command format: ERR_WRONG_FORMAT (-7)
- Number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Gathering must be configured: ERR_GATHERING_NOT_CONFIGURED (-32)
- Gathering must not be in progress: ERR_GATHERING_RUNNING (-43)
- Check gathering buffer size: ERR_GATHERING_BUFFER_FULL (-111)

DESCRIPTION

This function allows acquire manually only one data (configured by the “GatheringConfigurationSet” function).

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GATHERING_BUFFER_FULL (-111)
ERR_GATHERING_NOT_CONFIGURED (-32)
ERR_GATHERING_RUNNING (-43)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype

GatheringDataAcquire $SocketID

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

None

Return

TCL error (0 = success or 1 = syntax error) or Function error

C / C++

Prototype

int **GatheringDataAcquire** (int SocketID)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

None

Return

Function error

VISUAL BASIC

Prototype

Long **GatheringDataAcquire** (ByVal SocketID As Long)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

None

Return

Function error

MATLAB

Prototype

[Error] **GatheringDataAcquire** (int32 SocketID)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function

Return

Error ..... int32 ..... Function error

PYTHON

Prototype

[Error] **GatheringDataAcquire** (integer SocketID)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function

Return

Error ..... integer ..... Function error

2.10.1.5. GatheringDataGet

NAME

GatheringDataGet – Reads a data line from the current gathering buffer.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Check command format: ERR_WRONG_FORMAT (-7)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check input parameter type: ERR_WRONG_TYPE_INT (-15)
- Check output parameter type: ERR_WRONG_TYPE_CHAR (-13)
- Check gathering state: ERR_GATHERING_NOT_CONFIGURED (-32)
- Check index number: ERR_PARAMETER_OUT_OF_RANGE (-17)
  IndexPoint ≥ 0
  IndexPoint < currently gathered data number

DESCRIPTION

This function enables to read a data line from the current gathering buffer. The buffer line number is defined by the index of an acquired point.

The separator is ";" in the returned data line.

A gathering must be configured to use this function, else the ERR_GATHERING_NOT_CONFIGURED (-32) error is returned.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GATHERING_NOT_CONFIGURED (-32)
ERR_IN_INITIALIZATION (-21)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0): no error

TCL

Prototype

GatheringDataGet $SocketID $IndexPoint DataBufferLine

Input parameters

SocketID	integer	Socket identifier got from "TCP_ConnectToServer" function
IndexPoint	integer	Index of an acquired data from the current gathering buffer.

Output parameters

DataBufferLine	string	String contains values from the current buffer at the selected index.

Return

Error	integer	TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GatheringDataGet** (int SocketID, int IndexPoint, char *DataBufferLine)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
IndexPoint	int	Index of an acquired data from the current gathering buffer.

Output parameters

DataBufferLine	char *	String contains values from the current buffer at the selected index.

Return

Error	int	Function error code

VISUAL BASIC

Prototype

Long **GatheringDataGet** (ByVal SocketID As Long, ByVal IndexPoint As Long, DataBufferLine As String)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
IndexPoint	Long	Index of an acquired data from the current gathering buffer.

Output parameters

DataBufferLine	String	String contains values from the current buffer at the selected index.

Return

Error	Long	Function error code

MATLAB

Prototype

[Error, DataBufferLine] **GatheringDataGet** (int32 SocketID, cstring IndexPoint)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
IndexPoint	int32	Index of an acquired data from the current gathering buffer.

Return

Error	int32	Function error code
DataBufferLine	cstring	String contains values from the current buffer at the selected index.

PYTHON

Prototype

[Error, DataBufferLine] **GatheringDataGet** (integer SocketID, string UserName, string Password)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
IndexPoint	integer	Index of an acquired data from the current gathering buffer.

Return

Error	integer	Function error code
DataBufferLine	string	String contains values from the current buffer at the selected index.

2.10.1.6. GatheringDataMultipleLinesGet

NAME

GatheringDataMultipleLinesGet – Reads several data lines from the current gathering buffer in memory.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Check command format: ERR_WRONG_FORMAT (-7)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check input parameter type: ERR_WRONG_TYPE_INT (-15)
- Check output parameter type: ERR_WRONG_TYPE_CHAR (-13)
- Check gathering state: ERR_GATHERING_NOT_CONFIGURED (-32)
- Check index number: ERR_PARAMETER_OUT_OF_RANGE (-17)
  IndexPoint ≥ 0 (**Note:** index #0 = line #1)
  IndexPoint < currently gathered data number

DESCRIPTION

This function enables to read one or several data lines from the current gathering buffer. The buffer line number is defined by the index of an acquired point.
The separator is ";" in the returned data line and the end of each line is carriage return "\n".
A gathering must be configured to use this function, else the ERR_GATHERING_NOT_CONFIGURED (-32) error is returned.

Example of gathering buffer in memory:

index	Data1	Data2	Data3	Data4	Data5
0 →	1	10	0.1	21	100
1 →	2	20	0.2	22	102
2 →	3	30	0.3	23	103
3 →	4	40	0.4	24	104
5 →	5	50	0.5	25	105

GatheringDataMultipleLinesGet(0, 3, myString)
=> 0 = the start line is #1
=> 3 = the number of lines to read is 3
=> myString = buffer to get the part of buffer (65536 characters maximum)

index	Data1	Data2	Data3	Data4	Data5
0 →	1	10	0.1	21	100
1 →	2	20	0.2	22	102
2 →	3	30	0.3	23	103
3 →	4	40	0.4	24	104
5 →	5	50	0.5	25	105

"myString" result:

```
1;10;0.1;21;100
2;20;0.2;22;102
3;30;0.3;23;103
```

GatheringDataMultipleLinesGet(1, 4, myString)
=> 1 = the start line is #2
=> 4 = the number of lines to read is 4
=> myString = buffer to get the part of buffer (65536 characters maximum)

index	Data1	Data2	Data3	Data4	Data5
0 →	1	10	0.1	21	100
1 →	2	20	0.2	22	102
2 →	3	30	0.3	23	103
3 →	4	40	0.4	24	104
5 →	5	50	0.5	25	105

"myString" result:

```
2;20;0.2;22;102
3;30;0.3;23;103
4;40;0.4;24;104
5;50;0.5;25;105
```

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GATHERING_NOT_CONFIGURED (-32)
ERR_IN_INITIALIZATION (-21)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0): no error

TCL

Prototype

GatheringDataMultipleLinesGet $SocketID $IndexPoint DataBufferLine

Input parameters

SocketID ..... integer ..... Socket identifier gets by the "TCP_ConnectToServer" function
IndexPoint ..... integer ..... Index of an acquired data from the current gathering buffer.
NbLines ..... integer ..... Number of lines to get.

Output parameters

DataBufferLine ..... string ..... String contains lines from the current buffer at the selected index.

Return

Error ..... integer ..... TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GatheringDataMultipleLinesGet** (int SocketID, int IndexPoint, char *DataBufferLine)

Input parameters

SocketID ..... int ..... Socket identifier gets by the "TCP_ConnectToServer" function
IndexPoint ..... int ..... Index of an acquired data from the current gathering buffer.
NbLines ..... int ..... Number of lines to get.

Output parameters

DataBufferLine ..... char * ..... String contains lines from the current buffer at the selected index.

Return

Error ..... int ..... Function error code

VISUAL BASIC

Prototype

Long **GatheringDataMultipleLinesGet** (ByVal SocketID As Long, ByVal IndexPoint As Long, DataBufferLine As String)

Input parameters

SocketID	Long	Socket identifier gets by the “TCP_ConnectToServer” function
IndexPoint	Long	Index of an acquired data from the current gathering buffer.
NbLines	Long	Number of lines to get.

Output parameters

DataBufferLine	String	String contains lines from the current buffer at the selected index.

Return

Error	Long	Function error code

MATLAB

Prototype

[Error, DataBufferLine] **GatheringDataMultipleLinesGet** (int32 SocketID, cstring IndexPoint)

Input parameters

SocketID	int32	Socket identifier gets by the “TCP_ConnectToServer” function
IndexPoint	int32	Index of an acquired data from the current gathering buffer.
NbLines	int32	Number of lines to get.

Return

Error	int32	Function error code
DataBufferLine	cstring	String contains lines from the current buffer at the selected index.

PYTHON

Prototype

[Error, DataBufferLine] **GatheringDataMultipleLinesGet** (integer SocketID, string UserName, string Password)

Input parameters

SocketID	integer	Socket identifier gets by the “TCP_ConnectToServer” function
IndexPoint	integer	Index of an acquired data from the current gathering buffer.
NbLines	integer	Number of lines to get.

Return

Error	integer	Function error code
DataBufferLine	string	String contains lines from the current buffer at the selected index.

2.10.1.7. GatheringExternalConfigurationGet

NAME

GatheringExternalConfigurationGet – Returns the current configuration of the external triggered data gathering.

INPUT TESTS

- XPS configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Command format: ERR_WRONG_FORMAT (-7)
- Number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Output parameter type: ERR_WRONG_TYPE_ CHAR (-13)
- Gathering must be configured: ERR_GATHERING_NOT_CONFIGURED (-32)

DESCRIPTION

This function returns the current configuration of the external triggered data gathering.
Use the “GatheringExternalListGet” function to retrieve a complete list of external gathering types.

For a more thorough description of the external data gathering capability, please refer to the XPS Motion Tutorial, section named Data Gathering / External Data Gathering.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GATHERING_NOT_CONFIGURED (-32)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_TYPE_CHAR (-13)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype

GatheringExternalConfigurationGet $SocketID TypeList

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

TypeList ..... string ..... List of configured gathering types (separator is semicolon)

Return

TCL error (0 = success or 1 = syntax error) or Function error

C / C++

Prototype

int **GatheringExternalConfigurationGet** (int SocketID, char * TypeList)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

TypeList ..... char * ..... List of configured gathering types (separator is semicolon)

Return

Function error

VISUAL BASIC

Prototype

Long **GatheringExternalConfigurationGet** (ByVal SocketID As Long, ByVal TypeList As String)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function

Output parameters

TypeList	String	List of configured gathering types (separator is semicolon)

Return

Function error

MATLAB

Prototype

[Error, TypeList] **GatheringExternalConfigurationGet** (int32 SocketID)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function

Return

Error	int32	Function error
TypeList	cstring	List of configured gathering types (separator is semicolon)

PYTHON

Prototype

[Error, TypeList] **GatheringExternalConfigurationGet** (integer SocketID)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function

Return

Error	integer	Function error
TypeList	string	List of configured gathering types (separator is semicolon)

2.10.1.8. GatheringExternalConfigurationSet

NAME

GatheringExternalConfigurationSet – Configures an external gathering.

INPUT TESTS

- XPS configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Command format: ERR_WRONG_FORMAT (-7)
- Number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check input external gathering mnemonic: ERR_MNEMOTYPEGATHERING (-29)
- Gathering must not be in progress: ERR_GATHERING_RUNNING (-43)

DESCRIPTION

Defines one or several types of data gathered during the external triggered data gathering.

Maximum of 1000000 points can be acquired.
Maximum of 25 data types can be configured in a gathering.

External gathering data types:
PositionerName.ExternalLatchPosition
GPIO2.ADC1
GPIO2.ADC2
GPIO2.ADC3
GPIO2.ADC4
GPIO2.DAC1
GPIO2.DAC2
GPIO2.DAC3
GPIO2.DAC4

The “GatheringExternalListGet” function can be used to retrieve a complete list of gathering types.

For a more thorough description of the external data gathering capability, please refer to the XPS Motion Tutorial, section named Data Gathering / External Data Gathering.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GATHERING_RUNNING (-43)
ERR_IN_INITIALIZATION (-21)
ERR_MNEMOTYPEGATHERING (-29)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_TYPE_CHAR (-13)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype
GatheringExternalConfigurationSet $SocketID $GatheringType …

Input parameters
SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
GatheringType ..... string ..... Gathering type

Output parameters
None

Return
TCL error (0 = success or 1 = syntax error) or Function error

Note : *$GatheringType* can be repeted to set several gathering types.

C / C++

Prototype

int **GatheringExternalConfigurationSet** (int SocketID, int NbElements, char * TypeArray)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
NbElements	int	Number of types
GatheringTypes	char *	Gathering types (seperated by semicolon)

Output parameters

None

Return

Function error

Note : *GatheringTypes* must have elements number in accordance to *NbElements* to avoid array overrun.

VISUAL BASIC

Prototype

Long **GatheringExternalConfigurationSet** (ByVal SocketID As Long, ByVal NbElements As Long, ByVal GatheringTypes As String)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
NbElements	Long	Number of types
GatheringTypes	String	Gathering types (seperated by semicolon)

Output parameters

None

Return

Function error

Note : *GatheringTypes* must have elements number in accordance to *NbElements* to avoid array overrun.

MATLAB

Prototype

[Error] **GatheringExternalConfigurationSet** (int32 SocketID, cstring GatheringTypes)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
GatheringTypes	cstring	Gathering types (types separated by space (blank) character)

Return

Error	int32	Function error

Note : *GatheringTypes* can have some elements to set in one time several gathering types.

PYTHON

Prototype

[Error] **GatheringExternalConfigurationSet** (integer SocketID, string list GatheringTypes)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
GatheringTypes	string list	List of gathering types

Return

Error.....integer.....Function error

Note : *GatheringTypes* can have some elements to set in one time several gathering types.

2.10.1.9. GatheringExternalCurrentNumberGet

NAME

GatheringExternalCurrentNumberGet – Returns the current and maximum number of external gathered data.

INPUT TESTS

- XPS configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Command format: ERR_WRONG_FORMAT (-7)
- Number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Output parameter type: ERR_WRONG_TYPE_INT (-15)
- External gathering must be configured: ERR_GATHERING_NOT_CONFIGURED (-32)

DESCRIPTION

This function returns the current and maximum number of data gathered during the external triggered data gathering.

For a more thorough description of the external data gathering capability, please refer to the XPS Motion Tutorial, section named Data Gathering / External Data Gathering.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GATHERING_NOT_CONFIGURED (-32)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_TYPE_INT (-15)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype

GatheringExternalCurrentNumberGet $SocketID CurrentNumber MaxSamplesNumber

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

CurrentNumber ..... integer ..... Current number during acquisition
MaxSamplesNumber..... integer ..... Maximum number of samples

Return

TCL error (0 = success or 1 = syntax error) or Function error

C / C++

Prototype

int **GatheringExternalCurrentNumberGet** (int SocketID, int * CurrentNumber, int * MaxSamplesNumber)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

CurrentNumber ..... int * ..... Current number during acquisition
MaxSamplesNumber..... int * ..... Maximum number of samples

Return

Function error

VISUAL BASIC

Prototype

Long **GatheringExternalCurrentNumberGet** (ByVal SocketID As Long, CurrentNumber As Long, MaxSamplesNumber As Long)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

CurrentNumber ..... Long ..... Current number during acquisition
MaxSamplesNumber..... Long ..... Maximum number of samples

Return

Function error

MATLAB

Prototype

[Error, CurrentNumber, MaxSamplesNumber] **GatheringExternalCurrentNumberGet** (int32 SocketID)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function

Return

Error..... int32 ..... Function error
CurrentNumber ..... int32 ..... Current number during acquisition
MaxSamplesNumber..... int32 ..... Maximum number of samples

PYTHON

Prototype

[Error, CurrentNumber, MaxSamplesNumber] **GatheringExternalCurrentNumberGet** (integer SocketID)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function

Return

Error..... integer ..... Function error
CurrentNumber ..... integer ..... Current number during acquisition
MaxSamplesNumber..... integer ..... Maximum number of samples

2.10.1.10. GatheringExternalDataGet

NAME

GatheringExternalDataGet – Reads one data line from the current external gathering buffer.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Check command format: ERR_WRONG_FORMAT (-7)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check input parameter type: ERR_WRONG_TYPE_INT (-15)
- Check output parameter type: ERR_WRONG_TYPE_CHAR (-13)
- Check gathering state: ERR_GATHERING_NOT_CONFIGURED (-32)
- Check index number: ERR_PARAMETER_OUT_OF_RANGE (-17)
  IndexPoint ≥ 0
  IndexPoint < currently gathered data number

DESCRIPTION

This function enables to read a data line from the current gathering gathering buffer. The buffer line number is defined by the index of an acquired point.

The separator is ";" in the returned data line.

A gathering must be configured to use this function, else the ERR_GATHERING_NOT_CONFIGURED (-32) error is returned.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GATHERING_NOT_CONFIGURED (-32)
ERR_IN_INITIALIZATION (-21)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0): no error

TCL

Prototype

GatheringExternalDataGet $SocketID $IndexPoint DataBufferLine

Input parameters

SocketID ..... integer ..... Socket identifier gets by the "TCP_ConnectToServer" function
IndexPoint ..... integer ..... Index of an acquired data from the current gathering buffer.

Output parameters

DataBufferLine ..... string ..... String contains values from the current buffer at the selected index.

Return

Error ..... integer ..... TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **GatheringExternalDataGet** (int SocketID, int IndexPoint, char *DataBufferLine)

Input parameters

SocketID ..... int ..... Socket identifier gets by the “TCP_ConnectToServer” function
IndexPoint ..... int ..... Index of an acquired data from the current gathering buffer.

Output parameters

DataBufferLine ..... char * ..... String contains values from the current buffer at the selected index.

Return

Error ..... int ..... Function error code

VISUAL BASIC

Prototype

Long **GatheringExternalDataGet** (ByVal SocketID As Long, ByVal IndexPoint As Long, DataBufferLine As String)

Input parameters

SocketID ..... Long ..... Socket identifier gets by the “TCP_ConnectToServer” function
IndexPoint ..... Long ..... Index of an acquired data from the current gathering buffer.

Output parameters

DataBufferLine ..... String ..... String contains values from the current buffer at the selected index.

Return

Error ..... Long ..... Function error code

MATLAB

Prototype

[Error, DataBufferLine] **GatheringExternalDataGet** (int32 SocketID, cstring IndexPoint)

Input parameters

SocketID ..... int32 ..... Socket identifier gets by the “TCP_ConnectToServer” function
IndexPoint ..... int32 ..... Index of an acquired data from the current gathering buffer.

Return

Error ..... int32 ..... Function error code
DataBufferLine ..... cstring ..... String contains values from the current buffer at the selected index.

PYTHON

Prototype

[Error, DataBufferLine] **GatheringExternalDataGet** (integer SocketID, string UserName, string Password)

Input parameters

SocketID ..... integer ..... Socket identifier gets by the “TCP_ConnectToServer” function
IndexPoint ..... integer ..... Index of an acquired data from the current gathering buffer.

Return

Error ..... integer ..... Function error code
DataBufferLine ..... string ..... String contains values from the current buffer at the selected index.

2.10.1.11. GatheringExternalStopAndSave

NAME

GatheringExternalStopAndSave – Stops external triggered data gathering and saves data to the XPS controller.

INPUT TESTS

- XPS configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Command format: ERR_WRONG_FORMAT (-7)
- Number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check number of data (> 0): ERR_GATHERING_NOT_STARTED (-30)
- Check file opening: ERR_WRITE_FILE (-60)

DESCRIPTION

This function stops external triggered data gathering and saves data to the XPS controller. Gathered data are stored in the file “GatheringExternal.dat” in the “..\PUBLIC” folder of the XPS controller.

For a more thorough description of the external data gathering capability, please refer to the XPS Motion Tutorial, section named Data Gathering / External Data Gathering.
.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GATHERING_NOT_STARTED (-30)
ERR_IN_INITIALIZATION (-21)
ERR_WRITE_FILE (-60)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype

GatheringExternalStopAndSave $SocketID

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

None

Return

TCL error (0 = success or 1 = syntax error) or Function error

C / C++

Prototype

int **GatheringExternalStopAndSave** (int SocketID)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

None

Return

Function error

VISUAL BASIC

Prototype

Long **GatheringExternalStopAndSave** (ByVal SocketID As Long)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

None

Return

Function error

MATLAB

Prototype

[Error] **GatheringExternalStopAndSave** (int32 SocketID)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function

Return

Error ..... int32 ..... Function error

PYTHON

Prototype

[Error] **GatheringExternalStopAndSave** (integer SocketID)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function

Return

Error ..... integer ..... Function error

2.10.1.12. GatheringReset

NAME

GatheringReset – Resets gathered data to start new gathering from scratch.

INPUT TESTS

- XPS configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Command format: ERR_WRONG_FORMAT (-7)
- Number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Gathering must not be in progress: ERR_GATHERING_RUNNING (-43)

DESCRIPTION

This function allows reset gathered data to start new gathering from scratch.
The number of gathered data is setting to zero.

For a more thorough description of the internal data gathering capability, please refer to the XPS Motion Tutorial, section named Data Gathering / Internal Data Gathering.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GATHERING_RUNNING (-43)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype
GatheringReset $SocketID

Input parameters
SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters
None

Return
TCL error (0 = success or 1 = syntax error) or Function error

C / C++

Prototype
int **GatheringReset** (int SocketID)

Input parameters
SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters
None

Return
Function error

VISUAL BASIC

Prototype

Long **GatheringReset** (ByVal SocketID As Long)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

None

Return

Function error

MATLAB

Prototype

[Error] **GatheringReset** (int32 SocketID)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function

Return

Error ..... int32 ..... Function error

PYTHON

Prototype

[Error] **GatheringReset** (integer SocketID)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function

Return

Error ..... integer ..... Function error

2.10.1.13. GatheringRun

NAME

GatheringRun – Starts to gather data.

INPUT TESTS

- XPS configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Command format: ERR_WRONG_FORMAT (-7)
- Number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Gathering must not be in progress: ERR_GATHERING_RUNNING (-43)
- Gathering must be configured: ERR_GATHERING_NOT_CONFIGURED (-32)

DESCRIPTION

This function allows to start a new data gathering.
The data gathering needs to be configured before using this function (See GatheringConfigurationSet)

The parameters are the number of data to be gathered and the divisor of the frequency (servo frequency) at which the data gathering will be done.

For a more thorough description of the internal data gathering capability, please refer to the XPS Motion Tutorial, section named Data Gathering / Internal Data Gathering.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GATHERING_RUNNING (-43)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_GATHERING_NOT_CONFIGURED (-32)
SUCCESS (0) : no error

TCL

Prototype

GatheringRun $SocketID $DataNumber $Divisor

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
DataNumber	integer	The number of data line to gather
Divisor	integer	The divisor of the servo frequency

Output parameters

None

Return

TCL error (0 = success or 1 = syntax error) or Function error

C / C++

Prototype

int **GatheringRun** (int SocketID, int DataNumber, int Divisor)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
DataNumber	int	The number of data line to gather
Divisor	int	The divisor of the servo frequency

Output parameters

None

Return

Function error

VISUAL BASIC

Prototype

Long **GatheringRun** (ByVal SocketID As Long, ByVal DataNumber As Long, ByVal Divisor As Long)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
DataNumber	Long	The number of data line to gather
Divisor	Long	The divisor of the servo frequency

Output parameters

None

Return

Function error

MATLAB

Prototype

[Error] **GatheringRun** (int32 SocketID, int32 DataNumber, int32 Divisor)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
DataNumber	int32	The number of data line to gather
Divisor	int32	The divisor of the servo frequency

Return

Error	int32	Function error

PYTHON

Prototype

[Error] **GatheringRun** (integer SocketID, integer DataNumber, integer Divisor)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
DataNumber	integer	The number of data line to gather
Divisor	integer	The divisor of the servo frequency

Return

Error	integer	Function error

2.10.1.14. GatheringRunAppend

NAME

GatheringRunAppend – Do again the gathering, continuing from the last gathered data.

INPUT TESTS

- XPS configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Command format: ERR_WRONG_FORMAT (-7)
- Number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Gathering must not be in progress: ERR_GATHERING_RUNNING (-43)
- Gathering must be configured: ERR_GATHERING_NOT_CONFIGURED (-32)

DESCRIPTION

This function allows to do again the gathering from the data point that is previously stopped, if the gathering current data number has not reached the *DataNumber* specified before with the *GatheringRun()* function.
The gathering must to be configured, executed and stopped before using this function (see *GatheringConfigurationSet*, *GatheringRun*, *GatheringStop* functions)

For a more thorough description of the internal data gathering capability, please refer to the XPS Motion Tutorial, section named Data Gathering / Internal Data Gathering.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GATHERING_RUNNING (-43)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_GATHERING_NOT_CONFIGURED (-32)
SUCCESS (0) : no error

TCL

Prototype

GatheringRunAppend $SocketID

Input parameters

SocketID ..... integer ..... Socket identifier gets by the “TCP_ConnectToServer” Function

Output parameters

None

Return

TCL error (0 = success or 1 = syntax error) or Function error

C / C++

Prototype

int **GatheringRunAppend** (int SocketID)

Input parameters

SocketID ..... int ..... Socket identifier gets by the “TCP_ConnectToServer” function

Output parameters

None

Return

Function error

VISUAL BASIC

Prototype

Long **GatheringRunAppend** (ByVal SocketID As Long)

Input parameters

SocketID ..... Long ..... Socket identifier gets by the “TCP_ConnectToServer” Function

Output parameters

None

Return

Function error

MATLAB

Prototype

[Error] **GatheringRunAppend** (int32 SocketID)

Input parameters

SocketID ..... int32 ..... Socket identifier gets by the “TCP_ConnectToServer” Function

Return

Error ..... int32 ..... Function error

PYTHON

Prototype

[Error] **GatheringRunAppend** (integer SocketID)

Input parameters

SocketID ..... integer ..... Socket identifier gets by the “TCP_ConnectToServer” Function

Return

Error ..... integer ..... Function error

2.10.1.15. GatheringStop

NAME

GatheringStopAndSave – Stops internal and external triggered data gathering.

