# HXP

## Hexapod
## Motion Controller


Newport®

# Programmer's Manual

Intaller Pack Version #30002

# Table of Contents


# Hexapod Motion Controller HXP

## Introduction

The HXP is a high-performance motion controller, dedicated for the use with Newport Hexapods. One HXP is part of each Newport Hexapod system. The controller is preconfigured to the Hexapod mechanics.

The HXP bases on the same hardware as the Newport XPS Universal High-Performance Motion Controller/Driver, but uses a special firmware for Hexapod motion. The HXP has many common features with the XPS, but can control Hexapods and up to two SingleAxis motion systems.

## 1.0 TCP/IP Communication

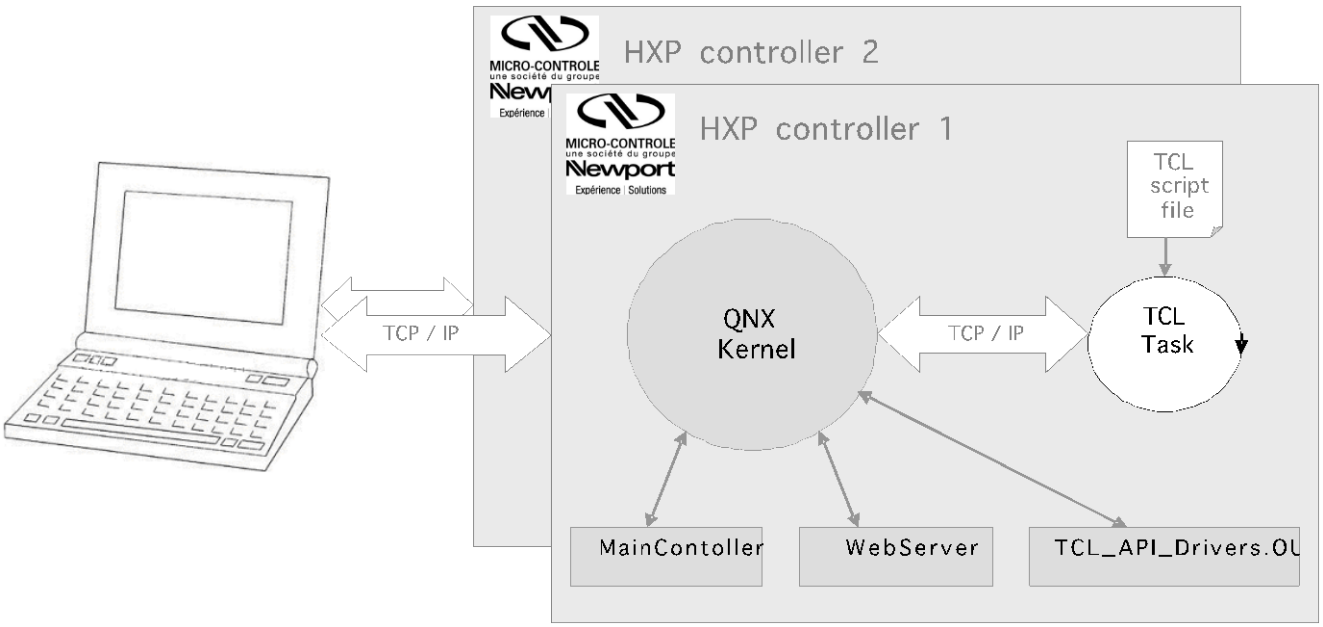
### 1.1 Description

HXP controller 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 HXP 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 HXP 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 HXP 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 HXP, 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 HXP 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)
- HXP 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".**

**CAUTION**

**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)
- HXP 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 HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 HXP 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**

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

**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**

Configures the TCP/IP communication and opens a socket to connect TCP server.

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.

**NOTE**

**OpenConnection function is used when users are in local, it only needs the timeout and socket number to open the connection with the HXP 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 HXP 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 HXP 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 HXP 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 HXP 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 HXP 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

# 2.0 XPS .NET Software Drivers

## 2.1 How to install .NET Drivers for XPS Controller

### 2.1.1 Requirements

.Net Framework is a programming infrastructure created by Microsoft for building, deploying, and running applications and services that use .NET technologies such as custom desktop applications.

The Windows PC computer requires having at least the .NET Framework 4.5.2 installed and you need to install either 32 bit (x86) or 64 bit (x64) .NET assembly depending on the Windows version you are using.

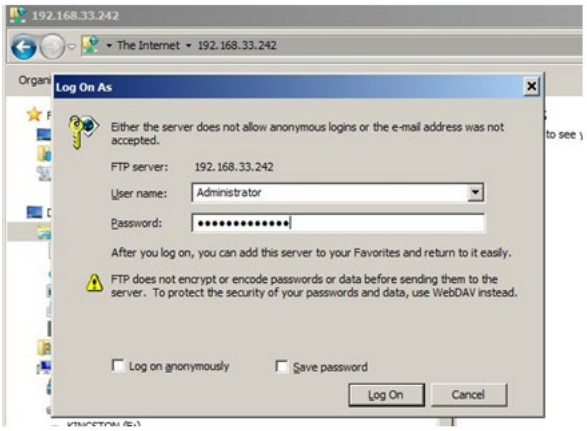
When developing your application, refer to the programming environment documentation to make the installed .NET assembly visible.

To communicate with the XPS controller you will need to:

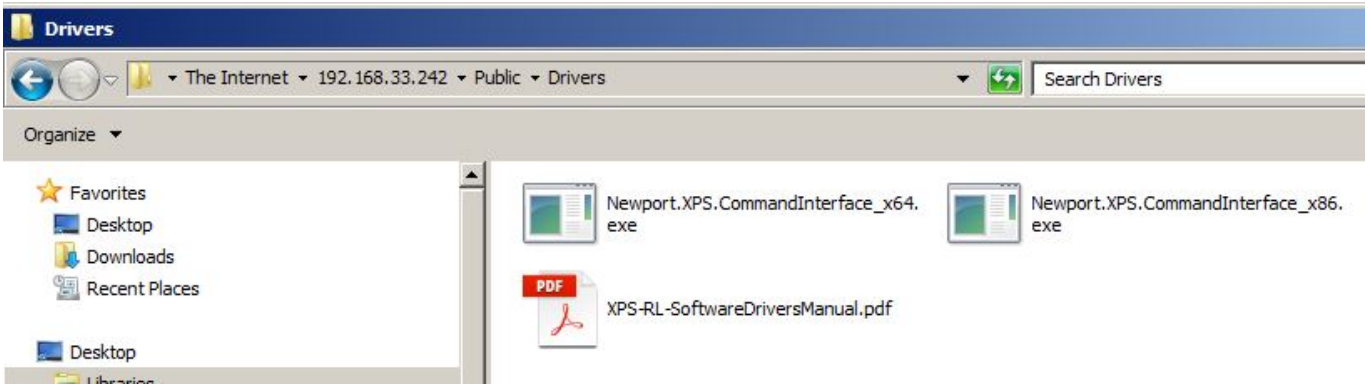
- Use the **OpenInstrument** method to connect to the controller
- Communicate with the controller using any of its API e.g. **FirmwareVersionGet**
- Once your application terminates it needs to disconnect from the controller using the **CloseInstrument** method. If it doesn't close the communication channel and runs many connections to the controller, it can run out of free channels and gets an error.

### 2.1.2 Installing the 32 bit (x86) Windows platform

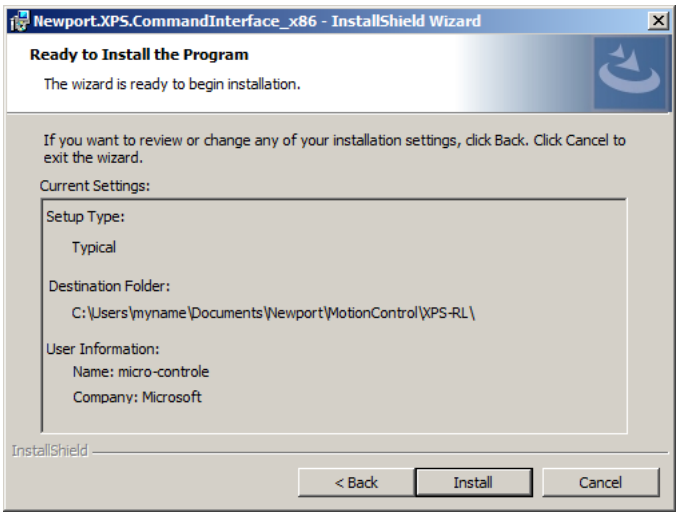
Connect to the XPS controller using the ftp protocol; refer to the XPS user's manual for more details:




Once connected, go to the "/Public/Drivers" folder and download the Newport.XPS.CommandInterface_x86.exe to your computer:



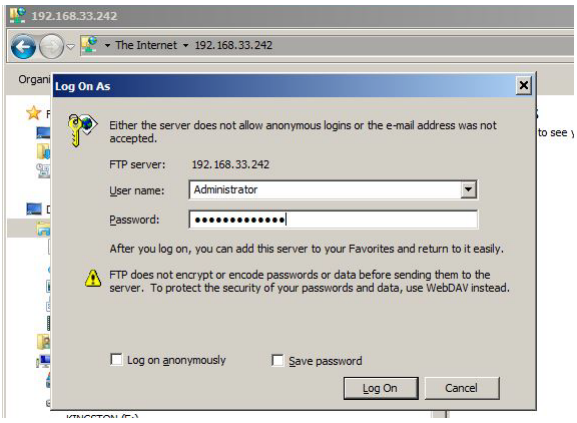

Once downloaded, run the **Newport.XPS.CommandInterface_x86** executable file.



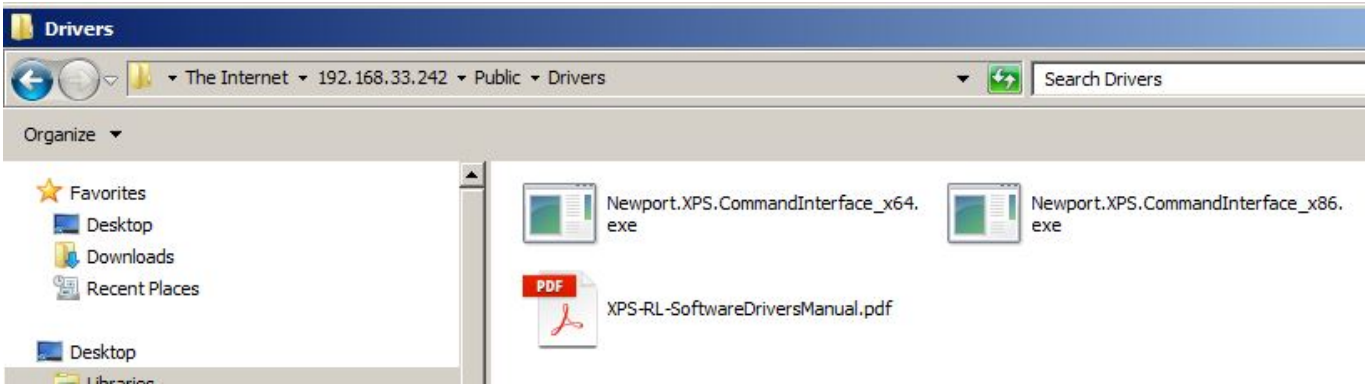

The .Net assembly "Newport.XPS.CommandInterface.dll" V1.0.x.x is installed in the GAC for x86 platforms:
*C:\Windows\Microsoft.NET\assembly\GAC_32\Newport.XPS.CommandInterface\v4.0_1.0.0.0__9a267756cf640dcf*

### 2.1.3 Installing the 64 bit (x64) Windows platform

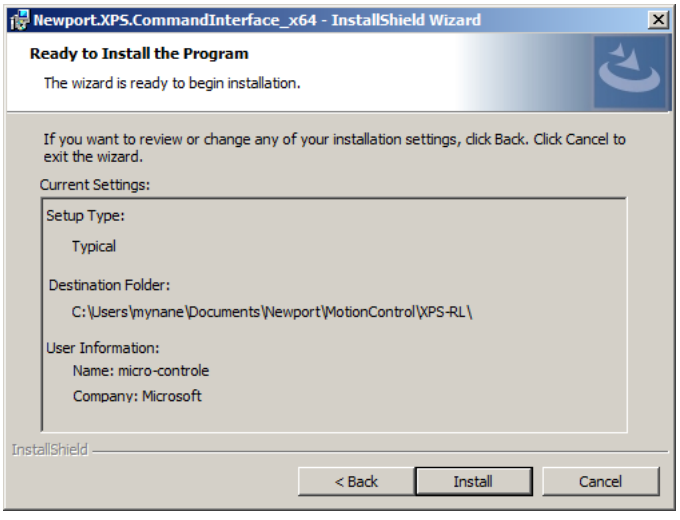
First connect to the XPS through the ftp protocol:




Once connected, go to the "/Public/Drivers" folder and download the Newport.XPS.CommandInterface_x64.exe to your computer:




Once downloaded, run the **Newport.XPS.CommandInterface_x64** executable file.




Once installed, the .Net assembly "Newport.XPS.CommandInterface.dll" V1.0.0.0 is located in GAC for x64 platforms:
*C:\Windows\Microsoft.NET\assembly\GAC_64\Newport.XPS.CommandInterface\v4.0_1.0.0.0__9a267756cf640dcf*

## 2.2 How to Use XPS .NET Assembly from Visual Studio C#?

Refer to Microsoft for more information on how to load and use a .NET assembly depending on your Visual Studio version.

### 2.2.1 Add Reference to .NET Assembly

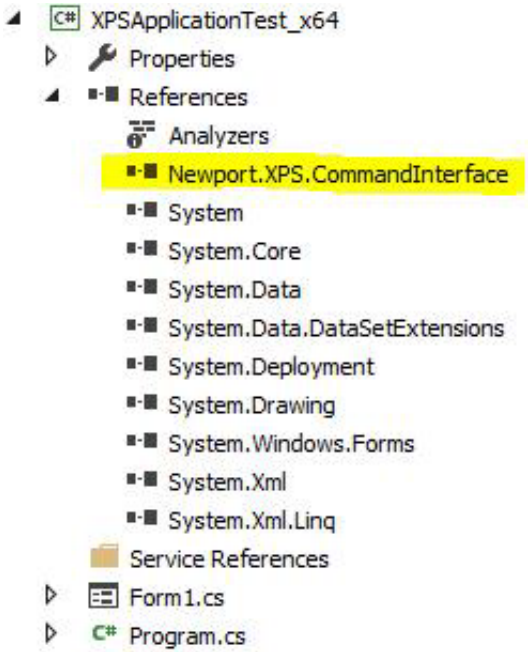
Add Newport.XPS.CommandInterface.dll in References to your project:

**x86:**

*C:\Windows\assembly\GAC_32\Newport.XPS.CommandInterface\1.0.0.0__9a267756cf640dcf*

**x64:**

*C:\Windows\assembly\GAC_64\Newport.XPS.CommandInterface\1.0.0.0__9a267756cf640dcf*



### 2.2.2 C# Code Sources

**C# Header**

```
using CommandInterfaceXPS; // Newport.XPS.CommandInterface .NET Assembly access
```

**Add a Variable to Declare an "XPS" Object**

```
CommandInterfaceXPS.XPS m_xpsInterface = null;
```

**Create an Instance of "XPS" Object**

```
m_xpsInterface = new CommandInterfaceXPS.XPS();
if (m_xpsInterface != null)
…
```

**Open XPS Connection**

```
if (m_xpsInterface != null)
int returnValue = m_xpsInterface.OpenInstrument(m_IPAddress, m_IPPort, DEFAULT_TIMEOUT);
```

**Call "XPS" Functions**

```
if (m_xpsInterface != null)
{
        string XPSVersion = string.Empty;
        string errorString = string.Empty;
        int result = m_xpsInterface.FirmwareVersionGet(out XPSVersion, out errorString);
        if (result == CommandInterfaceXPS.XPS.FAILURE)
        ...
```

**Close XPS Connection**

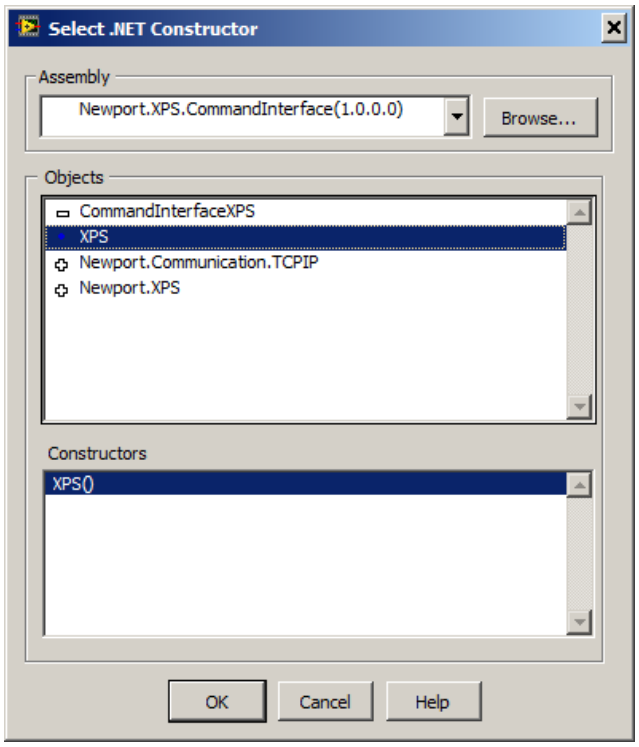
```
if (m_xpsInterface != null)
        m_xpsInterface.CloseInstrument();
```

## 2.3 How to use XPS .NET Assembly from LabVIEW?

Refer to LabVIEW for more information on how to load and use a .NET assembly depending on your LabVEW version.

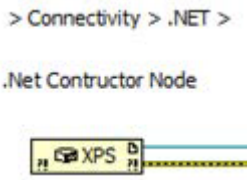
### 2.3.1 Add Reference to .NET Assembly

Select **CommandInterfaceXPS** and **XPS** constructor from a **.Net Constructor Node** (refer to Connectivity panel):

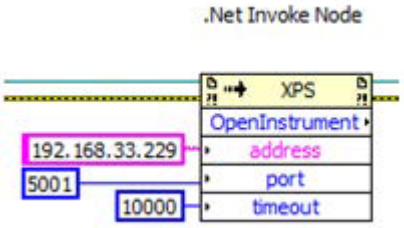


### 2.3.2 LabVIEW Code Sources

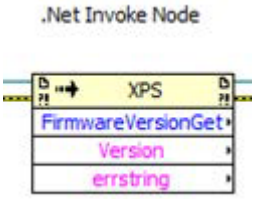
The instance of "XPS" object is created after configuration of **.Net Constructor Node**:



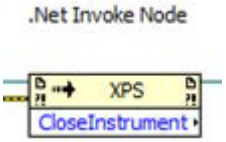

Open XPS connection (Use a **.Net Invoke Node** to select the XPS method "OpenInstrument"):



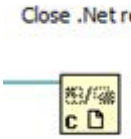

Call "XPS" functions (Use a **.Net Invoke Node** to select a XPS method):




Close XPS connection (Use a **.Net Invoke Node** to select the XPS method "CloseInstrument"):




Close .NET Reference:



## 2.4 How to use XPS .NET assembly from IronPython?

Refer to IronPython for more information on how to load and use a .NET assembly depending on your IronPython version.

### 2.4.1 Add Reference to .NET Assembly

Add Newport.XPS.CommandInterface.dll in References of your script:

**x86:**

```
import sys
sys.path.append(r'C:\Windows\Microsoft.NET\assembly\GAC_32\Newport.XPS.CommandInterface\v4.0_1.0.0.0__9a267756cf640dcf')
```

**x64:**

```
import sys
sys.path.append(r'C:\Windows\Microsoft.NET\assembly\GAC_64\Newport.XPS.CommandInterface\v4.0_1.0.0.0__9a267756cf640dcf')
```

### 2.4.2 IronPython Code Source

**IronPython Header**

```
# The CLR module provide functions for interacting with the underlying
# .NET runtime
import clr
# Add reference to assembly and import names from namespace (IronPython)
clr.AddReferenceToFile("Newport.XPS.CommandInterface.dll") from
CommandInterfaceXPS import *
```

**Create an Instance**

```
# Create XPS interface myXPS = XPS()
```

**Open XPS Connection**

```
def XPS_Open (address, port):
# Create XPS interface
myXPS = XPS()
# Open a socket
timeout = 1000
result = myXPS.OpenInstrument(address, port, timeout)
if result == 0:
    print 'Open ', address, ":", port, " => Successful"
else:
    print 'Open ', address, ":", port, " => failure ", result
return myXPS
```

**Call XPS Functions**

```
def XPS_GetControllerVersion (myXPS, flag):
result, version, errString = myXPS.FirmwareVersionGet()
if flag == 1:
    if result == 0:
    print 'XPS firmware version => ', version
    else:
    print 'FirmwareVersionGet Error => ',errString
    return result, version
def XPS_GetControllerState (myXPS, flag):
result, state, errString = myXPS.ControllerStatusGet()
if flag == 1:
    if result == 0:
    print 'XPS controller state => ', state
else:
    print 'ControllerStatusGet Error => ',errString
return result, state
```

**Close XPS Connection**

```
def XPS_Close(myXPS):
myXPS.CloseInstrument()
```

## 2.5 How to Use XPS .NET Assembly from Matlab?

Refer to Matlab for more information on how to load and use a .NET assembly depending on your Matlab version.

### 2.5.1 Add Reference to .NET Assembly

```
% Make the assembly visible from Matlab
asmInfo = NET.addAssembly('Newport.XPS.CommandInterface')
```

### 2.5.2 Matlab Code Source

**Create an Instance**

```
% Make the instantiation myxps=CommandInterfaceXPS.XPS();
```

**Open XPS Connection**

```
% Connect to the XPS controller
code=myxps.OpenInstrument('192.168.254.254',5001,1000);
```

**Call XPS Functions**

```
% Use API's
[code Version]=myxps.FirmwareVersionGet [code]=myxps.GroupKill('Group1')
[code]=myxps.GroupInitialize('Group1')
[code]=myxps.GroupHomeSearch('Group1')
```

**Close XPS Connection**

```
% Disconnect from the XPS controller
code=myxps.CloseInstrument;
```

# 3.0 HXP Features

## 3.1 General Features

### 3.1.1 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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |
| – Number | int32 | Index in the global array |

### Return

| | | |
|---|---|---|
| – Error | int32 | Function error code |
| – DoubleValue | doublePtr | Variable value |

## Python

### Prototype

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

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" function |
| – Number | integer | Index in the global array |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error code |
| – DoubleValue | doublePtr | Variable value |

### 3.1.2 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |
| – Number | integer | Index in the global array |
| – DoubleValue | Double | Variable value |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error code |

### 3.1.3 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 §3.15 Error ListError 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |
| – | ErrorCode | integer | Error code |

### Return

| | | | |
|---|---|---|---|
| – | Error | integer | Function error code |
| – | ErrorString | string | Error description |

### 3.1.4 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |

### Return

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

### 3.1.5 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 :

*"HXP Firmware V1.2.0"*

- Controller name is **HXP**
- Firmware version is **V1.2.0**

#### 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error code |
| – Version | string | Controller version |

### 3.1.6 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 §3.14 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |
| – | GroupStatusCode | integer | Group status code |

### Return

| | | | |
|---|---|---|---|
| – | Error | integer | Function error code |
| – | GroupStatusString | string | Group status description |

### 3.1.7 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |
| – Number | integer | Index in the global array |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error code |
| – StringValue | string | Variable value |

### 3.1.8 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |
| – Number | integer | Index in the global array |
| – StringValue | string | Variable value |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error code |

### 3.1.9 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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |

**Output parameters**

– None

**Return**

| | | |
|---|---|---|
| – Error | Long | Function error code |

## Matlab

### Prototype

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

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" function |

### Return

| | | |
|---|---|---|
| – Error | int32 | Function error code |

## Python

### Prototype

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

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" function |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error code |

### 3.1.10 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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |

**Output parameters**

– None

**Return**

| | | |
|---|---|---|
| – Error | Long | Function error code |

## Matlab

**Prototype**

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

**Input parameters**

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" function |

**Return**

| | | |
|---|---|---|
| – Error | int32 | Function error code |

## Python

**Prototype**

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

**Input parameters**

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" function |

**Return**

| | | |
|---|---|---|
| – Error | integer | Function error code |

### 3.1.11 RestartApplication

#### Name

**RestartApplication** – Restarts the controller's application and avoids hardware reboot.

#### Input tests

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

#### Description

This function allows restarting application of the controller without hardware reboot.