INPUT TESTS

- XPS configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Command format: ERR_WRONG_FORMAT (-7)
- Number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check number of data (> 0): ERR_GATHERING_NOT_STARTED (-30)
- Check file opening: ERR_WRITE_FILE (-60)

DESCRIPTION

This function stops internal and external triggered data gathering. To save it to a file, use GatheringStopAndSave function.

For a more thorough description of the internal data gathering capability, please refer to the XPS Motion Tutorial, section named Data Gathering / Internal Data Gathering.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GATHERING_NOT_STARTED (-30)
ERR_IN_INITIALIZATION (-21)
ERR_WRITE_FILE (-60)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype

GatheringStop $SocketID

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

None

Return

TCL error (0 = success or 1 = syntax error) or Function error

C / C++

Prototype

int **GatheringStop** (int SocketID)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

None

Return

Function error

VISUAL BASIC

Prototype

Long **GatheringStop** (ByVal SocketID As Long)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

None

Return

Function error

MATLAB

Prototype

[Error] **GatheringStop** (int32 SocketID)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function

Return

Error ..... int32 ..... Function error

PYTHON

Prototype

[Error] **GatheringStop** (integer SocketID)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function

Return

Error ..... integer ..... Function error

2.10.1.16. GatheringStopAndSave

NAME

GatheringStopAndSave – Stops internal triggered data gathering and saves data to the XPS controller.

INPUT TESTS

- XPS configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Command format: ERR_WRONG_FORMAT (-7)
- Number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check number of data (> 0): ERR_GATHERING_NOT_STARTED (-30)
- Check file opening: ERR_WRITE_FILE (-60)

DESCRIPTION

This function stops internal triggered data gathering and saves data to the XPS controller. Data is stored in the file GATHERING.DAT in the "..\PUBLIC" folder of the XPS controller.

For a more thorough description of the internal data gathering capability, please refer to the XPS Motion Tutorial, section named Data Gathering / Internal Data Gathering.

ERROR CODES

ERR_FATAL_INIT (-20)
ERR_GATHERING_NOT_STARTED (-30)
ERR_IN_INITIALIZATION (-21)
ERR_WRITE_FILE (-60)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype

GatheringStopAndSave $SocketID

Input parameters

SocketID ..... integer ..... Socket identifier got from "TCP_ConnectToServer" function

Output parameters

None

Return

TCL error (0 = success or 1 = syntax error) or Function error

C / C++

Prototype

int **GatheringStopAndSave** (int SocketID)

Input parameters

SocketID ..... int ..... Socket identifier got from "TCP_ConnectToServer" function

Output parameters

None

Return

Function error

VISUAL BASIC

Prototype

Long **GatheringStopAndSave** (ByVal SocketID As Long)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

None

Return

Function error

MATLAB

Prototype

[Error] **GatheringStopAndSave** (int32 SocketID)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function

Return

Error ..... int32 ..... Function error

PYTHON

Prototype

[Error] **GatheringStopAndSave** (integer SocketID)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function

Return

Error ..... integer ..... Function error

2.11. Events and actions

2.11.1. Functions description

2.11.1.1. EventExtendedAllGet

NAME

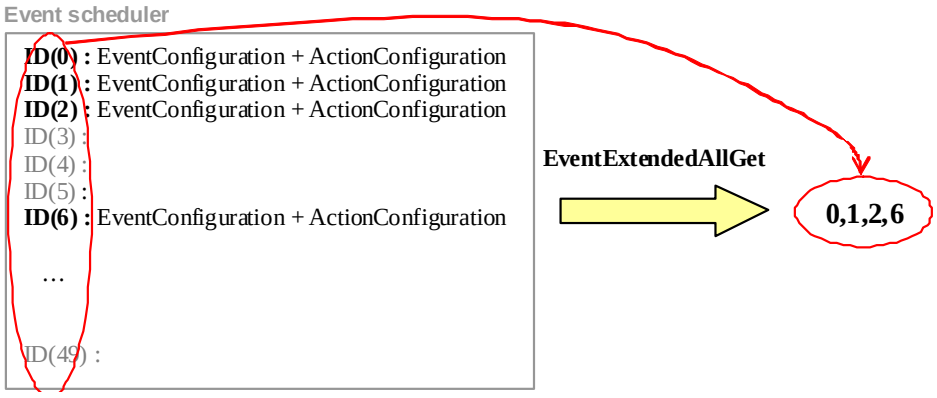
EventExtendedAllGet – Return all “event and action” identifiers in progress.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Output parameter type: ERR_WRONG_TYPE_CHAR (-13)

DESCRIPTION

Get the list of all “event and action” combination identifiers from the event scheduler (filled by the **ExtendedEventStart** or **ExtendedEventWait** function).
The list separator is the comma. If no “event and action” combination is in progress (in the event scheduler) then the error ERR_EVENT_ID_UNDEFINED (-83) is returned.



ERRORS

ERR_EVENT_ID_UNDEFINED (-83)
ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
SUCCESS (0) : no error

TCL

Prototype

EventExtendedAllGet $SocketID $EventID EventIdentifiersList

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
EventID ..... integer ..... “Event and action” identifier from “ExtendedEventStart”

Output parameters

EventIdentifiersList ..... string ..... List of “event and action” identifiers in scheduler

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **EventExtendedAllGet** (int SocketID, int EventID, char * EventIdentifiersList)

Input parameters

SocketID ..... int ... Socket identifier got from “TCP_ConnectToServer” function
EventID ..... int ... “Event and action” identifier from “ExtendedEventStart”

Output parameters

EventIdentifiersList ..... char * ..... List of “event and action” identifiers in scheduler

Return

Function error code

VISUAL BASIC

Prototype

Long **EventExtendedAllGet** (ByVal SocketID As Long, ByVal EventID As Long, EventIdentifiersList As String)

Input parameters

SocketID ..... Long Socket identifier got from “TCP_ConnectToServer” function
EventID ..... Long “Event and action” identifier from “ExtendedEventStart”

Output parameters

EventIdentifiersList ..... String ..... List of “event and action” identifiers in scheduler

Return

Function error code

MATLAB

Prototype

[Error, EventIdentifiersList] **EventExtendedAllGet** (int32 SocketID, int32 EventID)

Input parameters

SocketID ..... int32 Socket identifier got from “TCP_ConnectToServer” function
EventID ..... int32 “Event and action” identifier from “ExtendedEventStart”

Return

Error ..... integer ..... Function error code
EventIdentifiersList ..... cstring ..... List of “event and action” identifiers in scheduler

PYTHON

Prototype

[Error, EventIdentifiersList] **EventExtendedAllGet** (integer SocketID, integer EventID)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
EventID ..... integer ..... “Event and action” identifier from “ExtendedEventStart”

Return

Error ..... integer ..... Function error code
EventIdentifiersList ..... string ..... List of “event and action” identifiers in scheduler

2.11.1.2. EventExtendedConfigurationActionGet

NAME

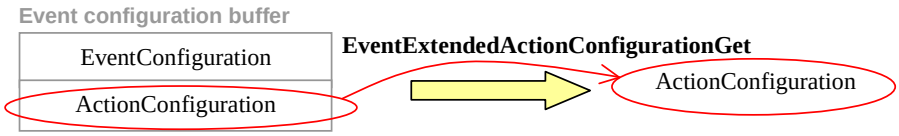
EventExtendedConfigurationActionGet – Return the action combination defined in buffer.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Output parameter type: ERR_WRONG_TYPE_CHAR (-13)
- Last action configuration in memory: ERR_ACTIONS_NOT_CONFIGURED (-81)

DESCRIPTION

Get the combination of action(s) defined by “EventExtendedConfigurationActionSet” function.
If no action is configured in buffer, the ERR_ACTIONS_NOT_CONFIGURED (-81) error is returned.



NOTE:
This function doesn’t return the last activated action. A combination of action(s) can be just defined in buffer and not activated…

ERRORS

ERR_ACTIONS_NOT_CONFIGURED (-81)
ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
SUCCESS (0) : no error

TCL

Prototype
EventExtendedConfigurationActionGet $SocketID ActionConfiguration

Input parameters
SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters
ActionConfiguration ..... string ..... Action combination configured in buffer

Return
TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype
int **EventExtendedConfigurationActionGet** (int SocketID, char * ActionConfiguration)

Input parameters
SocketID ..... int ... Socket identifier got from “TCP_ConnectToServer” function

Output parameters
ActionConfiguration ..... string ..... Action combination configured in buffer

Return
Function error code

VISUAL BASIC

Prototype
Long **EventExtendedConfigurationActionGet** (ByVal SocketID As Long, ActionConfiguration As String)

Input parameters
SocketID ..... Long Socket identifier got from “TCP_ConnectToServer” function

Output parameters
ActionConfiguration ..... String ..... Action combination configured in buffer

Return
Function error code

MATLAB

Prototype
[Error, ActionConfiguration] **EventExtendedConfigurationActionGet** (int32 SocketID)

Input parameters
SocketID ..... int32 Socket identifier got from “TCP_ConnectToServer” function

Return
Error ..... integer ..... Function error code
ActionConfiguration ..... cstring ..... Action combination configured in buffer

PYTHON

Prototype
[Error, ActionConfiguration] **EventExtendedConfigurationActionGet** (integer SocketID)

Input parameters
SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function

Return
Error ..... integer ..... Function error code
ActionConfiguration ..... string ..... Action combination configured in buffer

2.11.1.3. EventExtendedConfigurationActionSet

NAME

EventExtendedConfigurationActionSet – Define a combination of one or several actions in buffer.

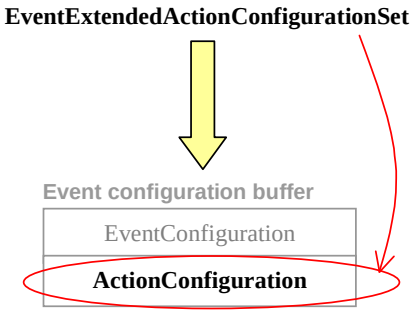
INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of arguments: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Output parameter type: ERR_WRONG_TYPE_CHAR (-13)
- Last action configured in memory: ERR_ACTIONS_NOT_CONFIGURED (-81)
- Action name: ERR_MNEMO_ACTION (-39)
- Action parameters: ERR_PARAMETER_OUT_OF_RANGE (-17), ERR_WRONG_OBJECT_TYPE (-8)
- Action to execute: ERR_GATHERING_NOT_CONFIGURED (-32) “Gathering” action.

DESCRIPTION

Just define a combination of one or several actions but don’t activate it. Use the “EventExtendedStart” function to activate this definition action(s). For each action, 4 parameters can be configured … see event specification to know if necessary. The actions are defined in § “Events and Actions” in the XPS user’s manual.

The number of actions in a combination is limited to 10 actions.



Action list

1. GPIOName.DOToggle
2. GPIOName.DOPulse
3. GPIOName.DOSet
4. GPIOName.DACSet.CurrentPosition
5. GPIOName.DACSet.CurrentVelocity
6. GPIOName.DACSet.SetpointPosition
7. GPIOName.DACSet.SetpointVelocity
8. GPIOName.DACSet.SetpointAcceleration
9. GPIOName.DACSet.Value
10. ExecuteTCLScript
11. KillTCLScript
12. ExternalGatheringRun
13. GatheringRun
14. GatheringRunAppend
15. GatheringOneData
16. GatheringStop
17. GroupName.MoveAbort

Action parameters

Actor				Action name	Parameters			
Group	Positioner	GPIO	Timer#		1	2	3	4
		•		DOToggle	Mask	0	0	0
		•		DOPulse	Mask	0	0	0
		•		DOSet	Mask	Value	0	0
		•		DACSet.SetpointPosition	Positioner name	Gain	Offset	0
		•		DACSet.SetpointVelocity	Positioner name	Gain	Offset	0
		•		DACSet.SetpointAcceleration	Positioner name	Gain	Offset	0
		•		DACSet.CurrentPosition	Positioner name	Gain	Offset	0
		•		DACSet.CurrentVelocity	Positioner name	Gain	Offset	0
		•		DACSet.Value	Value	0	0	0
				ExecuteTCLScript	TCL file name	Task name	Arguments	0
				KillTCLScript	Task name	0	0	0
				GatheringOneData	0	0	0	0
				GatheringRun	Nb of points	Divisor	0	0
				GatheringRunAppend	0	0	0	0
				GatheringStop	0	0	0	0
				ExternalGatheringRun	Nb of points	Divisor	0	0
•				MoveAbort	0	0	0	0

NOTE :
Before activating the defined actions, you must configure the events. Only then, you can use the "EventExtendedStart" or "EventExtendedWait" function.

For the "**ExecuteTCLScript**" action, the "**ActionParameter3**" represents a list of arguments. So, the **separator** must be a **semicolon** (;).

ERRORS

ERR_ACTIONS_NOT_CONFIGURED (-1)
ERR_FATAL_INIT (-20)
ERR_GATHERING_NOT_CONFIGURED (-32)
ERR_IN_INITIALIZATION (-21)
ERR_MNEMO_ACTION (-39)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
SUCCESS (0) : no error

TCL

Prototype

EventExtendedConfigurationActionSet $SocketID **{$**ActionName $ActionParameter1 $ActionParameter2 $ActionParameter3 $ActionParameter4**} …**

Input parameters

SocketID ..... integer ..... Socket identifier got from "TCP_ConnectToServer" function
ActionName ..... string ..... event full name (maximum size = 250) - see § Events -
ActionParameter1 ..... string ..... optional action parameter #1 (maximum size = 250)
ActionParameter2 ..... string ..... optional action parameter #2 (maximum size = 250)

ActionParameter3 ..... string ..... optional action parameter #3 (maximum size = 250)
ActionParameter4 ..... string ..... optional action parameter #4 (maximum size = 250)

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

Note : **$**ActionName, $ActionParameter1, $ActionParameter2, $ActionParameter3 and $ActionParameter4 can be repeted to set several action types.

C / C++

Prototype

int **EventExtendedConfigurationActionSet** (int SocketID, int NbElements, char* ActionNameList, char* ActionParameter1List, char* ActionParameter2List, char* ActionParameter3List, char* ActionParameter4List)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
NbElements..... int ..... number of events in configuration.
ActionNameList..... char* ..... event full name list (names separated by semicolon)
ActionParameter1List ..... char* ..... optional action parameter #1 list (separated by semicolon)
ActionParameter2List ..... char* ..... optional action parameter #2 list (separated by semicolon)
ActionParameter3List ..... char* ..... optional action parameter #3 list (separated by semicolon)
ActionParameter4List ..... char* ..... optional action parameter #4 list (separated by semicolon)

Output parameters

None

Return

Function error code

Note : ActionNameList, ActionParameter1List, ActionParameter2List, ActionParameter3List and ActionParameter4List must have elements number in accordance to *NbElements*.

VISUAL BASIC

Prototype

Long **EventExtendedConfigurationActionSet** (ByVal SocketID As Long, ByVal NbElements As Long, ByVal ActionNameList As String, ByVal ActionParameter1List As String, ByVal ActionParameter2List As String, ByVal ActionParameter3List As String, ByVal ActionParameter4List As String)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
NbElements..... Long ..... number of events in configuration.
ActionNameList..... String ..... event full name list (names separated by semicolon)
ActionParameter1List ..... String ..... optional action parameter #1 list (separated by semicolon)
ActionParameter2List ..... String ..... optional action parameter #2 list (separated by semicolon)
ActionParameter3List ..... String ..... optional action parameter #3 list (separated by semicolon)
ActionParameter4List ..... String ..... optional action parameter #4 list (separated by semicolon)

Output parameters

None

Return

Function error code

Note : ActionNameList, ActionParameter1List, ActionParameter2List, ActionParameter3List and ActionParameter4List must have elements number in accordance to *NbElements*.

MATLAB

Prototype

int32 **EventExtendedConfigurationActionSet** (int32 SocketID, int32 NbElements, cstring ActionNameList, cstring ActionParameter1List, cstring ActionParameter2List, cstring ActionParameter3List, cstring ActionParameter4List)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
ActionNameList	cstring	action full name list (separated by space (blank) character)
ActionParameter1List	cstring	optional action parameter #1 list (separated by space character)
ActionParameter2List	cstring	optional action parameter #2 list (separated by space character)
ActionParameter3List	cstring	optional action parameter #3 list (separated by space character)
ActionParameter4List	cstring	optional action parameter #4 list (separated by space character)

Output parameters

None

Return

Function error code

Note : ActionParameter1List, ActionParameter2List, ActionParameter3List and ActionParameter4List must have the same elements number as that of ActionNameList.

PYTHON

Prototype

integer **EventExtendedConfigurationActionSet** (integer SocketID, string list ActionNameList, string list ActionParameter1List, string list ActionParameter2List, string list ActionParameter3List, string list ActionParameter4List)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
ActionNameList	string list	event full name list
ActionParameter1List	string list	optional action parameter #1 list
ActionParameter2List	string list	optional action parameter #2 list
ActionParameter3List	string list	optional action parameter #3 list
ActionParameter4List	string list	optional action parameter #4 list

Output parameters

None

Return

Function error code

Note : ActionParameter1List, ActionParameter2List, ActionParameter3List and ActionParameter4List must have the same elements number as that of ActionNameList.

2.11.1.4. EventExtendedConfigurationTriggerGet

NAME

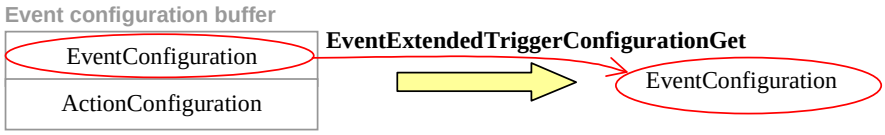
EventExtendedConfigurationTriggerGet – Return the trigger defined in buffer.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Parameter type: ERR_WRONG_TYPE_CHAR (-13)
- Last event configuration in memory: ERR_EVENTS_NOT_CONFIGURED (-80)

DESCRIPTION

Get the last event defined in buffer by “EventExtendedConfigurationTriggerSet” function.
If no event is defined in buffer, the ERR_EVENTS_NOT_CONFIGURED (-80) error is returned.



NOTE:
This function doesn’t return the last activated event. An event can be just configured and not activated…

ERRORS

SUCCESS (0) : no error
ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
ERR_EVENTS_NOT_CONFIGURED (-80)

TCL

Prototype
EventExtendedConfigurationTriggerGet $SocketID EventTriggerConfiguration

Input parameters
SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters
EventTriggerConfiguration..... string..... Event combination configured in buffer

Return
TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype
int **EventExtendedConfigurationTriggerGet** (int SocketID, char * EventTriggerConfiguration)

Input parameters
SocketID ..... int ... Socket identifier got from “TCP_ConnectToServer” function

Output parameters
EventTriggerConfiguration ..... string ..... Event combination configured in buffer

Return
Function error code

VISUAL BASIC

Prototype
Long **EventExtendedConfigurationTriggerGet** (ByVal SocketID As Long, EventTriggerConfiguration As String)

Input parameters
SocketID ..... Long Socket identifier got from “TCP_ConnectToServer” function

Output parameters
EventTriggerConfiguration ..... String ..... Event combination configured in buffer

Return
Function error code

MATLAB

Prototype
[Error, EventTriggerConfiguration] **EventExtendedConfigurationTriggerGet** (int32 SocketID)

Input parameters
SocketID ..... int32 Socket identifier got from “TCP_ConnectToServer” function

Return
Error ..... integer ..... Function error code
EventTriggerConfiguration ..... cstring ..... Event combination configured in buffer

PYTHON

Prototype
[Error, EventTriggerConfiguration] **EventExtendedConfigurationTriggerGet** (integer SocketID)

Input parameters
SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function

Return
Error ..... integer ..... Function error code
EventTriggerConfiguration ..... string ..... Event combination configured in buffer

2.11.1.5. EventExtendedConfigurationTriggerSet

NAME

EventExtendedConfigurationTriggerSet - Define a combination of one or several events in buffer.

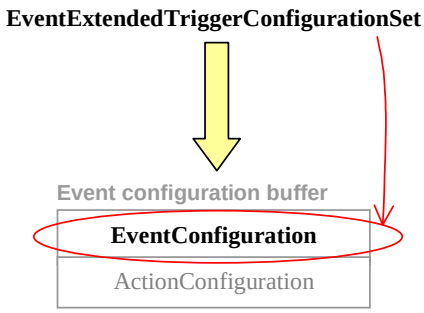
INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Event name: ERR_MNEMO_EVENT (-40)
- Event actor: ERR_WRONG_OBJECT_TYPE (-8)

DESCRIPTION

Just define one trigger (combination of one or several events) but don't activate it. Use the "EventExtendedStart" function to activate this definition event(s). For each event, 4 parameters can be configured … see event specification to know if necessary. The events are defined in § "Events and Actions" in the XPS user's manual.

The number of events in a combination is limited to 10 events.



Each full event name is defined as **[actor].[category].event** (see Event list):
[actor] - Optional actor name (Group name, Positioner name, GPIO name or Nothing)
[category] - Optional category name (Event category or Nothing)
event - Event name

Event list

1. Always
2. Immediate
3. Timer1
4. Timer2
5. Timer3
6. Timer4
7. Timer5
8. PositionerName.MotionDone
9. PositionerName.PositionerError
10. PositionerName.PositionerHardwareStatus
11. PositionerName.Category.ConstantVelocityStart
12. PositionerName.Category.ConstantVelocityEnd
13. PositionerName.Category.ConstantVelocityState
14. PositionerName.Category.ConstantAccelerationStart
15. PositionerName.Category.ConstantAccelerationEnd
16. PositionerName.Category.ConstantAccelerationState
17. PositionerName.Category.ConstantDecelerationStart
18. PositionerName.Category.ConstantDecelerationEnd
19. PositionerName.Category.ConstantDecelerationState
20. PositionerName.Category.MotionStart
21. PositionerName.Category.MotionEnd

22. PositionerName.Category.MotionState
23. GroupName.Category.TrajectoryStart
24. GroupName.Category.TrajectoryEnd
25. GroupName.Category.TrajectoryState
26. GroupName.Category.TrajectoryPulse
27. GroupName.Category.TrajectoryPulseState
28. GroupName.Category.ElementNumberStart
29. GroupName.Category.ElementNumberState
30. GPIOName.ADCHighLimit
31. GPIOName.ADCLowLimit
32. GPIOName.DILowHigh
33. GPIOName.DIHighLow
34. GPIOName.DIToggle

Category list for "profile" positioner events

1. SGamma
2. Slave
3. Spin
4. Jog
5. TrackingPosition
6. TrackingVelocity

Category list for "trajectory" group events

1. XYLineArc
2. Spline
3. PVT

Event parameters

Actor				Category					Event name	Parameters			
Group	Positioner	GPIO	Timer #	SGamma	Jog	XY LineArc	XYZ Spline	PVT		1	2	3	4
									Immediate	0	0	0	0
									Always	0	0	0	0
			●						Timer	0	0	0	0
	●			●	●				MotionStart	0	0	0	0
	●			●	●				MotionStop	0	0	0	0
	●			●	●				MotionState	0	0	0	0
	●			●	●				ConstantVelocityStart	0	0	0	0
	●			●					ConstantVelocityEnd	0	0	0	0
	●			●	●				ConstantVelocityState	0	0	0	0
	●			●					ConstantAccelerationStart	0	0	0	0
	●			●					ConstantAccelerationEnd	0	0	0	0
	●			●					ConstantAccelerationState	0	0	0	0
	●			●					ConstantDecelerationStart	0	0	0	0
	●			●					ConstantDecelerationEnd	0	0	0	0
	●			●					ConstantDecelerationState	0	0	0	0
●						●	●	●	TrajectoryStart	0	0	0	0
●						●	●	●	TrajectoryEnd	0	0	0	0
●						●	●	●	TrajectoryState	0	0	0	0
●						●	●	●	ElementNumberStart	Element #	0	0	0
●						●	●	●	ElementNumberState	Element #	0	0	0
●	●								MotionDone	0	0	0	0
●						●		●	TrajectoryPulse	0	0	0	0
●						●		●	TrajectoryPulseOutputState	0	0	0	0
		●							DILowHigh	Bit index	0	0	0
		●							DIHighLow	Bit index	0	0	0
		●							DIToggle	Bit index	0	0	0
		●							ADCHighLimit	Value	0	0	0
		●							ADCLowLimit	Value	0	0	0

	•								PositionerError	Mask	0	0	0
	•								PositionerHardwareStatus	Mask	0	0	0

NOTE:
Before activating this event combination, you must define one or several action(s) with the "EventExtendedConfigurationTriggerSet" function. Next, use the "EventExtendedStart" or "EventExtendedWait" function to launch these definitions "event and action".

ERRORS

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_MNEMO_EVENT (-40)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
SUCCESS (0) : no error

TCL

Prototype

EventExtendedConfigurationTriggerSet $**SocketID **{$EventName $EventParameter1 $EventParameter2 $EventParameter3 $EventParameter4**} …**

Input parameters

SocketID ..... integer ..... Socket identifier got from "TCP_ConnectToServer" function
EventName..... string ..... event full name (maximum size = 250) - see § Events -
EventParameter1 ..... string ..... optional event parameter #1 (maximum size = 250)
EventParameter2 ..... string ..... optional event parameter #2 (maximum size = 250)
EventParameter3 ..... string ..... optional event parameter #3 (maximum size = 250)
EventParameter4 ..... string ..... optional event parameter #4 (maximum size = 250)

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

Note : **$**EventName, $EventParameter1, $EventParameter2, $EventParameter3 and $EventParameter4 can be repeted to set several action types.

C / C++

Prototype

int **EventExtendedConfigurationTriggerSet** (int SocketID, int NbElements, char* EventNameList, char* EventParameter1List, char* EventParameter2List , char* EventParameter3List, char* EventParameter4List)

Input parameters

SocketID ..... int ..... Socket identifier got from "TCP_ConnectToServer" function
NbElements..... int ..... number of events in configuration.
EventNameList ..... char* ..... event full name list (names separated by semicolon)
EventParameter1List..... char* ..... optional event parameter #1 list (separated by semicolon)
EventParameter2List..... char* ..... optional event parameter #2 list (separated by semicolon)
EventParameter3List..... char* ..... optional event parameter #3 list (separated by semicolon)
EventParameter4List..... char* ..... optional event parameter #4 list (separated by semicolon)

Output parameters

None

Return

Function error code

Note : EventNameList, EventParameter1List, EventParameter2List, EventParameter3List and EventParameter4List must have elements number in accordance to *NbElements*.

VISUAL BASIC

Prototype

Long **EventExtendedConfigurationTriggerSet** (ByVal SocketID As Long, ByVal NbElements As Long, ByVal EventNameList As String, ByVal EventParameter1List As String, ByVal EventParameter2List As String, ByVal EventParameter3List As String, ByVal EventParameter4List As String)

Input parameters

SocketID ..... Long ..... Socket identifier got from "TCP_ConnectToServer" function
NbElements..... Long ..... number of events in configuration.
EventNameList ..... String ..... event full name list (names separated by semicolon)
EventParameter1List..... String ..... optional event parameter #1 list (separated by semicolon)
EventParameter2List..... String ..... optional event parameter #2 list (separated by semicolon)
EventParameter3List..... String ..... optional event parameter #3 list (separated by semicolon)
EventParameter4List..... String ..... optional event parameter #4 list (separated by semicolon)

Output parameters

None

Return

Function error code

Note : EventNameList, EventParameter1List, EventParameter2List, EventParameter3List and EventParameter4List must have elements number in accordance to *NbElements*.

MATLAB

Prototype

int32 **EventExtendedConfigurationTriggerSet** (int32 SocketID, cstring EventNameList, cstring EventParameter1List, cstring EventParameter2List, cstring EventParameter3List, cstring EventParameter4List)

Input parameters

SocketID ..... int32 ..... Socket identifier got from "TCP_ConnectToServer" function
EventNameList ..... cstring ..... event full name list (names separated by space (blank) character)
EventParameter1List..... cstring ..... optional event parameter #1 list (separated by space character)
EventParameter2List..... cstring ..... optional event parameter #2 list (separated by space character)
EventParameter3List..... cstring ..... optional event parameter #3 list (separated by space character)
EventParameter4List..... cstring ..... optional event parameter #4 list (separated by space character)

Output parameters

None

Return

Function error code

Note : EventParameter1List, EventParameter2List, EventParameter3List and EventParameter4List must have the same elements number as that of EventNameList.

PYTHON

Prototype

integer **EventExtendedConfigurationTriggerSet** (integer SocketID, string list EventNameList, string list EventParameter1List, string list EventParameter2List, string list EventParameter3List, string list EventParameter4List)

Input parameters

SocketID ..... integer ..... Socket identifier got from "TCP_ConnectToServer" function
EventNameList ..... string ..... event full name list

EventParameter1List.....string list .....optional event parameter #1 list
EventParameter2List.....string list .....optional event parameter #2 list
EventParameter3List.....string list .....optional event parameter #3 list
EventParameter4List.....string list .....optional event parameter #4 list

Output parameters

None

Return

Function error code

Note : EventParameter1List, EventParameter2List, EventParameter3List and EventParameter4List must have the same elements number as that of EventNameList.

2.11.1.6. EventExtendedGet

NAME

EventExtendedGet – Return the detail of “event and action” combination in scheduler defined by an identifier.

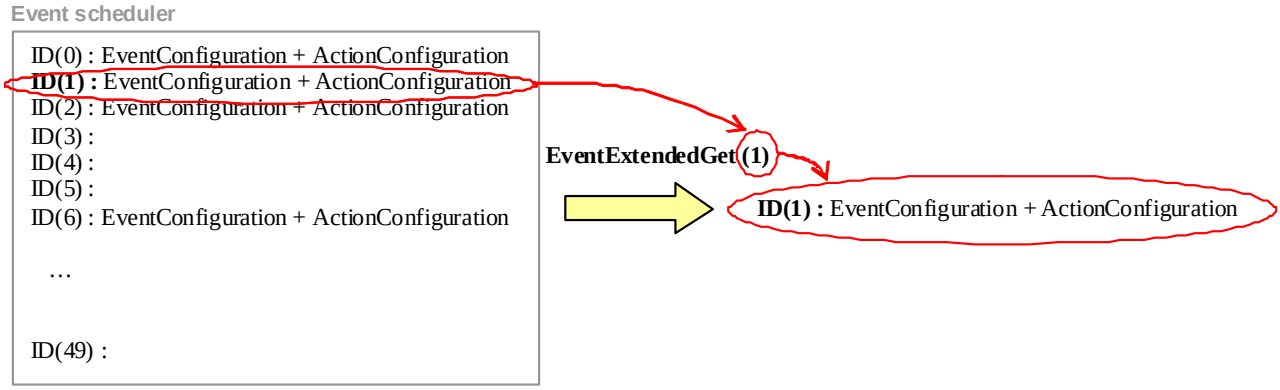
INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Parameter type: ERR_WRONG_TYPE_INT (-15), ERR_WRONG_TYPE_CHAR (-13)
- Event identifier [0:49]: ERR_EVENT_ID_UNDEFINED (-83)

DESCRIPTION

Get the composition of events and actions in progress defined by the identifier. This identifier is provided by the “EventExtendedStart” function.

The identifier must be defined between 0 and 49, if its value is “–1” then it’s not defined.
If the configured event is already deleted, the ERR_EVENT_ID_UNDEFINED (-83) error is returned.



ERRORS

SUCCESS (0) : no error
ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_INT (-15)
ERR_WRONG_TYPE_CHAR (-13)
ERR_EVENT_ID_UNDEFINED (-83)

TCL

Prototype

EventExtendedGet $SocketID $EventID EventConfiguration ActionConfiguration

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
EventID ..... integer ..... “Event and action” identifier from “ExtendedEventStart”

Output parameters

EventConfiguration ..... string ..... Event combination defined in scheduler
ActionConfiguration ..... string ..... Action combination defined in scheduler

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **EventExtendedGet** (int SocketID, int EventID, char * EventConfiguration, char * ActionConfiguration)

Input parameters

SocketID ..... int ... Socket identifier got from “TCP_ConnectToServer” function
EventID ..... int ... “Event and action” identifier from “ExtendedEventStart”

Output parameters

EventConfiguration ..... char * ..... Event combination defined in scheduler
ActionConfiguration ..... char * ..... Action combination defined in scheduler

Return

Function error code

VISUAL BASIC

Prototype

Long **EventExtendedGet** (ByVal SocketID As Long, ByVal EventID As Long, EventConfiguration As String, ActionConfiguration As String)

Input parameters

SocketID ..... Long Socket identifier got from “TCP_ConnectToServer” function
EventID ..... Long “Event and action” identifier from “ExtendedEventStart”

Output parameters

EventConfiguration ..... String ..... Event combination defined in scheduler
ActionConfiguration ..... String ..... Action combination defined in scheduler

Return

Function error code

MATLAB

Prototype

[Error, EventConfiguration, ActionConfiguration] **EventExtendedGet** (int32 SocketID, int32 EventID)

Input parameters

SocketID ..... int32 Socket identifier got from “TCP_ConnectToServer” function
EventID ..... int32 “Event and action” identifier from “ExtendedEventStart”

Return

Error ..... integer ..... Function error code
EventConfiguration ..... cstring ..... Event combination defined in scheduler
ActionConfiguration ..... cstring ..... Action combination defined in scheduler

PYTHON

Prototype

[Error, EventConfiguration, ActionConfiguration] **EventExtendedGet** (integer SocketID, integer EventID)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
EventID ..... integer ..... “Event and action” identifier from “ExtendedEventStart”

Return

Error ..... integer ..... Function error code
EventConfiguration ..... string ..... Event combination defined in scheduler
ActionConfiguration ..... string ..... Action combination defined in scheduler

2.11.1.7. EventExtendedRemove

NAME

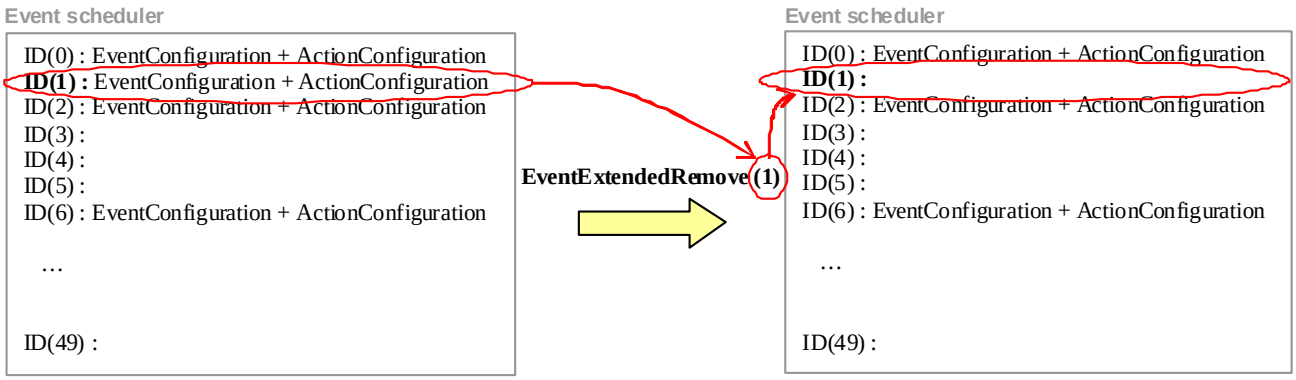
EventExtendedRemove – Remove an “event and action” combination in scheduler defined by an identifier.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of parameters [1]: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Parameter type: ERR_WRONG_TYPE_INT (-15)
- Event identifier [0:49]: ERR_EVENT_ID_UNDEFINED (-83), ERR_PARAMETER_OUT_OF_RANGE (-17)
- Actor event: ERR_WRONG_OBJECT_TYPE (-8)

DESCRIPTION

Delete the “event(s) and action(s)” combination in scheduler defined by an event identifier. This identifier is provided by the “EventExtendedStart” function.
The identifier must be defined between 0 and 49, if its value is “–1” then it’s not defined.
If the configured event is already deleted, the ERR_EVENT_ID_UNDEFINED (-83) error is returned.



ERRORS

ERR_EVENT_ID_UNDEFINED (-83)
ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

EventExtendedRemove $SocketID $EventID

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
EventID ..... integer ..... “Event and action” identifier

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **EventExtendedRemove** (int SocketID, int EventID)

Input parameters

SocketID ..... int ... Socket identifier got from “TCP_ConnectToServer” function
EventID ..... int ... “Event and action” identifier

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **EventExtendedRemove** (ByVal SocketID As Long, ByVal EventID As Long)

Input parameters

SocketID ..... Long Socket identifier got from “TCP_ConnectToServer” function
EventID ..... Long “Event and action” identifier

Output parameters

None

Return

Function error code

MATLAB

Prototype

[Error] **EventExtendedRemove** (int32 SocketID, int32 EventID)

Input parameters

SocketID ..... int32 Socket identifier got from “TCP_ConnectToServer” function
EventID ..... int32 “Event and action” identifier

Return

Error ..... int32 Function error code

PYTHON

Prototype

[Error] **EventExtendedRemove** (integer SocketID, integer EventID)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
EventID ..... integer ..... “Event and action” identifier

Return

Error ..... integer ..... Function error code

2.11.1.8. EventExtendedStart

NAME

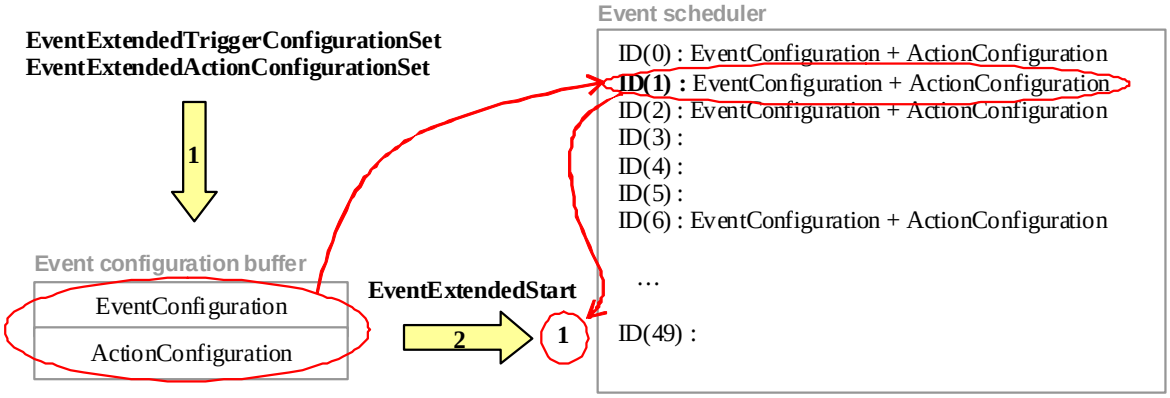
EventExtendedStart – Activate the “event and action” defined in buffer.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of arguments: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Output parameter type: ERR_WRONG_TYPE_INT (-15)
- Number of compositions in execution: ERR_EVENT_BUFFER_FULL (-82)
- Last event configuration in memory: ERR_EVENTS_NOT_CONFIGURED (-80)
- Last action configuration in memory: ERR_ACTIONS_NOT_CONFIGURED (-81)
- Event name to execute: ERR_MNEMO_EVENT (-40), ERR_WRONG_TYPE (-10), ERR_WRONG_OBJECT_TYPE (-8)

DESCRIPTION

Launch the configured event(s) and action(s) from the event configuration buffer to fill it in the event scheduler and gets an event identifier. The identifier must be defined between 0 and 49, if its value is “–1” then that means it’s not defined.
If no event is configured in buffer, the ERR_EVENTS_NOT_CONFIGURED (-80) error is returned.
If no action is configured in buffer, the ERR_ACTIONS_NOT_CONFIGURED error is returned.



NOTE:
In the event scheduler, when a configured event is occurred then it is deleted and free its space in event scheduler.

CAUTION:
If the configured event is PERMANENT then it is not deleted after to be occurred … it must use the “EventExtendedRemove” function to delete it.

ERRORS

ERR_ACTIONS_NOT_CONFIGURED (-81)
ERR_EVENT_BUFFER_FULL (-82)
ERR_EVENTS_NOT_CONFIGURED (-80)
ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_MNEMO_EVENT (-40)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE (-10)
ERR_WRONG_TYPE_INT (-15)
SUCCESS (0) : no error

TCL

Prototype

EventExtendedStart $SocketID EventID

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

EventID ..... integer ..... “Event and action” identifier

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **EventExtendedStart** (int SocketID, int * EventID)

Input parameters

SocketID ..... int ... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

EventID ..... int * “Event and action” identifier

Return

Function error code

VISUAL BASIC

Prototype

Long **EventExtendedStart** (ByVal SocketID As Long, EventID As Long)

Input parameters

SocketID ..... LongSocket identifier got from “TCP_ConnectToServer” function

Output parameters

EventID ..... Long“Event and action” identifier

Return

Function error code

MATLAB

Prototype

[Error, EventID] **EventExtendedStart** (int32 SocketID)

Input parameters

SocketID ..... int32 Socket identifier got from “TCP_ConnectToServer” function

Return

Error ..... int32 Function error code
EventID ..... int32 “Event and action” identifier

PYTHON

Prototype

[Error, EventID] **EventExtendedStart** (integer SocketID)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function

Return

Error ..... integer ..... Function error code
EventID ..... integer ..... “Event and action” identifier

2.11.1.9. EventExtendedWait

NAME

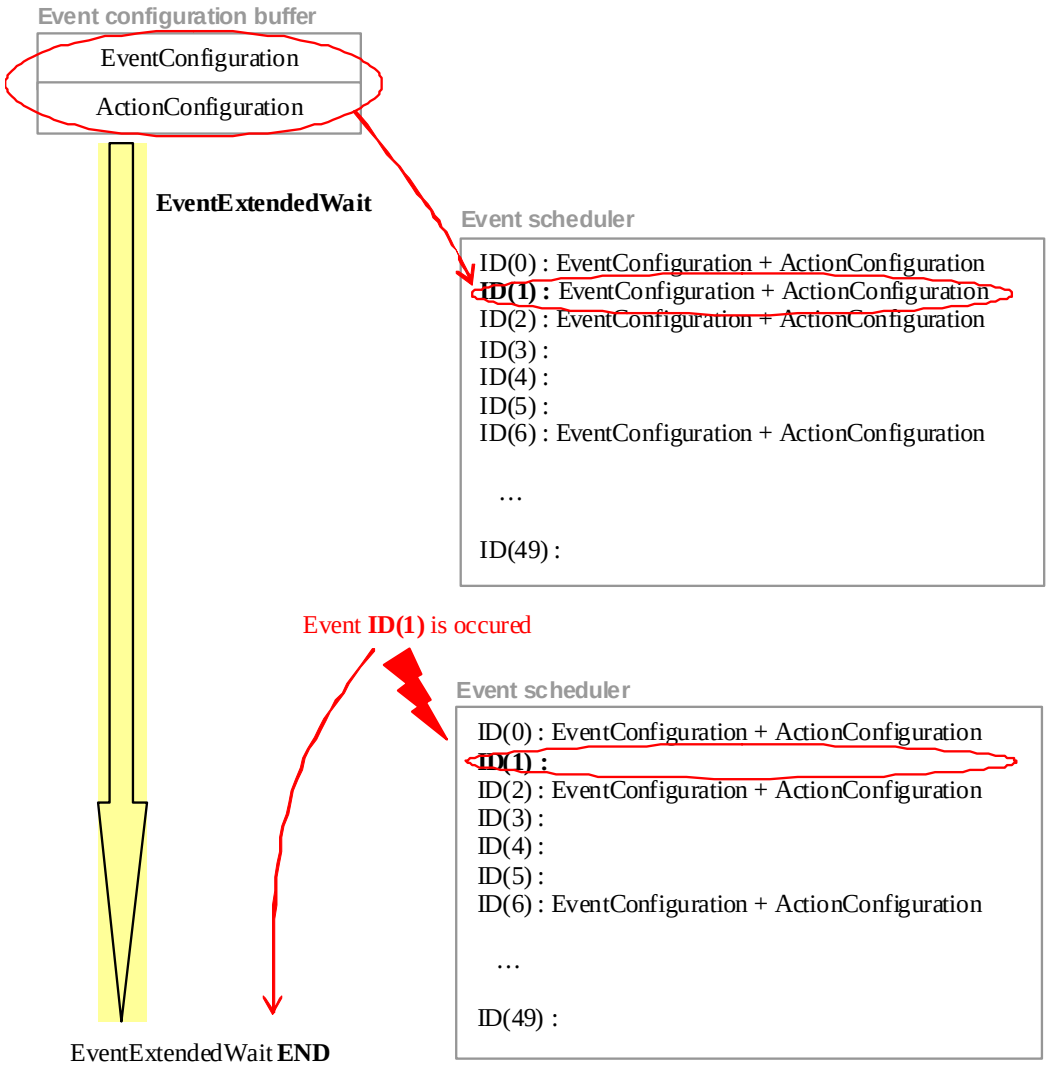
EventExtendedWait – Activate the last “event” configuration in memory and wait it occurs.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of arguments [0]: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Number of compositions in execution: ERR_EVENT_BUFFER_FULL (-82)
- Last event configuration in memory: ERR_EVENTS_NOT_CONFIGURED (-80)
- Event name to execute: ERR_MNEMO_EVENT (-40), ERR_WRONG_TYPE (-10)
- Event actor: ERR_WRONG_OBJECT_TYPE (-8)

DESCRIPTION

Launch the last configured event(s) to fill it in the event scheduler and wait it occurs to unlock the socket.
If no “event and action” combination is configured in the event configuration buffer, the ERR_EVENTS_NOT_CONFIGURED (-80) error is returned.



ERRORS

ERR_EVENT_BUFFER_FULL (-82)
ERR_EVENTS_NOT_CONFIGURED (-80)
ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_MNEMO_EVENT (-40)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_OBJECT_TYPE (-8)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_ (-10)
SUCCESS (0) : no error

TCL

Prototype
EventExtendedWait $SocketID

Input parameters
SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters
None

Return
TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype
int **EventExtendedWait** (int SocketID)

Input parameters
SocketID ..... int ... Socket identifier got from “TCP_ConnectToServer” function

Output parameters
None

Return
Function error code

VISUAL BASIC

Prototype
Long **EventExtendedWait** (ByVal SocketID As Long)

Input parameters
SocketID ..... Long Socket identifier got from “TCP_ConnectToServer” function

Output parameters
None

Return
Function error code

MATLAB

Prototype
[Error] **EventExtendedWait** (int32 SocketID)

Input parameters
SocketID ..... int32 Socket identifier got from “TCP_ConnectToServer” function

Return
Error ..... int32 Function error code

PYTHON

Prototype

[Error] **EventExtendedWait** (integer SocketID)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function

Return

Error ..... integer ..... Function error code

2.11.2. Obsolete Functions Description

Do not use for new projects! These Functions are only maintained for already written programs.
Use above described Functions.

2.11.2.1. EventAdd

TCL Prototype	int EventAdd (int SocketID, char FullPositionerName[250], char EventName[250], char EventParameter[250], char ActionName[250], char ActionParameter1[250], char ActionParameter2[250], char ActionParameter3[250])
Input parameters	int : SocketID (Socket identifier got from "TCP_ConnectToServer" function) char [250] : FullPositionerName char [250] : EventName (see § Events) char [250] : EventParameter char [250] : ActionName (see § Actions) char [250] : ActionParameter1 char [250] : ActionParameter2 char [250] : ActionParameter3
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or Function error

DLL Prototype	int EventAdd (int SocketID, char *FullPositionerName, char *EventName, char *EventParameter, char *ActionName, char *ActionParameter1, char *ActionParameter2, char *ActionParameter3)
Input parameters	int : SocketID (Socket identifier got from "TCP_ConnectToServer" function) char * : FullPositionerName char * : EventName (see § Events) char * : EventParameter char * : ActionName (see § Actions) char * : ActionParameter1 char * : ActionParameter2 char * : ActionParameter3
Output parameters	None
Return	Function error

Function Input tests	Verify the number of parameters. Verify the full positioner name, the event name and the action name. Verify the type of all output parameters. Parameters coherence test.
Function Description	Adds an action associated to an event for the defined positioner. For a more thorough description and a complete list of possible events and the actions, please refer to the XPS Motion Tutorial, section named Event triggers.
Function Errors	0 -7 -8 -9 -13 -39 -40

2.11.2.2. EventGet

TCL Prototype	int EventGet (int SocketID, char FullPositionerName [250], char EventList[250])
Input parameters	int : SocketID (Socket identifier got from “TCP_ConnectToServer” function) char [250] : FullPositionerName
Output parameters	char [250] : EventList
Return	TCL error (0 = success or 1 = syntax error) or Function error

DLL Prototype	int EventGet (int SocketID, char *FullPositionerName, char *EventList)
Input parameters	int : SocketID (Socket identifier got from “TCP_ConnectToServer” function) char * : FullPositionerName
Output parameters	char * : EventList
Return	Function error

Function Input tests	Verify the number of parameters. Verify the positioner name and the event name. Verify the type of all output parameters. Parameters coherence test.
Function Description	Returns the list of events and actions in progress for the selected positioner. For a more thorough description and a complete list of possible events and the actions, please refer to the XPS Motion Tutorial, section named Event triggers.
Function Errors	0 -7 -8 -9 -13

2.11.2.3. EventRemove

TCL Prototype	int EventRemove (int SocketID, char FullPositionerName[250], char EventName[250] , char EventParameter[250])
Input parameters	int : SocketID (Socket identifier got from “TCP_ConnectToServer” function) char [250] : FullPositionerName char [250] : EventName (see § Events) char [250] : EventParameter
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or Function error

DLL Prototype	int EventRemove (int SocketID, char *FullPositionerName, char *EventName , char *EventParameter)
Input parameters	int : SocketID (Socket identifier got from “TCP_ConnectToServer” function) char * : FullPositionerName char * : EventName (see § Events) char * : EventParameter
Output parameters	None
Return	Function error

Function Input tests	Verify the number of parameters. Verify the positioner name and the event name. Verify the event. Verify the type of all output parameters. Parameters coherence test.
Function Description	Deletes an action associated to an event for the defined positioner. For a more thorough description and a complete list of possible events and the actions, please refer to the XPS Motion Tutorial, section named Event triggers.
Function Errors	0 -7 -8 -9 -13 -40

2.11.2.4. EventWait

TCL Prototype	int EventWait (int SocketID, char FullPositionerName [250], char EventName[250] , char EventParameter[250])
Input parameters	int : SocketID (Socket identifier got from “TCP_ConnectToServer” function) char [250] : FullPositionerName char [250] : EventName (see § Events) char [250] : EventParameter
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or Function error

DLL Prototype	int EventWait (int SocketID, char *FullPositionerName, char *EventName , char *EventParameter)
Input parameters	int : SocketID (Socket identifier got from “TCP_ConnectToServer” function) char * : FullPositionerName char * : EventName (see § Events) char * : EventParameter
Output parameters	None
Return	Function error

Function Input tests	Verify the number of parameters. Verify the positioner name and the event name. Verify the event. Verify the type of all output parameters. Parameters coherence test.
Function Description	Waits for an event for the selected positioner. The socket is locked. As soon as the event occurs, the socket gets unlocked. For a more thorough description and a complete list of possible events and the actions, please refer to the XPS Motion Tutorial, section named Event triggers.
Function Errors	0 -7 -8 -9 -13 -40

2.12. TCL Programming

2.12.1. Function Description

2.12.1.1. TCLScriptExecute

NAME

TCLScriptExecute – Executes a TCL script.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of arguments: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check task name: ERR_WRONG_TCL_TASKNAME (-47)
- Check TCL file name: ERR_UNKNOWN_TCL_FILE (-36)
- Check TCL interpretor (task loading): ERR_TCL_INTERPRETOR (-37)

DESCRIPTION

This function executes a TCL script. The TCL script file must be saved in the folder "..\Public\Scripts" of the XPS controller.