#### Error codes

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

#### TCL

**Prototype**

**RestartApplication** $SocketID

**Input parameters**

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "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 **RestartApplication** (int SocketID)

**Input parameters**

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |

**Output parameters**

– None

**Return**

| | | |
|---|---|---|
| – Error | int | Function error code |

## Visual Basic

**Prototype**

– Long **RestartApplication** (ByVal SocketID As Long)

**Input parameters**

| | | |
|---|---|---|
| – SocketID | Long | Socket identifier gets by the "TCP_ConnectToServer" function |

**Output parameters**

– None

**Return**

| | | |
|---|---|---|
| – Error | Long | Function error code |

## Matlab

**Prototype**

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

**Input parameters**

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" function |

**Return**

| | | |
|---|---|---|
| – Error | int32 | Function error code |

## Python

**Prototype**

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

**Input parameters**

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" function |

**Return**

| | | |
|---|---|---|
| – Error | integer | Function error code |

### 3.1.12 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |
| – TimerName | string | Name of timer |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error code |
| – FrequencyTicks | integer | Number of frequency ticks |

### 3.1.13 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |
| – TimerName | string | Name of timer |
| – FrequencyTicks | integer | Number of frequency ticks |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error code |

## 3.2 Positioner

### 3.2.1 Description

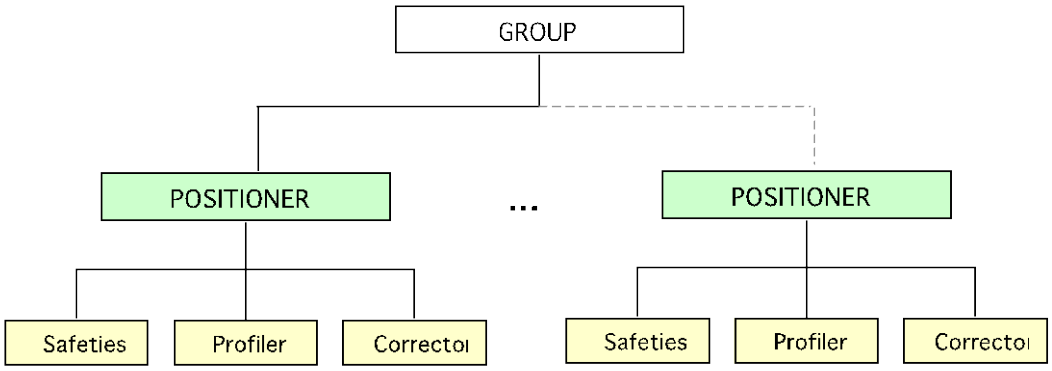
Positioner objects are used to define all motion specific configuration parameters.

The positioner includes a mapping correction : X = f (X)

The positioner includes the SGamma profile.

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

### 3.2.2 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:**

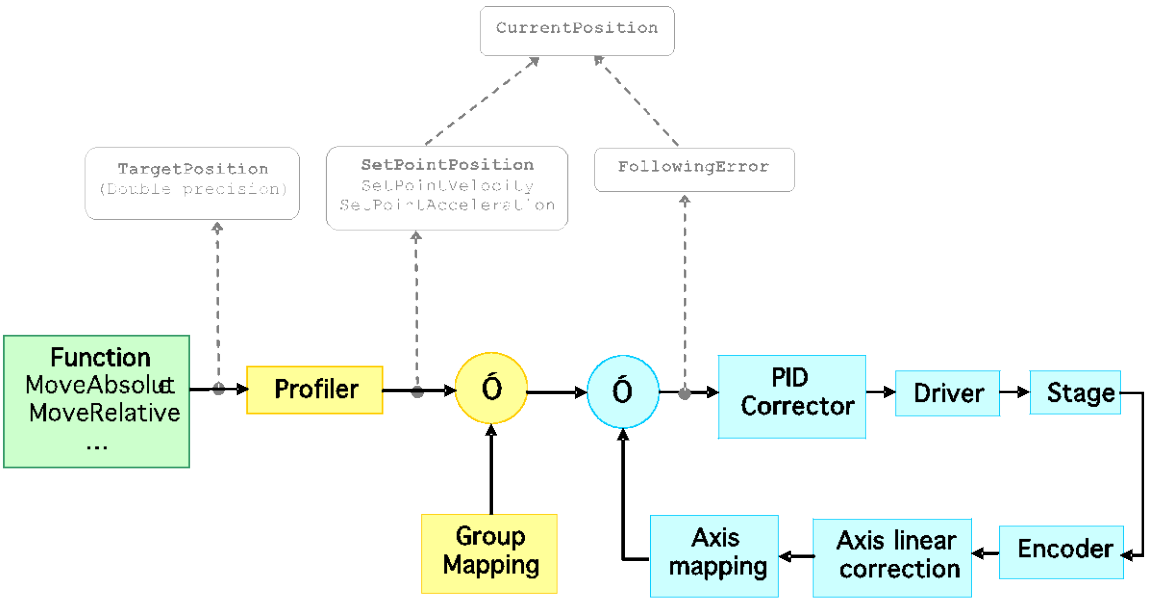
*GroupName.PositionerName*

### 3.2.3 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.*

### 3.2.4 Function description

#### 3.2.4.1 PositionerBacklashDisable

**Name**

**PositionerBacklashDisable** – Disables the backlash compensation.

**Input tests**

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer"function |
| – | FullPositionerName | string | Positioner name (maximum size = 250) |

### Return

– Function error code

### 3.2.4.2 PositionerBacklashEnable

#### Name

**PositionerBacklashEnable** – Enables the backlash compensation.

#### Input tests

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 HXP 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**

**If 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer"function |
| – FullPositionerName | string | Positioner name (maximum size = 250) |

### Return

– Function error code

### 3.2.4.3 PositionerBacklashGet

#### Name

**PositionerBacklashGet** – Gets the backlash compensation value.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "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") |

### 3.2.4.4 PositionerBacklashSet

#### Name

**PositionerBacklashSet** – Sets the backlash compensation value.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – BacklashValue | double | Backlash compensation value (units) |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error code |

### 3.2.4.5 PositionerCompensatedPCOAbort

#### Name

**PositionerCompensatedPCOAbort** – Abort the CIE08 compensated PCO pulses generation.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 the CIE board supports this function: ERR_HARDWARE_FUNCTION_NOT_SUPPORTED (-115)
- Check CIE08CompensatedPCOMode = Enabled (*system.ini*) : ERR_NOT_ALLOWED_MODE_DISABLED (-121)

#### Description

This function aborts the CIE08 compensated PCO pulses generation. The pulses generation is stopped immediately; no more pulses will be generated even if the scanning positioner continues to move across the predefined firing positions. To stop the scanning move, use *GroupMoveAbort()* function.

---

**NOTE**

- **The function works only when the CIE08 compensated PCO mode configuration is enabled (*system.ini* : CIE08CompensatedPCOMode = Enabled).**
- **This function can be used only with a position encoder ("AquadB" or "AnalogInterpolated"), elsewhere 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_HARDWARE_FUNCTION_NOT_SUPPORTED (-115)
- ERR_NOT_ALLOWED_MODE_DISABLED (-121)
- SUCCESS (0): no error

## TCL

**Prototype**

**PositionerCompensatedPCOAbort** $SocketID $FullPositionerName

**Input parameters**

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |

**Output parameters**

– None

**Return**

| | | | |
|---|---|---|---|
| – | Error | int | TCL error code (0 = success or 1 = syntax error) or function error code |

## C/C++

**Prototype**

int **PositionerCompensatedPCOAbort** (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 **PositionerCompensatedPCOAbort** (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] **PositionerCompensatedPCOAbort** (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] **PositionerCompensatedPCOAbort** (integer SocketID, string FullPositionerName)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |

### 3.2.4.6 PositionerCompensatedPCOCurrentStatusGet

#### Name

**PositionerCompensatedPCOCurrentStatusGet** – Get current status of CIE08 compensated PCO pulses generation.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 the CIE board supports this function: ERR_HARDWARE_FUNCTION_NOT_SUPPORTED (-115)
- Check CIE08CompensatedPCOMode = Enabled (*system.ini*) : ERR_NOT_ALLOWED_MODE_DISABLED (-121)

#### Description

This function gets the current status of CIE08 compensated PCO pulses generation.

Status possible values :

0 : Pulses generation inactive (*idle, no error*)

1 : Pulses generation activated (*running*)

-1 : Pulses generation aborted with errors.

---

**NOTE**

- **The function works only when the CIE08 compensated PCO mode configuration is enabled (*system.ini* : CIE08CompensatedPCOMode = Enabled).**
- **This function can be used only with a position encoder ("AquadB" or "AnalogInterpolated"), otherwise 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_HARDWARE_FUNCTION_NOT_SUPPORTED (-115)
- ERR_NOT_ALLOWED_MODE_DISABLED (-121)

- SUCCESS (0): no error

## TCL

### Prototype

**PositionerCompensatedPCOCurrentStatusGet** $SocketID $FullPositionerName Status

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |

### Output parameters

| | | | |
|---|---|---|---|
| – | Status | int | Mode status |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | TCL error code (0 = success or 1 = syntax error) or function error code |

## C/C++

### Prototype

int **PositionerCompensatedPCOCurrentStatusGet** (int SocketID, char FullPositionerName[250], int * Status)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | char * | Positioner name |

### Output parameters

| | | | |
|---|---|---|---|
| – | Status | int * | Mode status |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |

## Visual Basic

### Prototype

Long **PositionerCompensatedPCOCurrentStatusGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, Status As Long)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |

### Output parameters

| | | | |
|---|---|---|---|
| – | Status | long | Mode status |

### Return

| | | | |
|---|---|---|---|
| – | Error | long | Function error code |

## Matlab

### Prototype

[Error, Status] **PositionerCompensatedPCOCurrentStatusGet** (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 |
| – | Status | int32 | Mode status |

## Python

### Prototype

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

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |
| – | Status | int | Mode status |

### 3.2.4.7 PositionerCompensatedPCOEnable

#### Name

**PositionerCompensatedPCOEnable** – Activate the CIE08 compensated PCO pulses generation.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 the CIE board supports this function: ERR_HARDWARE_FUNCTION_NOT_SUPPORTED (-115)
- Check CIE08CompensatedPCOMode = Enabled (*system.ini*) : ERR_NOT_ALLOWED_MODE_DISABLED (-121)
- Check if data have been prepared by PositionerCompensatedPCOPrepare() : ERR_CHECK_DATA_INCORRECT (-122)
- Check current position is out and at the good side of scanning zone (*left of scanning zone if ScanDirection positive, right of scanning zone if ScanDirection negative*) : ERR_NOT_ALLOWED_ACTION (-22)
- Check CIE08 compensated PCO mode is running : ERR_NOT_ALLOWED_ACTION (-22)

#### Description

This function activates the CIE08 compensated PCO pulses generation *(status becomes running (value 1))*. The pulses will be generated when the scanning positioner will move across the predefined positions. When the last pulse is generated, the CIE08 compensated PCO mode will become inactive (*status becomes inactive (value 0))*. To get status of the CIE08 compensated PCO pulses generation, use *PositionerCompensatedPCOCurrentStatusGet()* function.

Note that only the scanning positioner positions are used to fire pulses: if you prepare a set of positions at a given location and then enable the pulses generation and start the move from a different location, the pulses could be generated but their accuracy will be impacted by the mapping difference between the two locations.

This function must be used after the firing pulses data preparation with the *PositionerCompensatedPCOPrepare*(), elsewhere the function fails and the error ERR_CHECK_DATA_INCORRECT (-122) will be returned.

---

**NOTE**

**The PCO pulses generation depends on the ScanVelocity and the pulse settling time (*set by PositionerPositionComparePulseParametersSet()*)**

---

Valid settings are shown in the table below:

| | PCO encoder frequency (kHz) | | | |
|---|---|---|---|---|
| **Pulse settling time (µs)** | 25 | 50 | 125 | > 500 |
| 0.075 | | | OK | OK |
| 1 | | OK | OK | |
| 4 | OK | OK | | |
| 12 | OK | | | |

How to determine the PCO encoder frequency :

- For AquadB encoder :
  *PCO encoder frequency = Velocity / EncoderResolution*
- For analog interpolated encoder :
  *PCO encoder frequency = Velocity * HardInterpolatorFactor / EncoderScalePitch*

Example: XML310 stage (EncoderScalePitch=0.004 mm, HardInterpolatorFactor=200).
With ScanVelocity=10mm/s => PCO encoder frequency = 10*200/0.004 = 500 kHz

**NOTE**

- **The function works only when the CIE08 compensated PCO mode configuration is enabled (*system.ini* : CIE08CompensatedPCOMode = Enabled).**
- **This function can be used only with a position encoder ("AquadB" or "AnalogInterpolated"), otherwise ERR_UNCOMPATIBLE (-24) error is returned.**

## Error codes

- ERR_FATAL_INIT (-20)
- ERR_NOT_ALLOWED_ACTION (-22)
- ERR_IN_INITIALIZATION (-21)
- ERR_UNCOMPATIBLE (-24)
- ERR_WRONG_FORMAT (-7)
- ERR_WRONG_OBJECT_TYPE (-8)
- ERR_WRONG_PARAMETERS_NUMBER (-9)
- ERR_HARDWARE_FUNCTION_NOT_SUPPORTED (-115)
- ERR_NOT_ALLOWED_MODE_DISABLED (-121)
- ERR_CHECK_DATA_INCORRECT (-122)
- SUCCESS (0): no error

## TCL

**Prototype**

**PositionerCompensatedPCOEnable** $SocketID $FullPositionerName

**Input parameters**

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

**Output parameter**

– None

**Return**

| | | |
|---|---|---|
| – Error | int | TCL error code (0 = success or 1 = syntax error) or function error code |

## C/C++

**Prototype**

int **PositionerCompensatedPCOEnable** (int SocketID, char FullPositionerName[250])

**Input parameters**

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | char * | Positioner name |

**Output parameter**

– None

**Return**

| | | |
|---|---|---|
| – Error | int | Function error code |

## Visual Basic

**Prototype**

Long **PositionerCompensatedPCOEnable** (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] **PositionerCompensatedPCOEnable** (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] **PositionerCompensatedPCOEnable** (integer SocketID, string FullPositionerName)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

### 3.2.4.8 PositionerCompensatedPCOFromFile

#### Name

**PositionerCompensatedPCOFromFile** – Read firing positions from a data file to controller's memory.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 the CIE board supports this function: ERR_HARDWARE_FUNCTION_NOT_SUPPORTED (-115)
- Check CIE08CompensatedPCOMode = Enabled (*system.ini*) : ERR_NOT_ALLOWED_MODE_DISABLED (-121)
- Check data file exists : ERR_READ_FILE (-61)
- Check data from file (must be $Position_i > Position_{i-1}$, $Width_i < Position_{i+1} - Position_i$) : ERR_CHECK_DATA_INCORRECT (-122)

#### Description

This function reads firing positions from a data file to the controller's memory.

The data file contains lines of data, formatted as follows:

$Position_i$<Space or Tabulation>$Width_i$<CRLF or LF>

Example :

$Position_1 Width_1$

$Position_2 Width_2$

… …

$Position_N Width_N$

Data conditions : $Position_i > Position_{i-1}$, $Width_i < Position_{i+1} - Position_i$

**NOTE**

- **$Position_i$ (i=1..N) are the offset values relative to the scanning positioner start position that is defined in the PositionerCompensatedPCOPrepare().**
- **The function works only when the CIE08 compensated PCO mode configuration is enabled (*system.ini* : CIE08CompensatedPCOMode = Enabled).**
- **This function can be used only with a position encoder ("AquadB" or "AnalogInterpolated"), otherwise 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_HARDWARE_FUNCTION_NOT_SUPPORTED (-115)
- ERR_NOT_ALLOWED_MODE_DISABLED (-121)
- ERR_READ_FILE (-61)
- ERR_CHECK_DATA_INCORRECT (-122)
- SUCCESS (0): no error

**TCL**

**Prototype**

**PositionerCompensatedPCOFromFile** $SocketID $FullPositionerName $DataFileName

**Input parameters**

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – DataFileName | string | Data file name |

**Output parameter**

– None

**Return**

| | | |
|---|---|---|
| – Error | int | TCL error code (0 = success or 1 = syntax error) or function error code |

## C/C++

**Prototype**

int **PositionerCompensatedPCOFromFile** (int SocketID, char FullPositionerName[250] , char DataFileName[250])

**Input parameters**

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | char * | Positioner name |
| – DataFileName | char * | Data file name |

**Output parameter**

– None

**Return**

| | | |
|---|---|---|
| – Error | int | Function error code |

## Visual Basic

**Prototype**

Long **PositionerCompensatedPCOFromFile** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal DataFileName As String)

**Input parameters**

| | | |
|---|---|---|
| – SocketID | long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – DataFileName | string | Data file name |

**Output parameter**

– None

**Return**

| | | |
|---|---|---|
| – Error | long | Function error code |

## Matlab

**Prototype**

[Error] **PositionerCompensatedPCOFromFile** (int32 SocketID, cstring FullPositionerName, cstring DataFileName)

**Input parameters**

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | cstring | Positioner name |
| – DataFileName | cstring | Data file name |

**Return**

| | | |
|---|---|---|
| – Error | int32 | Function error code |

## Python

### Prototype

[Error] **PositionerCompensatedPCOFromFile** (integer SocketID, string FullPositionerName, string DataFileName)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |
| – | DataFileName | string | Data file name |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |

### 3.2.4.9 PositionerCompensatedPCOLoadToMemory

#### Name

**PositionerCompensatedPCOLoadToMemory** – Append firing positions to controller's memory..

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 the CIE board supports this function: ERR_HARDWARE_FUNCTION_NOT_SUPPORTED (-115)
- Check CIE08CompensatedPCOMode = Enabled (*system.ini*) : ERR_NOT_ALLOWED_MODE_DISABLED (-121)
- Check data lines (must be $Position_i > Position_{i-1}$, $Width_i < Position_{i+1} - Position_i$) : ERR_CHECK_DATA_INCORRECT (-122)

#### Description

This function appends firing positions to controller's memory from *DataLines* parameter.

To reset the controller's memory, the *PositionerCompensatedPCOMemoryReset()* function is provided.

The data line format must be :

$Position_i$<Space or Tabulation>$Width_i$<CRLF, LF or ;>

Example :

$Position_1 Width_1$

$Position_2 Width_2$

… …

$Position_N Width_N$

Or :

$Position_1 Width_1$;$Position_2 Width_2$; …;$Position_N Width_N$

Data conditions : $Position_i > Position_{i-1}$, $Width_i < Position_{i+1} - Position_i$

Example : Send PositionerCompensatedLoadToMemory (XY.X,0 0.1;1 0.1;2 0.1;3 0.1)

**NOTE**

- **Position$_i$ (i=1..N) are the offset values relative to the scanning positioner start position that is defined in the PositionerCompensatedPCOPrepare().**
- **The function works only when the CIE08 compensated PCO mode configuration is enabled (*system.ini* : CIE08CompensatedPCOMode = Enabled).**
- **This function can be used only with a position encoder ("AquadB" or "AnalogInterpolated"), otherwise 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_HARDWARE_FUNCTION_NOT_SUPPORTED (-115)
- ERR_NOT_ALLOWED_MODE_DISABLED (-121)
- ERR_CHECK_DATA_INCORRECT (-122)
- SUCCESS (0): no error

## TCL

### Prototype

**PositionerCompensatedPCOLoadToMemory** $SocketID $FullPositionerName $DataLines

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |
| – | DataLines | string | Some data lines |

### Output parameter

– None

### Return

| | | | |
|---|---|---|---|
| – | Error | int | TCL error code (0 = success or 1 = syntax error) or function error code |

## C/C++

**Prototype**

int **PositionerCompensatedPCOLoadToMemory** (int SocketID, char FullPositionerName[250] , char DataLines[500])

**Input parameters**

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | char * | Positioner name |
| – DataLines | char * | Some data lines |

**Output parameters**

– None

**Return**

| | | |
|---|---|---|
| – Error | int | Function error code |

## Visual Basic

**Prototype**

Long **PositionerCompensatedPCOLoadToMemory** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal DataLines As String)

**Input parameters**

| | | |
|---|---|---|
| – SocketID | long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – DataLines | string | Some data lines |

**Output parameter**

– None

**Return**

| | | |
|---|---|---|
| – Error | long | Function error code |

## Matlab

**Prototype**

[Error] **PositionerCompensatedPCOLoadToMemory** (int32 SocketID, cstring FullPositionerName, cstring DataLines)

**Input parameters**

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | cstring | Positioner name |
| – DataLines | cstring | Some data lines |

**Return**

| | | |
|---|---|---|
| – Error | int32 | Function error code |

## Python

### Prototype

[Error] **PositionerCompensatedPCOLoadToMemory** (integer SocketID, string FullPositionerName, string DataLines)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |
| – | DataLines | string | Some data lines |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |

### 3.2.4.10 PositionerCompensatedPCOMemoryReset

#### Name

**PositionerCompensatedPCOMemoryReset** – Reset CIE08 compensated PCO data memory.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 the CIE board supports this function: ERR_HARDWARE_FUNCTION_NOT_SUPPORTED (-115)
- Check CIE08CompensatedPCOMode = Enabled (*system.ini*) : ERR_NOT_ALLOWED_MODE_DISABLED (-121)

#### Description

This function resets the CIE08 compensated PCO data memory. This function is useful to remove the data that was previously entered with the *PositionerCompensatedPCOLoadToMemory()* function.