- ✓ *TaskName* is a user designation for the TCL script in execution. If two TCL scripts are executed at the same time with the same task name, The ERR_WRONG_TCL_TASKNAME (-47) is returned because it is not allowed to have the same TaskName.
- ✓ *InputArguments* represents the input arguments to the TCL script to be executed. The number of these input arguments is not limited but the string length is limited to 250 characters. The argument separator is a comma.

ERRORS

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_TCL_INTERPRETOR (-37)
ERR_UNKNOWN_TCL_FILE (-36)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TCL_TASKNAME (-47)
SUCCESS (0) : no error

TCL

Prototype

TCLScriptExecute $SocketID $TCLFileName $TaskName $InputArguments

Input parameters

SocketID ..... integer ..... Socket identifier got from "TCP_ConnectToServer" function
TCLFileName ..... string ..... File name contains the TCL script
TaskName ..... string ..... Task name
InputArguments ..... string ..... Input argument string (separator is a comma)

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **TCLScriptExecute** (int SocketID, char *TCLFileName, char *TaskName, char *InputArguments)

Input parameters

SocketID	int	Socket identifier got from “TCP_ConnectToServer” function
TCLFileName	char *	File name contains the TCL script
TaskName	char *	Task name
InputArguments	char *	Input argument string (separator is a comma)

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **TCLScriptExecute** (ByVal SocketID As Long, ByVal TCLFileName As String, ByVal TaskName As String, ByVal InputArguments As String)

Input parameters

SocketID	Long	Socket identifier got from “TCP_ConnectToServer” function
TCLFileName	String	File name contains the TCL script
TaskName	String	Task name
InputArguments	String	Input argument string (separator is a comma)

Output parameters

None

Return

Function error code

MATLAB

Prototype

[Error] **TCLScriptExecute** (int32 SocketID, cstring TCLFileName, cstring TaskName, cstring InputArguments)

Input parameters

SocketID	int32	Socket identifier got from “TCP_ConnectToServer” function
TCLFileName	cstring	File name contains the TCL script
TaskName	cstring	Task name
InputArguments	cstring	Input argument string (separator is a comma)

Return

Error	int32	Function error code

PYTHON

Prototype

[Error] **TCLScriptExecute** (integer SocketID, string TCLFileName, string TaskName, string InputArguments)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
TCLFileName	string	File name contains the TCL script
TaskName	string	Task name
InputArguments	string	Input argument string (separator is a comma)

Return

Error	integer	Function error code

2.12.1.2. TCLScriptExecuteAndWait

NAME

TCLScriptExecuteAndWait – Executes a TCL script and waits the end of execution.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of arguments: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check task name: ERR_WRONG_TCL_TASKNAME (-47)
- Check TCL file name: ERR_UNKNOWN_TCL_FILE (-36)
- Check TCL interpretor (task loading): ERR_TCL_INTERPRETOR (-37)

DESCRIPTION

This function executes a TCL program. The "TCLScriptExecuteAndWait" function is different than the "TCLScriptExecute" function because it blocks the socket until the script teminates. The TCL script file must be saved in the folder "..\Public\Scripts" of the XPS controller. The file extension is ".tcl".

- ✓ *TaskName* is a user designation for the TCL script in execution. If two TCL scripts are executed at the same time with the same task name, The ERR_WRONG_TCL_TASKNAME (-47) is returned because it is not allowed to have the same TaskName.
- ✓ *InputArguments* represents the input arguments to the TCL script to be executed. The number of these input arguments is not limited but the string length is limited to 250 characters. The argument separator is a comma.
- ✓ *OutputArguments* represents the output arguments to the TCL script to be executed. The number of these output arguments is not limited but the string length is limited to 250 characters. The argument separator is a comma.

ERRORS

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_TCL_INTERPRETOR (-37)
ERR_UNKNOWN_TCL_FILE (-36)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TCL_TASKNAME (-47)
SUCCESS (0) : no error

TCL

Prototype

TCLScriptExecuteAndWait $SocketID $TCLFileName $TaskName $InputArguments OutputArguments

Input parameters

SocketID ..... integer ..... Socket identifier got from "TCP_ConnectToServer" function
TCLFileName ..... string ..... File name contains the TCL script
TaskName ..... string ..... Task name
InputArguments ..... string ..... Input argument string (separator is a comma)

Output parameters

OutputArguments..... string ..... Output argument string (separator is a comma)

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **TCLScriptExecuteAndWait** (int SocketID, char *TCLFileName, char *TaskName, char *InputArguments, char *OutputArguments)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
TCLFileName ..... char * ..... File name contains the TCL script
TaskName ..... char * ..... Task name
InputArguments ..... char * ..... Input argument string (separator is a comma)

Output parameters

OutputArguments..... char * ..... Output argument string (separator is a comma)

Return

Function error code

VISUAL BASIC

Prototype

Long **TCLScriptExecuteAndWait** (ByVal SocketID As Long, ByVal TCLFileName As String, ByVal TaskName As String, ByVal InputArguments As String)

Input parameters

SocketID ..... Long..... Socket identifier got from “TCP_ConnectToServer” function
TCLFileName ..... String ..... File name contains the TCL script
TaskName ..... String ..... Task name
InputArguments ..... String ..... Input argument string (separator is a comma)

Output parameters

OutputArguments..... String ..... Output argument string (separator is a comma)

Return

Function error code

MATLAB

Prototype

[Error] **TCLScriptExecuteAndWait** (int32 SocketID, cstring TCLFileName, cstring TaskName, cstring InputArguments, cstring OutputArguments)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
TCLFileName ..... cstring ..... File name contains the TCL script
TaskName ..... cstring ..... Task name
InputArguments ..... cstring ..... Input argument string (separator is a comma)

Return

Error..... int32 ..... Function error code
OutputArguments..... cstring ..... Output argument string (separator is a comma)

PYTHON

Prototype

[Error,OutputArguments] **TCLScriptExecuteAndWait** (integer SocketID, string TCLFileName, string TaskName, string InputArguments)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
TCLFileName ..... string ..... File name contains the TCL script
TaskName ..... string ..... Task name
InputArguments ..... string ..... Input argument string (separator is a comma)

Return

Error..... integer ..... Function error code
OutputArguments..... string ..... Output argument string (separator is a comma)

2.12.1.3. TCLScriptExecuteWithPriority

NAME

TCLScriptExecuteWithPriority – Execute a TCL script with TCL task given priority.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of arguments: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check task name: ERR_WRONG_TCL_TASKNAME (-47)
- Priority mnemonic incorrect: ERR_PARAMETERS_OUT_OF_RANGE (-17)
- Check TCL file name: ERR_UNKNOWN_TCL_FILE (-36)
- Check TCL interpretor (task loading): ERR_TCL_INTERPRETOR (-37)

DESCRIPTION

This function executes a TCL script with the TCL task that its priority level is defined by user. The TCL script file must be saved in the folder "..\Public\Scripts" of the XPS controller.

- ✓ *TaskName* is a user designation for the TCL script in execution. If two TCL scripts are executed at the same time with the same task name, The ERR_WRONG_TCL_TASKNAME (-47) is returned because it is not allowed to have the same TaskName.

- ✓ *InputArguments* represents the input arguments to the TCL script to be executed. The number of these input arguments is not limited but the string length is limited to 250 characters. The argument separator is a comma.

- ✓ *PriorityLevel* has three possible values : "HIGH", "MEDIUM" and "LOW"
  - o *HIGH* (task prioriry = 30) : TCL task with priority higher than FTP (55), DHCP (56), HTTP (200), Telnet (215) tasks's one.
  - o *MEDIUM* (task prioriry = 105) : TCL task with priority lower than FTP, DHCP tasks's one but higher than HTTP and Telnet tasks's.
  - o *LOW* (task prioriry = 205) : TCL task piority lower than FTP, DHCP, HTTP tasks's one, but higher Telnet tasks's. It is the same priority of the TCL task created with TCLScriptExecute() function.

ERRORS

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_TCL_INTERPRETOR (-37)
ERR_UNKNOWN_TCL_FILE (-36)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_PARAMETER_OUT_OF_RANGE (-17)
ERR_WRONG_TCL_TASKNAME (-47)
SUCCESS (0) : no error

TCL

Prototype

TCLScriptExecuteWithPriority $SocketID $TCLFileName $TaskName $Priority $InputArguments

Input parameters

SocketID ..... integer ..... Socket identifier gets by the "TCP_ConnectToServer" Function
TCLFileName ..... string ..... File name contains the TCL script
TaskName ..... string ..... Task name
Priority ..... string ..... TCL task priority (HIGH, MEDIUM or LOW)

InputArguments ..... string ..... Input argument string (separator is a comma)

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **TCLScriptExecuteWithPriority** (int SocketID, char *TCLFileName, char *TaskName, char *Priority, char *InputArguments)

Input parameters

SocketID ..... int ..... Socket identifier gets by the “TCP_ConnectToServer” function
TCLFileName ..... char * ..... File name contains the TCL script
TaskName ..... char * ..... Task name
Priority ..... char * ..... TCL task priority (HIGH, MEDIUM or LOW)
InputArguments ..... char * ..... Input argument string (separator is a comma)

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **TCLScriptExecuteWithPriority** (ByVal SocketID As Long, ByVal TCLFileName As String, ByVal TaskName As String, ByVal Priority As String , ByVal InputArguments As String)

Input parameters

SocketID ..... Long ..... Socket identifier gets by the “TCP_ConnectToServer” function
TCLFileName ..... String ..... File name contains the TCL script
TaskName ..... String ..... Task name
Priority ..... String ..... TCL task priority (HIGH, MEDIUM or LOW)
InputArguments ..... String ..... Input argument string (separator is a comma)

Output parameters

None

Return

Function error code

MATLAB

Prototype

[Error] **TCLScriptExecuteWithPriority** (int32 SocketID, cstring TCLFileName, cstring TaskName, cstring Priority, cstring InputArguments)

Input parameters

SocketID ..... int32 ..... Socket identifier gets by the “TCP_ConnectToServer” function
TCLFileName ..... cstring ..... File name contains the TCL script
TaskName ..... cstring ..... Task name
Priority ..... cstring ..... TCL task priority (HIGH, MEDIUM or LOW)
InputArguments ..... cstring ..... Input argument string (separator is a comma)

Return

Error ..... int32 Function error code

PYTHON

Prototype

[Error] **TCLScriptExecuteWithPriority** (integer SocketID, string TCLFileName, string TaskName, string Priority, string InputArguments)

Input parameters

SocketID	integer	Socket identifier gets by the “TCP_ConnectToServer” Function
TCLFileName	string	File name contains the TCL script
TaskName	string	Task name
Priority	string	TCL task priority (HIGH, MEDIUM or LOW)
InputArguments	string	Input argument string (separator is a comma)

Return

Error	integer	Function error code

2.12.1.4. TCLScriptKill

NAME

TCLScriptKill – Kills a TCL script.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of arguments: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check TCL interpretor (task loading) and task name: ERR_TCL_SCRIPT_KILL (-38)
- Check semaphore to use the TCL interpretor: ERR_TCL_INTERPRETOR (-37)

DESCRIPTION

This function kills a running TCL script selected by its task name. The task name is a user designation for the TCL script in execution.

NOTE:
For the boot script, the task name is “BootScript”.

ERRORS

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_TCL_INTERPRETOR (-37)
ERR_TCL_SCRIPT_KILL (-38)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TCL_TASKNAME (-47)
SUCCESS (0) : no error

TCL

Prototype

TCLScriptKill $SocketID $TaskName

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
TaskName ..... string ..... Task name to kill

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **TCLScriptKill** (int SocketID, char *TaskName)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
TaskName ..... char * ..... Task name to kill

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **TCLScriptKill** (ByVal SocketID As Long, ByVal TaskName As String)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
TaskName ..... String ..... Task name to kill

Output parameters

None

Return

Function error code

MATLAB

Prototype

[Error] **TCLScriptKill** (int32 SocketID, cstring TaskName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
TaskName ..... cstring ..... Task name to kill

Return

Error ..... int32 ..... Function error code

PYTHON

Prototype

[Error] **TCLScriptKill** (integer SocketID, string TaskName)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
TaskName ..... string ..... Task name to kill

Return

Error ..... integer ..... Function error code

2.13. Optional module programming

The optional module programming allows to manage (load, execute and stop execution) the written by user program blocks (optional modules) inside the XPS controller, with the following conditions :

- Every optional module is written in C language (GNU with WindRiver Tornado) and built under the form *"*.out"*.
- The module name must begin with *"OptionalModule"*, for example *"OptionalModule_LaserManage.out"*.
- The module main function must has the same name that the module name, for example :

```
void OptionalModule_LaserManage(void)
{
    …
}
```

- The optional module file must be in the "/ADMIN/Firmware/" on the XPS controller.
- All optional modules need the "clib.out" module file (in the "/ADMIN/Firmware/") to interface with the XPS controller. The "clib.out" is loaded automatically with the OptionalModuleExecute() function. Don't put the "clib.out" module in the "system.ref" file, if you do so the OptionalModuleExecute() function will not work (return ERR_OPTIONAL_EXTERNAL_MODULE_LOAD).

2.13.1. Function Description

2.13.1.1. OptionalModuleExecute

NAME

OptionalModuleExecute – Executes an optional (user) external module.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of arguments: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check task name: ERR_OPTIONAL_EXTERNAL_MODULE_EXECUTE (-95)
- Check module file name: ERR_OPTIONAL_EXTERNAL_MODULE_FILE (-94)
- Check module loading: ERR_OPTIONAL_EXTERNAL_MODULE_LOAD (-97)

DESCRIPTION

This function executes a optinal (user) module. The optional module file must be saved in the folder "\ADMIN\Firmware" of the XPS controller.

- ✓ An optinal module has always a main function whose name is identical to the module name. The term "execute a module" means executing its main function.
- ✓ *TaskName* is a user designation for the module in execution. It is not possible to execute a module at the same time with a same task name, elsewhere the ERR_OPTIONAL_EXTERNAL_MODULE_EXECUTE (-95) is returned.

ERRORS

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_OPTIONAL_EXTERNAL_MODULE_FILE (-94)
ERR_OPTIONAL_EXTERNAL_MODULE_EXECUTE (-95)
ERR_OPTIONAL_EXTERNAL_MODULE_LOAD (-97)
SUCCESS (0) : no error

TCL

Prototype

OptionalModuleExecute $SocketID $ModuleFileName $TaskName

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
ModuleFileName ..... string ..... Module file name
TaskName ..... string ..... Task name

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **OptionalModuleExecute** (int SocketID, char * ModuleFileName, char *TaskName)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
ModuleFileName ..... char * ..... Module file name
TaskName ..... char * ..... Task name

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **OptionalModuleExecute** (ByVal SocketID As Long, ByVal ModuleFileName As String, ByVal TaskName As String)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
ModuleFileName ..... String ..... Module file name
TaskName ..... String ..... Task name

Output parameters

None

Return

Function error code

MATLAB

Prototype

[Error] **OptionalModuleExecute** (int32 SocketID, cstring ModuleFileName, cstring TaskName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
ModuleFileName ..... cstring ..... Module file name
TaskName ..... cstring ..... Task name

Return

Error ..... int32 Function error code

PYTHON

Prototype

[Error] **OptionalModuleExecute** (integer SocketID, string ModuleFileName, string TaskName)

Input parameters

SocketID	integer	Socket identifier got from “TCP_ConnectToServer” function
ModuleFileName	string	Module file name
TaskName	string	Task name

Return

Error	integer	Function error code

2.13.1.2. OptionalModuleKill

NAME

OptionalModuleKill – Kill the execution of a optional module.

INPUT TESTS

- Configuration files reading: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of arguments: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check task name and task killing : ERR_OPTIONAL_EXTERNAL_MODULE_KILL (-96)

DESCRIPTION

This function kills a running optional module selected by its task name. The task name is a user designation for the optional module in execution *(see OptionalModuleExecute)*.

ERRORS

ERR_FATAL_INIT (-20)
ERR_IN_INITIALIZATION (-21)
ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_OPTIONAL_EXTERNAL_MODULE_KILL (-96)
SUCCESS (0) : no error

TCL

Prototype

OptionalModuleKill $SocketID $TaskName

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
TaskName ..... string ..... Task name to kill

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **OptionalModuleKill** (int SocketID, char *TaskName)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function
TaskName ..... char * ..... Task name to kill

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **OptionalModuleKill** (ByVal SocketID As Long, ByVal TaskName As String)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function
TaskName ..... String ..... Task name to kill

Output parameters

None

Return

Function error code

MATLAB

Prototype

[Error] **OptionalModuleKill** (int32 SocketID, cstring TaskName)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function
TaskName ..... cstring ..... Task name to kill

Return

Error ..... int32 ..... Function error code

PYTHON

Prototype

[Error] **OptionalModuleKill** (integer SocketID, string TaskName)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function
TaskName ..... string ..... Task name to kill

Return

Error ..... integer ..... Function error code

2.14. Hardware date and time setting

2.14.1. Function Description

2.14.1.1. HardwareDateAndTimeGet

NAME

HardwareDateAndTimeGet – Get the current date and time.

INPUT TESTS

- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check output parameter type: ERR_WRONG_TYPE_CHAR (-13)
- Check output format: ERR_WRONG_FORMAT (-7)

DESCRIPTION

This function gets the current date and time of the XPS controller under the form "***WeekDay Month Day Hour:Minute:Second Year*** ", for example "*Tue Jan 15 10:28:06 2008*".

ERRORS

ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
ERR_WRONG_FORMAT (-7)
SUCCESS (0) : no error

TCL

Prototype

HardwareDateAndTimeGet $SocketID DateAndTime

Input parameters

SocketID .....integer.....Socket identifier gets by the "TCP_ConnectToServer" function

Output parameters

DateAndTime.....string .....Controller date and time

Return

Error.....integer.....TCL error code (0 = success or 1 = syntax error) or function error code

C / C++

Prototype

int **HardwareDateAndTimeGet** (int SocketID, char * DateAndTime)

Input parameters

SocketID .....int .....Socket identifier gets by the "TCP_ConnectToServer" function

Output parameters

DateAndTime.....char *.....Controller date and time

Return

Error.....int .....Function error code

VISUAL BASIC

Prototype

Long **HardwareDateAndTimeGet** (ByVal SocketID As Long, ByVal DateAndTime As String)

Input parameters

SocketID ..... Long ..... Socket identifier gets by the “TCP_ConnectToServer” function

Output parameters

DateAndTime..... String ..... Controller date and time

Return

Error..... Long..... Function error code

MATLAB

Prototype

[Error, DateAndTime] **HardwareDateAndTimeGet** (int32 SocketID)

Input parameters

SocketID ..... int32 ..... Socket identifier gets by the “TCP_ConnectToServer” function

Return

Error..... int32 ..... Function error code
DateAndTime..... cstring ..... Controller date and time

PYTHON

Prototype

[Error, DateAndTime] **HardwareDateAndTimeGet** (integer SocketID)

Input parameters

SocketID ..... integer ..... Socket identifier gets by the “TCP_ConnectToServer” function

Return

Error..... integer ..... Function error code
DateAndTime..... string ..... Controller date and time

2.14.1.2. HardwareDateAndTimeSet

NAME

HardwareDateAndTimeSet – Set the date and time.

INPUT TESTS

- Check command format: ERR_WRONG_FORMAT (-7)
- Verify the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check intput parameter type: ERR_WRONG_TYPE_CHAR (-13)

DESCRIPTION

This function sets the date and time of the XPS controller. The setting form must be "***WeekDay Month Day Hour:Minute:Second Year*** ", for example "*Tue Jan 15 10:28:06 2008*".

ERRORS

ERR_WRONG_FORMAT (-7)
ERR_WRONG_PARAMETERS_NUMBER (-9)
ERR_WRONG_TYPE_CHAR (-13)
SUCCESS (0) : no error

TCL

Prototype

HardwareDateAndTimeSet $SocketID $DateAndTime

Input parameters

SocketID ..... integer ..... Socket identifier gets by the "TCP_ConnectToServer" Function
DateAndTime..... string ..... Date and time to set

Output parameters

None

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

int **HardwareDateAndTimeSet** (int SocketID, char * DateAndTime)

Input parameters

SocketID ..... int ..... Socket identifier gets by the "TCP_ConnectToServer" function
DateAndTime..... char * ..... Date and time to set

Output parameters

None

Return

Function error code

VISUAL BASIC

Prototype

Long **HardwareDateAndTimeSet** (ByVal SocketID As Long, ByVal DateAndTime As String)

Input parameters

SocketID ..... Long ..... Socket identifier gets by the "TCP_ConnectToServer" function
DateAndTime..... String ..... Date and time to set

Output parameters
None

Return
Function error code

MATLAB

Prototype
[Error] **HardwareDateAndTimeSet** (int32 SocketID, cstring DateAndTime)

Input parameters
SocketID ..... int32 ..... Socket identifier gets by the “TCP_ConnectToServer” function
DateAndTime..... cstring ..... Date and time to set

Return
Error..... int32 ..... Function error code

PYTHON

Prototype
[Error] **HardwareDateAndTimeSet** (integer SocketID, string DateAndTime)

Input parameters
SocketID ..... integer ..... Socket identifier gets by the “TCP_ConnectToServer” Function
DateAndTime..... string ..... Date and time to set

Return
Error..... integer ..... Function error code

2.15. Version

2.15.1. Function Description

2.15.1.1. GetLibraryVersion

NAME

GetLibraryVersion – Gets the version of DLL library.

INPUT TESTS

None

DESCRIPTION

This function returns the version of DLL library.
The library version represents the firmware version that used to build the library.

ERRORS

None

TCL

Prototype

GetLibraryVersion $SocketID LibVersion

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

LibVersion ..... string ..... DLL library version

Return

TCL error code (0 = success or 1 = syntax error) or Function error code

C / C++

Prototype

char * **GetLibraryVersion** (int SocketID)

Input parameters

SocketID ..... int ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

None

Return

LibVersion ..... char * ..... DLL library version

VISUAL BASIC

Prototype

String **GetLibraryVersion** (ByVal SocketID As Long)

Input parameters

SocketID ..... Long ..... Socket identifier got from “TCP_ConnectToServer” function

Output parameters

None

Return

LibVersion ..... String ..... DLL library version

MATLAB

Prototype

[LibVersion] **GetLibraryVersion** (int32 SocketID)

Input parameters

SocketID ..... int32 ..... Socket identifier got from “TCP_ConnectToServer” function

Return

LibVersion ..... cstring ..... DLL library version

PYTHON

Prototype

[LibVersion] **GetLibraryVersion** (integer SocketID)

Input parameters

SocketID ..... integer ..... Socket identifier got from “TCP_ConnectToServer” function

Return

LibVersion ..... string ..... DLL library version

2.16. Positioner error list

code	Error description
0	
0x80000001	General inhibition detected
0x80000002	Fatal following error detected
0x80000004	Home search time out
0x80000008	Motion done time out
0x80000010	Requested position exceed travel limits in trajectory or slave mode
0x80000020	Requested velocity exceed maximum value in trajectory or slave mode
0x80000040	Requested acceleration exceed maximum value in trajectory or slave mode
0x80000100	Minus end of run activated
0x80000200	Plus end of run activated
0x80000400	Minus end of run glitch
0x80000800	Plus end of run glitch
0x80001000	Encoder quadrature error
0x80002000	Encoder frequency and coherancy error
0x80010000	Hard interpolator encoder error
0x80020000	Hard interpolator encoder quadrature error
0x80100000	First driver in fault
0x80200000	Second driver in fault
0x81000000	Home search mechanical zero inconsistency
0x88000000	Fatal internal error

NOTE: The most significant bit is always set to 1. So, all positioner errors are negative.