**NOTE**

- **The function works only when the CIE08 compensated PCO mode configuration is enabled (*system.ini* : CIE08CompensatedPCOMode = Enabled).**
- **This function can be used only with a position encoder ("AquadB" or "AnalogInterpolated"), otherwise 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_HARDWARE_FUNCTION_NOT_SUPPORTED (-115)
- ERR_NOT_ALLOWED_MODE_DISABLED (-121)
- SUCCESS (0): no error

## TCL

### Prototype

**PositionerCompensatedPCOMemoryReset** $SocketID $FullPositionerName

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |

### Output parameter

– None

### Return

| | | | |
|---|---|---|---|
| – | Error | int | TCL error code (0 = success or 1 = syntax error) or function error code |

## C/C++

### Prototype

int **PositionerCompensatedPCOMemoryReset** (int SocketID, char FullPositionerName[250])

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | char * | Positioner name |

### Output parameter

– None

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |

## Visual Basic

### Prototype

Long **PositionerCompensatedPCOMemoryReset** (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 parameter

– None

### Return

| | | | |
|---|---|---|---|
| – | Error | long | Function error code |

## Matlab

### Prototype

[Error] **PositionerCompensatedPCOMemoryReset** (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] **PositionerCompensatedPCOMemoryReset** (integer SocketID, string FullPositionerName)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |

### 3.2.4.11 PositionerCompensatedPCOPrepare

#### Name

**PositionerCompensatedPCOPrepare** – Prepare data for CIE08 compensated PCO pulses generation.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 the CIE board supports this function: ERR_HARDWARE_FUNCTION_NOT_SUPPORTED (-115)
- Check CIE08CompensatedPCOMode = Enabled (*system.ini*) : ERR_NOT_ALLOWED_MODE_DISABLED (-121)
- Check data have been set, loaded or read from file to buffer (DataNumber must > 0 and < CIE08CompensatedPCOMaximumDataNumber (*system.ini)*) : ERR_CHECK_DATA_INCORRECT (-122)
- Check scanning zone exceed stage travel limits : ERR_TRAVEL_LIMITS (-35)
- Check input parameter values (ScanDirection value must equal to 1 *(positive direction)* or -1 *(negative direction)*) : ERR_PARAMETER_OUT_OF_RANGE (-17)

#### Description

This function calculates the firing for absolute positions*, in user's coordinate system* and converts them to firing absolute raw PCO positions, *in encoder's coordinate system*.

When mappings are enabled, the correction between user's coordinate system position and raw encoder position will be different at each location. For this reason, the prepare function must know the location (*positions of all positioners in the scanning group*) where the scan will be done.

This function must be called before the use of *PositionerCompensatedPCOEnable()* function.

Parameters :

- ScanDirection : Scan direction,(value : 1 (*positive*) or -1 (*negative*)).
- StartPosition1 : Group 1st positioner start position.
- StartPosition2 : Group 2nd positioner start position
- StartPosition3 : Group 3rd positioner start position
- …… etc ……

**NOTE**

- **The function works only when the CIE08 compensated PCO mode configuration is enabled (*system.ini* : CIE08CompensatedPCOMode = Enabled)..**
- **This function can be used only with a position encoder ("AquadB" or "AnalogInterpolated"), otherwise 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_HARDWARE_FUNCTION_NOT_SUPPORTED (-115)
- ERR_NOT_ALLOWED_MODE_DISABLED (-121)
- ERR_CHECK_DATA_INCORRECT (-122)
- ERR_TRAVEL_LIMITS (-35)
- SUCCESS (0): no error

## TCL

### Prototype

**PositionerCompensatedPCOPrepare** $SocketID $FullPositionerName $ScanDirection $StartPosition1 $StartPosition2 $StartPosition3 …

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |
| – | ScanDirection | int | Scan direction (1 or -1) |
| – | StartPosition1 | double | Group 1st positioner start position (units) |
| – | StartPosition2 | double | Group 2nd positioner start position (units) |
| – | StartPosition3 | double | Group 3rd positioner start position (units) |
| – | | ……. | ……. ……. |

### Output parameter

– None

### Return

| | | | |
|---|---|---|---|
| – | Error | int | TCL error code (0 = success or 1 = syntax error) or function error code |

## C/C++

### Prototype

int **PositionerCompensatedPCOPrepare** (int SocketID, char FullPositionerName[250], int ScanDirection, double StartPosition1, double StartPosition2, double StartPosition3, …)

### Input parameters

| | | | |
|---|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function | |
| – FullPositionerName | char * | Positioner name | |
| – ScanDirection | int | Scan direction (1 or -1) | |
| – StartPosition1 | double | Group 1st positioner start position (units) | |
| – StartPosition2 | double | Group 2nd positioner start position (units) | |
| – StartPosition3 | double | Group 3rd positioner start position (units) | |
| – | ……. | ……. | ……. |

### Output parameter

– None

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

## Visual Basic

### Prototype

Long **PositionerCompensatedPCOPrepare** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal ScanDirection As Double, ByVal StartPosition1 As Double, ByVal StartPosition2 As Double, ByVal StartPosition3 As Double, …)

### Input parameters

| | | | |
|---|---|---|---|
| – SocketID | long | Socket identifier gets by the "TCP_ConnectToServer" function | |
| – FullPositionerName | string | Positioner name | |
| – ScanDirection | long | Scan direction (1 or -1) | |
| – StartPosition1 | double | Group 1st positioner start position (units) | |
| – StartPosition2 | double | Group 2nd positioner start position (units) | |
| – StartPosition3 | double | Group 3rd positioner start position (units) | |
| – | ……. | ……. | ……. |

### Output parameter

– None

### Return

| | | |
|---|---|---|
| – Error | long | Function error code |

## Matlab

### Prototype

[Error] **PositionerCompensatedPCOPrepare** (int32 SocketID, cstring FullPositionerName, int32 ScanDirection, double StartPosition1, double StartPosition2, double StartPosition3, …)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | cstring | Positioner name |
| – | ScanDirection | int32 | Scan direction (1 or -1) |
| – | StartPosition1 | double | Group 1st positioner start position (units) |
| – | StartPosition2 | double | Group 2nd positioner start position (units) |
| – | StartPosition3 | double | Group 3rd positioner start position (units) |
| – | ……. | ……. | ……. |

### Return

| | | | |
|---|---|---|---|
| – | Error | int32 | Function error code |

## Python

### Prototype

[Error] **PositionerCompensatedPCOPrepare** (integer SocketID, string FullPositionerName, integer ScanDirection, double StartPosition1, double StartPosition2, double StartPosition3, …)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |
| – | ScanDirection | int32 | Scan direction (1 or -1) |
| – | StartPosition1 | double | Group 1st positioner start position (units) |
| – | StartPosition2 | double | Group 2nd positioner start position (units) |
| – | StartPosition3 | double | Group 3rd positioner start position (units) |
| – | ……. | ……. | ……. |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |

### 3.2.4.12 PositionerCompensatedPCOSet

#### Name

**PositionerCompensatedPCOSet** – Calculate a set of evenly spaced firing positions to the controller's memory.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 input parameter values: ERR_PARAMETER_OUT_OF_RANGE (-17)
- (Must be: Start < Stop, Distance > 0, Width > 0, Width < Distance, Width < Stop-Start)
- Check the position encoder ("AquadB" or "AnalogInterpolated"): ERR_UNCOMPATIBLE (-24)
- Check the CIE board supports this function: ERR_HARDWARE_FUNCTION_NOT_SUPPORTED (-115)
- Check CIE08CompensatedPCOMode = Enabled (*system.ini*) : ERR_NOT_ALLOWED_MODE_DISABLED (-121)
- Check DataNumber : NData = integer((Stop-Start)/Distance) + 1
- If NData > CIE08CompensatedPCOMaximumDataNumber (*system.ini*): ERR_CHECK_DATA_INCORRECT (-122)

#### Description

This function calculates a set of evenly spaced firing positions to the controller's memory.

Parameters :

- Start: Relative distance to the start position where PCO pulses start.
- Stop : Relative distance to the start position where PCO pulses stop.
- Distance : Step of pulses (distance between two consecutive pulses)
- Width : Width of the pulse enable signal at the firing positions.

Example : Send PositionerCompensatedPCOSet (XY.X, 0, 3, 1, 0.1)

**NOTE**

- **Start and Stop are the offset values relative to the scanning positioner start position that is defined in the PositionerCompensatedPCOPrepare().**
- **The function works only when the CIE08 compensated PCO mode configuration is enabled (*system.ini* : CIE08CompensatedPCOMode = Enabled).**
- **This function can be used only with a position encoder ("AquadB" or "AnalogInterpolated"), otherwise 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_HARDWARE_FUNCTION_NOT_SUPPORTED (-115)
- ERR_NOT_ALLOWED_MODE_DISABLED (-121)
- ERR_CHECK_DATA_INCORRECT (-122)
- SUCCESS (0): no error

## TCL

**Prototype**

**PositionerCompensatedPCOSet** $SocketID $FullPositionerName $Start $Stop $Distance $Width

**Input parameters**

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – Start | double | Start position (units) |
| – Stop | double | Stop position (units) |
| – Distance | double | Distance between two consecutive pulses (units) |
| – Width | double | Width of pulse enable signal (units) |

**Output parameter**

– None

**Return**

| | | |
|---|---|---|
| – Error | int | TCL error code (0 = success or 1 = syntax error) or function error code |

## C/C++

### Prototype

int **PositionerCompensatedPCOSet** (int SocketID, char FullPositionerName[250] , double Start, double Stop, double Distance, double Width)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | char * | Positioner name |
| – Start | double | Start position (units) |
| – Stop | double | Stop position (units) |
| – Distance | double | Distance between two consecutive pulses (units) |
| – Width | double | Width of pulse enable signal (units) |

### Output parameter

– None

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

## Visual Basic

### Prototype

Long **PositionerCompensatedPCOSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal Start As Double, ByVal Stop As Double, ByVal Distance As Double, ByVal Width As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – Start | double | Start position (units) |
| – Stop | double | Stop position (units) |
| – Distance | double | Distance between two consecutive pulses (units) |
| – Width | double | Width of pulse enable signal (units) |

### Output parameter

– None

### Return

| | | |
|---|---|---|
| – Error | long | Function error code |

## Matlab

### Prototype

[Error] **PositionerCompensatedPCOSet** (int32 SocketID, cstring FullPositionerName, double Start, double Stop, double Distance, double Width)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | cstring | Positioner name |
| – | Start | double | Start position (units) |
| – | Stop | double | Stop position (units) |
| – | Distance | double | Distance between two consecutive pulses (units) |
| – | Width | double | Width of pulse enable signal (units) |

### Return

| | | | |
|---|---|---|---|
| – | Error | int32 | Function error code |

## Python

### Prototype

[Error] **PositionerCompensatedPCOSet** (integer SocketID, string FullPositionerName, double Start, double Stop, double Distance, double Width)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |
| – | Start | double | Start position (units) |
| – | Stop | double | Stop position (units) |
| – | Distance | double | Distance between two consecutive pulses (units) |
| – | Width | double | Width of pulse enable signal (units) |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |

### 3.2.4.13 PositionerCompensationFrequencyNotchsGet

#### Name

**PositionerCompensationFrequencyNotchsGet** – Gets pre-feedforward compensation notch filters parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 corrector type: ERR_WRONG_OBJECT_TYPE (-8)

#### Description

This functions returns the *CompensationSystemPreFeedForward* frequency notch filters parameters. These notch filters allow the user to reduce external perturbations such as base motion or floor vibrations. Note that the *CompensationSystemPreFeedForward* feature is available for all corrector types (acceleration, velocity, voltage or position) functioning *in closed loop configuration*.

- NotchFrequency1
- NotchsBandwidth1
- NotchsGain1
- NotchFrequency2
- NotchsBandwidth2
- NotchsGain2
- NotchFrequency3
- NotchsBandwidth3
- NotchsGain3

---

**NOTE**

**This function can be used only with the XPS-Qn Precision Platform controller.**

---

#### 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

**PositionerCompensationFrequencyNotchsGet** $SocketID $FullPositionerName NotchFrequency1 NotchBandwidth1 NotchGain1 NotchFrequency2 NotchBandwidth2 NotchGain2 NotchFrequency3 NotchBandwidth3 NotchGain3

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Output parameters

| | | |
|---|---|---|
| – NotchFrequency1 | double | Notch frequency for filter #1 (Hz) |
| – NotchBandwidth1 | double | Notch bandwidth for filter #1 (Hz) |
| – NotchGain1 | double | Notch gain for filter #1 |
| – NotchFrequency2 | double | Notch frequency for filter #2 (Hz) |
| – NotchBandwidth2 | double | Notch bandwidth for filter #2 (Hz) |
| – NotchGain2 | double | Notch gain for filter #2 |
| – NotchFrequency3 | double | Notch frequency for filter #3 (Hz) |
| – NotchBandwidth3 | double | Notch bandwidth for filter #3 (Hz) |
| – NotchGain3 | double | Notch gain for filter #3 |

### Return

| | | |
|---|---|---|
| – Error | int | TCL error code (0 = success or 1 = syntax error) or function error code |

## C/C++

### Prototype

int **PositionerCompensationFrequencyNotchsGet** (int SocketID, char FullPositionerName [250], double* NotchFrequency1, double* NotchBandwidth1, double* NotchGain1, double* NotchFrequency2, double* NotchBandwidth2, double* NotchGain2, double* NotchFrequency3, double* NotchBandwidth3, double* NotchGain3)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | char * | Positioner name |

### Output parameters

| | | |
|---|---|---|
| – NotchFrequency1 | double * | Notch frequency for filter #1 (Hz) |
| – NotchBandwidth1 | double * | Notch bandwidth for filter #1 (Hz) |
| – NotchGain1 | double * | Notch gain for filter #1 |
| – NotchFrequency2 | double * | Notch frequency for filter #2 (Hz) |
| – NotchBandwidth2 | double * | Notch bandwidth for filter #2 (Hz) |
| – NotchGain2 | double * | Notch gain for filter #2 |
| – NotchFrequency3 | double * | Notch frequency for filter #3 (Hz) |
| – NotchBandwidth3 | double * | Notch bandwidth for filter #3 (Hz) |
| – NotchGain3 | double * | Notch gain for filter #3 |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

## Visual Basic

### Prototype

Long **PositionerCompensationFrequencyNotchsGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, NotchFrequency1 As Double, NotchBandwidth1 As Double, NotchGain1 As Double, NotchFrequency2 As Double, NotchBandwidth2 As Double, NotchGain2 As Double, NotchFrequency3 As Double, NotchBandwidth3 As Double, NotchGain3 As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Output parameters

| | | |
|---|---|---|
| – NotchFrequency1 | double | Notch frequency for filter #1 (Hz) |
| – NotchBandwidth1 | double | Notch bandwidth for filter #1 (Hz) |
| – NotchGain1 | double | Notch gain for filter #1 |
| – NotchFrequency2 | double | Notch frequency for filter #2 (Hz) |
| – NotchBandwidth2 | double | Notch bandwidth for filter #2 (Hz) |
| – NotchGain2 | double | Notch gain for filter #2 |
| – NotchFrequency3 | double | Notch frequency for filter #3 (Hz) |
| – NotchBandwidth3 | double | Notch bandwidth for filter #3 (Hz) |
| – NotchGain3 | double | Notch gain for filter #3 |

### Return

| | | |
|---|---|---|
| – Error | long | Function error code |

## Matlab

### Prototype

[Error, NotchFrequency1, NotchBandwidth1, NotchGain1, NotchFrequency2, NotchBandwidth2, NotchGain2, NotchFrequency3, NotchBandwidth3, NotchGain3] **PositionerCompensationFrequencyNotchsGet** (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 |
| – NotchFrequency1 | double | Notch frequency for filter #1 (Hz) |
| – NotchBandwidth1 | double | Notch bandwidth for filter #1 (Hz) |
| – NotchGain1 | double | Notch gain for filter #1 |
| – NotchFrequency2 | double | Notch frequency for filter #2 (Hz) |
| – NotchBandwidth2 | double | Notch bandwidth for filter #2 (Hz) |
| – NotchGain2 | double | Notch gain for filter #2 |
| – NotchFrequency3 | double | Notch frequency for filter #3 (Hz) |
| – NotchBandwidth3 | double | Notch bandwidth for filter #3 (Hz) |
| – NotchGain3 | double | Notch gain for filter #3 |

## Python

### Prototype

[Error, NotchFrequency1, NotchBandwidth1, NotchGain1, NotchFrequency2, NotchBandwidth2, NotchGain2, NotchFrequency3, NotchBandwidth3, NotchGain3] **PositionerCompensationFrequencyNotchsGet** (integer SocketID, string FullPositionerName)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |
| – NotchFrequency1 | double | Notch frequency for filter #1 (Hz) |
| – NotchBandwidth1 | double | Notch bandwidth for filter #1 (Hz) |
| – NotchGain1 | double | Notch gain for filter #1 |
| – NotchFrequency2 | double | Notch frequency for filter #2 (Hz) |
| – NotchBandwidth2 | double | Notch bandwidth for filter #2 (Hz) |
| – NotchGain2 | double | Notch gain for filter #2 |
| – NotchFrequency3 | double | Notch frequency for filter #3 (Hz) |
| – NotchBandwidth3 | double | Notch bandwidth for filter #3 (Hz) |

| | | | |
|---|---|---|---|
| – | NotchGain3 | double | Notch gain for filter #3 |

### 3.2.4.14 PositionerCompensationFrequencyNotchsSet

#### Name

**PositionerCompensationFrequencyNotchsSet** – Sets pre-feedforward compensation notch filters parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 corrector type: ERR_WRONG_OBJECT_TYPE (-8)
- Check parameter values: ERR_PARAMETER_OUT_OF_RANGE (-17)
- NotchFrequency $\in \left[0 : \frac{0.5}{\text{CorrectorISRPeriod}}\right]$
- NotchBandwidth $\in \left[0 : \frac{0.5}{\text{CorrectorISRPeriod}}\right]$

**NOTE**

**Refer to *system.ref* file to get CorrectorISRPeriod value.**

**This function can be used only with the XPS-Q8 Precision Platform controller.**

#### Description

This functions sets the *CompensationSystemPreFeedForward* frequency notch filters parameters. These notch filters all the user to reduce external perturbations such as base motion or floor vibrations. Note that the *CompensationSystemPreFeedForward* feature is available for all corrector types (acceleration, velocity, voltage or position) functioning *in closed loop configuration*.

NotchFrequency1

NotchsBandwidth1

NotchsGain1

NotchFrequency2

NotchsBandwidth2

NotchsGain2

NotchFrequency3

NotchsBandwidth3

NotchsGain3

**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**

**PositionerCompensationFrequencyNotchsSet** $SocketID $FullPositionerName $NotchFrequency1 $NotchBandwidth1 $NotchGain1 $NotchFrequency2 $NotchBandwidth2 $NotchGain2 $NotchFrequency3 $NotchBandwidth3 $NotchGain3

**Input parameters**

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – NotchFrequency1 | double | Notch frequency for filter #1 (Hz) |
| – NotchBandwidth1 | double | Notch bandwidth for filter #1 (Hz) |
| – NotchGain1 | double | Notch gain for filter #1 |
| – NotchFrequency2 | double | Notch frequency for filter #2 (Hz) |
| – NotchBandwidth2 | double | Notch bandwidth for filter #2 (Hz) |
| – NotchGain2 | double | Notch gain for filter #2 |
| – NotchFrequency3 | double | Notch frequency for filter #3 (Hz) |
| – NotchBandwidth3 | double | Notch bandwidth for filter #3 (Hz) |
| – NotchGain3 | double | Notch gain for filter #3 |

**Output parameters**

– None

**Return**

| | | |
|---|---|---|
| – Error | int | TCL error code (0 = success or 1 = syntax error) or function error code |

## C/C++

### Prototype

int **PositionerCompensationFrequencyNotchsSet** (int SocketID, char FullPositionerName [250], double NotchFrequency1, double NotchBandwidth1, double NotchGain1, double NotchFrequency2, double NotchBandwidth2, double NotchGain2, double NotchFrequency3, double NotchBandwidth3, double NotchGain3)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | char * | Positioner name |
| – NotchFrequency1 | double | Notch frequency for filter #1 (Hz) |
| – NotchBandwidth1 | double | Notch bandwidth for filter #1 (Hz) |
| – NotchGain1 | double | Notch gain for filter #1 |
| – NotchFrequency2 | double | Notch frequency for filter #2 (Hz) |
| – NotchBandwidth2 | double | Notch bandwidth for filter #2 (Hz) |
| – NotchGain2 | double | Notch gain for filter #2 |
| – NotchFrequency3 | double | Notch frequency for filter #3 (Hz) |
| – NotchBandwidth3 | double | Notch bandwidth for filter #3 (Hz) |
| – NotchGain3 | double | Notch gain for filter #3 |

### Output parameters

– None

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

## Visual Basic

### Prototype

Long **PositionerCompensationFrequencyNotchsSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal NotchFrequency1 As Double, ByVal NotchBandwidth1 As Double, ByVal NotchGain1 As Double, ByVal NotchFrequency2 As Double, ByVal NotchBandwidth2 As Double, ByVal NotchGain2 As Double, ByVal NotchFrequency3 As Double, ByVal NotchBandwidth3 As Double, ByVal NotchGain3 As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – NotchFrequency1 | double | Notch frequency for filter #1 (Hz) |
| – NotchBandwidth1 | double | Notch bandwidth for filter #1 (Hz) |
| – NotchGain1 | double | Notch gain for filter #1 |
| – NotchFrequency2 | double | Notch frequency for filter #2 (Hz) |
| – NotchBandwidth2 | double | Notch bandwidth for filter #2 (Hz) |
| – NotchGain2 | double | Notch gain for filter #2 |
| – NotchFrequency3 | double | Notch frequency for filter #3 (Hz) |
| – NotchBandwidth3 | double | Notch bandwidth for filter #3 (Hz) |
| – NotchGain3 | double | Notch gain for filter #3 |

### Output parameters

– None

### Return

| | | |
|---|---|---|
| – Error | long | Function error code |

## Matlab

### Prototype

[Error] **PositionerCompensationFrequencyNotchsSet** (int32 SocketID, cstring FullPositionerName, double NotchFrequency1, double NotchBandwidth1, double NotchGain1, double NotchFrequency2, double NotchBandwidth2, double NotchGain2, double NotchFrequency3, double NotchBandwidth3, double NotchGain3)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | cstring | Positioner name |
| – NotchFrequency1 | double | Notch frequency for filter #1 (Hz) |
| – NotchBandwidth1 | double | Notch bandwidth for filter #1 (Hz) |
| – NotchGain1 | double | Notch gain for filter #1 |
| – NotchFrequency2 | double | Notch frequency for filter #2 (Hz) |
| – NotchBandwidth2 | double | Notch bandwidth for filter #2 (Hz) |
| – NotchGain2 | double | Notch gain for filter #2 |
| – NotchFrequency3 | double | Notch frequency for filter #3 (Hz) |
| – NotchBandwidth3 | double | Notch bandwidth for filter #3 (Hz) |
| – NotchGain3 | double | Notch gain for filter #3 |

### Return

| | | |
|---|---|---|
| – Error | int32 | Function error code |

## Python

### Prototype

[Error] **PositionerCompensationFrequencyNotchsSet** (integer SocketID, string FullPositionerName, double NotchFrequency1, double NotchBandwidth1, double NotchGain1, double NotchFrequency2, double NotchBandwidth2, double NotchGain2, double NotchFrequency3, double NotchBandwidth3, double NotchGain3)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – NotchFrequency1 | double | Notch frequency for filter #1 (Hz) |
| – NotchBandwidth1 | double | Notch bandwidth for filter #1 (Hz) |
| – NotchGain1 | double | Notch gain for filter #1 |
| – NotchFrequency2 | double | Notch frequency for filter #2 (Hz) |
| – NotchBandwidth2 | double | Notch bandwidth for filter #2 (Hz) |
| – NotchGain2 | double | Notch gain for filter #2 |
| – NotchFrequency3 | double | Notch frequency for filter #3 (Hz) |
| – NotchBandwidth3 | double | Notch bandwidth for filter #3 (Hz) |
| – NotchGain3 | double | Notch gain for filter #3 |

### Return

– Error int Function error code

### 3.2.4.15 PositionerCompensationLowPassTwoFilterGet

#### Name

**PositionerCompensationLowPassTwoFilterGet** – Gets the post-feedforward compensation second order lowpass filter parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 corrector type: ERR_WRONG_OBJECT_TYPE (-8)

#### Description

This functions returns the system compensation parameters defined for the post-feedforward compensation second order low-pass filter.

CutOffFrequency

**NOTE**

**This function can be used only with the XPS-Q8 Precision Platform controller.**

#### 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

**PositionerCompensationLowPassTwoFilterGet** $SocketID $FullPositionerName CutOffFrequency

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Output parameters

| | | |
|---|---|---|
| – CutOffFrequency | double | Second order filter cut-off frequency (Herz) |

### Return

| | | |
|---|---|---|
| – Error | int | TCL error code (0 = success or 1 = syntax error) or function error code |

## C/C++

### Prototype

int **PositionerCompensationLowPassTwoFilterGet** (int SocketID, char FullPositionerName [250], double* CutOffFrequency)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | char * | Positioner name |

### Output parameters

| | | |
|---|---|---|
| – CutOffFrequency | double * | Second order filter cut-off frequency (Herz) |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

## Visual Basic

### Prototype

Long **PositionerCompensationLowPassTwoFilterGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, CutOffFrequency As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Output parameters

| | | |
|---|---|---|
| – CutOffFrequency | double | Second order filter cut-off frequency (Herz) |

### Return

| | | |
|---|---|---|
| – Error | long | Function error code |

## Matlab

### Prototype

[Error, CutOffFrequency] **PositionerCompensationLowPassTwoFilterGet** (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 |
| – CutOffFrequency | double | Second order filter cut-off frequency (Herz) |

## Python

### Prototype

[Error, CutOffFrequency] **PositionerCompensationLowPassTwoFilterGet** (integer SocketID, string FullPositionerName)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |
| – CutOffFrequency | double | Second order filter cut-off frequency (Herz) |

### 3.2.4.16 PositionerCompensationLowPassTwoFilterSet

#### Name

**PositionerCompensationLowPassTwoFilterSet** – Sets the post-feedforward compensation second order lowpass filter parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 correcor type: ERR_WRONG_OBJECT_TYPE (-8)
- Check parameter values: ERR_PARAMETER_OUT_OF_RANGE (-17)
- CutOffFrequency $\in \left[ 0 : \frac{0.5}{\text{CorrectorISRPeriod}} \right]$

**NOTE**

**Refer to *system.ref* file to get CorrectorISRPeriod value.**

#### Description

This function configures the parameters defined for the post-feedforward compensation second order low-pass filter.

CutOffFrequency

**NOTE**

**This function can be used only with the XPS-Qn Precision Platform controller.**

**If the "CutOffFrequency" value = 0 then the second order low-pass filter is not activated.**

#### 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

**PositionerCompensationLowPassTwoFilterSet** $SocketID $FullPositionerName $CutOffFrequency

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – CutOffFrequency | double | Second order filter cut-off frequency (Herz) |

### Output parameter

– None

### Return

| | | |
|---|---|---|
| – Error | int | TCL error code (0 = success or 1 = syntax error) or function error code |

## C/C++

### Prototype

int **PositionerCompensationLowPassTwoFilterSet** (int SocketID, char FullPositionerName [250], double CutOffFrequency)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | char * | Positioner name |
| – CutOffFrequency | double | Second order filter cut-off frequency (Herz) |

### Output parameter

– None

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

## Visual Basic

### Prototype

Long **PositionerCompensationLowPassTwoFilterSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal CutOffFrequency As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – CutOffFrequency | double | Second order filter cut-off frequency (Herz) |

### Output parameters

– None

### Return

| | | |
|---|---|---|
| – Error | long | Function error code |

## Matlab

### Prototype

[Error] **PositionerCompensationLowPassTwoFilterSet** (int32 SocketID, cstring FullPositionerName, double CutOffFrequency)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | cstring | Positioner name |
| – CutOffFrequency | double | Second order filter cut-off frequency (Herz) |

### Return

| | | |
|---|---|---|
| – Error | int32 | Function error code |

## Python

### Prototype

[Error] **PositionerCompensationLowPassTwoFilterSet** (integer SocketID, string FullPositionerName, double CutOffFrequency)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – CutOffFrequency | double | Second order filter cut-off frequency (Herz) |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

### 3.2.4.17 PositionerCompensationNotchModeFiltersGet

#### Name

**PositionerCompensationNotchModeFiltersGet** – Gets the post-feedforward compensation notch mode filter parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 corrector type: ERR_WRONG_OBJECT_TYPE (-8)

#### Description

This functions returns the system compensation parameters defined for two post-feedforward compensation notch mode filters.

First notch mode filter parameters:

- NotchModeFr1
- NotchModeFa1
- NotchModeZr1
- NotchModeZa1

Second notch mode filter parameters:

- NotchModeFr2
- NotchModeFa2
- NotchModeZr2
- NotchModeZa2

---

**NOTE**

**This function can be used only with the XPS-Q8 Precision Platform controller.**

---

#### 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

**PositionerCompensationNotchModeFiltersGet** $SocketID $FullPositionerName NotchModeFr1 NotchModeFa1 NotchModeZr1 NotchModeZa1 NotchModeFr2 NotchModeFa2 NotchModeZr2 NotchModeZa2

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Output parameters

| | | |
|---|---|---|
| – NotchModeFr1 | double | Resonance frequency (Herz) for notch mode filter #1 |
| – NotchModeFa1 | double | Anti-resonance frequency (Herz) for notch mode filter #1 |
| – NotchModeZr1 | double | Resonance damping factor for notch mode filter #1 |
| – NotchModeZa1 | double | Anti-resonance damping factor for notch mode filter #1 |
| – NotchModeFr2 | double | Resonance frequency (Herz) for notch mode filter #2 |
| – NotchModeFa2 | double | Anti-resonance frequency (Herz) for notch mode filter #2 |
| – NotchModeZr2 | double | Resonance damping factor for notch mode filter #2 |
| – NotchModeZa2 | double | Anti-resonance damping factor for notch mode filter #2 |

### Return

| | | |
|---|---|---|
| – Error | int | TCL error code (0 = success or 1 = syntax error) or function error code |

## C/C++

### Prototype

int **PositionerCompensationNotchModeFiltersGet** (int SocketID, char FullPositionerName [250], double* NotchModeFr1,double* NotchModeFa1, double* NotchModeZr1, double* NotchModeZa1, double* NotchModeFr2,double* NotchModeFa2, double* NotchModeZr2, double* NotchModeZa2)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | char * | Positioner name |

### Output parameters

| | | |
|---|---|---|
| – NotchModeFr1 | double * | Resonance frequency (Herz) for notch mode filter #1 |
| – NotchModeFa1 | double * | Anti-resonance frequency (Herz) for notch mode filter #1 |
| – NotchModeZr1 | double * | Resonance damping factor for notch mode filter #1 |
| – NotchModeZa1 | double * | Anti-resonance damping factor for notch mode filter #1 |
| – NotchModeFr2 | double * | Resonance frequency (Herz) for notch mode filter #2 |
| – NotchModeFa2 | double * | Anti-resonance frequency (Herz) for notch mode filter #2 |
| – NotchModeZr2 | double * | Resonance damping factor for notch mode filter #2 |
| – NotchModeZa2 | double * | Anti-resonance damping factor for notch mode filter #2 |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

## Visual Basic

### Prototype

Long **PositionerCompensationNotchModeFiltersGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, NotchModeFr1 As Double, NotchModeFa1 As Double, NotchModeZr1 As Double, NotchModeZa1 As Double, NotchModeFr2 As Double, NotchModeFa2 As Double, NotchModeZr2 As Double, NotchModeZa2 As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Output parameters

| | | |
|---|---|---|
| – NotchModeFr1 | double | Resonance frequency (Herz) for notch mode filter #1 |
| – NotchModeFa1 | double | Anti-resonance frequency (Herz) for notch mode filter #1 |
| – NotchModeZr1 | double | Resonance damping factor for notch mode filter #1 |
| – NotchModeZa1 | double | Anti-resonance damping factor for notch mode filter #1 |
| – NotchModeFr2 | double | Resonance frequency (Herz) for notch mode filter #2 |
| – NotchModeFa2 | double | Anti-resonance frequency (Herz) for notch mode filter #2 |
| – NotchModeZr2 | double | Resonance damping factor for notch mode filter #2 |
| – NotchModeZa2 | double | Anti-resonance damping factor for notch mode filter #2 |

### Return

| | | |
|---|---|---|
| – Error | long | Function error code |

## Matlab

### Prototype

[Error, NotchModeFr1, NotchModeFa1, NotchModeZr1, NotchModeZa1, NotchModeFr2, NotchModeFa2, NotchModeZr2, NotchModeZa2] **PositionerCompensationNotchModeFiltersGet** (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 |
| – NotchModeFr1 | double | Resonance frequency (Herz) for notch mode filter #1 |
| – NotchModeFa1 | double | Anti-resonance frequency (Herz) for notch mode filter #1 |
| – NotchModeZr1 | double | Resonance damping factor for notch mode filter #1 |
| – NotchModeZa1 | double | Anti-resonance damping factor for notch mode filter #1 |
| – NotchModeFr2 | double | Resonance frequency (Herz) for notch mode filter #2 |
| – NotchModeFa2 | double | Anti-resonance frequency (Herz) for notch mode filter #2 |
| – NotchModeZr2 | double | Resonance damping factor for notch mode filter #2 |
| – NotchModeZa2 | double | Anti-resonance damping factor for notch mode filter #2 |

## Python

### Prototype

[Error, NotchModeFr1, NotchModeFa1, NotchModeZr1, NotchModeZa1, NotchModeFr2, NotchModeFa2, NotchModeZr2, NotchModeZa2] **PositionerCompensationNotchModeFiltersGet** (integer SocketID, string FullPositionerName)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |
| – | NotchModeFr1 | double | Resonance frequency (Herz) for notch mode filter #1 |
| – | NotchModeFa1 | double | Anti-resonance frequency (Herz) for notch mode filter #1 |
| – | NotchModeZr1 | double | Resonance damping factor for notch mode filter #1 |
| – | NotchModeZa1 | double | Anti-resonance damping factor for notch mode filter #1 |
| – | NotchModeFr2 | double | Resonance frequency (Herz) for notch mode filter #2 |
| – | NotchModeFa2 | double | Anti-resonance frequency (Herz) for notch mode filter #2 |
| – | NotchModeZr2 | double | Resonance damping factor for notch mode filter #2 |
| – | NotchModeZa2 | double | Anti-resonance damping factor for notch mode filter #2 |

### 3.2.4.18 PositionerCompensationNotchModeFiltersSet

#### Name

**PositionerCompensationNotchModeFiltersSet** – Sets the post-feedforward compensation notch mode filter parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 corrector type: ERR_WRONG_OBJECT_TYPE (-8)
- Check parameter values: ERR_PARAMETER_OUT_OF_RANGE (-17)
- NotchModeFr $\in \left[ 0 : \frac{0.5}{\text{CorrectorISRPeriod}} \right]$
- NotchModeFa $\in \left[ 0 : \frac{0.5}{\text{CorrectorISRPeriod}} \right]$

**NOTE**

**This function can be used only with the XPS-Q8 Precision Platform controller.**

**Refer to *system.ref* file to get CorrectorISRPeriod value.**

#### Description

This functions configures the parameters defined for two post-feedforward compensation notch mode filters.

First notch mode filter parameters:

- NotchModeFr1
- NotchModeFa1
- NotchModeZr1
- NotchModeZa1

Second notch mode filter parameters:

- NotchModeFr2
- NotchModeFa2
- NotchModeZr2
- NotchModeZa2

**NOTE**

**If the "NotchModeFr" value = 0 or the "NotchModeFa" value = 0, then the notch mode filter is not activated.**

### 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

**PositionerCompensationNotchModeFiltersSet** $SocketID $FullPositionerName $NotchModeFr1 $NotchModeFa1 $NotchModeZr1 $NotchModeZa1 $NotchModeFr2 $NotchModeFa2 $NotchModeZr2 $NotchModeZa2

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – NotchModeFr1 | double | Resonance frequency (Herz) for notch mode filter #1 |
| – NotchModeFa1 | double | Anti-resonance frequency (Herz) for notch mode filter #1 |
| – NotchModeZr1 | double | Resonance damping factor for notch mode filter #1 |
| – NotchModeZa1 | double | Anti-resonance damping factor for notch mode filter #1 |
| – NotchModeFr2 | double | Resonance frequency (Herz) for notch mode filter #2 |
| – NotchModeFa2 | double | Anti-resonance frequency (Herz) for notch mode filter #2 |
| – NotchModeZr2 | double | Resonance damping factor for notch mode filter #2 |
| – NotchModeZa2 | double | Anti-resonance damping factor for notch mode filter #2 |

### Output parameter

– None

### Return

| | | |
|---|---|---|
| – Error | int | TCL error code (0 = success or 1 = syntax error) or function error code |

## C/C++

### Prototype

int **PositionerCompensationNotchModeFiltersSet** (int SocketID, char FullPositionerName [250], double NotchModeFr1,double NotchModeFa1, double NotchModeZr1, double NotchModeZa1, double NotchModeFr2,double NotchModeFa2, double NotchModeZr2, double NotchModeZa2)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | char * | Positioner name |
| – NotchModeFr1 | double | Resonance frequency (Herz) for notch mode filter #1 |
| – NotchModeFa1 | double | Anti-resonance frequency (Herz) for notch mode filter #1 |
| – NotchModeZr1 | double | Resonance damping factor for notch mode filter #1 |
| – NotchModeZa1 | double | Anti-resonance damping factor for notch mode filter #1 |
| – NotchModeFr2 | double | Resonance frequency (Herz) for notch mode filter #2 |
| – NotchModeFa2 | double | Anti-resonance frequency (Herz) for notch mode filter #2 |
| – NotchModeZr2 | double | Resonance damping factor for notch mode filter #2 |
| – NotchModeZa2 | double | Anti-resonance damping factor for notch mode filter #2 |

### Output parameter

– None

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

## Visual Basic

### Prototype

Long **PositionerCompensationNotchModeFiltersSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal NotchModeFr1 As Double, ByVal NotchModeFa1 As Double, ByVal NotchModeZr1 As Double, ByVal NotchModeZa1 As Double, ByVal NotchModeF2 As Double, ByVal NotchModeFa2 As Double, ByVal NotchModeZr2 As Double, ByVal NotchModeZa2 As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – NotchModeFr1 | double | Resonance frequency (Herz) for notch mode filter #1 |
| – NotchModeFa1 | double | Anti-resonance frequency (Herz) for notch mode filter #1 |
| – NotchModeZr1 | double | Resonance damping factor for notch mode filter #1 |
| – NotchModeZa1 | double | Anti-resonance damping factor for notch mode filter #1 |
| – NotchModeFr2 | double | Resonance frequency (Herz) for notch mode filter #2 |
| – NotchModeFa2 | double | Anti-resonance frequency (Herz) for notch mode filter #2 |
| – NotchModeZr2 | double | Resonance damping factor for notch mode filter #2 |
| – NotchModeZa2 | double | Anti-resonance damping factor for notch mode filter #2 |

### Output parameter

– None

### Return

| | | |
|---|---|---|
| – Error | long | Function error code |

## Matlab

### Prototype

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

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | cstring | Positioner name |
| – NotchModeFr1 | double | Resonance frequency (Herz) for notch mode filter #1 |
| – NotchModeFa1 | double | Anti-resonance frequency (Herz) for notch mode filter #1 |
| – NotchModeZr1 | double | Resonance damping factor for notch mode filter #1 |
| – NotchModeZa1 | double | Anti-resonance damping factor for notch mode filter #1 |
| – NotchModeFr2 | double | Resonance frequency (Herz) for notch mode filter #2 |
| – NotchModeFa2 | double | Anti-resonance frequency (Herz) for notch mode filter #2 |
| – NotchModeZr2 | double | Resonance damping factor for notch mode filter #2 |
| – NotchModeZa2 | double | Anti-resonance damping factor for notch mode filter #2 |

### Return

| | | |
|---|---|---|
| – Error | int32 | Function error code |

## Python

### Prototype

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

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |
| – | NotchModeFr1 | double | Resonance frequency (Herz) for notch mode filter #1 |
| – | NotchModeFa1 | double | Anti-resonance frequency (Herz) for notch mode filter #1 |
| – | NotchModeZr1 | double | Resonance damping factor for notch mode filter #1 |
| – | NotchModeZa1 | double | Anti-resonance damping factor for notch mode filter #1 |
| – | NotchModeFr2 | double | Resonance frequency (Herz) for notch mode filter #2 |
| – | NotchModeFa2 | double | Anti-resonance frequency (Herz) for notch mode filter #2 |
| – | NotchModeZr2 | double | Resonance damping factor for notch mode filter #2 |
| – | NotchModeZa2 | double | Anti-resonance damping factor for notch mode filter #2 |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |

### 3.2.4.19 PositionerCompensationPhaseCorrectionFiltersGet

#### Name

**PositionerCompensationPhaseCorrectionFiltersGet** – Gets the post-feedforward compensation phase correction filter parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 corrector type: ERR_WRONG_OBJECT_TYPE (-8)

#### Description

This function returns the system compensation parameters defined for two post-feedforward compensation phase correction filters.

First phase correction filter parameters:

- PhaseCorrectionFn1
- PhaseCorrectionFd1
- PhaseCorrectionGain1

Second phase correction filter parameters:

- PhaseCorrectionFn2
- PhaseCorrectionFd2
- PhaseCorrectionGain2

---

**NOTE**

**This function can be used only with the XPS-Q8 Precision Platform controller.**

---

#### 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

**PositionerCompensationPhaseCorrectionFiltersGet** $SocketID $FullPositionerName PhaseCorrectionFn1 PhaseCorrectionFd1 PhaseCorrectionGain1 PhaseCorrectionFn2 PhaseCorrectionFd2 PhaseCorrectionGain2

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Output parameters

| | | |
|---|---|---|
| – PhaseCorrectionFn1 | double | Numerator frequency (Herz) for phase correction filter #1 |
| – PhaseCorrectionFd1 | double | Denominator frequency (Herz) for phase correction filter #1 |
| – PhaseCorrectionGain1 | double | Gain for phase correction filter #1 |
| – PhaseCorrectionFn2 | double | Numerator frequency (Herz) for phase correction filter #2 |
| – PhaseCorrectionFd2 | double | Denominator frequency (Herz) for phase correction filter #2 |
| – PhaseCorrectionGain2 | double | Gain for phase correction filter #2 |

### Return

| | | |
|---|---|---|
| – Error | int | TCL error code (0 = success or 1 = syntax error) or function error code |

## C/C++

### Prototype

int **PositionerCompensationPhaseCorrectionFiltersGet** (int SocketID, char FullPositionerName [250], double* PhaseCorrectionFn1,double* PhaseCorrectionFd1, double* PhaseCorrectionGain1, double* PhaseCorrectionFn2,double* PhaseCorrectionFd2, double* PhaseCorrectionGain2)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | char * | Positioner name |

### Output parameters

| | | |
|---|---|---|
| – PhaseCorrectionFn1 | double * | Numerator frequency (Herz) for phase correction filter #1 |
| – PhaseCorrectionFd1 | double * | Denominator frequency (Herz) for phase correction filter #1 |
| – PhaseCorrectionGain1 | double * | Gain for phase correction filter #1 |
| – PhaseCorrectionFn2 | double * | Numerator frequency (Herz) for phase correction filter #2 |
| – PhaseCorrectionFd2 | double * | Denominator frequency (Herz) for phase correction filter #2 |
| – PhaseCorrectionGain2 | double * | Gain for phase correction filter #2 |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

## Visual Basic

### Prototype

Long **PositionerCompensationPhaseCorrectionFiltersGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, PhaseCorrectionFn1 As Double, PhaseCorrectionFd1 As Double, PhaseCorrectionGain1 As Double, PhaseCorrectionFn2 As Double, PhaseCorrectionFd2 As Double, PhaseCorrectionGain2 As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Output parameters

| | | |
|---|---|---|
| – PhaseCorrectionFn1 | double | Numerator frequency (Herz) for phase correction filter #1 |
| – PhaseCorrectionFd1 | double | Denominator frequency (Herz) for phase correction filter #1 |
| – PhaseCorrectionGain1 | double | Gain for phase correction filter #1 |
| – PhaseCorrectionFn2 | double | Numerator frequency (Herz) for phase correction filter #2 |
| – PhaseCorrectionFd2 | double | Denominator frequency (Herz) for phase correction filter #2 |
| – PhaseCorrectionGain2 | double | Gain for phase correction filter #2 |

### Return

| | | |
|---|---|---|
| – Error | long | Function error code |

## Matlab

### Prototype

[Error, PhaseCorrectionFn1, PhaseCorrectionFd1, PhaseCorrectionGain1, PhaseCorrectionFn2, PhaseCorrectionFd2, PhaseCorrectionGain2]
**PositionerCompensationPhaseCorrectionFiltersGet** (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 |
| – PhaseCorrectionFn1 | double | Numerator frequency (Herz) for phase correction filter #1 |
| – PhaseCorrectionFd1 | double | Denominator frequency (Herz) for phase correction filter #1 |
| – PhaseCorrectionGain1 | double | Gain for phase correction filter #1 |
| – PhaseCorrectionFn2 | double | Numerator frequency (Herz) for phase correction filter #2 |
| – PhaseCorrectionFd2 | double | Denominator frequency (Herz) for phase correction filter #2 |
| – PhaseCorrectionGain2 | double | Gain for phase correction filter #2 |

## Python

### Prototype

[Error, PhaseCorrectionFn1, PhaseCorrectionFd1, PhaseCorrectionGain1, PhaseCorrectionFn2, PhaseCorrectionFd2, PhaseCorrectionGain2]
**PositionerCompensationPhaseCorrectionFiltersGet** (integer SocketID, string FullPositionerName)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |
| – PhaseCorrectionFn1 | double | Numerator frequency (Herz) for phase correction filter #1 |
| – PhaseCorrectionFd1 | double | Denominator frequency (Herz) for phase correction filter #1 |
| – PhaseCorrectionGain1 | double | Gain for phase correction filter #1 |
| – PhaseCorrectionFn2 | double | Numerator frequency (Herz) for phase correction filter #2 |
| – PhaseCorrectionFd2 | double | Denominator frequency (Herz) for phase correction filter #2 |
| – PhaseCorrectionGain2 | double | Gain for phase correction filter #2 |

### 3.2.4.20 PositionerCompensationPhaseCorrectionFiltersSet

#### Name

**PositionerCompensationPhaseCorrectionFiltersSet** – Sets the post-feedforward compensation phase correction filter parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 corrector type: ERR_WRONG_OBJECT_TYPE (-8)
- Check parameter values: ERR_PARAMETER_OUT_OF_RANGE (-17)
- PhaseCorrectionFn $\in \left[ 0 : \frac{0.5}{\text{CorrectorISRPeriod}} \right]$
- PhaseCorrectionFd $\in \left[ 0 : \frac{0.5}{\text{CorrectorISRPeriod}} \right]$

**NOTE**

**This function can be used only with the XPS-Qn Precision Platform controller.**

**Refer to *system.ref* file to get CorrectorISRPeriod value.**

#### Description

This functions configures the parameters defined for two post-feedforward compensation phase correction filters.