2.17. Positioner hardware status list

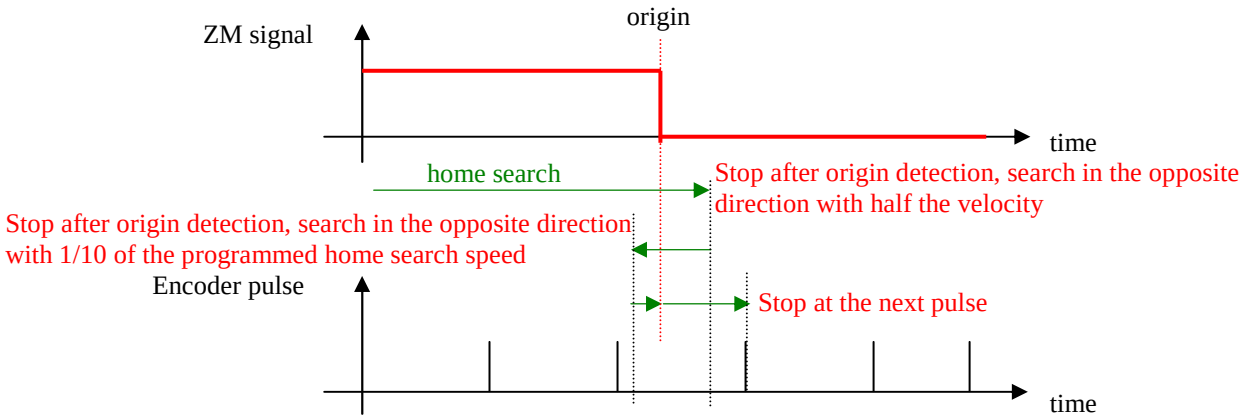
code	Error description
0x00000001	General inhibition detected
0x00000004	ZM high level
0x00000100	Minus end of run activated
0x00000200	Plus end of run activated
0x00000400	Minus end of run glitch
0x00000800	Plus end of run glitch
0x00001000	Encoder quadrature error
0x00002000	Encoder frequency or coherancy error
0x00010000	Hard interpolator encoder error
0x00020000	Hard interpolator encoder quadrature error
0x00100000	First driver in fault
0x00200000	Second driver in fault
0x00400000	First driver powered on
0x00800000	Second driver powered on

NOTE: Positioner errors are used to trigger consequences on the system, for instance disable, emergency break, etc. Positioner hardware status information is mainly provided for information purposes.

Explanation about positioner hardware status:

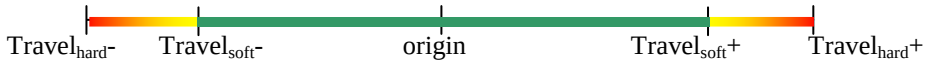
General inhibition detected: This refers to the general Inhibition connector at the rear panel or the Stop All button at the front panel of the XPS controller. The General Inhibiton connector is a safety features and can be used for a custom STOP ALL emergency switch. Inhibition (pin#2), must always be connected to GND during normal operation of the controller. In this case, inhibition is not detected. An open circuit is equivalent to pressing STOP ALL on the front panel. In that case, inhibition is detected.

ZM high level: This refers to the mechanical zero signal used with some stages. The ZM signal is high during one part of the travel and low during the other part of the travel. The detection of the ZM high/low transition in combination with an encoder index pulse signal allows a fast and repeatable origin search (MechanicalZeroAndIndexHomeSearch).



Minus end of run activated: Refers to the hardware minus end of run limit switch. During normal operation, this end of run switch gets ever actuated and any motion will be latest stopped by the detection of the minus software limit.

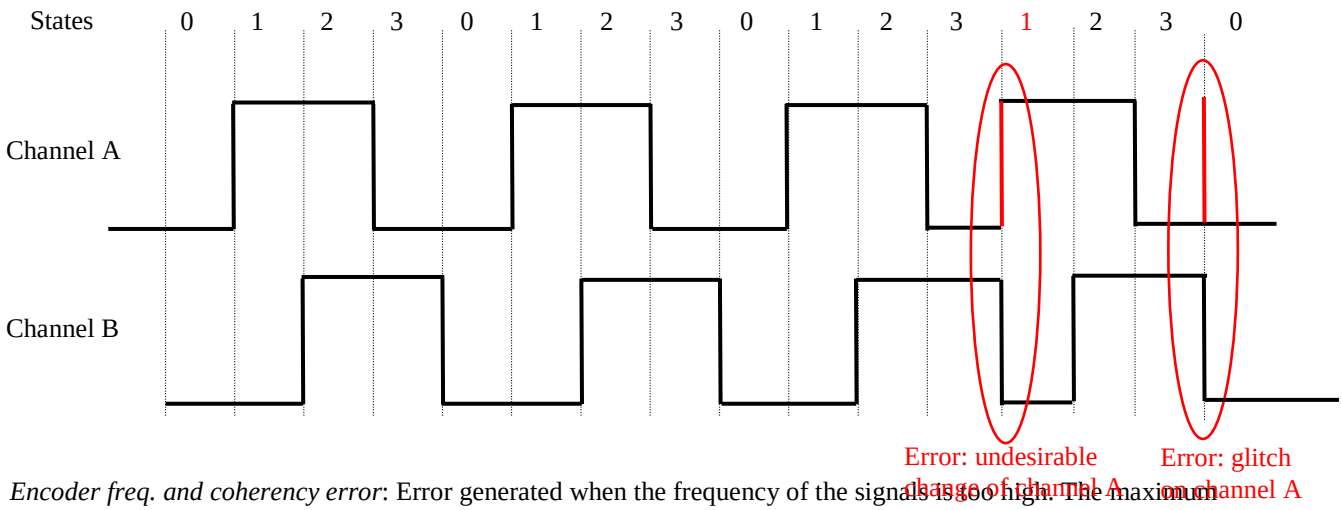
Plus end of run activated: Refers to the hardware positive end of run limit switch. During normal operation, this end of run switch gets ever actuated and any motion will be latest stopped by the detection of the positive software limit.



Minus end of run glitch: Undesirable, momentary instability of the hardware minus end of run signal, for instance generated by ripple or noise.

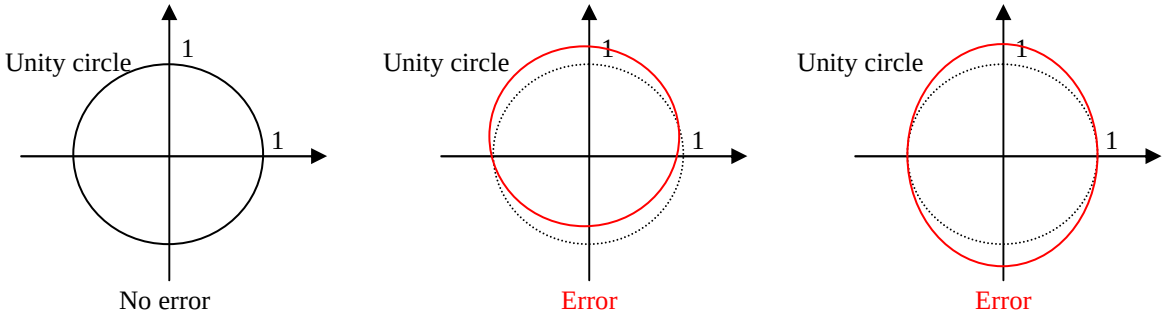
Plus end of run glitch: Undesirable, momentary instability of the hardware positive end of run signal, for instance generated by ripple or noise.

Encoder quadrature error: Error generated when the signals of both encoder channels simultaneously changes. In normal operation, only one quadrature signal changes state at once. This error can occur due to an undesirable level change or a glitch.



Encoder freq. and coherency error: Error generated when the frequency of the signals is too high. The maximum frequency of the encoder input is 25MHz.

Hard interpolator encoder error: Error generated when the difference of the sine/cosine encoder signals from a unity circle is too large (for instance when signals are phase shifted or amplitude modified).



Hard interpolator quad. encoder error: Error generated when the signals of both encoder channels of the hardware interpolated encoder output simultaneously changes. Same error as *Encoder quadrature error* except that the quadrature signals are those converted from the sine / cosine signals of the hard interpolator. The hardware interpolator is used only with AnalogInterpolated encoders to trigger the position compare output and to gather positions during external data gathering.

First driver in fault: problem with the first driver.

Second driver in fault: problem with the second driver in case two drivers are connected to one axis.

First driver powered on: First driver on motor ON, after initialization.

Second driver powered on: Second driver on motor ON, after initialization, in case two drivers are connected to one axis.

2.18. Positioner driver status list

Bit	code	DRV00x	DRV01	DRV02x	D6U	DRV03	DRVP1	DRVM
0	a			Short-circuit	Short-circuit	Short-circuit		
1	b			Broken fuse	Broken fuse	Broken fuse	Voltage out of range	
2	c			Thermistor fault	Thermistor fault		Over temperature	
3	d			Initialization error	Initialization error	Initialization error	Initialization error	
4	e			I^2T	I^2T	I^2T	Dynamic error	
5	f			Current limit	Current limit			
6	g					TG is opened	No stage connected	
7	h	Inhibition input	Inhibition input	Inhibition input	Inhibition input	Inhibition input	Inhibition input	Inhibition input
8	i	Driver in fault	Driver in fault	Driver in fault	Driver in fault	Driver in fault	Driver in fault	Driver in fault

2.19. Group status list

Code	Description
0	Not initialized state
1	Not initialized state due to an emergency brake : see positioner status
2	Not initialized state due to an emergency stop : see positioner status
3	Not initialized state due to a following error during homing
4	Not initialized state due to a following error
5	Not initialized state due to an homing timeout
6	Not initialized state due to a motion done timeout during homing
7	Not initialized state due to a KillAll command
8	Not initialized state due to an end of run after homing
9	Not initialized state due to an encoder calibration error
10	Ready state due to an AbortMove command
11	Ready state from homing
12	Ready state from motion
13	Ready State due to a MotionEnable command
14	Ready state from slave
15	Ready state from jogging
16	Ready state from analog tracking
17	Ready state from trajectory
18	Ready state from spinning
20	Disable state
21	Disabled state due to a following error on ready state
22	Disabled state due to a following error during motion
23	Disabled state due to a motion done timeout during moving
24	Disabled state due to a following error on slave state
25	Disabled state due to a following error on jogging state
26	Disabled state due to a following error during trajectory
27	Disabled state due to a motion done timeout during trajectory
28	Disabled state due to a following error during analog tracking
29	Disabled state due to a slave error during motion
30	Disabled state due to a slave error on slave state
31	Disabled state due to a slave error on jogging state
32	Disabled state due to a slave error during trajectory
33	Disabled state due to a slave error during analog tracking
34	Disabled state due to a slave error on ready state
35	Disabled state due to a following error on spinning state
36	Disabled state due to a slave error on spinning state
37	Disabled state due to a following error on auto-tuning
38	Disabled state due to a slave error on auto-tuning
40	Emergency braking
41	Motor initialization state
42	Not referenced state
43	Homing state
44	Moving state
45	Trajectory state
46	Slave state due to a SlaveEnable command
47	Jogging state due to a JogEnable command
48	Analog tracking state due to a TrackingEnable command
49	Analog interpolated encoder calibrating state
50	Not initialized state due to a mechanical zero inconsistency during homing
51	Spinning state due to a SpinParametersSet command
63	Not initialized state due to a motor initialization error
64	Referencing state

66	Not initialized state due to a perpendicularity error homing
67	Not initialized state due to a master/slave error during homing
68	Auto-tuning state
69	Scaling calibration state
70	Ready state from auto-tuning
71	Not initialized state from scaling calibration
72	Not initialized state due to a scaling calibration error
73	Excitation signal generation state
74	Disable state due to a following error on excitation signal generation state
75	Disable state due to a master/slave error on excitation signal generation state
76	Disable state due to an emergency stop on excitation signal generation state
77	Ready state from excitation signal generation

2.20. Error list

Error mnemonic	code	Error description
ERR_TCL_INTERPRETOR_ERROR	1	TCL interpretor error : wrong syntax
SUCCESS	0	Successful command
ERR_BUSY_SOCKET	-1	Busy socket : previous command not yet finished
ERR_TCP_TIMEOUT	-2	TCP timeout
ERR_STRING_TOO_LONG	-3	String command too long
ERR_UNKNOWN_COMMAND	-4	Unknown command
ERR_POSITIONER_ERROR	-5	Not allowed due to a positioner error
ERR_WRONG_FORMAT	-7	Wrong format in the command string
ERR_WRONG_OBJECT_TYPE	-8	Wrong object type for this command
ERR_WRONG_PARAMETERS_NUMBER	-9	Wrong number of parameters in the command
ERR_WRONG_TYPE	-10	Wrong parameter type in the command string
ERR_WRONG_TYPE_BIT_WORD	-11	Wrong parameters type in the command string : word or word * expected
ERR_WRONG_TYPE_BOOL	-12	Wrong parameter type in the command string : bool or bool * expected
ERR_WRONG_TYPE_CHAR	-13	Wrong parameter type in the command string : char * expected
ERR_WRONG_TYPE_DOUBLE	-14	Wrong parameter type in the command string : double or double * expected
ERR_WRONG_TYPE_INT	-15	Wrong parameter type in the command string : int, short, int * or short * expected
ERR_WRONG_TYPE_UNSIGNEDINT	-16	Wrong parameter type in the command string : unsigned int, unsigned short, unsigned int * or unsigned short * expected
ERR_PARAMETER_OUT_OF_RANGE	-17	Parameter out of range
ERR_POSITIONER_NAME	-18	Positioner Name doesn't exist
ERR_GROUP_NAME	-19	GroupName doesn't exist or unknown command
ERR_FATAL_INIT	-20	Fatal Error during initialization, read the error.log file for more details
ERR_IN_INITIALIZATION	-21	Controller in initialization
ERR_NOT_ALLOWED_ACTION	-22	Not allowed action
ERR_POSITION_COMPARE_NOT_SET	-23	Position compare not set
ERR_UNCOMPATIBLE	-24	Not available in this configuration
ERR_FOLLOWING_ERROR	-25	Following Error
ERR_EMERGENCY_SIGNAL	-26	Emergency signal
ERR_GROUP_ABORT_MOTION	-27	Move Aborted
ERR_GROUP_HOME_SEARCH_TIMEOUT	-28	Home search timeout
ERR_MNEMOTYPEGATHERING	-29	Mnemonique gathering type doesn't exist
ERR_GATHERING_NOT_STARTED	-30	Gathering not started
ERR_HOME_OUT_RANGE	-31	Home position is out of user travel limits
ERR_GATHERING_NOT_CONFIGURED	-32	Gathering not configurated
ERR_GROUP_MOTION_DONE_TIMEOUT	-33	Motion done timeout
ERR_TRAVEL_LIMITS	-35	Not allowed : home preset outside travel limits
ERR_UNKNOWN_TCL_FILE	-36	Unknown TCL file
ERR_TCL_SCRIPT_KILL	-37	TCL interpretor doesn't run
ERR_TCL_INTERPRETOR	-38	TCL script can't be killed
ERR_MNEMO_ACTION	-39	Mnemonique action doesn't exist
ERR_MNEMO_EVENT	-40	Mnemonique event doesn't exist
ERR_SLAVE_CONFIGURATION	-41	Slave-Master mode not configurated
ERR_JOG_OUT_OF_RANGE	-42	Jog value out of range
ERR_GATHERING_RUNNING	-43	Gathering running
ERR_SLAVE	-44	Slave error disabling master
ERR_END_OF_RUN	-45	End of run activated
ERR_NOT_ALLOWED_BACKLASH	-46	Not allowed action due to backlash
ERR_WRONG_TCL_TASKNAME	-47	Wrong TCL task name : each TCL task name must be different
ERR_BASE_VELOCITY	-48	BaseVelocity must be null
ERR_GROUP_HOME_SEARCH_ZM_ERROR	-49	Inconsistent mechanical zero during home search
ERR_MOTOR_INITIALIZATION_ERROR	-50	Motor initialization error : check InitializationAcceleration
ERR_SPIN_OUT_OF_RANGE	-51	Spin value out of range

ERR_WRITE_FILE	-60	Error during file writing or file doesn't exist
ERR_READ_FILE	-61	Error during file reading or file doesn't exist
ERR_TRAJ_ELEM_TYPE	-62	Wrong trajectory element type
ERR_TRAJ_ELEM_RADIUS	-63	Wrong XY trajectory element arc radius
ERR_TRAJ_ELEM_SWEEP	-64	Wrong XY trajectory element sweep angle
ERR_TRAJ_ELEM_LINE	-65	Trajectory line element discontinuity error or new element is too small
ERR_TRAJ_EMPTY	-66	Trajectory doesn't content any element or not loaded
ERR_TRAJ_VEL_LIMIT	-68	Velocity on trajectory is too high
ERR_TRAJ_ACC_LIMIT	-69	Acceleration on trajectory is too high
ERR_TRAJ_FINAL_VELOCITY	-70	Final velocity on trajectory is not zero
ERR_MSG_QUEUE	-71	Error write or read from message queue
ERR_TRAJ_INITIALIZATION	-72	Error during trajectory initialization
ERR_END_OF_FILE	-73	End of file
ERR_READ_FILE_PARAMETER_KEY	-74	Error file parameter key not found
ERR_TRAJ_TIME	-75	Time delta of trajectory element is negative or null
ERR_EVENTS_NOT_CONFIGURED	-80	Event not configured
ERR_ACTIONS_NOT_CONFIGURED	-81	Action not configured
ERR_EVENT_BUFFER_FULL	-82	Event buffer is full
ERR_EVENT_ID_UNDEFINED	-83	Event ID not defined
ERR_HOME_SEARCH_GANTRY_TOLERANCE_ERROR	-85	Secondary positioner index is too far from first positioner
ERR_OPTIONAL_EXTERNAL_MODULE_FILE	-94	Module file doesn't exist or module name incorrect (must be OptionalModule... .out)
ERR_OPTIONAL_EXTERNAL_MODULE_EXECUTE	-95	Error of executing an optional module
ERR_OPTIONAL_EXTERNAL_MODULE_KILL	-96	Error of stopping an optional module
ERR_OPTIONAL_EXTERNAL_MODULE_LOAD	-97	Error of loading an optional module
ERR_OPTIONAL_EXTERNAL_MODULE_UNLOAD	-98	Error of unloading an optional module
ERR_FATAL_EXTERNAL_MODULE_LOAD	-99	Fatal external module load : see error.log
ERR_INTERNAL_ERROR	-100	Internal error due to a memory allocation failure
ERR_RELAY_FEEDBACK_TEST_NO_OSCILLATION	-101	Relay Feedback Test failed : No oscillation
ERR_RELAY_FEEDBACK_TEST_SIGNAL_NOISY	-102	Relay Feedback Test failed : Signal too noisy
ERR_SIGNAL_POINTS_NOT_ENOUGH	-103	Relay Feedback Test failed: Signal data not enough for analyse
ERR_PID_TUNING_INITIALIZATION	-104	Error of tuning process initialization
ERR_SCALING_CALIBRATION	-105	Error of scaling calibration initialization
ERR_WRONG_USERNAME_OR_PASSWORD	-106	Wrong user name or password
ERR_NEED_ADMINISTRATOR_RIGHTS	-107	This function requires to be logged in with Administrator rights
ERR_SOCKET_CLOSED_BY_ADMIN	-108	The TCP/IP connection was closed by an administrator
ERR_NEED_TO_BE_HOMED_AT_LEAST_ONCE	-109	Group need to be homed at least once to use this function (distance mesured during home search)
ERR_NOT_ALLOWED_FOR_GANTRY	-110	Execution not allowed for Gantry configuration
ERR_GATHERING_BUFFER_FULL	-111	Gathering buffer is full
ERR_EXCITATION_SIGNAL_INITIALIZATION	-112	Error of excitation signal generation initialization
ERR_BOTH_ENDS_OF_RUNS_ACTIVATED	-113	Both end of runs are activated
ERR_HARDWARE_FUNCTION_NOT_SUPPORTED	-115	Function is not supported by current hardware
ERR_FUNCTION_ONLY_ALLOWED_IN_DISABLED_STATE	-117	Function is only allowed in DISABLED group state
ERR_NOT_ALLOWED_DRIVER_NOT_INITIALIZED	-118	Not allowed action driver not initialized

2.21. Controller status list

Controller status code	code	Controller status description
CONTROLLER_STATUS_OK	0x00000000	Controller status OK
CONTROLLER_STATUS_INITIALIZATION_FAILED	0x00000001	Controller status initialization failed
CONTROLLER_STATUS_NB_OPENED_SOCKETS_REACHED_MAXIMUM_ALLOWED	0x00000002	Number of currently opened sockets reached maximum allowed number
CONTROLLER_STATUS_CPU_OVERLOAD	0x00000004	Controller CPU is overloaded
CONTROLLER_STATUS_CORRECTOR_OVER_CALCULATED	0x00000008	Current measured corrector period exceeds 2 * IRSCorrectorPeriod
CONTROLLER_STATUS_PROFILER_OVER_CALCULATED	0x00000010	Profile generator calculating time exceeds ProfileGeneratorISRRatio * IRSCorrectorPeriod

Note :

- Within about 5 minutes after the controller startup, due to the hardware thermal stabilization, the CONTROLLER_STATUS_CORRECTOR_OVER_CALCULATED, CONTROLLER_STATUS_PROFILER_OVER_CALCULATED, CONTROLLER_STATUS_CPU_OVERLOAD or CONTROLLER_STATUS_NB_OPENED_SOCKETS_REACHED_MAXIMUM_ALLOWED status flags may be raised.
- These flags are automatically reset after a controller status reading using the *ControllerStatusGet()* command.
- Another way to avoid these flags raised duning 5 first minutes after boot is to set the following parameter in system.ref to 300 (seconds):
    - DelayBeforeStartup = 300 ; Controller is booted completely after 300 seconds

2.22. Function list classed in categories

General function

1. CloseAllOtherSockets
2. ControllerMotionKernelTimeLoadGet
3. ControllerStatusGet
4. ControllerStatusStringGet
5. DoubleGlobalArrayGet
6. DoubleGlobalArraySet
7. ElapsedTimeGet
8. ErrorStringGet
9. EventAdd
10. EventGet
11. EventRemove
12. EventWait
13. EventExtendedConfigurationTriggerSet
14. EventExtendedConfigurationTriggerGet
15. EventExtendedConfigurationActionSet
16. EventExtendedConfigurationActionGet
17. EventExtendedStart
18. EventExtendedAllGet
19. EventExtendedGet
20. EventExtendedRemove
21. EventExtendedWait
22. FirmwareVersionGet
23. GatheringConfigurationGet
24. GatheringConfigurationSet
25. GatheringCurrentNumberGet
26. GatheringDataAcquire
27. GatheringDataGet
28. GatheringDataMultipleLinesGet
29. GatheringExternalConfigurationSet
30. GatheringExternalConfigurationGet
31. GatheringExternalCurrentNumberGet
32. GatheringExternalDataGet
33. GatheringExternalStopAndSave
34. GatheringReset
35. GatheringRun
36. GatheringRunAppend
37. GatheringStop
38. GatheringStopAndSave
39. GlobalArrayGet
40. GlobalArraySet
41. GPIOAnalogGet
42. GPIOAnalogSet
43. GPIOAnalogGainGet
44. GPIOAnalogGainSet
45. GPIODigitalGet
46. GPIODigitalSet
47. HardwareDateAndTimeGet
48. HardwareDateAndTimeSet
49. KillAll
50. Login
51. OtionalModuleExecute
52. OtionalModuleKill
53. Reboot
54. TCLScriptExecute

55. TCLScriptExecuteAndWait
56. TCLScriptExecuteWithPriority
57. TCLScriptKill
58. TimerGet
59. TimerSet

Positioner functions

1. PositionerAccelerationAutoScaling
2. PositionerAnalogTrackingPositionParametersGet
3. PositionerAnalogTrackingPositionParametersSet
4. PositionerAnalogTrackingVelocityParametersGet
5. PositionerAnalogTrackingVelocityParametersSet
6. PositionerBacklashGet
7. PositionerBacklashSet
8. PositionerBacklashEnable
9. PositionerBacklashDisable
10. PositionerCorrectorAutoTuning
11. PositionerCorrectorNotchFiltersSet
12. PositionerCorrectorNotchFiltersGet
13. PositionerCorrectorPIDFFAccelerationSet
14. PositionerCorrectorPIDFFAccelerationGet
15. PositionerCorrectorPIDFFVelocitySet
16. PositionerCorrectorPIDFFVelocityGet
17. PositionerCorrectorPIDDualFFVoltageSet
18. PositionerCorrectorPIDDualFFVoltageGet
19. PositionerCorrectorPIPositionSet
20. PositionerCorrectorPIPositionGet
21. PositionerCorrectorTypeGet
22. PositionerCurrentVelocityAccelerationFiltersSet
23. PositionerCurrentVelocityAccelerationFiltersGet
24. PositionerDriverFiltersGet
25. PositionerDriverFiltersSet
26. PositionerDriverPositionOffsetsGet
27. PositionerDriverFiltersGet
28. PositionerDriverFiltersSet
29. PositionerDriverPositionOffsetsGet
30. PositionerDriverStatusGet
31. PositionerDriverStatusStringGet
32. PositionerEncoderAmplitudeValuesGet
33. PositionerEncoderCalibrationParametersGet
34. PositionerErrorGet
35. PositionerErrorRead
36. PositionerErrorStringGet
37. PositionerExcitationSignalGet
38. PositionerExcitationSignalSet
39. PositionerHardwareStatusGet
40. PositionerHardwareStatusStringGet
41. PositionerHardInterpolatorFactorGet
42. PositionerHardInterpolatorFactorSet
43. PositionerMaximumVelocityAndAccelerationGet
44. PositionerMotionDoneGet
45. PositionerMotionDoneSet
46. PositionerPositionCompareDisable
47. PositionerPositionCompareEnable
48. PositionerPositionCompareGet
49. PositionerPositionCompareSet
50. PositionerPositionComparePulseParametersGet