First phase correction filter parameters:

- PhaseCorrectionFn1
- PhaseCorrectionFd1
- PhaseCorrectionGain1

Second phase correction filter parameters:

- PhaseCorrectionFn2
- PhaseCorrectionFd2
- PhaseCorrectionGain2

**NOTE**

**If the "PhaseCorrectionFn" value = 0 or the "PhaseCorrectionFd" value = 0 then the phase correction filter is not activated.**

**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**

**PositionerCompensationPhaseCorrectionFiltersSet** $SocketID $FullPositionerName $PhaseCorrectionFn1 $PhaseCorrectionFd1 $PhaseCorrectionGain1 $PhaseCorrectionFn2 $PhaseCorrectionFd2 $PhaseCorrectionGain2

**Input parameters**

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – PhaseCorrectionFn1 | double | Numerator frequency (Herz) for phase correction filter #1 |
| – PhaseCorrectionFd1 | double | Denominator frequency (Herz) for phase correction filter #1 |
| – PhaseCorrectionGain1 | double | Gain for phase correction filter #1 |
| – PhaseCorrectionFn2 | double | Numerator frequency (Herz) for phase correction filter #2 |
| – PhaseCorrectionFd2 | double | Denominator frequency (Herz) for phase correction filter #2 |
| – PhaseCorrectionGain2 | double | Gain for phase correction filter #2 |

**Output parameter**

– None

**Return**

| | | |
|---|---|---|
| – Error | int | TCL error code (0 = success or 1 = syntax error) or function error code |

## C/C++

### Prototype

int **PositionerCompensationPhaseCorrectionFiltersSet** (int SocketID, char FullPositionerName [250], double PhaseCorrectionFn1,double PhaseCorrectionFd1, double PhaseCorrectionGain1, double PhaseCorrectionFn2,double PhaseCorrectionFd2, double PhaseCorrectionGain2)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | char * | Positioner name |
| – | PhaseCorrectionFn1 | double | Numerator frequency (Herz) for phase correction filter #1 |
| – | PhaseCorrectionFd1 | double | Denominator frequency (Herz) for phase correction filter #1 |
| – | PhaseCorrectionGain1 | double | Gain for phase correction filter #1 |
| – | PhaseCorrectionFn2 | double | Numerator frequency (Herz) for phase correction filter #2 |
| – | PhaseCorrectionFd2 | double | Denominator frequency (Herz) for phase correction filter #2 |
| – | PhaseCorrectionGain2 | double | Gain for phase correction filter #2 |

### Output parameter

– None

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |

## Visual Basic

### Prototype

Long **PositionerCompensationPhaseCorrectionFiltersSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal PhaseCorrectionFn1 As Double, ByVal PhaseCorrectionFd1 As Double, ByVal PhaseCorrectionGain1 As Double, ByVal PhaseCorrectionFn2 As Double, ByVal PhaseCorrectionFd2 As Double, ByVal PhaseCorrectionGain2 As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – PhaseCorrectionFn1 | double | Numerator frequency (Herz) for phase correction filter #1 |
| – PhaseCorrectionFd1 | double | Denominator frequency (Herz) for phase correction filter #1 |
| – PhaseCorrectionGain1 | double | Gain for phase correction filter #1 |
| – PhaseCorrectionFn2 | double | Numerator frequency (Herz) for phase correction filter #2 |
| – PhaseCorrectionFd2 | double | Denominator frequency (Herz) for phase correction filter #2 |
| – PhaseCorrectionGain2 | double | Gain for phase correction filter #2 |

### Output parameter

– None

### Return

| | | |
|---|---|---|
| – Error | long | Function error code |

## Matlab

### Prototype

[Error] **PositionerCompensationPhaseCorrectionFiltersSet** (int32 SocketID, cstring FullPositionerName, double PhaseCorrectionFn1,double PhaseCorrectionFd1, double PhaseCorrectionGain1, double PhaseCorrectionFn2,double PhaseCorrectionFd2, double PhaseCorrectionGain2)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | cstring | Positioner name |
| – PhaseCorrectionFn1 | double | Numerator frequency (Herz) for phase correction filter #1 |
| – PhaseCorrectionFd1 | double | Denominator frequency (Herz) for phase correction filter #1 |
| – PhaseCorrectionGain1 | double | Gain for phase correction filter #1 |
| – PhaseCorrectionFn2 | double | Numerator frequency (Herz) for phase correction filter #2 |
| – PhaseCorrectionFd2 | double | Denominator frequency (Herz) for phase correction filter #2 |
| – PhaseCorrectionGain2 | double | Gain for phase correction filter #2 |

### Return

| | | |
|---|---|---|
| – Error | int32 | Function error code |

## Python

### Prototype

[Error] **PositionerCompensationPhaseCorrectionFiltersSet** (integer SocketID, string FullPositionerName, double PhaseCorrectionFn1,double PhaseCorrectionFd1, double PhaseCorrectionGain1, double PhaseCorrectionFn2,double PhaseCorrectionFd2, double PhaseCorrectionGain2)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – PhaseCorrectionFn1 | double | Numerator frequency (Herz) for phase correction filter #1 |
| – PhaseCorrectionFd1 | double | Denominator frequency (Herz) for phase correction filter #1 |
| – PhaseCorrectionGain1 | double | Gain for phase correction filter #1 |
| – PhaseCorrectionFn2 | double | Numerator frequency (Herz) for phase correction filter #2 |
| – PhaseCorrectionFd2 | double | Denominator frequency (Herz) for phase correction filter #2 |
| – PhaseCorrectionGain2 | double | Gain for phase correction filter #2 |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

### 3.2.4.21 PositionerCompensationSpatialPeriodicNotchsGet

#### Name

**PositionerCompensationSpatialPeriodicNotchsGet** – Gets pre-feedforward compensation spatial periodic filters parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 corrector type: ERR_WRONG_OBJECT_TYPE (-8)

#### Description

This functions returns the *CompensationSystemPreFeedForward* spatial periodic filters parameters. These filters reduce the spatial periodic perturbations coming from screw pitch or cogging.

Note that the *CompensationSystemPreFeedForward* feature is available for all corrector types (acceleration, velocity, voltage or position) functioning *in closed loop configuration*.

- SpatialNotchStep1
- SpatialNotchsBandwidth1
- SpatialNotchsGain1
- SpatialNotchStep2
- SpatialNotchsBandwidth2
- SpatialNotchsGain2
- SpatialNotchStep3
- SpatialNotchsBandwidth3
- SpatialNotchsGain3

---

**NOTE**

**This function can be used only with the XPS-Qn Precision Platform controller.**

---

#### 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

**PositionerCompensationSpatialPeriodicNotchsGet** $SocketID $FullPositionerName SpatialNotchStep1 SpatialNotchBandwidth1 SpatialNotchGain1 SpatialNotchStep2 SpatialNotchBandwidth2 SpatialNotchGain2 SpatialNotchStep3 SpatialNotchBandwidth3 SpatialNotchGain3

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Output parameters

| | | |
|---|---|---|
| – SpatialNotchStep1 | double | Spatial periodic step for filter #1 (units) |
| – SpatialNotchBandwidth1 | double | Spatial periodic bandwidth for filter #1 (Hz) |
| – SpatialNotchGain1 | double | Spatial periodic gain for filter #1 |
| – SpatialNotchStep2 | double | Spatial periodic step for filter #2 (units) |
| – SpatialNotchBandwidth2 | double | Spatial periodic bandwidth for filter #2 (Hz) |
| – SpatialNotchGain2 | double | Spatial periodic gain for filter #2 |
| – SpatialNotchStep3 | double | Spatial periodic step for filter #3 (units) |
| – SpatialNotchBandwidth3 | double | Spatial periodic bandwidth for filter #3 (Hz) |
| – SpatialNotchGain3 | double | Spatial periodic gain for filter #3 |

### Return

| | | |
|---|---|---|
| – Error | int | TCL error code (0 = success or 1 = syntax error) or function error code |

## C/C++

### Prototype

int **PositionerCompensationSpatialPeriodicNotchsGet** (int SocketID, char FullPositionerName [250], double* SpatialNotchStep1, double* SpatialNotchBandwidth1, double* SpatialNotchGain1, double* SpatialNotchStep2, double* SpatialNotchBandwidth2, double* SpatialNotchGain2, double* SpatialNotchStep3, double* SpatialNotchBandwidth3, double* SpatialNotchGain3)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | char * | Positioner name |

### Output parameters

| | | |
|---|---|---|
| – SpatialNotchStep1 | double * | Spatial periodic step for filter #1 (units) |
| – SpatialNotchBandwidth1 | double * | Spatial periodic bandwidth for filter #1 (Hz) |
| – SpatialNotchGain1 | double * | Spatial periodic gain for filter #1 |
| – SpatialNotchStep2 | double * | Spatial periodic step for filter #2 (units) |
| – SpatialNotchBandwidth2 | double * | Spatial periodic bandwidth for filter #2 (Hz) |
| – SpatialNotchGain2 | double * | Spatial periodic gain for filter #2 |
| – SpatialNotchStep3 | double * | Spatial periodic step for filter #3 (units) |
| – SpatialNotchBandwidth3 | double * | Spatial periodic bandwidth for filter #3 (Hz) |
| – SpatialNotchGain3 | double * | Spatial periodic gain for filter #3 |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

## Visual Basic

### Prototype

Long **PositionerCompensationSpatialPeriodicNotchsGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, SpatialNotchStep1 As Double, SpatialNotchBandwidth1 As Double, SpatialNotchGain1 As Double, SpatialNotchStep2 As Double, SpatialNotchBandwidth2 As Double, SpatialNotchGain2 As Double, SpatialNotchStep3 As Double, SpatialNotchBandwidth3 As Double, SpatialNotchGain3 As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Output parameters

| | | |
|---|---|---|
| – SpatialNotchStep1 | double | Spatial periodic step for filter #1 (units) |
| – SpatialNotchBandwidth1 | double | Spatial periodic bandwidth for filter #1 (Hz) |
| – SpatialNotchGain1 | double | Spatial periodic gain for filter #1 |
| – SpatialNotchStep2 | double | Spatial periodic step for filter #2 (units) |
| – SpatialNotchBandwidth2 | double | Spatial periodic bandwidth for filter #2 (Hz) |
| – SpatialNotchGain2 | double | Spatial periodic gain for filter #2 |
| – SpatialNotchStep3 | double | Spatial periodic step for filter #3 (units) |
| – SpatialNotchBandwidth3 | double | Spatial periodic bandwidth for filter #3 (Hz) |
| – SpatialNotchGain3 | double | Spatial periodic gain for filter #3 |

### Return

| | | |
|---|---|---|
| – Error | long | Function error code |

## Matlab

### Prototype

[Error, SpatialNotchStep1, SpatialNotchBandwidth1, SpatialNotchGain1, SpatialNotchStep2, SpatialNotchBandwidth2, SpatialNotchGain2, SpatialNotchStep3, SpatialNotchBandwidth3, SpatialNotchGain3]
**PositionerCompensationSpatialPeriodicNotchsGet** (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 |
| – | SpatialNotchStep1 | double | Spatial periodic step for filter #1 (units) |
| – | SpatialNotchBandwidth1 | double | Spatial periodic bandwidth for filter #1 (Hz) |
| – | SpatialNotchGain1 | double | Spatial periodic gain for filter #1 |
| – | SpatialNotchStep2 | double | Spatial periodic step for filter #2 (units) |
| – | SpatialNotchBandwidth2 | double | Spatial periodic bandwidth for filter #2 (Hz) |
| – | SpatialNotchGain2 | double | Spatial periodic gain for filter #2 |
| – | SpatialNotchStep3 | double | Spatial periodic step for filter #3 (units) |
| – | SpatialNotchBandwidth3 | double | Spatial periodic bandwidth for filter #3 (Hz) |
| – | SpatialNotchGain3 | double | Spatial periodic gain for filter #3 |

## Python

### Prototype

[Error, SpatialNotchStep1, SpatialNotchBandwidth1, SpatialNotchGain1, SpatialNotchStep2, SpatialNotchBandwidth2, SpatialNotchGain2, SpatialNotchStep3, SpatialNotchBandwidth3, SpatialNotchGain3]
**PositionerCompensationSpatialPeriodicNotchsGet** (integer SocketID, string FullPositionerName)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |
| – SpatialNotchStep1 | double | Spatial periodic step for filter #1 (units) |
| – SpatialNotchBandwidth1 | double | Spatial periodic bandwidth for filter #1 (Hz) |
| – SpatialNotchGain1 | double | Spatial periodic gain for filter #1 |
| – SpatialNotchStep2 | double | Spatial periodic step for filter #2 (units) |
| – SpatialNotchBandwidth2 | double | Spatial periodic bandwidth for filter #2 (Hz) |
| – SpatialNotchGain2 | double | Spatial periodic gain for filter #2 |
| – SpatialNotchStep3 | double | Spatial periodic step for filter #3 (units) |
| – SpatialNotchBandwidth3 | double | Spatial periodic bandwidth for filter #3 (Hz) |
| – SpatialNotchGain3 | double | Spatial periodic gain for filter #3 |

### 3.2.4.22 PositionerCompensationSpatialPeriodicNotchsSet

#### Name

**PositionerCompensationSpatialPeriodicNotchsSet** – Sets pre-feedforward compensation spatial periodic filters parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 correcor type: ERR_WRONG_OBJECT_TYPE (-8)
- Check parameter values: ERR_PARAMETER_OUT_OF_RANGE (-17)
- SpatialNotchStep $\in \left[0 : MaximumVelocity * CorrectorISRPeriod\right]$
- SpatialNotchBandwidth $\in \left[0 : \frac{0.5}{CorrectorISRPeriod}\right]$

**NOTE**

**Refer to system.ref file to get CorrectorISRPeriod and stages.ini for MaximumVelocity values.**

#### Description

This function sets the *CompensationSystemPreFeedForward* spatial periodic filters parameters. These filters reduce the spatial periodic perturbations coming from screw pitch or cogging.

Note that the *CompensationSystemPreFeedForward* feature is available for all corrector types (acceleration, velocity, voltage or position) functioning *in closed loop configuration*.

- SpatialNotchStep1
- SpatialNotchsBandwidth1
- SpatialNotchsGain1
- SpatialNotchStep2
- SpatialNotchsBandwidth2
- SpatialNotchsGain2
- SpatialNotchStep3
- SpatialNotchsBandwidth3
- SpatialNotchsGain3

**NOTE**

**This function can be used only with the XPS-Qn Precision Platform controller.**

### 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

**PositionerCompensationSpatialPeriodicNotchsSet** $SocketID $FullPositionerName $SpatialNotchStep1 $SpatialNotchBandwidth1 $SpatialNotchGain1 $SpatialNotchStep2 $SpatialNotchBandwidth2 $SpatialNotchGain2 $SpatialNotchStep3 $SpatialNotchBandwidth3 $SpatialNotchGain3

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – SpatialNotchStep1 | double | Spatial periodic step for filter #1 (units) |
| – SpatialNotchBandwidth1 | double | Spatial periodic bandwidth for filter #1 (Hz) |
| – SpatialNotchGain1 | double | Spatial periodic gain for filter #1 |
| – SpatialNotchStep2 | double | Spatial periodic step for filter #2 (units) |
| – SpatialNotchBandwidth2 | double | Spatial periodic bandwidth for filter #2 (Hz) |
| – SpatialNotchGain2 | double | Spatial periodic gain for filter #2 |
| – SpatialNotchStep3 | double | Spatial periodic step for filter #3 (units) |
| – SpatialNotchBandwidth3 | double | Spatial periodic bandwidth for filter #3 (Hz) |
| – SpatialNotchGain3 | double | Spatial periodic gain for filter #3 |

### Output parameters

– None

### Return

| | | |
|---|---|---|
| – Error | int | TCL error code (0 = success or 1 = syntax error) or function error code |

## C/C++

### Prototype

int **PositionerCompensationSpatialPeriodicNotchsSet** (int SocketID, char FullPositionerName [250], double SpatialNotchStep1, double SpatialNotchBandwidth1, double SpatialNotchGain1, double SpatialNotchStep2, double SpatialNotchBandwidth2, double SpatialNotchGain2, double SpatialNotchStep3, double SpatialNotchBandwidth3, double SpatialNotchGain3)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | char * | Positioner name |
| – SpatialNotchStep1 | double | Spatial periodic step for filter #1 (units) |
| – SpatialNotchBandwidth1 | double | Spatial periodic bandwidth for filter #1 (Hz) |
| – SpatialNotchGain1 | double | Spatial periodic gain for filter #1 |
| – SpatialNotchStep2 | double | Spatial periodic step for filter #2 (units) |
| – SpatialNotchBandwidth2 | double | Spatial periodic bandwidth for filter #2 (Hz) |
| – SpatialNotchGain2 | double | Spatial periodic gain for filter #2 |
| – SpatialNotchStep3 | double | Spatial periodic step for filter #3 (units) |
| – SpatialNotchBandwidth3 | double | Spatial periodic bandwidth for filter #3 (Hz) |
| – SpatialNotchGain3 | double | Spatial periodic gain for filter #3 |

### Output parameters

– None

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

## Visual Basic

### Prototype

Long **PositionerCompensationSpatialPeriodicNotchsSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal SpatialNotchStep1 As Double, ByVal SpatialNotchBandwidth1 As Double, ByVal SpatialNotchGain1 As Double, ByVal SpatialNotchStep2 As Double, ByVal SpatialNotchBandwidth2 As Double, ByVal SpatialNotchGain2 As Double, ByVal SpatialNotchStep3 As Double, ByVal SpatialNotchBandwidth3 As Double, ByVal SpatialNotchGain3 As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – SpatialNotchStep1 | double | Spatial periodic step for filter #1 (units) |
| – SpatialNotchBandwidth1 | double | Spatial periodic bandwidth for filter #1 (Hz) |
| – SpatialNotchGain1 | double | Spatial periodic gain for filter #1 |
| – SpatialNotchStep2 | double | Spatial periodic step for filter #2 (units) |
| – SpatialNotchBandwidth2 | double | Spatial periodic bandwidth for filter #2 (Hz) |
| – SpatialNotchGain2 | double | Spatial periodic gain for filter #2 |
| – SpatialNotchStep3 | double | Spatial periodic step for filter #3 (units) |
| – SpatialNotchBandwidth3 | double | Spatial periodic bandwidth for filter #3 (Hz) |
| – SpatialNotchGain3 | double | Spatial periodic gain for filter #3 |

### Output parameters

– None

### Return

| | | |
|---|---|---|
| – Error | long | Function error code |

## Matlab

### Prototype

[Error] **PositionerCompensationSpatialPeriodicNotchsSet** (int32 SocketID, cstring FullPositionerName, double SpatialNotchStep1, double SpatialNotchBandwidth1, double SpatialNotchGain1, double SpatialNotchStep2, double SpatialNotchBandwidth2, double SpatialNotchGain2, double SpatialNotchStep3, double SpatialNotchBandwidth3, double SpatialNotchGain3)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | cstring | Positioner name |
| – SpatialNotchStep1 | double | Spatial periodic step for filter #1 (units) |
| – SpatialNotchBandwidth1 | double | Spatial periodic bandwidth for filter #1 (Hz) |
| – SpatialNotchGain1 | double | Spatial periodic gain for filter #1 |
| – SpatialNotchStep2 | double | Spatial periodic step for filter #2 (units) |
| – SpatialNotchBandwidth2 | double | Spatial periodic bandwidth for filter #2 (Hz) |
| – SpatialNotchGain2 | double | Spatial periodic gain for filter #2 |
| – SpatialNotchStep3 | double | Spatial periodic step for filter #3 (units) |
| – SpatialNotchBandwidth3 | double | Spatial periodic bandwidth for filter #3 (Hz) |
| – SpatialNotchGain3 | double | Spatial periodic gain for filter #3 |

### Return

| | | |
|---|---|---|
| – Error | int32 | Function error code |

## Python

### Prototype

[Error] **PositionerCompensationSpatialPeriodicNotchsSet** (integer SocketID, string FullPositionerName, double SpatialNotchStep1, double SpatialNotchBandwidth1, double SpatialNotchGain1, double SpatialNotchStep2, double SpatialNotchBandwidth2, double SpatialNotchGain2, double SpatialNotchStep3, double SpatialNotchBandwidth3, double SpatialNotchGain3)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – SpatialNotchStep1 | double | Spatial periodic step for filter #1 (units) |
| – SpatialNotchBandwidth1 | double | Spatial periodic bandwidth for filter #1 (Hz) |
| – SpatialNotchGain1 | double | Spatial periodic gain for filter #1 |
| – SpatialNotchStep2 | double | Spatial periodic step for filter #2 (units) |
| – SpatialNotchBandwidth2 | double | Spatial periodic bandwidth for filter #2 (Hz) |
| – SpatialNotchGain2 | double | Spatial periodic gain for filter #2 |
| – SpatialNotchStep3 | double | Spatial periodic step for filter #3 (units) |
| – SpatialNotchBandwidth3 | double | Spatial periodic bandwidth for filter #3 (Hz) |
| – SpatialNotchGain3 | double | Spatial periodic gain for filter #3 |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

### 3.2.4.23 PositionerCorrectorNotchFiltersGet

#### Name

**PositionerCorrectorNotchFiltersGet** – Gets the notch filter parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 |

### 3.2.4.24 PositionerCorrectorNotchFiltersSet

#### Name

**PositionerCorrectorNotchFiltersSet** – Sets the notch filter parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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)

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

$\Rightarrow [0 : 5000]$

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

$\Rightarrow [0 : 5000]$

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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 |

### 3.2.4.25 PositionerCorrectorPIDDualFFVoltageGet

#### Name

**PositionerCorrectorPIDDualFFVoltageGet** – Gets the corrector "PIDDualFFVoltage" parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 |

### 3.2.4.26 PositionerCorrectorPIDDualFFVoltageSet

#### Name

**PositionerCorrectorPIDDualFFVoltageSet** – Configures the corrector "PIDDualFFVoltage" parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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)

$KP \geq 0$

$KI \geq 0$

$KD \geq 0$

$0 \leq KS \leq 1$

$IntegrationTime \geq CorrectorPeriod\ (0.0001\ s)$

$GKP > -1$

$GKI > -1$

$GKD > -1$

$KForm \geq 0$

$KFeedForwardVelocity \geq 0$

$KFeedForwardAcceleration \geq 0$

$Friction \geq 0$

$$DerivativeFilterCutOffFrequency \in \left[0 : \frac{0.5}{CorrectorPeriod}\right] \Rightarrow [0 : 5000] \text{ with}$$

$CorrectorPeriod = 0.0001\ s$

#### 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 |

### 3.2.4.27 PositionerCorrectorPIDFFAccelerationGet

#### Name

**PositionerCorrectorPIDFFAccelerationGet** – Gets the corrector "PIDFFAcceleration" parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "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) |

### 3.2.4.28 PositionerCorrectorPIDFFAccelerationSet

#### Name

**PositionerCorrectorPIDFFAccelerationSet** – Gets the corrector "PIDFFAcceleration" parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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*

*KD* ≥ *0*

*0* ≤ *KS* ≤ *1*

*IntegrationTime* ≥ *CorrectorPeriod (0.0001 s)*

*GKP* > *-1*

*GKI* > *-1*

*GKD* > *-1*

*KForm* ≥ *0*

*KFeedForwardAcceleration* ≥ *0*

$$DerivativeFilterCutOffFrequency \in \left[0 : \frac{0.5}{CorrectorPeriod}\right] \Rightarrow [0 : 5000] \text{ with}$$

*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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 |

### 3.2.4.29 PositionerCorrectorPIDFFVelocityGet

#### Name

**PositionerCorrectorPIDFFVelocityGet** – Gets the corrector "PIDFFVelocity" parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "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) |

### 3.2.4.30 PositionerCorrectorPIDFFVelocitySet

#### Name

**PositionerCorrectorPIDFFVelocitySet** – Configures the corrector "PIDFFVelocity" parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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*

*KD ≥ 0*

*0 ≤ KS ≤ 1*

*IntegrationTime ≥ CorrectorPeriod (0.0001 s)*

*GKP > -1*

*GKI > -1*

*GKD > -1*

*KForm ≥ 0*

*KFeedForwardVelocity ≥ 0*

$$DerivativeFilterCutOffFrequency \in \left[0 : \frac{0.5}{CorrectorPeriod}\right] \Rightarrow [0 : 5000] \text{ with}$$

*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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 |

### 3.2.4.31 PositionerCorrectorPIPositionGet

#### Name

**PositionerCorrectorPIPositionGet** – Gets the corrector "PIPosition" parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "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) |

### 3.2.4.32 PositionerCorrectorPIPositionSet

#### Name

**PositionerCorrectorPIPositionSet** – Configures the corrector "PIPosition" parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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)

$KP \geq 0$

$KI \geq 0$

$IntegrationTime \geq CorrectorPeriod\ (0.0001\ s)$

#### 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 |

### 3.2.4.33 PositionerCorrectorSR1AccelerationGet

#### Name

**PositionerCorrectorSR1AccelerationGet** – Gets the corrector "SR1Acceleration" parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 returns the SR1 corrector parameter current values.

**NOTE**

**This function can be used only with the XPS-Qn Precision Platform controller.**

**The "CorrectorType" must be "SR1Acceleration" 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

**PositionerCorrectorSR1AccelerationGet** $SocketID $FullPositionerName ClosedLoopStatus KP KI KV ObserverFrequency CompensationGainVelocity CompensationGainAcceleration CompensationGainJerk

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Output parameters

| | | |
|---|---|---|
| – ClosedLoopStatus | bool | Position servo loop status (true=closed and false=opened) |
| – KP | double | SR1 corrector proportional gain ($sec^{-2}$) |
| – KI | double | SR1 corrector integral gain ($sec^{-3}$) |
| – KV | double | SR1 corrector velocity gain ($sec^{-1}$) |
| – ObserverFrequency | double | SR1 observer frequency (Hz) |
| – CompensationGainVelocity | double | Velocity compensation gain (sec) |
| – CompensationGainAcceleration | double | Acceleration compensation gain ($sec^2$) |
| – CompensationGainJerk | double | Jerk compensation gain ($sec^3$) |

### Return

| | | |
|---|---|---|
| – Error | int | TCL error code (0 = success or 1 = syntax error) or function error code |

## C/C++

### Prototype

int **PositionerCorrectorSR1AccelerationGet** (int SocketID, char FullPositionerName[250], bool* ClosedLoopStatus, double* KP, double* KI, double* KV, double* ObserverFrequency, double* CompensationGainVelocity, double* CompensationGainJerk)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | char * | Positioner name |

### Output parameters

| | | |
|---|---|---|
| – ClosedLoopStatus | bool * | Position servo loop status (true=closed and false=opened) |
| – KP | double * | SR1 corrector proportional gain ($sec^{-2}$) |
| – KI | double * | SR1 corrector integral gain ($sec^{-3}$) |
| – KV | double * | SR1 corrector velocity gain ($sec^{-1}$) |
| – ObserverFrequency | double * | SR1 observer frequency (Hz) |
| – CompensationGainVelocity | double * | Velocity compensation gain (sec) |
| – CompensationGainAcceleration | double * | Acceleration compensation gain ($sec^2$) |
| – CompensationGainJerk | double * | Jerk compensation gain ($sec^3$) |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

## Visual Basic

### Prototype

Long **PositionerCorrectorSR1AccelerationGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ClosedLoopStatus As Boolean, KP As Double, KI As Double, KV As Double, ObserverFrequency As Double, CompensationGainVelocity As Double, CompensationGainAcceleration As Double, CompensationGainJerk As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Output parameters

| | | |
|---|---|---|
| – ClosedLoopStatus | bool | Position servo loop status (true=closed and false=opened) |
| – KP | double | SR1 corrector proportional gain ($sec^{-2}$) |
| – KI | double | SR1 corrector integral gain ($sec^{-3}$) |
| – KV | double | SR1 corrector velocity gain ($sec^{-1}$) |
| – ObserverFrequency | double | SR1 observer frequency (Hz) |
| – CompensationGainVelocity | double | Velocity compensation gain (sec) |
| – CompensationGainAcceleration | double | Acceleration compensation gain ($sec^2$) |
| – CompensationGainJerk | double | Jerk compensation gain ($sec^3$) |

### Return

| | | |
|---|---|---|
| – Error | long | Function error code |

## Matlab

### Prototype

[Error, ClosedLoopStatus, KP, KI, KV, ObserverFrequency, CompensationGainVelocity, CompensationGainAcceleration, CompensationGainJerk] **PositionerCorrectorSR1AccelerationGet** (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 |
| – ClosedLoopStatus | bool | Position servo loop status (true=closed and false=opened) |
| – KP | double | SR1 corrector proportional gain ($sec^{-2}$) |
| – KI | double | SR1 corrector integral gain ($sec^{-3}$) |
| – KV | double | SR1 corrector velocity gain ($sec^{-1}$) |
| – ObserverFrequency | double | SR1 observer frequency (Hz) |
| – CompensationGainVelocity | double | Velocity compensation gain (sec) |
| – CompensationGainAcceleration | double | Acceleration compensation gain ($sec^2$) |
| – CompensationGainJerk | double | Jerk compensation gain ($sec^3$) |

## Python

### Prototype

[Error, ClosedLoopStatus, KP, KI, KV, ObserverFrequency, CompensationGainVelocity, CompensationGainAcceleration, CompensationGainJerk] **PositionerCorrectorSR1AccelerationGet** (integer SocketID, string FullPositionerName)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |
| – ClosedLoopStatus | bool | Position servo loop status (true=closed and false=opened) |
| – KP | double | SR1 corrector proportional gain ($sec^{-2}$) |
| – KI | double | SR1 corrector integral gain ($sec^{-3}$) |
| – KV | double | SR1 corrector velocity gain ($sec^{-1}$) |
| – ObserverFrequency | double | SR1 observer frequency (Hz) |
| – CompensationGainVelocity | double | Velocity compensation gain (sec) |
| – CompensationGainAcceleration | double | Acceleration compensation gain ($sec^2$) |
| – CompensationGainJerk | double | Jerk compensation gain ($sec^3$) |

### 3.2.4.34 PositionerCorrectorSR1AccelerationSet

#### Name

**PositionerCorrectorSR1AccelerationSet** – Sets the corrector "SR1Acceleration" parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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
- KV > 0
- ObserverFrequency $\in \left[0 : \frac{0.5}{\text{CorrectorISRPeriod}}\right]$

#### Description

This function configures the "SR1Acceleration" corrector parameters.

**NOTES**

**This function can be used only with the XPS-Qn Precision Platform controller.**

**The "CorrectorType" parameter must be defined as "SR1Acceleration" in the "stages.ini" file, else ERR_WRONG_OBJECT_TYPE (-8) 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

**PositionerCorrectorSR1AccelerationSet** $SocketID $FullPositionerName $ClosedLoopStatus $KP $KI $KV $ObserverFrequency $CompensationGainVelocity $CompensationGainAcceleration $CompensationGainJerk

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – ClosedLoopStatus | bool | Position servo loop status (true=closed and false=opened) |
| – KP | double | SR1 corrector proportional gain ($sec^{-2}$) |
| – KI | double | SR1 corrector integral gain ($sec^{-3}$) |
| – KV | double | SR1 corrector velocity gain ($sec^{-1}$) |
| – ObserverFrequency | double | SR1 observer frequency (Hz) |
| – CompensationGainVelocity | double | Velocity compensation gain (sec) |
| – CompensationGainAcceleration | double | Acceleration compensation gain ($sec^2$) |
| – CompensationGainJerk | double | Jerk compensation gain ($sec^3$) |

### Output parameters

– None

### Return

| | | |
|---|---|---|
| – Error | int | TCL error code (0 = success or 1 = syntax error) or function error code |

## C/C++

### Prototype

int **PositionerCorrectorSR1AccelerationSet** (int SocketID, char FullPositionerName[250], bool ClosedLoopStatus, double KP, double KI, double KV, double ObserverFrequency, double CompensationGainVelocity, double CompensationGainAcceleration, double CompensationGainJerk)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | char * | Positioner name |
| – ClosedLoopStatus | bool | Position servo loop status (true=closed and false=opened) |
| – KP | double | SR1 corrector proportional gain ($sec^{-2}$) |
| – KI | double | SR1 corrector integral gain ($sec^{-3}$) |
| – KV | double | SR1 corrector velocity gain ($sec^{-1}$) |
| – ObserverFrequency | double | SR1 observer frequency (Hz) |
| – CompensationGainVelocity | double | Velocity compensation gain (sec) |
| – CompensationGainAcceleration | double | Acceleration compensation gain ($sec^2$) |
| – CompensationGainJerk | double | Jerk compensation gain ($sec^3$) |

### Output parameters

– None

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

## Visual Basic

### Prototype

Long **PositionerCorrectorSR1AccelerationSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal ClosedLoopStatus As Boolean, ByVal KP As Double, ByVal KI As Double, ByVal KV As Double, ByVal ObserverFrequency As Double, ByVal CompensationGainVelocity As Double, ByVal CompensationGainAcceleration As Double, ByVal CompensationGainJerk As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – ClosedLoopStatus | bool | Position servo loop status (true=closed and false=opened) |
| – KP | double | SR1 corrector proportional gain ($sec^{-2}$) |
| – KI | double | SR1 corrector integral gain ($sec^{-3}$) |
| – KV | double | SR1 corrector velocity gain ($sec^{-1}$) |
| – ObserverFrequency | double | SR1 observer frequency (Hz) |
| – CompensationGainVelocity | double | Velocity compensation gain (sec) |
| – CompensationGainAcceleration | double | Acceleration compensation gain ($sec^2$) |
| – CompensationGainJerk | double | Jerk compensation gain ($sec^3$) |

### Output parameters

– None

### Return

| | | |
|---|---|---|
| – Error | long | Function error code |

## Matlab

### Prototype

[Error] **PositionerCorrectorSR1AccelerationSet** (int32 SocketID, cstring FullPositionerName, bool ClosedLoopStatus, double KP, double KI, double KV, double ObserverFrequency, double CompensationGainVelocity, double CompensationGainAcceleration, double CompensationGainJerk)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | cstring | Positioner name |
| – ClosedLoopStatus | bool | Position servo loop status (true=closed and false=opened) |
| – KP | double | SR1 corrector proportional gain ($sec^{-2}$) |
| – KI | double | SR1 corrector integral gain ($sec^{-3}$) |
| – KV | double | SR1 corrector velocity gain ($sec^{-1}$) |
| – ObserverFrequency | double | SR1 observer frequency (Hz) |
| – CompensationGainVelocity | double | Velocity compensation gain (sec) |
| – CompensationGainAcceleration | double | Acceleration compensation gain ($sec^2$) |
| – CompensationGainJerk | double | Jerk compensation gain ($sec^3$) |

### Return

| | | |
|---|---|---|
| – Error | int32 | Function error code |

## Python

### Prototype

[Error] **PositionerCorrectorSR1AccelerationSet** (integer SocketID, string FullPositionerName, bool ClosedLoopStatus, double KP, double KI, double KV, double ObserverFrequency, double CompensationGainVelocity, double CompensationGainAcceleration, double CompensationGainJerk)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |
| – | ClosedLoopStatus | bool | Position servo loop status (true=closed and false=opened) |
| – | KP | double | SR1 corrector proportional gain ($sec^{-2}$) |
| – | KI | double | SR1 corrector integral gain ($sec^{-3}$) |
| – | KV | double | SR1 corrector velocity gain ($sec^{-1}$) |
| – | ObserverFrequency | double | SR1 observer frequency (Hz) |
| – | CompensationGainVelocity | double | Velocity compensation gain (sec) |
| – | CompensationGainAcceleration | double | Acceleration compensation gain ($sec^2$) |
| – | CompensationGainJerk | double | Jerk compensation gain ($sec^3$) |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |

### 3.2.4.35 PositionerCorrectorSR1ObserverAccelerationSet

#### Name

**PositionerCorrectorSR1ObserverAccelerationSet** – Sets the corrector "SR1Acceleration" observer parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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_DOUBLE (-14)

#### Description

This function sets the SR1 observer parameters.

---

**NOTES**

**This function can be used only with the XPS-Qn Precision Platform controller.**

**This function set the SR1 observer parameters, so it erases all of the observer parameter values previously calculated by the PositionerCorrectorSR1AccelerationSet**

**The "CorrectorType" parameter must be defined as "SR1Acceleration" in the "stages.ini" file, else ERR_WRONG_OBJECT_TYPE (-8) 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_WRONG_FORMAT (-7)
- ERR_WRONG_OBJECT_TYPE (-8)
- ERR_WRONG_PARAMETERS_NUMBER (-9)
- ERR_WRONG_TYPE_DOUBLE (-14)
- SUCCESS (0): no error

## TCL

### Prototype

**PositionerCorrectorSR1ObserverAccelerationSet** $SocketID $FullPositionerName $ParameterA $ParameterB $ParameterC

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – ParameterA | double | SR1 observer parameter A |
| – ParameterB | double | SR1 observer parameter B |
| – ParameterC | double | SR1 observer parameter C |

### Output parameters

– None

### Return

| | | |
|---|---|---|
| – Error | int | TCL error code (0 = success or 1 = syntax error) or function error code |

## C/C++

### Prototype

int **PositionerCorrectorSR1ObserverAccelerationSet** (int SocketID, char FullPositionerName[250], double ParameterA, double ParameterB, double ParameterC)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | char * | Positioner name |
| – ParameterA | double | SR1 observer parameter A |
| – ParameterB | double | SR1 observer parameter B |
| – ParameterC | double | SR1 observer parameter C |

### Output parameters

– None

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

## Visual Basic

### Prototype

Long **PositionerCorrectorSR1ObserverAccelerationSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal ParameterA As Double, ByVal ParameterB As Double, ByVal ParameterC As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – ParameterA | double | SR1 observer parameter A |
| – ParameterB | double | SR1 observer parameter B |
| – ParameterC | double | SR1 observer parameter C |

### Output parameters

– None

– Return

| | | |
|---|---|---|
| – Error | long | Function error code |

## Matlab

### Prototype

[Error] **PositionerCorrectorSR1ObserverAccelerationSet** (int32 SocketID, cstring FullPositionerName, double ParameterA, double ParameterB, double ParameterC)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | cstring | Positioner name |
| – ParameterA | double | SR1 observer parameter A |
| – ParameterB | double | SR1 observer parameter B |
| – ParameterC | double | SR1 observer parameter C |

### Return

| | | |
|---|---|---|
| – Error | int32 | Function error code |

## Python

### Prototype

[Error] **PositionerCorrectorSR1ObserverAccelerationSet** (integer SocketID, string FullPositionerName, double ParameterA, double ParameterB, double ParameterC)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |
| – | ParameterA | double | SR1 observer parameter A |
| – | ParameterB | double | SR1 observer parameter B |
| – | ParameterC | double | SR1 observer parameter C |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |

### 3.2.4.36 PositionerCorrectorSR1OffsetAccelerationGet

#### Name

**PositionerCorrectorSR1OffsetAccelerationGet** – Gets the corrector "SR1Acceleration" acceleration output offset.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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_DOUBLE (-14)

#### Description

This function gets the SR1 corrector acceleration output offset current value.

**NOTE**

**This function can be used only with the XPS-Qn Precision Platform controller.**

**The "CorrectorType" must be "SR1Acceleration" 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_DOUBLE (-14)
- SUCCESS (0): no error

## TCL

### Prototype

**PositionerCorrectorSR1OffsetAccelerationGet** $SocketID $FullPositionerName AccelerationOffset

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Output parameters

| | | |
|---|---|---|
| – AccelerationOffset | double | SR1 corrector acceleration output offset |

### Return

| | | |
|---|---|---|
| – Error | int | TCL error code (0 = success or 1 = syntax error) or function error code |

## C/C++

### Prototype

int **PositionerCorrectorSR1OffsetAccelerationGet** (int SocketID, char FullPositionerName[250], double* AccelerationOffset)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | char * | Positioner name |

### Output parameters

| | | |
|---|---|---|
| – AccelerationOffset | double * | SR1 corrector acceleration output offset |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

## Visual Basic

### Prototype

Long **PositionerCorrectorSR1OffsetAccelerationGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, AccelerationOffset As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Output parameters

| | | |
|---|---|---|
| – AccelerationOffset | double | SR1 corrector acceleration output offset |
| – Return | | |
| – Error | long | Function error code |

## Matlab

### Prototype

[Error, AccelerationOffset] **PositionerCorrectorSR1OffsetAccelerationGet** (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 |
| – AccelerationOffset | double | SR1 corrector acceleration output offset |

## Python

### Prototype

[Error, AccelerationOffset] **PositionerCorrectorSR1OffsetAccelerationGet** (integer SocketID, string FullPositionerName)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |
| – AccelerationOffset | double | SR1 corrector acceleration output offset |

### 3.2.4.37 PositionerCorrectorSR1OffsetAccelerationSet

#### Name

**PositionerCorrectorSR1OffsetAccelerationSet** – Sets the corrector "SR1Acceleration" acceleration output offset.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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_DOUBLE (-14)

#### Description

This function sets the SR1 acceleration output offset.

---

**NOTE**

**This function can be used only with the XPS-Qn Precision Platform controller.**

**The default value of the SR1 acceleration output offset is zero (0) at controller reboot.**

**The "CorrectorType" parameter must be defined as "SR1Acceleration" in the "stages.ini" file, else ERR_WRONG_OBJECT_TYPE (-8) 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_WRONG_FORMAT (-7)
- ERR_WRONG_OBJECT_TYPE (-8)
- ERR_WRONG_PARAMETERS_NUMBER (-9)
- ERR_WRONG_TYPE_DOUBLE (-14)
- SUCCESS (0): no error

## TCL

### Prototype

**PositionerCorrectorSR1OffsetAccelerationSet** $SocketID $FullPositionerName $AccelerationOffset

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – AccelerationOffset | double | SR1 corrector acceleration output offset |

### Output parameters

– None

### Return

| | | |
|---|---|---|
| – Error | int | TCL error code (0 = success or 1 = syntax error) or function error code |

## C/C++

### Prototype

int **PositionerCorrectorSR1OffsetAccelerationSet** (int SocketID, char FullPositionerName[250], double AccelerationOffset)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | char * | Positioner name |
| – AccelerationOffset | double | SR1 corrector acceleration output offset |

### Output parameters

– None

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

## Visual Basic

### Prototype

Long **PositionerCorrectorSR1OffsetAccelerationSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal AccelerationOffset As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – AccelerationOffset | double | SR1 corrector acceleration output offset |

### Output parameters

– None

### Return

| | | |
|---|---|---|
| – Error | long | Function error code |

## Matlab

### Prototype

[Error] **PositionerCorrectorSR1OffsetAccelerationSet** (int32 SocketID, cstring FullPositionerName, double AccelerationOffset)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | cstring | Positioner name |
| – AccelerationOffset | double | SR1 corrector acceleration output offset |

### Return

| | | |
|---|---|---|
| – Error | int32 | Function error code |

## Python

### Prototype

[Error] **PositionerCorrectorSR1OffsetAccelerationSet** (integer SocketID, string FullPositionerName, double AccelerationOffset)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – AccelerationOffset | double | SR1 corrector acceleration output offset |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

### 3.2.4.38 PositionerCorrectorTypeGet

#### Name

**PositionerCorrectorTypeGet** – Returns the corrector type.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error code |
| – CorrectorType | string | Corrector type |

### 3.2.4.39 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)
- HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "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) |

### 3.2.4.40 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)
- HXP 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)

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

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

#### 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 |

### 3.2.4.41 PositionerDriverFiltersGet

#### Name

**PositionerDriverFiltersGet** – Gets the piezo driver notch and lowpass filters parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 returns 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 | int | 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 | int | 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 | int | Socket identifier got from "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | 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) |

### 3.2.4.42 PositionerDriverFiltersSet

#### Name

**PositionerDriverFiltersSet** – Sets the piezo driver filters parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 sets 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 | int | 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 | int | 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 | int | 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 | int | Function error code |

### 3.2.4.43 PositionerDriverPositionOffsetsGet

#### Name

**PositionerDriverPositionOffsetsGet** – Gets the current value of piezo driver stage and gage position offsets.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 returns 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 | int | 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 | int | 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 | int | Socket identifier got from "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

**Return**

| | | |
|---|---|---|
| – Error | int | Function error code |
| – StagePositionOffset | double | Driver stage position offset (units) |
| – GagePositionOffset | double | Driver gage position offset (units) |

### 3.2.4.44 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)
- HXP 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 §3.13.**

#### 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error code |
| – PositionerDriverStatus | integer | Driver status code |

### 3.2.4.45 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)
- HXP 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 §3.13.**

#### 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |
| – | PositionerDriverStatus | integer | Driver status code |

### Return

| | | | |
|---|---|---|---|
| – | Error | integer | Function error code |
| – | PositionerDriverStatusString | string | Driver status description |

### 3.2.4.46 PositionerEncoderAmplitudeValuesGet

#### Name

**PositionerEncoderAmplitudeValuesGet** – Gets the encoder amplitude values.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "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) |

### 3.2.4.47 PositionerEncoderCalibrationParametersGet

#### Name

**PositionerEncoderCalibrationParametersGet** – Gets the encoder calibration parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "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) |

### 3.2.4.48 PositionerErrorGet

#### Name

**PositionerErrorGet** – Returns the positioner error code and clears it.

#### Input tests

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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" §3.11. 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer"function |
| – PositionerName | string | Positioner name |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error code |
| – PositionerError | integer | Positioner error code |

### 3.2.4.49 PositionerErrorRead

#### Name

**PositionerErrorRead** – Returns the positioner error code without clears it.

#### Input tests

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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" §3.11. 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer"function |
| – PositionerName | string | Positioner name |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error code |
| – PositionerError | integer | Positioner error code |

### 3.2.4.50 PositionerErrorStringGet

#### Name

**PositionerErrorStringGet** – Gets the positioner error description.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 §3.11.**

---

#### 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |
| – | PositionerErrorCode | integer | Positioner error code |

### Return

| | | | |
|---|---|---|---|
| – | Error | integer | Function error code |
| – | PositionerErrorString | string | Positioner error description |

### 3.2.4.51 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)
- HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |

### Return

| | | | |
|---|---|---|---|
| – | Error | integer | Function error code |
| – | InterpolationFactor | integer | Interpolation factor |

### 3.2.4.52 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)
- HXP 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 from these values then ERR_PARAMETER_OUT_OF_RANGE 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error code |
| – InterpolationFactor | integer | Interpolation factor |

### 3.2.4.53 PositionerHardInterpolatorPositionGet

#### Name

**PositionerHardInterpolatorPositionGet** – Gets interpolated position from the encoder hard interpolator.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 type (must be "AnalogInterpolated"): ERR_WRONG_OBJECT_TYPE (-8)
- Check output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

#### Description

This function returns the position interpolated by the encoder hard interpolator.