51. PositionerPositionComparePulseParametersSet
52. PositionerPositionCompareAquadBAlwaysEnable
53. PositionerPositionCompareAquadBWindowedGet
54. PositionerPositionCompareAquadBWindowedSet
55. PositionerRawEncoderPositionGet
56. PositionersEncoderIndexDifferenceGet
57. PositionerSGammaExactVelocityAjustedDisplacementGet
58. PositionerSGammaParametersGet
59. PositionerSGammaParametersSet
60. PositionerSGammaPreviousMotionTimesGet
61. PositionerStageParameterGet
62. PositionerStageParameterSet
63. PositionerTimeFlasherDisable
64. PositionerTimeFlasherEnable
65. PositionerTimeFlasherGet
66. PositionerTimeFlasherSet
67. PositionerUserTravelLimitsGet
68. PositionerUserTravelLimitsSet

Group functions

	SingleAxis GROUP	Spindle GROUP	XY GROUP	XYZ GROUP	MultipleAxes GROUP	POSITIONER
1. GroupAccelerationSetpointGet	×	×	×	×	×	×
2. GroupAnalogTrackingModeEnable	×	×	×	×	×	
3. GroupAnalogTrackingModeDisable	×	×	×	×	×	
4. GroupCorrectorOutputGet	×	×	×	×	×	×
5. GroupCurrentFollowingErrorGet	×	×	×	×	×	×
6. GroupHomeSearch	×	×	×	×	×	
7. GroupHomeSearchAndRelativeMove	×	×	×	×	×	
8. GroupInitialize	×	×	×	×	×	
9. GroupInitializeWithEncoderCalibration	×	×	×	×	×	
10. GroupJogParametersSet	×		×	×	×	×
11. GroupJogParametersGet	×		×	×	×	×
12. GroupJogCurrentGet	×		×	×	×	×
13. GroupJogModeEnable	×		×	×	×	
14. GroupJogModeDisable	×		×	×	×	
15. GroupKill	×	×	×	×	×	
16. GroupMoveAbort	×	×	×	×	×	×
17. GroupMoveAbsolute	×	×	×	×	×	×
18. GroupMoveRelative	×	×	×	×	×	×
19. GroupMotionDisable	×	×	×	×	×	
20. GroupMotionEnable	×	×	×	×	×	
21. GroupPositionCorrectedProfilerGet			×			
22. GroupPositionCurrentGet	×	×	×	×	×	×
23. GroupPositionSetpointGet	×	×	×	×	×	×
24. GroupPositionTargetGet	×	×	×	×	×	×
25. GroupPositionPCORawEncoderGet			×			
26. GroupReferencingActionExecute						×
27. GroupReferencingStart	×	×	×	×	×	

28. GroupReferencingStop	×	×	×	×	×	
29. GroupStatusGet	×	×	×	×	×	
30. GroupStatusStringGet	×	×	×	×	×	
31. GroupVelocityCurrentGet	×	×	×	×	×	×

SingleAxes group functions

1. SingleAxisSlaveModeDisable
2. SingleAxisSlaveModeEnable
3. SingleAxisSlaveParametersGet
4. SingleAxisSlaveParametersSet

Spindle group functions

1. GroupSpinCurrentGet
2. GroupSpinModeStop
3. GroupSpinParametersGet
4. GroupSpinParametersSet
5. SpindleSlaveModeDisable
6. SpindleSlaveModeEnable
7. SpindleSlaveParametersGet
8. SpindleSlaveParametersSet

XY group functions

1. GroupPositionCorrectedProfilerGet
2. XYLineArcExecution
3. XYLineArcParametersGet
4. XYLineArcPulseOutputGet
5. XYLineArcPulseOutputSet
6. XYLineArcVerification
7. XYLineArcVerificationResultGet

XYZ group functions

1. XYZSplineExecution
2. XYZSplineParametersGet
3. XYZSplineVerification
4. XYZSplineVerificationResultGet

MultipleAxes group functions

1. MultipleAxesPVTExecution
2. MultipleAxesPVTParametersGet
3. MultipleAxesPVTPulseOutputGet
4. MultipleAxesPVTPulseOutputSet
5. MultipleAxesPVTVerification
6. MultipleAxesPVTVerificationResultGet

3. Process examples

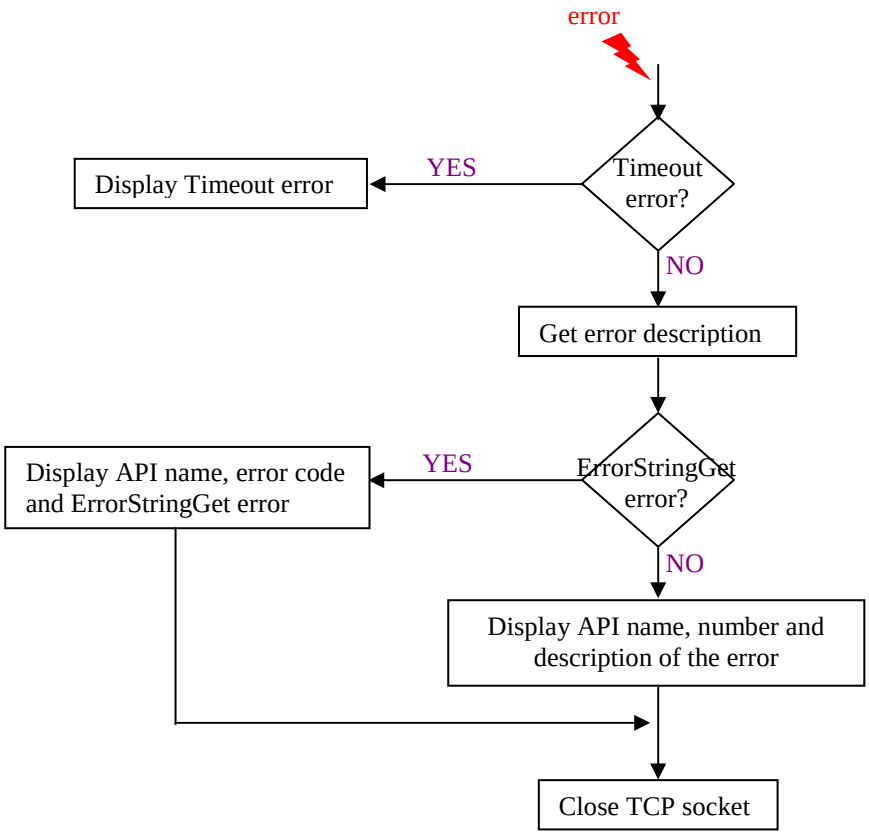
This part provides examples of programming sequences. Graph7 diagrams are presented to show you the order of use of the different Functions. To see the programming code examples, please refer to:

- The TCL Manual for TCL scripts (part 7. Examples of TCL programs with XPS).
- The Software Drivers Manual for C++ sequences (part 1.3 Example of C++ programs using the XPS DLL).

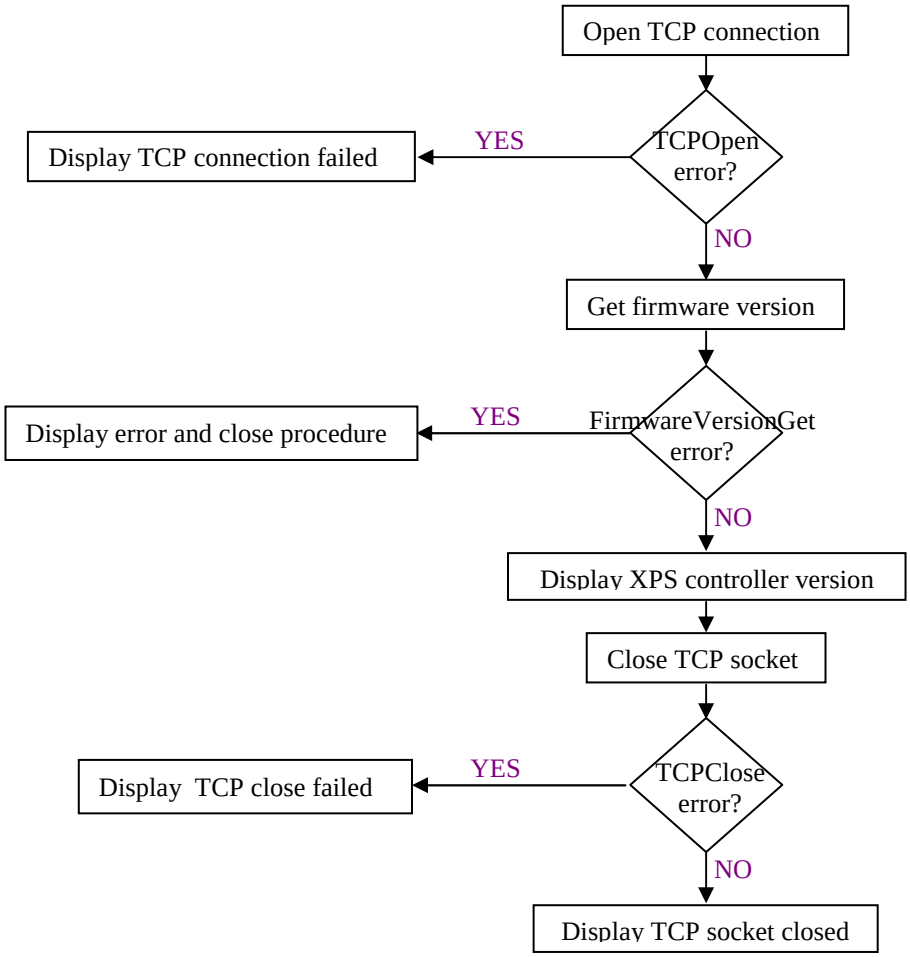
3.1. Management of the errors

In good practices of programming, when an error occurs, it is desirable to analyze and treat it. The following display error and close procedure could be useful to detect and display the errors during the execution of a program. This sequence could be added to each program and called each time users need to test certain parts of a program.

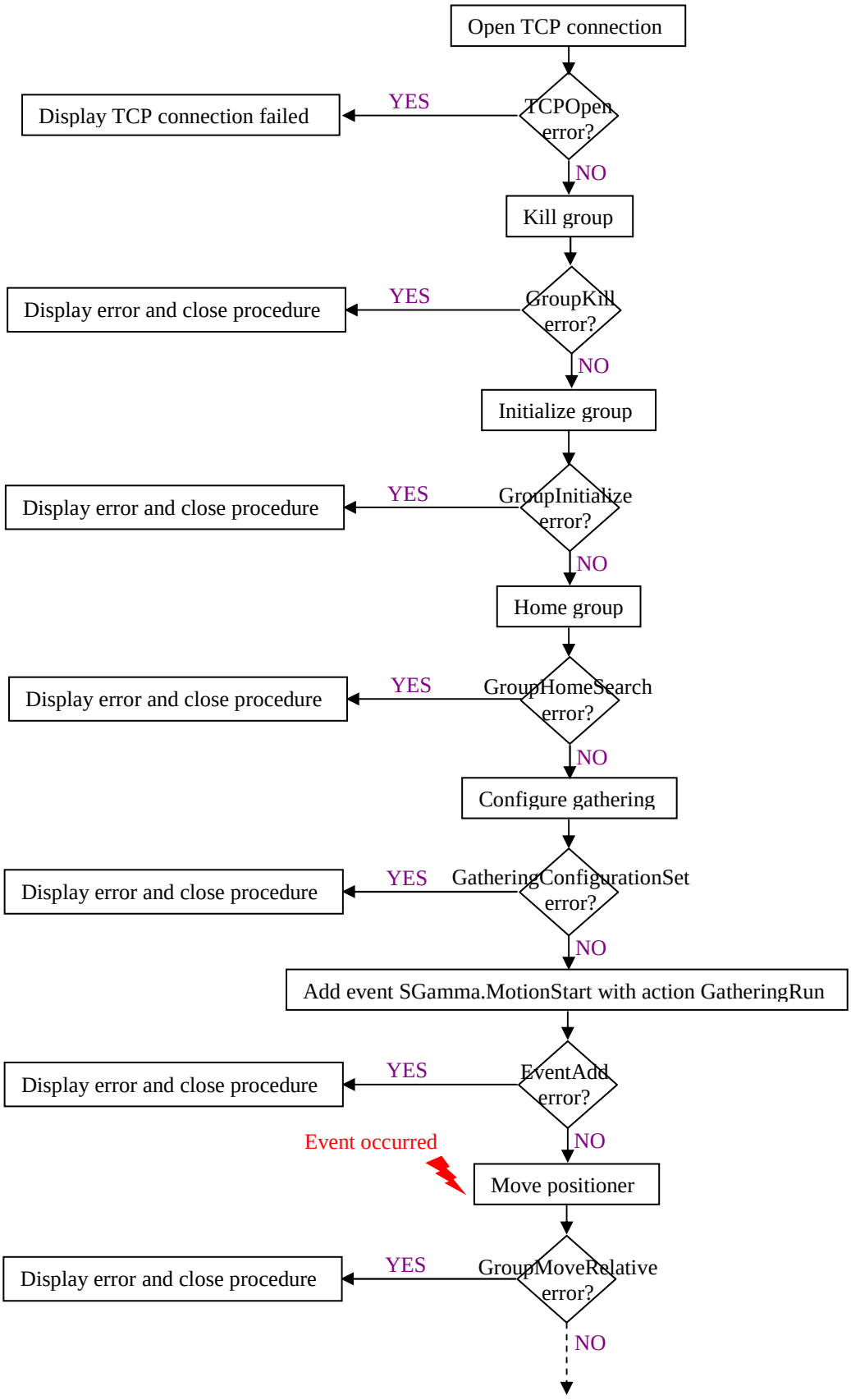
Display error and close procedure:

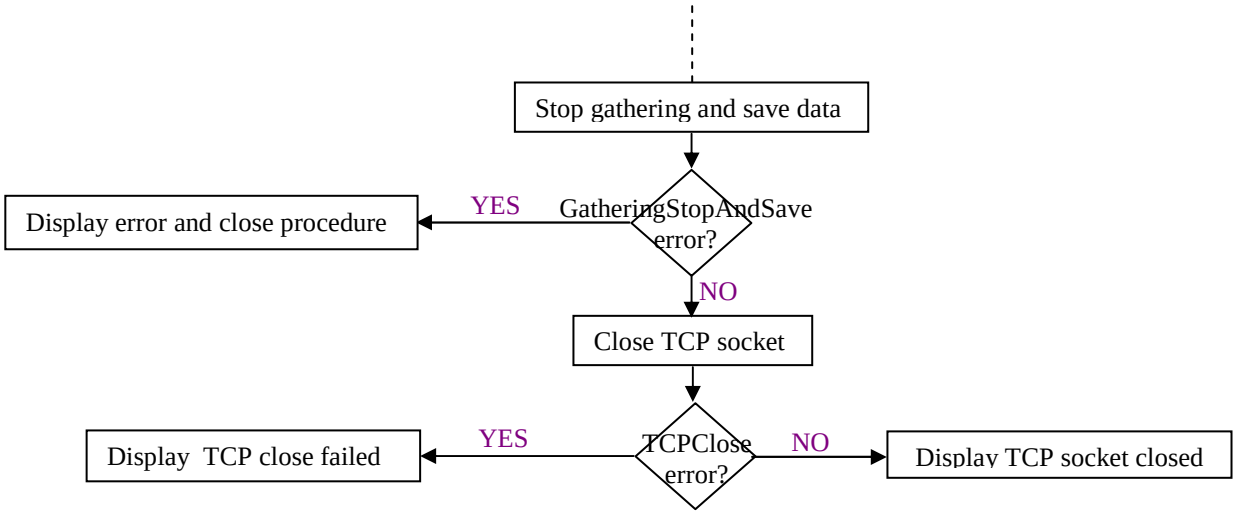


3.2. Firmware version

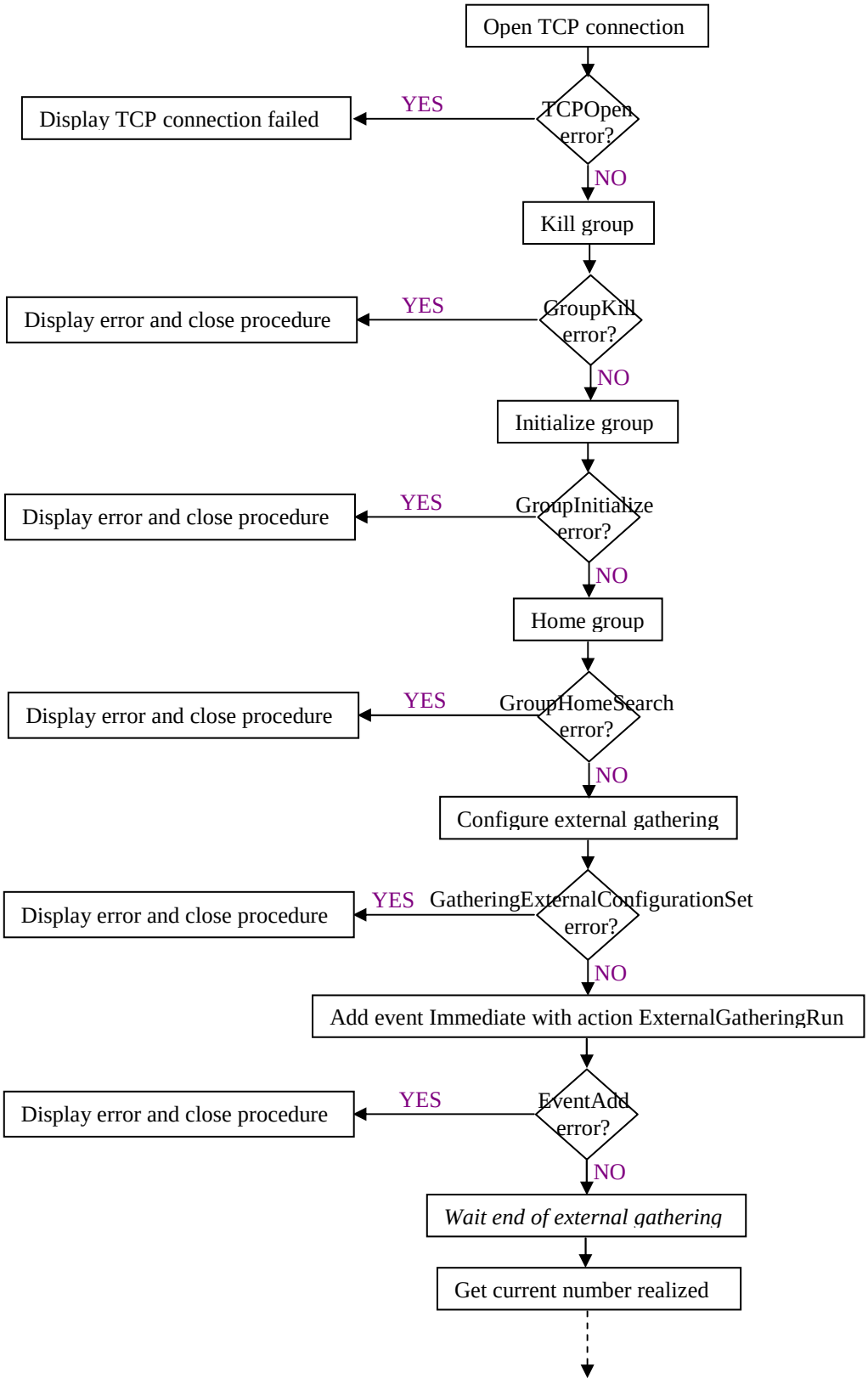


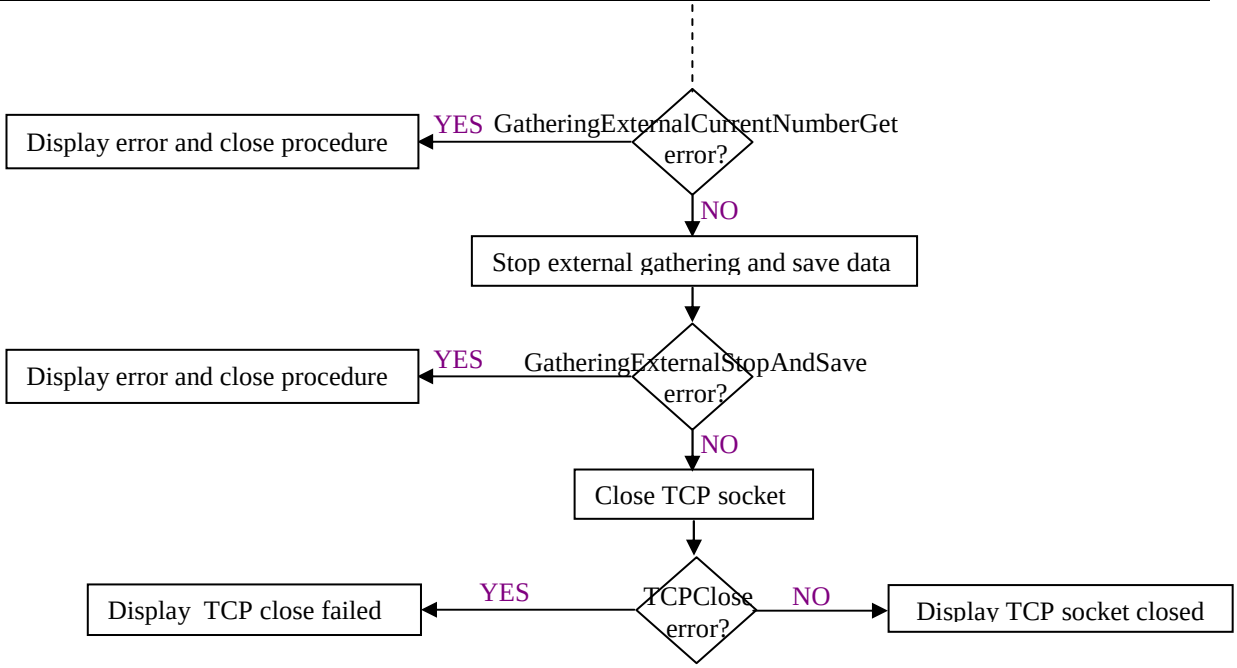
3.3. Gathering with motion



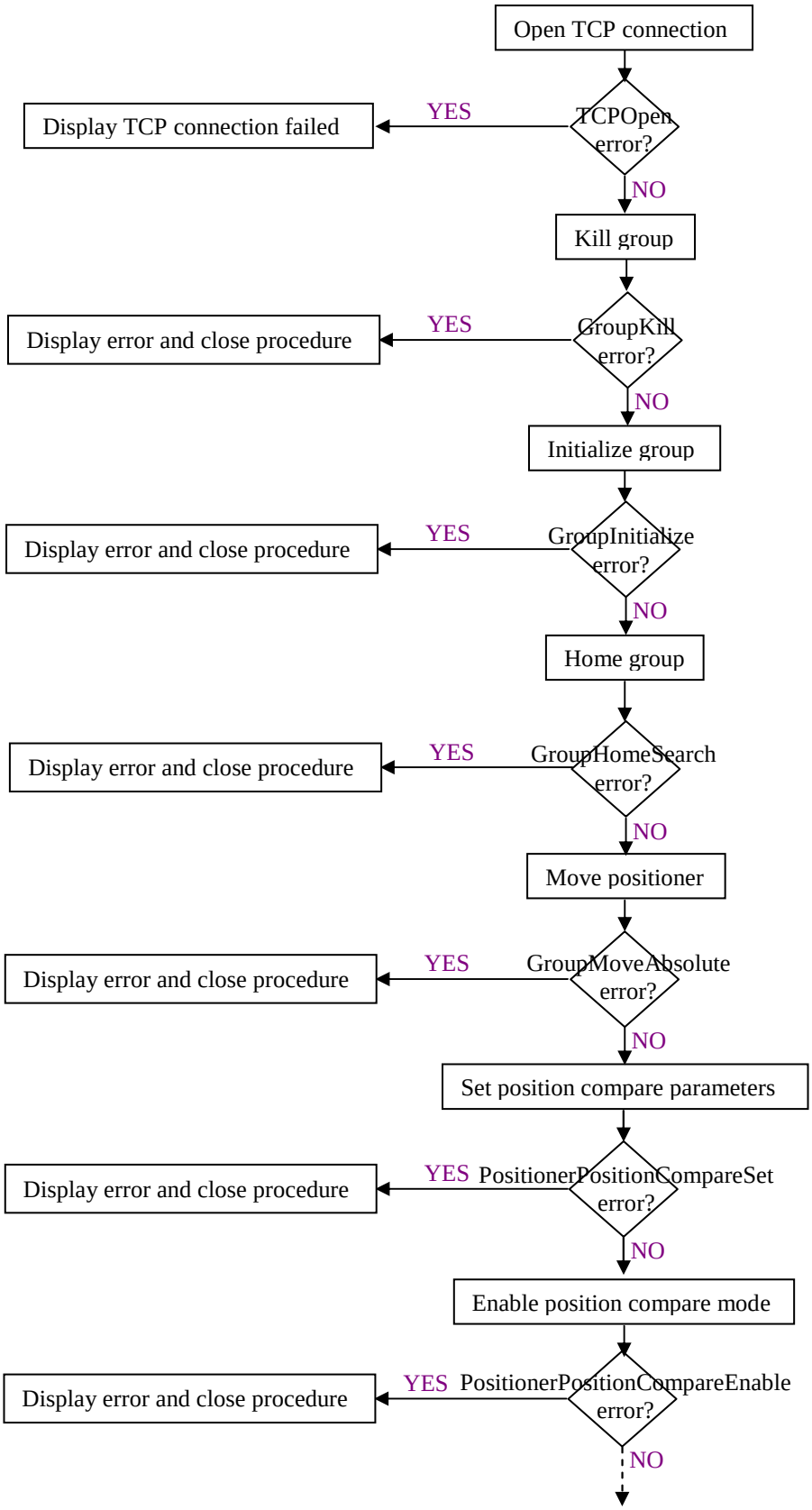

Stop gathering and save data
GatheringStopAndSave error?
YES
Display error and close procedure
NO
Close TCP socket
TCPClose error?
YES
Display TCP close failed
NO
Display TCP socket closed

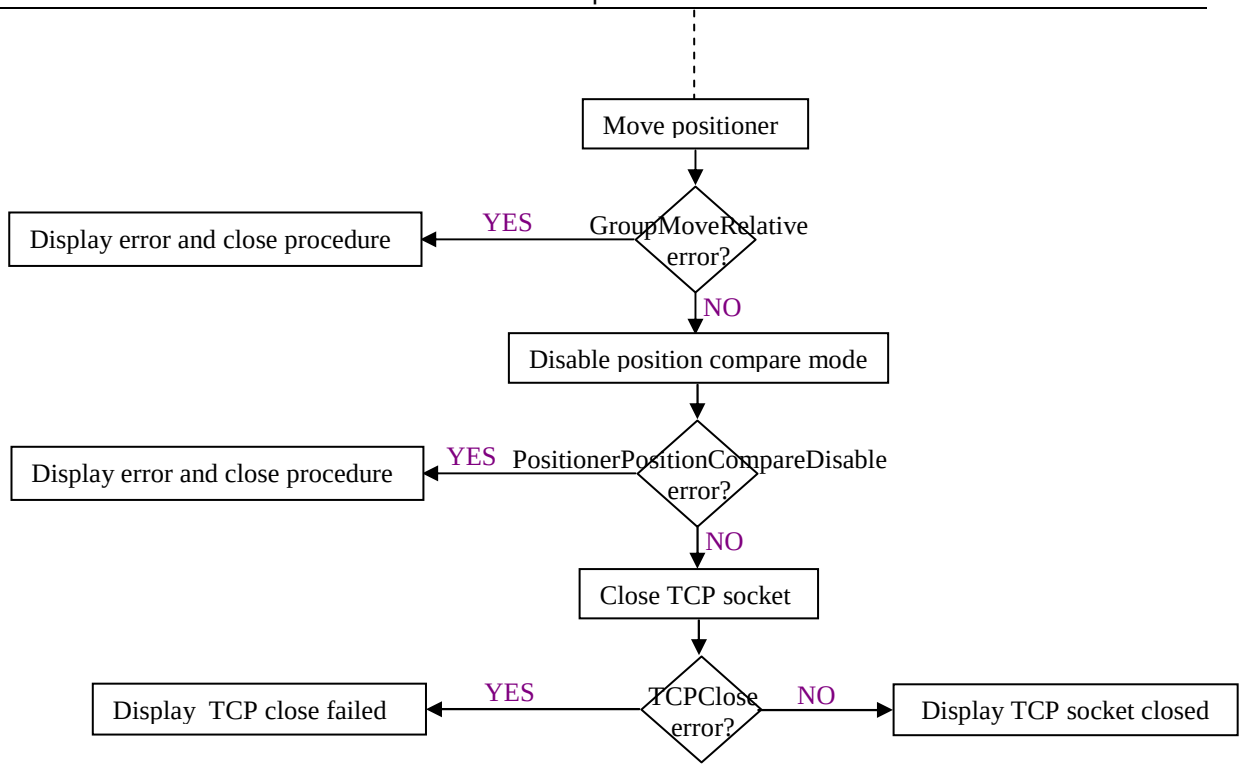
3.4. External gathering



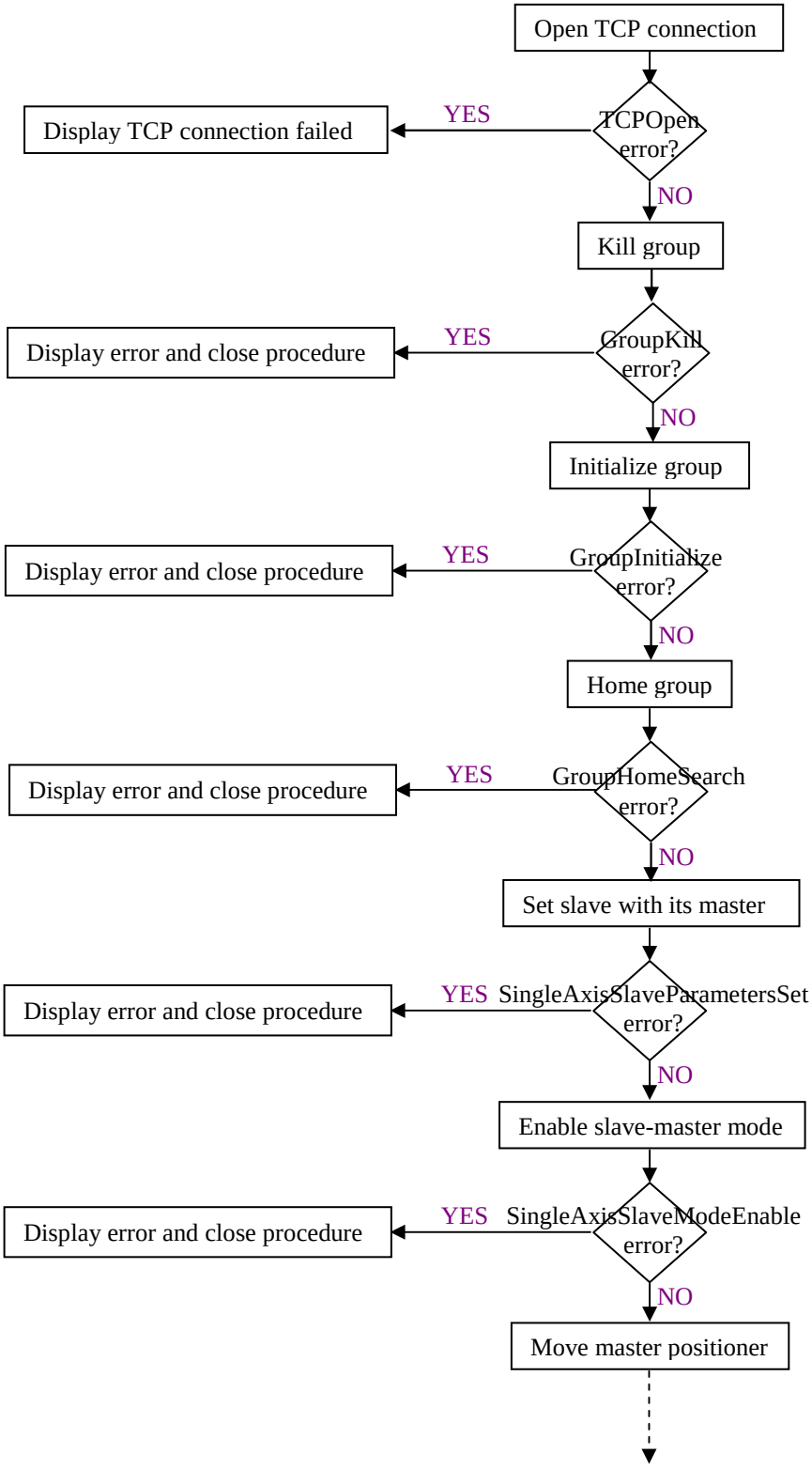

GatheringExternalCurrentNumberGet
error?
YES
Display error and close procedure
NO
Stop external gathering and save data
GatheringExternalStopAndSave
error?
YES
Display error and close procedure
NO
Close TCP socket
TCPClose
error?
YES
Display TCP close failed
NO
Display TCP socket closed

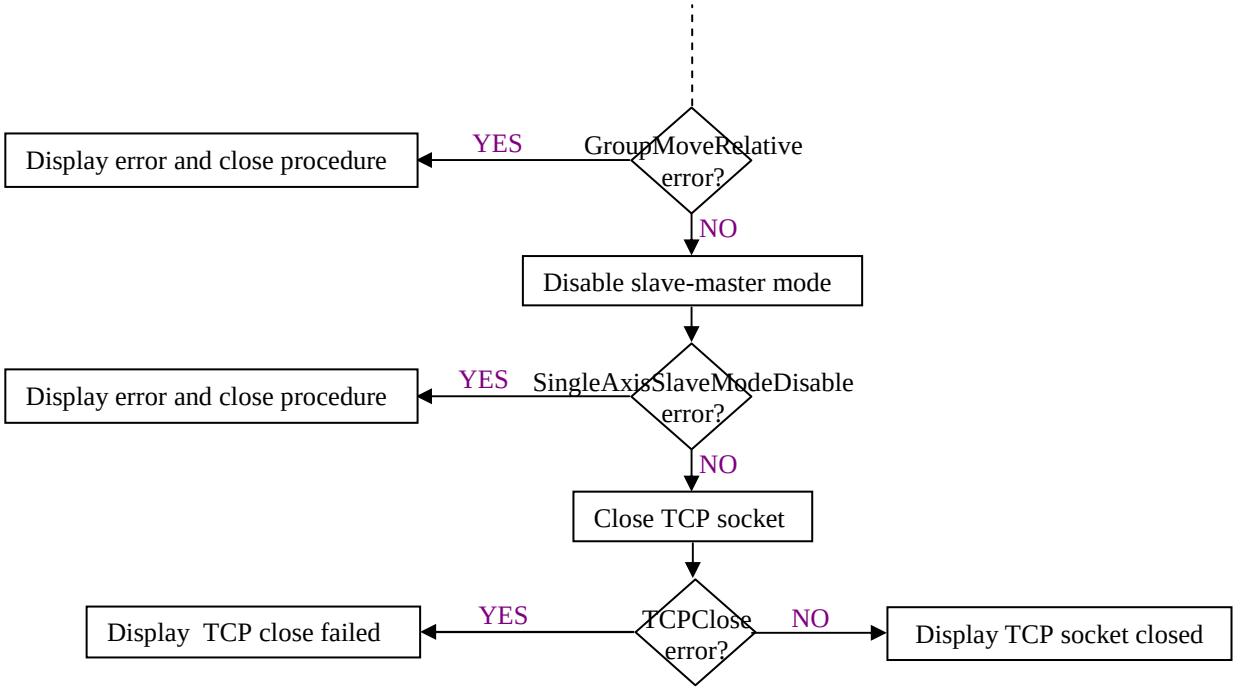
3.5. Output compare



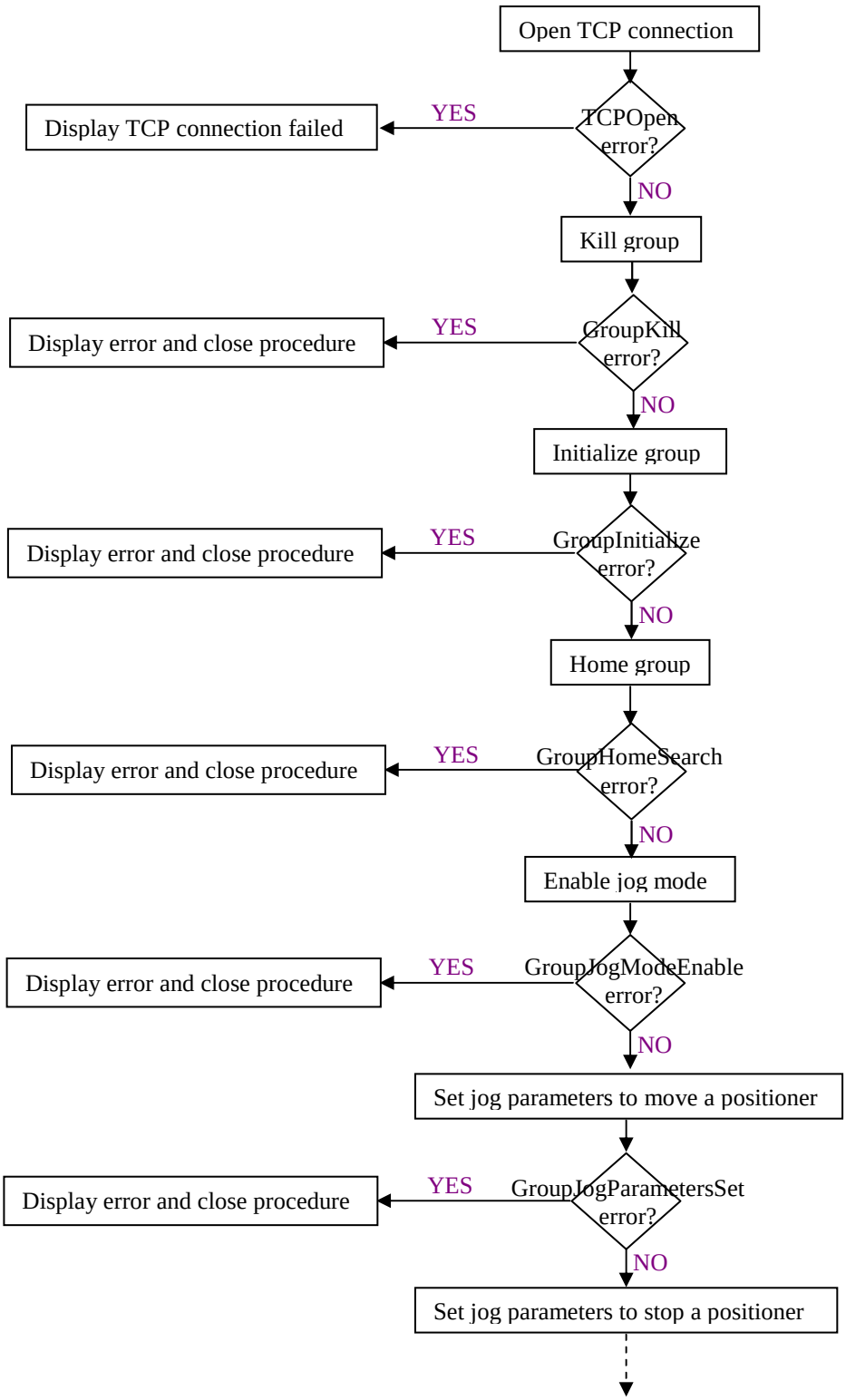

Move positioner
GroupMoveRelative error?
YES
Display error and close procedure
NO
Disable position compare mode
PositionerPositionCompareDisable error?
YES
Display error and close procedure
NO
Close TCP socket
TCPClose error?
YES
Display TCP close failed
NO
Display TCP socket closed

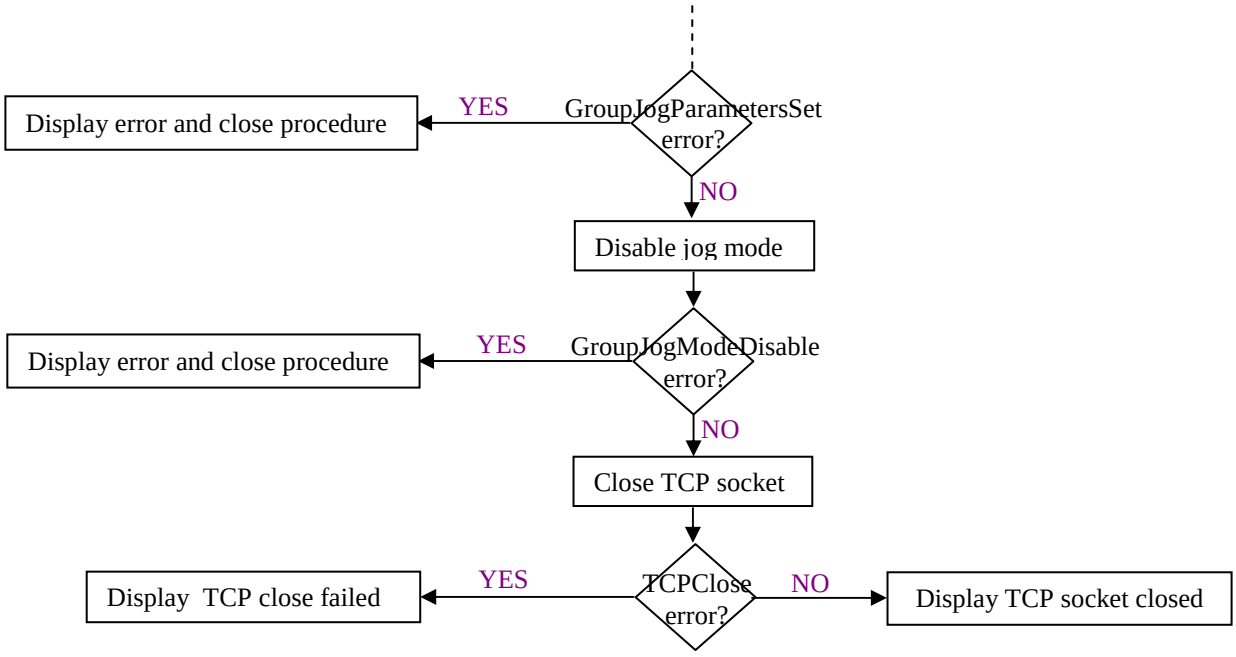
3.6. Slave-master mode



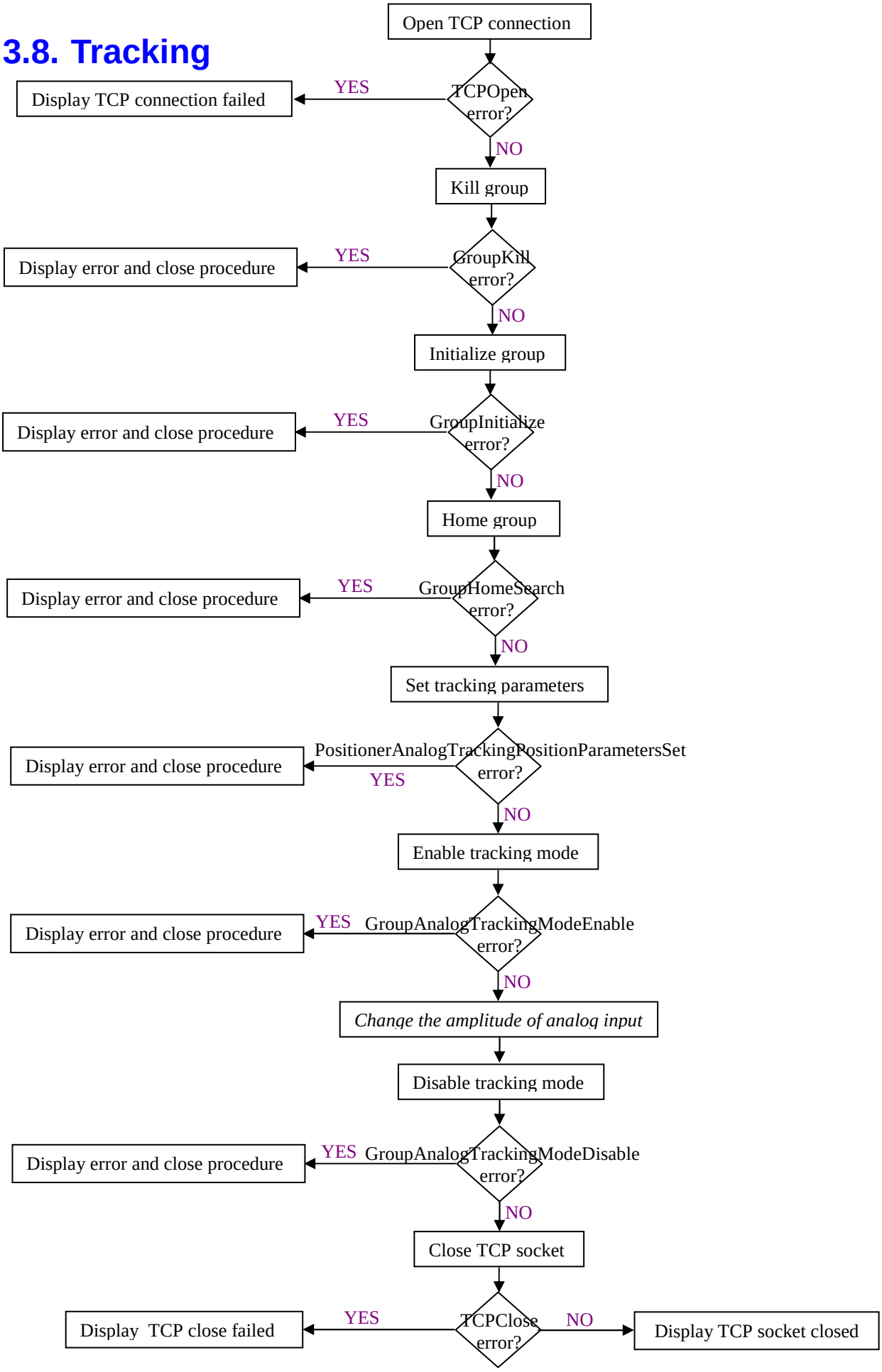

GroupMoveRelative error?
YES
Display error and close procedure
NO
Disable slave-master mode
SingleAxisSlaveModeDisable error?
YES
Display error and close procedure
NO
Close TCP socket
TCPClose error?
YES
Display TCP close failed
NO
Display TCP socket closed

3.7. Jogging

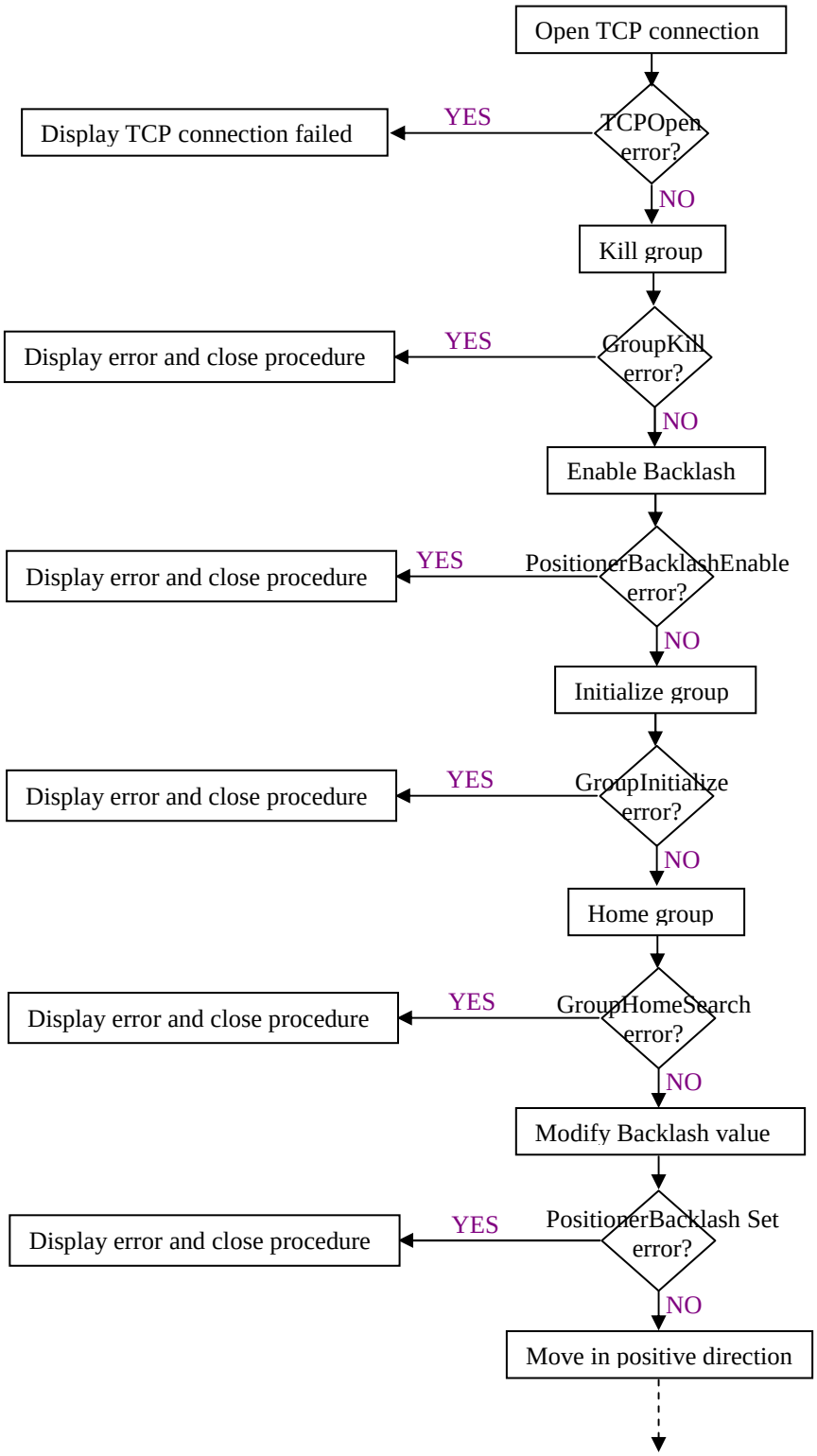

Open TCP connection
TCPOpen error?
YES
Display TCP connection failed
NO
Kill group
GroupKill error?
YES
Display error and close procedure
NO
Initialize group
GroupInitialize error?
YES
Display error and close procedure
NO
Home group
GroupHomeSearch error?
YES
Display error and close procedure
NO
Enable jog mode
GroupJogModeEnable error?
YES
Display error and close procedure
NO
Set jog parameters to move a positioner
GroupJogParametersSet error?
YES
Display error and close procedure
NO
Set jog parameters to stop a positioner