**NOTE**

**This function can be used only with the XPS-Qn Precision Platform controller.**

**The encoder type must be "*AnalogInterpolated*" or "*AnalogInterpolatedTheta*" 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**

**PositionerHardInterpolatorPositionGet** $SocketID $FullPositionerName Position

**Input parameters**

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |

**Output parameters**

| | | | |
|---|---|---|---|
| – | Position | double | Interpolated position |

**Return**

| | | | |
|---|---|---|---|
| – | Error | int | TCL error code (0=success or 1=syntax error) or function error code |

## C/C++

**Prototype**

int **PositionerHardInterpolatorPositionGet** (int SocketID, char FullPositionerName[250] , double * Position)

**Input parameters**

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | char * | Positioner name |

**Output parameters**

| | | | |
|---|---|---|---|
| – | Position | double * | Interpolated position |

**Return**

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |

## Visual Basic

**Prototype**

Long **PositionerHardInterpolatorPositionGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, Position As Double)

**Input parameters**

| | | | |
|---|---|---|---|
| – | SocketID | long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |

**Output parameters**

| | | | |
|---|---|---|---|
| – | Position | double | Interpolated position |

**Return**

| | | | |
|---|---|---|---|
| – | Error | long | Function error code |

## Matlab

### Prototype

[Error, Position] **PositionerHardInterpolatorPositionGet** (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 |
| – Position | double | Interpolated position |

## Python

### Prototype

[Error, Position] **PositionerHardInterpolatorPositionGet** (integer SocketID, string FullPositionerName)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |
| – Position | double | Interpolated position |

### 3.2.4.54 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)
- HXP 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 §3.12.**

---

#### 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error code |
| – PositionerHardwareStatus | integer | Hardware status code |

### 3.2.4.55 PositionerHardwareStatusStringGet

#### Name

**PositionerHarwareStatusStringGet** – Gets the positioner error description.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 §3.12.**

#### 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 | |

### 3.2.4.56 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)
- HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "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) |

### 3.2.4.57 PositionerMotionDoneGet

#### Name

**PositionerMotionDoneGet** – Gets the motion done parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "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) |

### 3.2.4.58 PositionerMotionDoneSet

#### Name

**PositionerMotionDoneSet** – Sets the motion done parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 |

### 3.2.4.59 PositionerPositionCompareDisable

#### Name

**PositionerPositionCompareDisable** – Disables the position compare mode.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 disables the position compare mode.

For a more thorough description of the position compare output, please refer to the XPS Motion Tutorial section 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 | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |

##### Output parameters

– None

##### Return

| | | | |
|---|---|---|---|
| – | Error | int | 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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | cstring | Positioner name |

**Return**

| | | | |
|---|---|---|---|
| – | Error | int32 | Function error code |

## Python

### Prototype

[Error] **PositionerPositionCompareDisable** (integer SocketID, string FullPositionerName)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |

### 3.2.4.60 PositionerPositionCompareEnable

#### Name

**PositionerPositionCompareEnable** – Enables the position compare mode.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 enables the position compare mode. The group must be in READY state to use this function else ERR_NOT_ALLOWED_ACTION (-22) is returned.

If the position compare parameters are not configured (by the "PositionerPositionCompareSet" function) then ERR_NOT_ALLOWED_ACTION (-22) is returned.

For a more thorough description of the position compare output, please refer to the XPS Motion Tutorial section 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 | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |

**Output parameters**

– None

**Return**

| | | | |
|---|---|---|---|
| – | Error | int | 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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | cstring | Positioner name |

### Return

| | | |
|---|---|---|
| – Error | int32 | Function error code |

## Python

### Prototype

[Error] **PositionerPositionCompareEnable** (integer SocketID, string FullPositionerName)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

### 3.2.4.61 PositionerPositionCompareGet

#### Name

**PositionerPositionCompareGet** – Gets the position compare parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 PositionCompare): ERR_UNCOMPATIBLE (-24)
- Check output parameter type: ERR_WRONG_TYPE_DOUBLE (-14), ERR_WRONG_TYPE_BOOL (-12)

#### Description

This function returns the real value (without correction) of parameters of the position compare output trigger and returns current state (enabled or disabled).

For a more thorough description of the position compare output, please refer to the XPS Motion Tutorial section Triggers/Position Compare Output.

#### 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

**PositionerPositionCompareGet** $SocketID $FullPositionerName MinimumPosition MaximumPosition PositionStep EnableState

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "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 | int | 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 gets by the "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 gets by the "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 | long | Function error code |

## Matlab

### Prototype

[Error, MinimumPosition, MaximumPosition, PositionStep, EnableState] **PositionerPositionCompareGet** (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) |
| – 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 | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Return

| | | |
|---|---|---|
| – Error | int | 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) |

### 3.2.4.62 PositionerPositionCompareSet

#### Name

**PositionerPositionCompareSet** – Sets the position compare parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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
- 0 <= PositionStep <= (MaximumPosition – MinimumPosition)
- Check position compare state (must be disabled): ERR_NOT_ALLOWED_ACTION (-22)

#### Description

This function sets 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 Triggers/Position Compare Output.

---

**NOTE**

**This function can be used only with a position encoder. If no position encoder then ERR_UNCOMPATIBLE (-24) is returned.**

**In the PositionCompare mode (activated with PositionerPositionCompareEnable() function), during the move (relative or absolute) and inside the zone set by PositionerPositionCompareSet(), if the current following error exceeds the Warning Following Error value, the PositionCompareWarningFollowingErrorFlag is activated and the move returns an error (-120: Warning following error during move with position compare enabled). To reset the PositionCompareWarningFollowingErrorFlag, send the PositionerPositionCompareDisable() function. The WarningFollowingError is set to FatalFollowingError (defined in stages.ini file) by default, but it can be modified by PositionerWarningErrorSet().**

**In the PositionCompare mode (activated with PositionerPositionCompareEnable() function), during the move (relative or absolute) and inside the zone set by PositionerPositionCompareSet(), the CorrectorOutput is limited to MaximumAcceleration (*defined in stages.ini*).**

---

**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)
- SUCCESS (0): no error

**TCL**

**Prototype**

**PositionerPositionCompareSet** $SocketID $FullPositionerName $MinimumPosition $MaximumPosition $PositionStep

**Input parameters**

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "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 | int | 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 gets by the "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 gets by the "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 gets by the "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 | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – MinimumPosition | double | Minimum position (units) |
| – MaximumPosition | double | Maximum position (units) |
| – PositionStep | double | Position step (units) |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

### 3.2.4.63 PositionerPositionComparePulseParametersGet

#### Name

**PositionerPositionComparePulseParametersGet** – Gets the position compare PCO pulse parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 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 | int | Socket identifier gets by the "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 | int | 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 gets by the "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 gets by the "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 gets by the "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 | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |
| – | PCOPulseWidth | double | Width of PCO pulses (µs) |
| – | EncoderSettlingTime | double | Encoder signal settling time (µs) |

### 3.2.4.64 PositionerPositionComparePulseParametersSet

#### Name

**PositionerPositionComparePulseParametersSet** – Sets the position compare PCO pulse parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 sets 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 stabilize after a change of state is detected.

These parameters are used only when using the position compare mode. For a more thorough description of the position compare output, please refer to the XPS Motion Tutorial section Triggers/Position Compare Output.

---

**NOTE**

**When changing the PCO Pulse settle time you must limit the maximum velocity of the stage accordingly, otherwise you will loose the PCO position and generate the wrong number of pulses at wrong positions. For example, if you set the EncoderSettlingTime to 4 µs, the maximum PCO encoder frequency need to be limited to less than 0.25 / $4e^{-6}$ = 62.5 kHz. So, if EncoderScalePitch = 0.004 mm and HardInterpolatorFactor = 200 then the stage maximum velocity must ≤ $62.5e^{3}$ * 0.004 / 200 = 1.25 mm/s, otherwise the PCO will not work properly.**

---

How to determine PCO encoder frequency:

1. ● For AquadB encoder:
   *PCO encoder frequency = Velocity / EncoderResolution*
2. ● For analog interpolated encoder:
   *PCO encoder frequency = Velocity * HardInterpolatorFactor / EncoderScalePitch*

Example: XML310 stage (EncoderScalePitch=0.004 mm, HardInterpolatorFactor=200).
If Velocity=10mm/s => PCO encoder frequency = 10*200/0.004 = 500 kHz

**NOTE**

**This function can be used only with a position encoder. If no position encoder then ERR_UNCOMPATIBLE (-24) 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 | int | Socket identifier gets by the "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 | int | 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 gets by the "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 gets by the "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 gets by the "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 | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – PCOPulseWidth | double | Width of PCO pulses (µs) |
| – EncoderSettlingTime | double | Encoder signal settling time (µs) |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

### 3.2.4.65 PositionerPositionCompareScanAccelerationLimitGet

#### Name

**PositionerPositionCompareScanAccelerationLimitGet** – Get the position compare scan acceleration limit.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 corrector type (must be *PIDFFAcceleration* corrector): ERR_UNCOMPATIBLE (-24)

#### Description

This function returns the position compare scan acceleration limit.

During scan of position compare, the motor output will be limited to this value instead of the *AccelerationLimit*.

The position compare scan acceleration limit takes effect only with *PIDFFAcceleration* corrector type.

This function can be used only with a *PIDFFAcceleration* corrector else ERR_UNCOMPATIBLE is returned.

For a more thorough description of the position compare output, please refer to the XPS Motion Tutorial section Triggers/Position Compare Output.

---

**NOTE**

**This function can be used only with the XPS-Qn Precision Platform controller.**

---

#### Error codes

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

## TCL

### Prototype

PositionerPositionCompareScanAccelerationLimitGet $SocketID $FullPositionerName ScanAccelerationLimit

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Output parameters

| | | |
|---|---|---|
| – ScanAccelerationLimit | double | Limit of position compare scan acceleration (units/s$^2$) |

### Return

| | | |
|---|---|---|
| – Error | int | TCL error code (0=success or 1=syntax error) or function error code |

## C/C++

### Prototype

int **PositionerPositionCompareScanAccelerationLimitGet** (int SocketID, char FullPositionerName[250] , double* ScanAccelerationLimit)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | char * | Positioner name |

### Output parameters

| | | |
|---|---|---|
| – ScanAccelerationLimit | double * | Limit of position compare scan acceleration (units/s$^2$) |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

## Visual Basic

### Prototype

Long **PositionerPositionCompareScanAccelerationLimitGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ScanAccelerationLimit As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Output parameters

| | | |
|---|---|---|
| – ScanAccelerationLimit | double | Limit of position compare scan acceleration (units/s$^2$) |

### Return

| | | |
|---|---|---|
| – Error | long | Function error code |

## Matlab

### Prototype

[Error, ScanAccelerationLimit]
**PositionerPositionCompareScanAccelerationLimitGet** (int32 SocketID, cstring FullPositionerName)

### Input parameter

| | | | |
|---|---|---|---|
| – | SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | cstring | Positioner name |

### Return

| | | | |
|---|---|---|---|
| – | Error | int32 | Function error code |
| – | ScanAccelerationLimit | double | Limit of position compare scan acceleration (units/s$^2$) |

## Python

### Prototype

[Error, ScanAccelerationLimit]
**PositionerPositionCompareScanAccelerationLimitGet** (integer SocketID, string FullPositionerName)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |
| – | ScanAccelerationLimit | double | Limit of position compare scan acceleration (units/s$^2$) |

### 3.2.4.66 PositionerPositionCompareScanAccelerationLimitSet

#### Name

PositionerPositionCompareScanAccelerationLimitSet – Sets the position compare acceleration limit.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: ERR_FATAL_INIT (-20)
- XPS initialization in progress: ERR_IN_INITIALIZATION (-21)
- Check the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Check the corrector type (must be *PIDFFAcceleration* corrector): ERR_UNCOMPATIBLE (-24)
- Check input parameter values: ERR_PARAMETER_OUT_OF_RANGE (-17)
- PositionCompareScanAccelerationLimit value must > 0 and <= MaximumAcceleration (value in stages.ini)

#### Description

This function sets the position compare scan acceleration limit.

During position compare, the motor output will be limited to this value instead of *AccelerationLimit*.

The position compare scan acceleration limit takes effect only with *PIDFFAcceleration* corrector type.

This function can be used only with a *PIDFFAcceleration* corrector otherwise ERR_UNCOMPATIBLE is returned.

For a more thorough description of the position compare output, please refer to the XPS Motion Tutorial section Triggers/Position Compare Output.

**NOTE**

**This function can be used only with the XPS-Qn Precision Platform controller.**

#### Error codes

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

## TCL

### Prototype

**PositionerPositionCompareScanAccelerationLimitSet** $SocketID $FullPositionerName $ScanAccelerationLimit

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |
| – | ScanAccelerationLimit | double | Limit of position compare scan acceleration (units/s$^2$) |

### Output parameters

– None

### Return

| | | | |
|---|---|---|---|
| – | Error | int | TCL error code (0 = success or 1 = syntax error) or function error code |

## C/C++

### Prototype

int **PositionerPositionCompareScanAccelerationLimitSet** (int SocketID, char FullPositionerName[250] , double ScanAccelerationLimit)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | char * | Positioner name |
| – | ScanAccelerationLimit | double | Limit of position compare scan acceleration (units/s$^2$) |

### Output parameters

– None

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |

## Visual Basic

### Prototype

Long **PositionerPositionCompareScanAccelerationLimitSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal ScanAccelerationLimit As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – ScanAccelerationLimit | double | Limit of position compare scan acceleration (units/s$^2$) |

### Output parameters

– None

### Return

| | | |
|---|---|---|
| – Error | long | Function error code |

## Matlab

### Prototype

[Error] **PositionerPositionCompareScanAccelerationLimitSet** (int32 SocketID, cstring FullPositionerName, double ScanAccelerationLimit)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | cstring | Positioner name |
| – ScanAccelerationLimit | double | Limit of position compare scan acceleration (units/s$^2$) |

### Return

| | | |
|---|---|---|
| – Error | int32 | Function error code |

## Python

### Prototype

[Error] **PositionerPositionCompareScanAccelerationLimitSet** (integer SocketID, string FullPositionerName, double ScanAccelerationLimit)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – ScanAccelerationLimit | double | Limit of position compare scan acceleration (units/s$^2$) |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

### 3.2.4.67 PositionerPositionCompareAquadBAlwaysEnable

#### Name

**PositionerPositionCompareAquadBAlwaysEnable** – Enables the AquadB signal in the always mode.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 signals 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 ERR_UNCOMPATIBLE (-24) 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 | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |

### Output parameters

– None

### Return

| | | | |
|---|---|---|---|
| – | Error | int | 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 | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |

### 3.2.4.68 PositionerPositionCompareAquadBWindowedGet

#### Name

**PositionerPositionCompareAquadBWindowedGet** – Gets the windowed AquadB mode parameters and state..

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 | int | 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 | int | 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 | bool | 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 | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |
| – | MinimumPosition | double | Minimum position (units) |
| – | MaximumPosition | double | Maximum position (units) |
| – | EnableState | bool | Windowed AquadB state (true=enabled or false=disabled) |

### 3.2.4.69 PositionerPositionCompareAquadBWindowedSet

#### Name

**PositionerPositionCompareAquadBWindowedSet** – Sets the windowed AquadB signal parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 sets 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 in effect 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 ERR_UNCOMPATIBLE (-24) 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 | int | 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 | int | 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 | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – MinimumPosition | double | Minimum position (units) |
| – MaximumPosition | double | Maximum position (units) |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

### 3.2.4.70 PositionerRawEncoderPositionGet

#### Name

**PositionerRawEncoderPositionGet** – Returns the raw encoder position for a positioner.

#### Input tests

- Controller initialization failed: 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 returns the raw encoder position from a 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 | int | 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 | doubletPtr | Raw encoder position |

## Python

### Prototype

[Error, RawEncoderPosition] **PositionerRawEncoderPositionGet** (integer SocketID, string PositionerName, double UserEncoderPosition)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer"function |
| – | PositionerName | string | Positioner name |
| – | UserEncoderPosition | double | User corrected encoder position |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |
| – | RawEncoderPosition | doubletPtr | Raw encoder position |

### 3.2.4.71 PositionersEncoderIndexDifferenceGet

#### Name

**PositionersEncoderIndexDifferenceGet** – Gets the distance between the two index encoders (gantry).

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 encoders indexes of a "primary positioner – secondary positioner" couple. To use this function, the positioner must be configured in "gantry" mode else ERR_UNCOMPATIBLE (-24) is returned.

For further information about gantry mode, refer to the "SYSTEM – Manual Configuration – Gantries (Secondary Positioners)" section in the XPS 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 | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Output parameters

| | | |
|---|---|---|
| – Distance | double | Distance between the two index encoders (units) |

### Return

| | | |
|---|---|---|
| – Error | int | 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 gets by the "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 gets by the "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 gets by the "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 | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |
| – Distance | double | Distance between the two index encoders (units) |

### 3.2.4.72 PositionerSGammaExactVelocityAjustedDisplacementGet

#### Name

**PositionerSGammaExactVelocityAjustedDisplacementGet** – Gets the adjusted displacement to get exact velocity.

#### Input tests

- Controller initialization failed: 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 the 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 | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |
| – | DesiredDisplacement | double | Desired displacement (units) |

##### Output parameters

| | | | |
|---|---|---|---|
| – | AdjustedDisplacement | double | Ajusted displacement (units) |

##### Return

| | | | |
|---|---|---|---|
| – | Error | int | 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 gets by the "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 gets by the "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 gets by the "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 | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |
| – | DesiredDisplacement | double | Desired displacement (units) |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |
| – | AdjustedDisplacement | double | Ajusted displacement (units) |

### 3.2.4.73 PositionerSGammaParametersDistanceGet

#### Name

**PositionerSGammaParametersDistanceGet** – Returns distance during acceleration phase and distance during constant velocity phase.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 allows getting the distance during acceleration phase and the distance during constant velocity phase of the SGamma motion profile.

#### 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

**PositionerSGammaParametersDistanceGet** $SocketID $FullPositionerName $Displacement, $Velocity $Acceleration $MinJerkTime $MaxJerkTime DisplacementDuringAcc DisplacementDuringVel

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |
| – | Displacement | double | displacement value (units) |
| – | 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

| | | | |
|---|---|---|---|
| – | DisplacementDuringAcc | double | displacement value during acceleration phase (units) |
| – | DisplacementDuringVel | double | displacement value during constant velocity phase (units) |

### Return

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

## C/C++

### Prototype

int **PositionerSGammaParametersDistanceGet** (int SocketID, char FullPositionerName[250], double Displacement, double Velocity, double Acceleration, double MinJerkTime, double MaxJerkTime, double *DisplacementDuringAcc, double *DisplacementDuringVel)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | char * | Positioner name |
| – | Displacement | double | displacement value (units) |
| – | 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

| | | | |
|---|---|---|---|
| – | DisplacementDuringAcc | double * | displacement value during acceleration phase (units) |
| – | DisplacementDuringVel | double * | displacement value during constant velocity phase (units) |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |

## Visual Basic

### Prototype

Long **PositionerSGammaParametersDistanceGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal Displacement As Double, ByVal Velocity As Double, ByVal Acceleration As Double, ByVal MinimumJerkTime As Double, ByVal MaximumJerkTime As Double, MinimumJerkTime As Double, MaximumJerkTime As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | Long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | String | Positioner name |
| – Displacement | Double | displacement value (units) |
| – 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

| | | |
|---|---|---|
| – DisplacementDuringAcc | Double | displacement value during acceleration phase (units) |
| – DisplacementDuringVel | Double | displacement value during constant velocity phase (units) |

### Return

| | | |
|---|---|---|
| – Error | Long | Function error code |

## Matlab

### Prototype

[Error, DisplacementDuringAcc, DisplacementDuringVel]
**PositionerSGammaParametersDistanceGet** (int32 SocketID, cstring FullPositionerName, double Displacement, double Velocity, double Acceleration, double MinimumJerkTime, double MaximumJerkTime )

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | cstring | Positioner name |
| – Displacement | double | displacement value (units) |
| – 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 |
| – DisplacementDuringAcc | double | displacement value during acceleration phase (units) |
| – DisplacementDuringVel | double | displacement value during constant velocity phase (units) |

## Python

### Prototype

[Error, DisplacementDuringAcc, DisplacementDuringVel]
**PositionerSGammaParametersDistanceGet** (integer SocketID, string FullPositionerName, double Displacement, double Velocity, double Acceleration, double MinimumJerkTime, double MaximumJerkTime)

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "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 |
| – DisplacementDuringAcc | double | displacement value during acceleration phase (units) |
| – DisplacementDuringVel | double | displacement value during constant velocity phase (units) |

### 3.2.4.74 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)
- HXP 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 < **NewVelocity** ≤ MaximumVelocity*

*0 < **NewAcceleration** ≤ MaximumAcceleration*

*2 * ISRProfileGeneratorPeriod ≤ **NewMinimumJerkTime** ≤ **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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 |

### 3.2.4.75 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)
- HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "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) |

### 3.2.4.76 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)
- HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error code |
| – SettingTime | double | setting time (seconds) |
| – SettlingTime | double | settling time (seconds) |

### 3.2.4.77 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)
- HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | | Positioner name |
| – | ParameterName | string | Parameter name |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error code |
| – ParameterValue | string | Parameter value |

### 3.2.4.78 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)
- HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – ParameterName | string | Parameter name |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error code |
| – ParameterValue | string | Parameter value |

### 3.2.4.79 PositionerTimeFlasherDisable

#### Name

**PositionerTimeFlasherDisable** – Disables the time flasher mode.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |

##### Output parameters

– None

##### Return

| | | | |
|---|---|---|---|
| – | Error | int | 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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | cstring | Positioner name |

**Return**

| | | | |
|---|---|---|---|
| – | Error | int32 | Function error code |

## Python

### Prototype

[Error] **PositionerTimeFlasherDisable** (integer SocketID, string FullPositionerName)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |

### 3.2.4.80 PositionerTimeFlasherEnable

#### Name

**PositionerTimeFlasherEnable** – Enables the time flasher mode.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 ERR_NOT_ALLOWED_ACTION (-22) 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 | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

**Output parameters**

– None

**Return**

| | | |
|---|---|---|
| – Error | int | 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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | cstring | Positioner name |

### Return

| | | | |
|---|---|---|---|
| – | Error | int32 | Function error code |

## Python

### Prototype

[Error] **PositionerTimeFlasherEnable** (integer SocketID, string FullPositionerName)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |

### 3.2.4.81 PositionerTimeFlasherGet

#### Name

**PositionerTimeFlasherGet** – Gets the time flasher parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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)
- Check the configured mode type (must be TimeFlasher): ERR_UNCOMPATIBLE (-24)

#### 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 Triggers/Position Compare Output.

#### 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

**PositionerTimeFlasherGet** $SocketID $FullPositionerName MinimumPosition MaximumPosition TimePeriod EnableState

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |

### 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 | 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 gets by the "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 gets by the "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 | long | Function error code |

## Matlab

### Prototype

[Error, MinimumPosition, MaximumPosition, TimePeriod, EnableState] **PositionerTimeFlasherGet** (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) |
| – 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 | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | 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) |

### 3.2.4.82 PositionerTimeFlasherSet

#### Name

**PositionerTimeFlasherSet** – Sets the time flasher parameters.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Controller initialization failed: 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 configures 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 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 | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – MinimumPosition | double | Minimum position (units) |
| – MaximumPosition | double | Maximum position (units) |
| – TimePeriod | double | Time period (seconds) |

**Output parameters**

– None

**Return**

| | | |
|---|---|---|
| – Error | int | 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 gets by the "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 gets by the "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 gets by the "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 | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – MinimumPosition | double | Minimum position (units) |
| – MaximumPosition | double | Maximum position (units) |
| – TimePeriod | double | Time period (seconds) |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

### 3.2.4.83 PositionerUserTravelLimitsGet

#### Name

**PositionerUserTravelLimitsGet** – Gets the user travel limits.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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)

#### 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)
- SUCCESS (0) : no error

#### TCL

##### Prototype

**PositionerUserTravelLimitsGet** $SocketID $FullPositionerName UserMinimumTarget UserMaximumTarget

##### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | integer | Socket identifier gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | String | Positioner name |

### Output parameters

| | | |
|---|---|---|
| – UserMinimumTarget | Double | User minimum travel limit (units) |
| – UserMaximumTarget | Double | User maximum travel limit (units) |

### Return

| | | | |
|---|---|---|---|
| – | rror | Long | Function error code |

## Matlab

### Prototype

[Error, UserMinimumTarget, UserMaximumTarget] **PositionerUserTravelLimitsGet** (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 |
| – | 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 gets by the "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) |

### 3.2.4.84 PositionerUserTravelLimitsSet

#### Name

**PositionerUserTravelLimitsSet** – Sets the user travel limits.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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*

#### 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)
- SUCCESS (0) : no error

## TCL

### Prototype

**PositionerUserTravelLimitsSet** $SocketID $FullPositionerName $UserMinimumTarget $UserMaximumTarget

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 |

### 3.2.4.85 PositionerWarningFollowingErrorGet

#### Name

**PositionerWarningFollowingErrorGet** – Returns the warning following error for a positioner.

#### Input tests

- Controller initialization failed: 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_POSITIONER_NAME (-18)
- Valid output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

#### Description

This function gets the current value of the warning following error for a positioner.