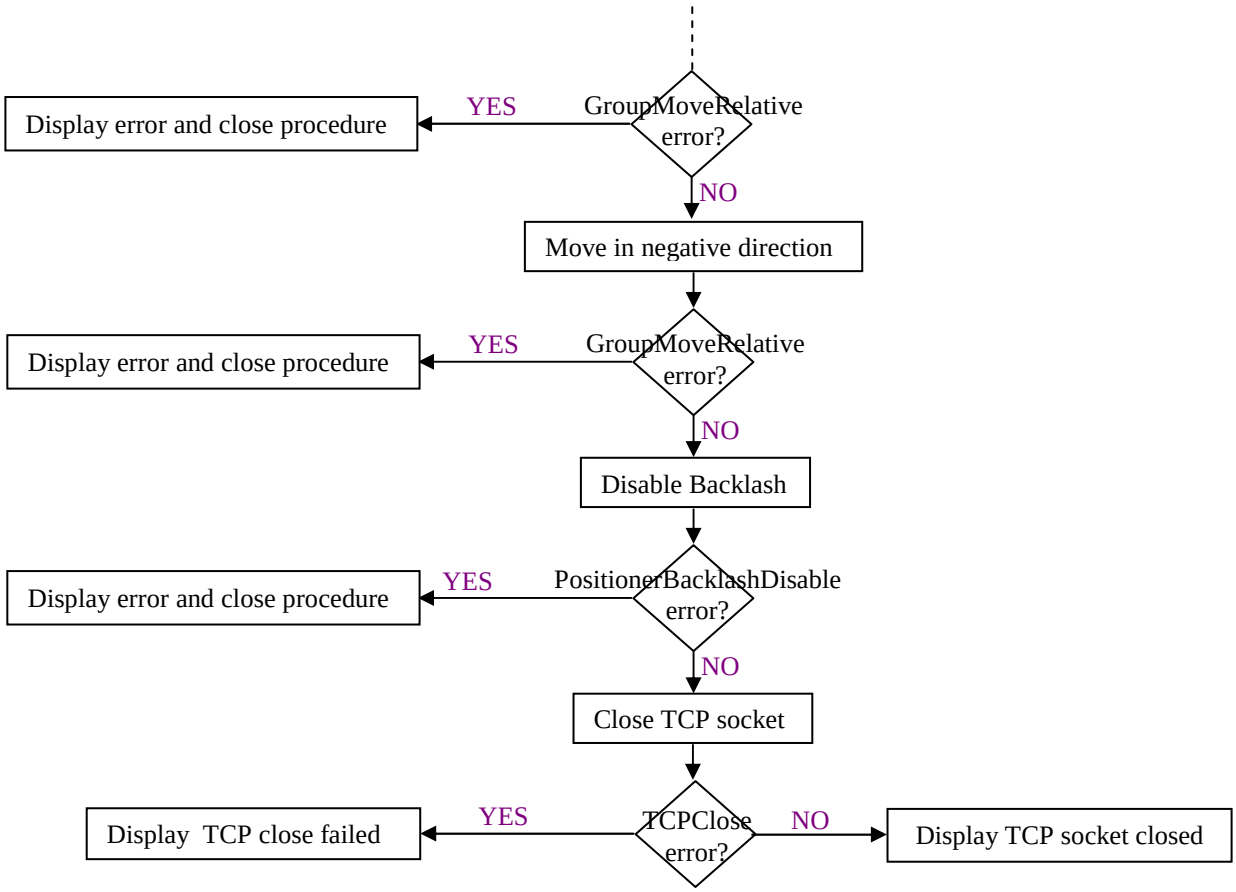

GroupJogParametersSet error?
YES
Display error and close procedure
NO
Disable jog mode
GroupJogModeDisable error?
YES
Display error and close procedure
NO
Close TCP socket
TCPClose error?
YES
Display TCP close failed
NO
Display TCP socket closed

3.8. Tracking

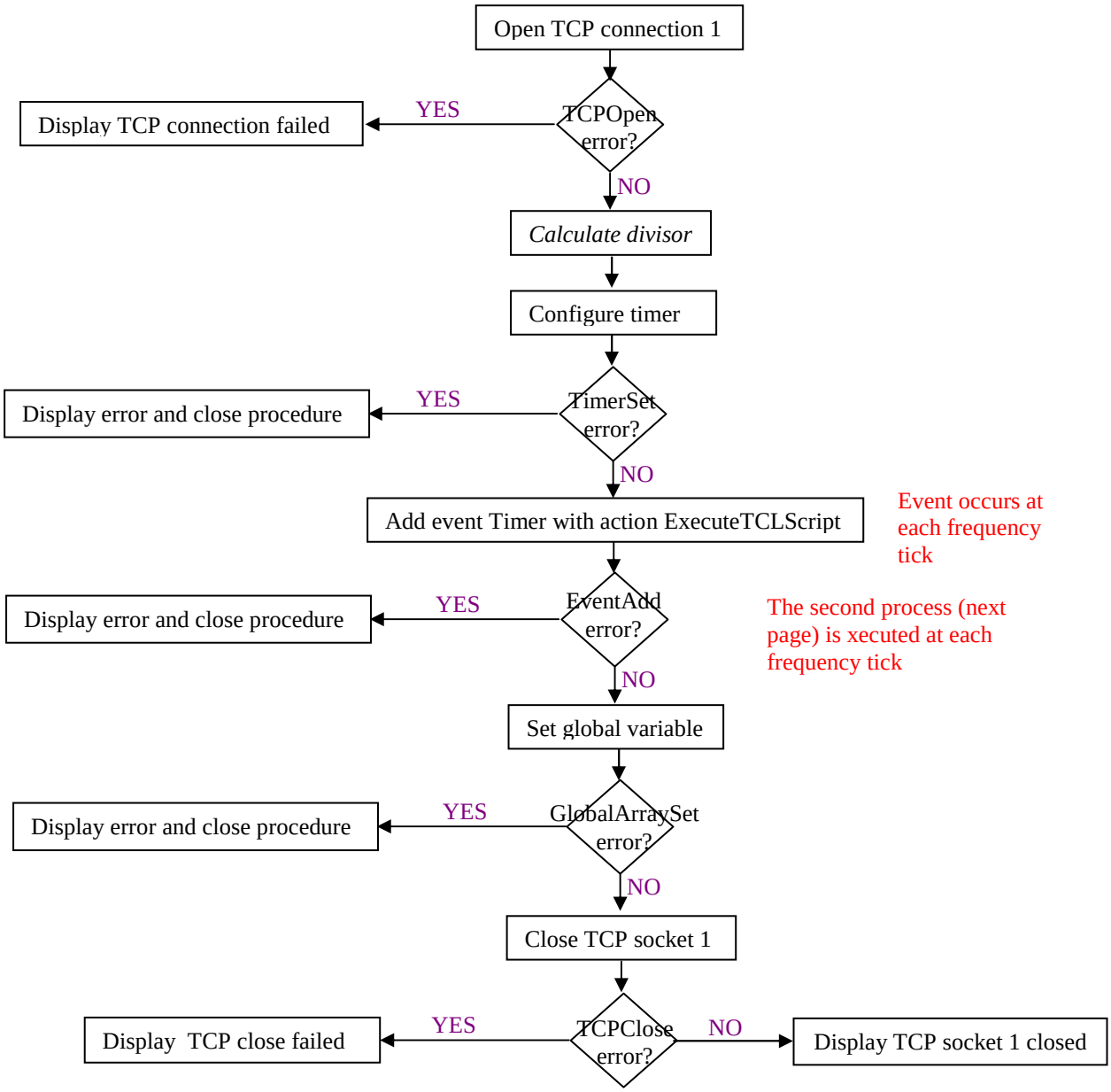

Open TCP connection
TCPOpen error?
YES
Display TCP connection failed
NO
Kill group
GroupKill error?
YES
Display error and close procedure
NO
Initialize group
GroupInitialize error?
YES
Display error and close procedure
NO
Home group
GroupHomeSearch error?
YES
Display error and close procedure
NO
Set tracking parameters
PositionerAnalogTrackingPositionParametersSet error?
YES
Display error and close procedure
NO
Enable tracking mode
GroupAnalogTrackingModeEnable error?
YES
Display error and close procedure
NO
Change the amplitude of analog input
Disable tracking mode
GroupAnalogTrackingModeDisable error?
YES
Display error and close procedure
NO
Close TCP socket
TCPClose error?
YES
Display TCP close failed
NO
Display TCP socket closed

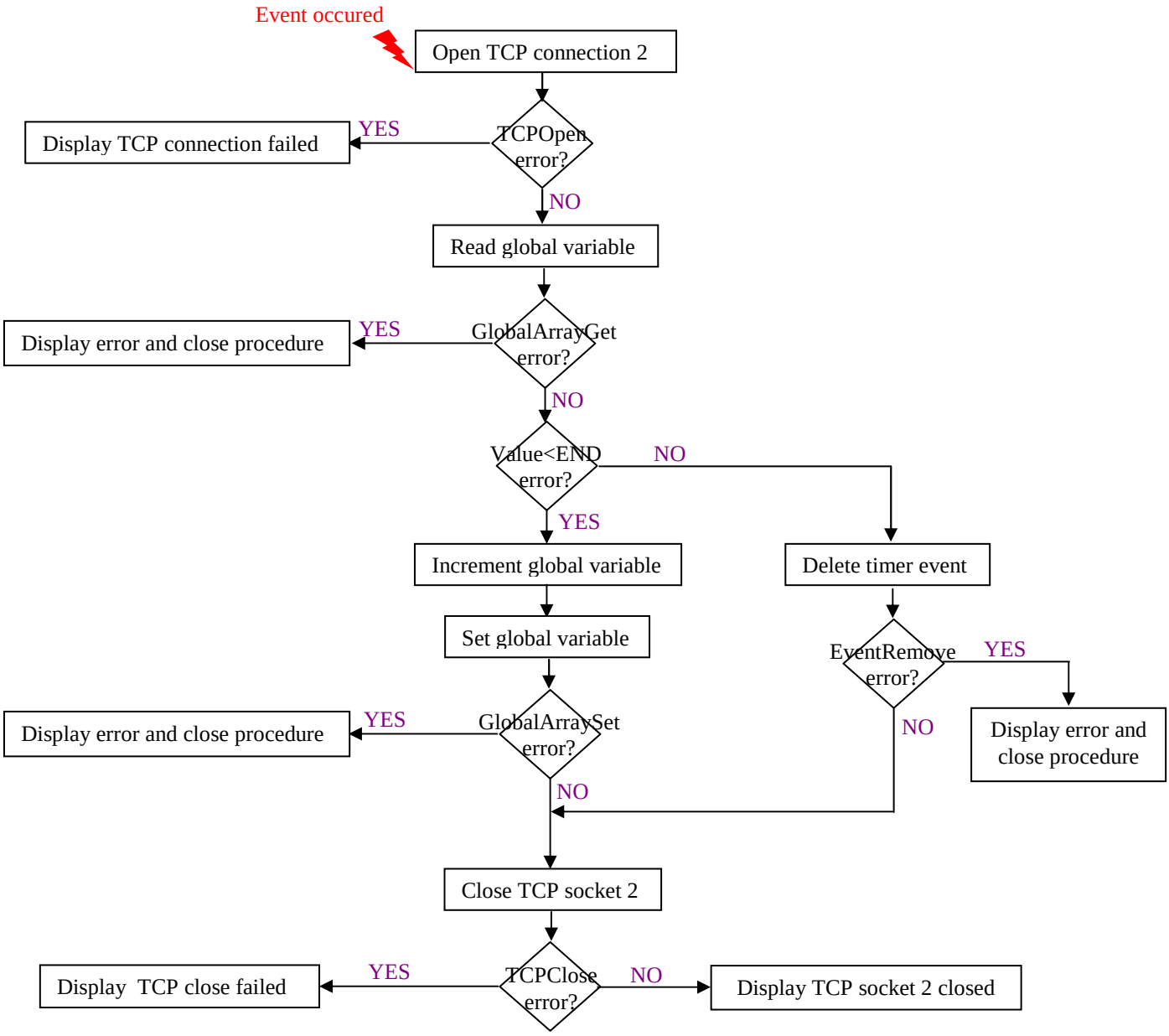
3.9. Backlash


Open TCP connection
TCPOpen error?
YES
Display TCP connection failed
NO
Kill group
GroupKill error?
YES
Display error and close procedure
NO
Enable Backlash
PositionerBacklashEnable error?
YES
Display error and close procedure
NO
Initialize group
GroupInitialize error?
YES
Display error and close procedure
NO
Home group
GroupHomeSearch error?
YES
Display error and close procedure
NO
Modify Backlash value
PositionerBacklash Set error?
YES
Display error and close procedure
NO
Move in positive direction

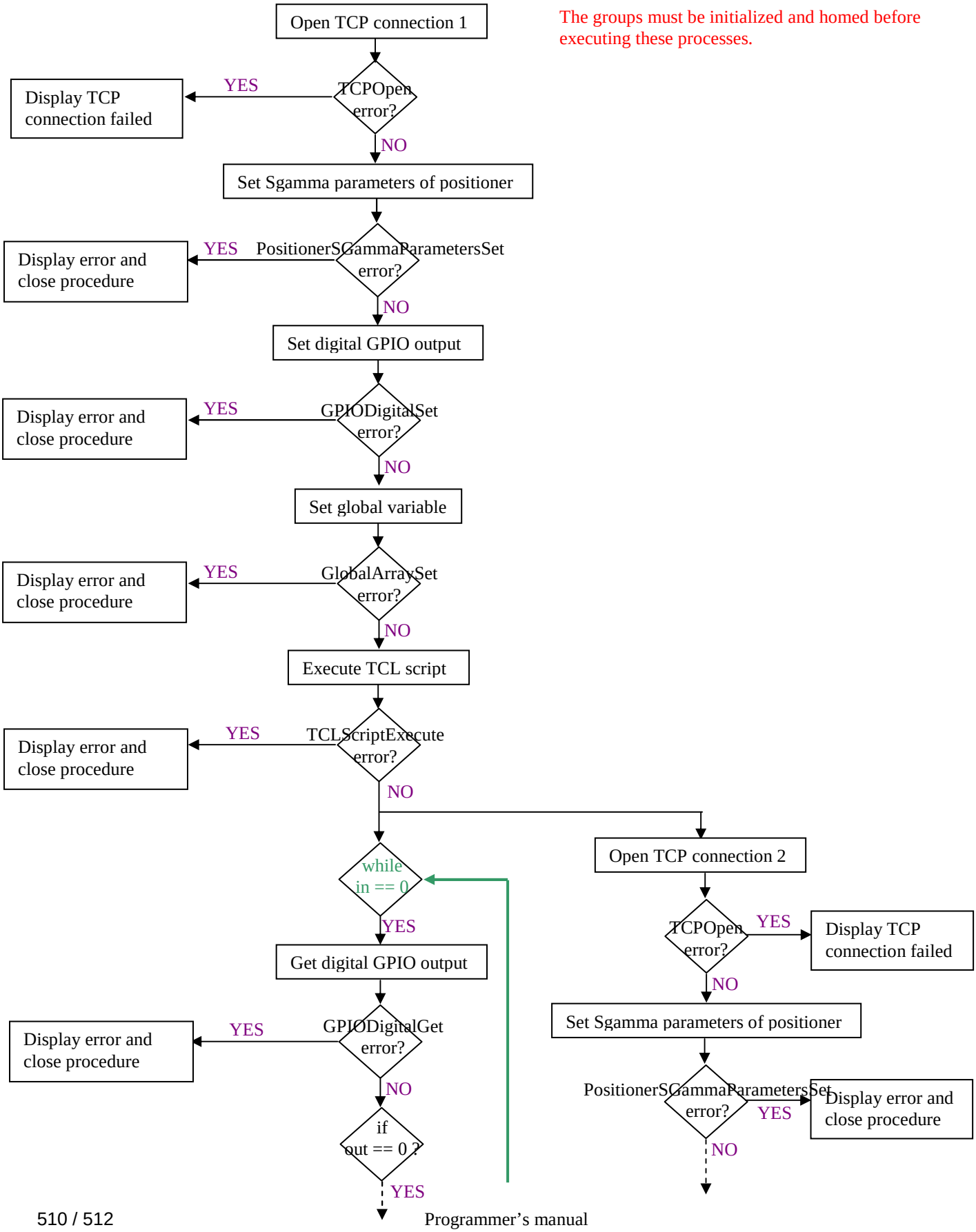

GroupMoveRelative error?
YES
Display error and close procedure
NO
Move in negative direction
GroupMoveRelative error?
YES
Display error and close procedure
NO
Disable Backlash
PositionerBacklashDisable error?
YES
Display error and close procedure
NO
Close TCP socket
TCPClose error?
YES
Display TCP close failed
NO
Display TCP socket closed

3.10. Timer event and global variables

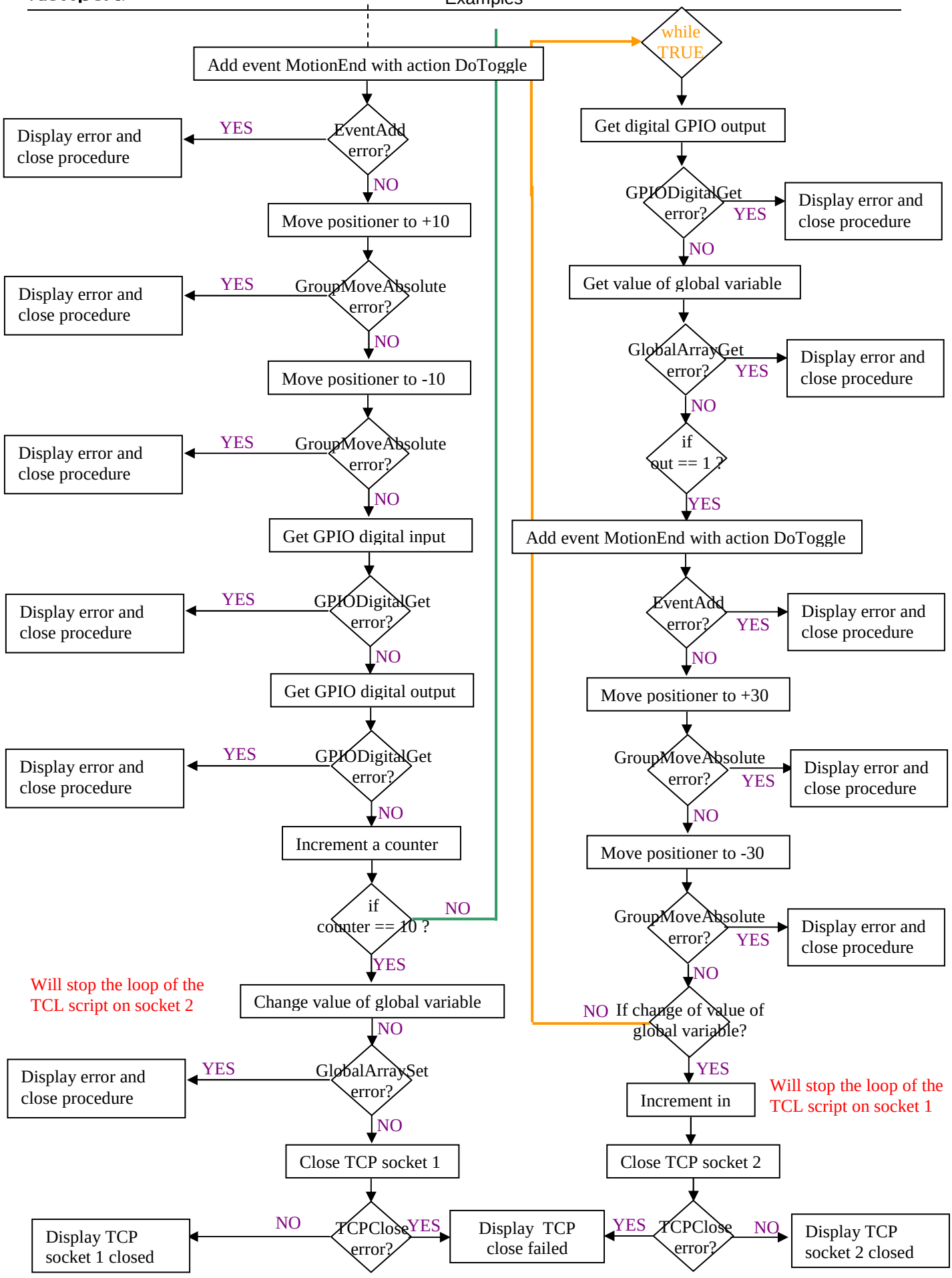



Event occured
Open TCP connection 2
TCPOpen error?
YES
Display TCP connection failed
NO
Read global variable
GlobalArrayGet error?
YES
Display error and close procedure
NO
Value<END error?
NO
YES
Increment global variable
Delete timer event
Set global variable
EventRemove error?
YES
Display error and close procedure
GlobalArraySet error?
YES
Display error and close procedure
NO
NO
Close TCP socket 2
TCPClose error?
YES
Display TCP close failed
NO
Display TCP socket 2 closed

3.11. Running simultaneously several motion processes



The groups must be initialized and homed before executing these processes.


Add event MotionEnd with action DoToggle
EventAdd error?
YES
Display error and close procedure
NO
Move positioner to +10
GroupMoveAbsolute error?
YES
Display error and close procedure
NO
Move positioner to -10
GroupMoveAbsolute error?
YES
Display error and close procedure
NO
Get GPIO digital input
GPIODigitalGet error?
YES
Display error and close procedure
NO
Get GPIO digital output
GPIODigitalGet error?
YES
Display error and close procedure
NO
Increment a counter
if counter == 10 ?
NO
YES
Will stop the loop of the TCL script on socket 2
Change value of global variable
NO
GlobalArraySet error?
YES
Display error and close procedure
NO
Close TCP socket 1
TCPClose error?
NO
Display TCP socket 1 closed
YES
Display TCP close failed
while TRUE
Get digital GPIO output
GPIODigitalGet error?
YES
Display error and close procedure
NO
Get value of global variable
GlobalArrayGet error?
YES
Display error and close procedure
NO
if out == 1 ?
YES
Add event MotionEnd with action DoToggle
EventAdd error?
YES
Display error and close procedure
NO
Move positioner to +30
GroupMoveAbsolute error?
YES
Display error and close procedure
NO
Move positioner to -30
GroupMoveAbsolute error?
YES
Display error and close procedure
NO
If change of value of global variable?
NO
YES
Increment in
Will stop the loop of the TCL script on socket 1
Close TCP socket 2
TCPClose error?
YES
Display TCP close failed
NO
Display TCP socket 2 closed

Newport Corporation
Worldwide Headquarters

North America & Asia:
Newport Corporation
1791 Deere Avenue
Irvine, CA 92606
USA

Tel: (949)-863-3144 or
(800)-222-6440
Fax: (949)-253-1680

Email: sales@newport.com
tech@newport.com

Europe:
Newport / Micro-Contrôle S.A.
11 Rue du Bois Sauvage
F-91055 Evry Cedex
FRANCE

Tel: 33-(0)1-60-91-68-68
Fax: 33-(0)1-60-91-68-69

Email: france@newport-fr.com
tech_europe@newport.com

Newport

Visit Newport Online at: **www.newport.com**

ISO 9001
FM 27207

Newport Corporation, Irvine, California, has been certified compliant with ISO 9001 by the British Standards Institution