**NOTE**

**This function can be used only with the XPS-Qn Precision Platform controller.**

#### 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

**PositionerWarningFollowingErrorGet** $SocketID $FullPositionerName WarningFollowingError

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |

### Output parameters

| | | |
|---|---|---|
| – WarningFollowingError | double | Warning following error (units) |

### Return

| | | |
|---|---|---|
| – Error | int | TCL error code (0=success or 1=syntax error) or function error code |

## C/C++

### Prototype

int **PositionerWarningFollowingErrorGet** (int SocketID, char FullPositionerName[250] , double * WarningFollowingError)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | char * | Positioner name |

### Output parameters

| | | |
|---|---|---|
| – WarningFollowingError | double * | Warning following error (units) |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

## Visual Basic

### Prototype

Long **PositionerWarningFollowingErrorGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, WarningFollowingError As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Output parameters

| | | |
|---|---|---|
| – WarningFollowingError | double | Warning following error (units) |

### Return

| | | |
|---|---|---|
| – Error | long | Function error code |

## Matlab

### Prototype

[Error, WarningFollowingError] **PositionerWarningFollowingErrorGet** (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 |
| – | WarningFollowingError | double | Warning following error limit (units) |

## Python

### Prototype

[Error, WarningFollowingError] **PositionerWarningFollowingErrorGet** (integer SocketID, string FullPositionerName)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |
| – | WarningFollowingError | double | Warning following error (units) |

### 3.2.4.86 PositionerWarningFollowingErrorSet

#### Name

**PositionerWarningFollowingErrorSet** – Sets value of the warning following error for a positioner.

#### Input tests

- Controller initialization failed: 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_POSITIONER_NAME (-18)
- Valid output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)
- Check input parameter values: ERR_PARAMETER_OUT_OF_RANGE (-17)
- 0 < WarningFollowingError <= FatalFollowingError

#### Description

This function sets a new value of the warning following error for a positioner.

**NOTE**

**This function can be used only with the XPS-Qn Precision Platform controller.**

#### 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_PARAMETER_OUT_OF_RANGE (-17)
- SUCCESS (0): no error

## TCL

### Prototype

**PositionerWarningFollowingErrorSet** $SocketID $FullPositionerName $WarningFollowingError

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – WarningFollowingError | double | Warning following error (units) |

### Output parameters

– None

### Return

| | | |
|---|---|---|
| – Error | int | TCL error code (0=success or 1=syntax error) or function error code |

## C/C++

### Prototype

int **PositionerWarningFollowingErrorSet** (int SocketID, char FullPositionerName[250] , double WarningFollowingError)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | char * | Positioner name |
| – WarningFollowingError | double | Warning following error (units) |

### Output parameters

– None

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

## Visual Basic

### Prototype

Long **PositionerWarningFollowingErrorSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, WarningFollowingError As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – WarningFollowingError | double | Warning following error (units) |

### Output parameters

– None

### Return

| | | |
|---|---|---|
| – Error | long | Function error code |

## Matlab

### Prototype

[Error] **PositionerWarningFollowingErrorSet** (int32 SocketID, cstring FullPositionerName, double WarningFollowingError)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | cstring | Positioner name |
| – | WarningFollowingError | double | Warning following error (units) |

### Return

| | | | |
|---|---|---|---|
| – | Error | int32 | Function error code |

## Python

### Prototype

[Error] **PositionerWarningFollowingErrorSet** (integer SocketID, string FullPositionerName, double WarningFollowingError)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |
| – | WarningFollowingError | double | Warning following error (units) |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |

### 3.2.5 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.
3. The **Geometry_RRPS.ini** file defines the Hexapod geometry parameters and the name of the text file which contains base and carriage error compensation matrix.

- **System.ref file:**

```
[GENERAL]
BootScriptFileName =
BootScriptArguments =                    ; the separator is the comma

[GROUPS]
HexapodInUse = MyGROUP                   ; the separator is the comma

[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
;--- 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)
; If DriverName = XPS-DRV01 driver
;---------  If  MotorDriverInterface = AnalogVelocity
DriverPWMFrequency =
DriverErrorAmplifierGain =
;---------  If  MotorDriverInterface = AnalogVoltage
PWMFrequency =
;---------  If  MotorDriverInterface = AnalogStepper
PWMFrequency =
HalfWinding =
; If DriverName = XPS-DRV02 driver
DriverBridgeFreeWheel =
DriverStepperWinding =
;--- MOTOR DRIVER INTERFACE
MotorDriverInterface =              ; AnalogStepperPosition
                                    ; AnalogVelocity
                                    ; AnalogVoltage
                                    ; AnalogAcceleration
                                    ; AnalogSin60Acceleration
                                    ; AnalogSin90Acceleration
                                    ; AnalogSin120Acceleration
                                    ; AnalogDualSin60Acceleration
                                    ; AnalogDualSin90Acceleration
                                    ; AnalogDualSin120Acceleration
                                    ; AnalogPosition
                                    ; 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 = AnalogStepperPosition
DisplacementPerFullStep =           ; units
ScalingCurrent =                    ; amps for 10 volts
PeakCurrentPerPhase =               ; amps
StandbyPeakCurrentPerPhase =        ; amps
; 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
MinimumTargetPosition =                 ; units
HomePreset =                            ; units
MaximumTargetPosition =                 ; units
;--- Profiler
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
```

- **Geometry_RRPS.ini file:**

```
[HEXAPOD]
; Base
BaseDiameter = 240.0                          ; units
BaseAlpha = 15.0                              ; degrees
BaseThickness = 25.0                          ; units
; Carriage
CarriageDiameter = 150.0                      ; units
CarriageAlpha = 20.0                          ; degrees
CarriageThickness = 25.0                      ; units
; Actuator
ActuatorLengthAtHomePreset = 167.5410634      ; units (Actuators length at "zero" encoder position (not home
                                              ;  position defined using HomePreset)
; Hexapod type
HexapodType = RRPS                            ; RRPRR (rotation around the actuator lead screw = degree of
                                               ; freedom) or RRPS
; Compensation
HexapodCorrectionFile = HexapodMatrix.txt
```

**The** HexapodMatrix.txt **file defines base and carriage error compensation matrix as below:**

```
; Base compensation
Joint1_Xcoord_ ModelError    Joint1_Ycoord_ ModelError    joint1_Zcoord_ ModelError
Joint2_Xcoord_ ModelError    Joint2_Ycoord_ ModelError    Joint2_Zcoord_ ModelError
Joint3_Xcoord_ ModelError    Joint3_Ycoord_ ModelError    Joint3_Zcoord_ ModelError
Joint4_Xcoord_ ModelError    Joint4_Ycoord_ ModelError    Joint4_Zcoord_ ModelError
Joint5_Xcoord_ ModelError    Joint5_Ycoord_ ModelError    Joint5_Zcoord_ ModelError
Joint6_Xcoord_ ModelError    Joint6_Ycoord_ ModelError    Joint6_Zcoord_ ModelError
; Carriage compensation
Joint1_Xcoord_ ModelError    Joint1_Ycoord_ ModelError    joint1_Zcoord_ ModelError
Joint2_Xcoord_ ModelError    Joint2_Ycoord_ ModelError    Joint2_Zcoord_ ModelError
Joint3_Xcoord_ ModelError    Joint3_Ycoord_ ModelError    Joint3_Zcoord_ ModelError
Joint4_Xcoord_ ModelError    Joint4_Ycoord_ ModelError    Joint4_Zcoord_ ModelError
```

| | | |
|---|---|---|
| Joint5_Xcoord_ ModelError | Joint5_Ycoord_ ModelError | Joint5_Zcoord_ ModelError |
| Joint6_Xcoord_ ModelError | Joint6_Ycoord_ ModelError | Joint6_Zcoord_ ModelError |

A matrix values refer to error compensation for every Hexapod's joint ; For exempl in base compensation matrix:

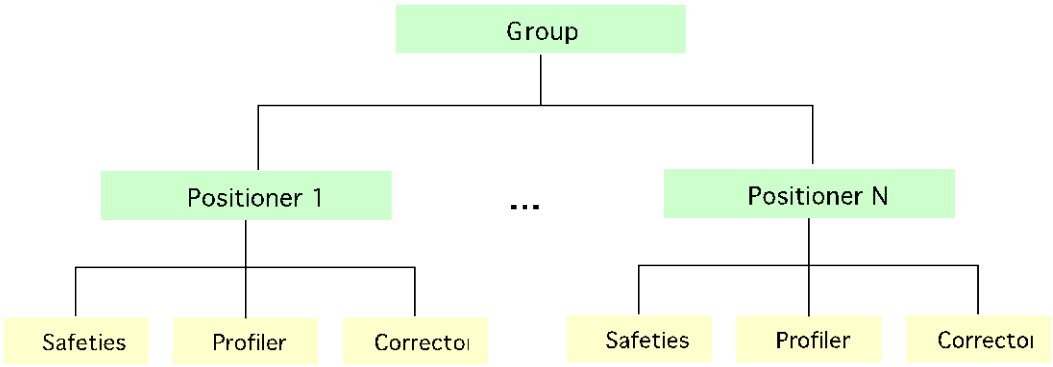
Joint1_Xcoord_ ModelError : this value refers to error compensation for joint 1 and X coordinate.

## 3.3 Group

### 3.3.1 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 for the HXP is **Hexapod** (6 axes).

### 3.3.2 Object structure




A motion “**Group**” is built in relation to a **group type** (Hexapod).

A group is defined by a **group name**.

#### 3.3.2.1 To defined a new group see §3.2.4.85 (

## PositionerWarningFollowingErrorGet

### Name

**PositionerWarningFollowingErrorGet** – Returns the warning following error for a positioner.

### Input tests

- Controller initialization failed: 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_POSITIONER_NAME (-18)
- Valid output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)

### Description

This function gets the current value of the warning following error for a positioner.

**NOTE**

**This function can be used only with the XPS-Qn Precision Platform controller.**

### 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

**PositionerWarningFollowingErrorGet** $SocketID $FullPositionerName WarningFollowingError

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |

### Output parameters

| | | | |
|---|---|---|---|
| – | WarningFollowingError | double | Warning following error (units) |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | TCL error code (0=success or 1=syntax error) or function error code |

## C/C++

### Prototype

int **PositionerWarningFollowingErrorGet** (int SocketID, char FullPositionerName[250] , double * WarningFollowingError)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | char * | Positioner name |

### Output parameters

| | | | |
|---|---|---|---|
| – | WarningFollowingError | double * | Warning following error (units) |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |

## Visual Basic

### Prototype

Long **PositionerWarningFollowingErrorGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, WarningFollowingError As Double)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |

### Output parameters

| | | | |
|---|---|---|---|
| – | WarningFollowingError | double | Warning following error (units) |

### Return

| | | | |
|---|---|---|---|
| – | Error | long | Function error code |

## Matlab

### Prototype

[Error, WarningFollowingError] **PositionerWarningFollowingErrorGet** (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 |
| – WarningFollowingError | double | Warning following error limit (units) |

## Python

### Prototype

[Error, WarningFollowingError] **PositionerWarningFollowingErrorGet** (integer SocketID, string FullPositionerName)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |
| – WarningFollowingError | double | Warning following error (units) |

### 3.3.2.2 PositionerWarningFollowingErrorSet

#### Name

**PositionerWarningFollowingErrorSet** – Sets value of the warning following error for a positioner.

#### Input tests

- Controller initialization failed: 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_POSITIONER_NAME (-18)
- Valid output parameter type: ERR_WRONG_TYPE_DOUBLE (-14)
- Check input parameter values: ERR_PARAMETER_OUT_OF_RANGE (-17)
- 0 < WarningFollowingError <= FatalFollowingError

#### Description

This function sets a new value of the warning following error for a positioner.

**NOTE**

**This function can be used only with the XPS-Qn Precision Platform controller.**

#### 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_PARAMETER_OUT_OF_RANGE (-17)
- SUCCESS (0): no error

## TCL

### Prototype

**PositionerWarningFollowingErrorSet** $SocketID $FullPositionerName $WarningFollowingError

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | WarningFollowingError | double | Warning following error (units) |

### Output parameters

– None

### Return

| | | | |
|---|---|---|---|
| – | Error | int | TCL error code (0=success or 1=syntax error) or function error code |

## C/C++

### Prototype

int **PositionerWarningFollowingErrorSet** (int SocketID, char FullPositionerName[250] , double WarningFollowingError)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | char * | Positioner name |
| – | WarningFollowingError | double | Warning following error (units) |

### Output parameters

– None

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |

## Visual Basic

### Prototype

Long **PositionerWarningFollowingErrorSet** (ByVal SocketID As Long, ByVal FullPositionerName As String, WarningFollowingError As Double)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |
| – | WarningFollowingError | double | Warning following error (units) |

### Output parameters

– None

### Return

| | | | |
|---|---|---|---|
| – | Error | long | Function error code |

## Matlab

### Prototype

[Error] **PositionerWarningFollowingErrorSet** (int32 SocketID, cstring FullPositionerName, double WarningFollowingError)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | cstring | Positioner name |
| – WarningFollowingError | double | Warning following error (units) |

### Return

| | | |
|---|---|---|
| – Error | int32 | Function error code |

## Python

### Prototype

[Error] **PositionerWarningFollowingErrorSet** (integer SocketID, string FullPositionerName, double WarningFollowingError)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | string | Positioner name |
| – WarningFollowingError | double | Warning following error (units) |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

Configuration Files).

**NOTE**

**The maximum number of positioners in a same group is limited to 8.**

### 3.3.3 Function Description

#### 3.3.3.1 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)
- HXP 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 gets by the "TCP_ConnectToServer" function |
| – | GroupName | string | Group name or Positioner name (maximum size = 250) |

**Output parameters**

| | | | |
|---|---|---|---|
| – | CorrectorOutput | floating point | Corrector output |

**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 * CorrectorOutput)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer"function |
| – GroupName | char * | Group name or Positioner name |
| – NbPositioners | int | Number of positioners in the selected group (1 if positioner) |

### Output parameters

| | | |
|---|---|---|
| – CorrectorOutput | double * | Corrector output array |

### Return

– Function error code

## Visual Basic

### Prototype

Long **GroupCorrectorOutputGet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal NbPositioners As Long, CorrectorOutput As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | Long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – GroupName | String | Group name or Positioner name |
| – NbPositioners | Long | Number of positioners in the selected group (1 if positioner) |

### Output parameters

| | | |
|---|---|---|
| – CorrectorOutput | DoublePtr | Corrector output array |

### Return

– Function error code

## Matlab

### Prototype

[Error, CorrectorOutput] **GroupCorrectorOutputGet** (int32 SocketID, cstring GroupName, int32 NbPositioners )

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "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 | double | Corrector output array |

## Python

### Prototype

[Error, CorrectorOutput] **GroupCorrectorOutputGet** (integer SocketID, string GroupName, integer NbPositioners)

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "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 |
| – CorrectorOutput | double | Corrector output array |

### 3.3.3.2 GroupInitialize

#### Name

**GroupInitialize** - Initializes the motor and activates the servo loop of the selected group.

#### Input tests

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer"function |
| – | GroupName | string | Group name |

**Return**

– Function error code

### 3.3.3.3 GroupInitializeWithEncoderCalibration

#### Name

**GroupInitializeWithEncoderCalibration** - Initializes motor, calibrates encoder and activates servo loop.

#### Input tests

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer"function |
| – GroupName | string | Group name |

### Return

– Function error code

### 3.3.3.4 GroupHomeSearch

#### Name

**GroupHomeSearch** - Initiates a home search.

#### Input tests

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer"function |
| – | GroupName | string | Group name |

**Return**

– Function error code

### 3.3.3.5 GroupHomeSearchAndRelativeMove

#### Name

**GroupHomeSearchAndRelativeMove** - Initiates a home search followed by a relative move.

#### Input tests

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 gets by the "TCP_ConnectToServer" function |
| – | GroupName | string | Group name (maximum size = 250) |
| – | Displacement | floating point | Relative displacement (must be repeated for each positioner of the selected 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 gets by the "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

## 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 gets by the "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

## Matlab

### Prototype

int32 **GroupHomeSearchAndRelativeMove** (int32 SocketID, cstring GroupName, doublePtr Displacement)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer"function |
| – GroupName | cstring | SingleAxis group name |
| – Displacement | doublePtr | Relative displacement array |

### Return

– Function error code

## Python

### Prototype

integer **GroupHomeSearchAndRelativeMove** (integer SocketID, string GroupName, doublePtr Displacement)

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer"function |
| – GroupName | string | SingleAxis group name |
| – Displacement | doublePtr | Relative displacement array |

### Return

– Function error code

### 3.3.3.6 GroupKill

#### Name

**GroupKill** - Kills the selected group to go in the "not initialized" status.

#### Input tests

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer"function |
| – | GroupName | string | Group name |

### Return

– Function error code

### 3.3.3.7 GroupMotionDisable

#### Name

**GroupMotionDisable** – Disables a "ready" group.

#### Input tests

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer"function |
| – | GroupName | string | Group name |

### Return

– Function error code

### 3.3.3.8 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)
- HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer"function |
| – | GroupName | string | Group name |

### Return

– Function error code

### 3.3.3.9 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)
- HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer"function |
| – | GroupName | string | Group name |

**Return**

– Function error code

### 3.3.3.10 GroupMoveAbsolute

#### Name

**GroupMoveAbsolute** - Initiates an absolute move for a positioner or a group.

#### Input tests

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 gets by the "TCP_ConnectToServer" function |
| – | GroupName | string | Group name (maximum size = 250) |
| – | TargetPosition | floating point | Target position (must be repeated for each positioner of the selected 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 gets by the "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

## 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 gets by the "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

## Matlab

### Prototype

int32 **GroupMoveAbsolute** (int32 SocketID, cstring GroupName, DoublePtr TargetPosition)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer"function |
| – GroupName | cstring | SingleAxis group name |
| – TargetPosition | doublePtr | Target position array |

### Return

– Function error code

## Python

### Prototype

integer **GroupMoveAbsolute** (integer SocketID, string GroupName, doublePtr TargetPosition)

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer"function |
| – GroupName | string | SingleAxis group name |
| – TargetPosition | doublePtr | Target position array |

### Return

– Function error code

### 3.3.3.11 GroupMoveRelative

#### Name

**GroupMoveRelative** - Initiates a relative move for a positioner or a group.

#### Input tests

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 gets by the "TCP_ConnectToServer" function |
| – | GroupName | string | Group name (maximum size = 250) |
| – | Displacement | floating point | Relative displacement (must be repeated for each positioner of the selected 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 gets by the "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

## 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 gets by the "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

## Matlab

### Prototype

– int32 **GroupMoveRelative** (int32 SocketID, cstring GroupName, DoublePtr Displacement)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer"function |
| – | GroupName | cstring | SingleAxis group name |
| – | Displacement | doublePtr | Relative displacement array |

### Return

– Function error code

## Python

### Prototype

integer **GroupMoveRelative** (integer SocketID, string GroupName, doublePtr Displacement)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer"function |
| – | GroupName | string | SingleAxis group name |
| – | Displacement | doublePtr | Relative displacement array |

### Return

– Function error code

### 3.3.3.12 GroupPositionCurrentGet

#### Name

**GroupPositionCurrentGet** – When queried for a group, returns current position of the Tool coordinate system in the Work coordinate system with X, Y, Z, U, V, W values. Otherwise, can return the current position for one positioners or coordinate of the selected group.

#### Input tests

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 gets by the "TCP_ConnectToServer" function |
| – GroupName | string | Group name / Positioner 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 gets by the "TCP_ConnectToServer"function |
| – GroupName | char * | Group name / Positioner Name |
| – NbPositioners | int | Number of positioners in the selected group |

### Output parameters

| | | |
|---|---|---|
| – CurrentPosition | double * | Current position array |

### Return

– Function error code

## 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 gets by the "TCP_ConnectToServer" function |
| – GroupName | String | Group name / Positioner Name |
| – NbPositioners | Long | Number of positioners in the selected group |

### Output parameters

| | | |
|---|---|---|
| – CurrentPosition | Double | Current position array |

### Return

– Function error code

## Matlab

### Prototype

[Error, CurrentPosition] **GroupPositionCurrentGet** (int32 SocketID, cstring GroupName)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer"function |
| – GroupName | cstring | Group name / Positioner Name |

### Return

| | | |
|---|---|---|
| – Error | int32 | Function error code |
| – CurrentPosition | doublePtr | Current position array |

## Python

### Prototype

[Error, CurrentPosition] **GroupPositionCurrentGet** (integer SocketID, string GroupName)

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer"function |
| – GroupName | string | Group name / Positioner Name |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error code |
| – CurrentPosition | doublePtr | Current position array |

### 3.3.3.13 GroupPositionSetpointGet

#### Name

**GroupPositionSetpointGet** – When queried for a group, returns setpoint position of the Tool coordinate system in the Work coordinate system with X, Y, Z, U, V, W values. Otherwise, can return the setpoint position for one positioners or coordinate of the selected group.

#### Input tests

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 gets by the "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 gets by the "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

## 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 gets by the "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

## Matlab

### Prototype

[Error, SetpointPosition] **GroupPositionSetpointGet** (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 |
| – SetpointPosition | doublePtr | Setpoint position array |

## Python

### Prototype

[Error, SetpointPosition] **GroupPositionSetpointGet** (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 |
| – SetpointPosition | doublePtr | Setpoint position array |

### 3.3.3.14 GroupPositionTargetGet

#### Name

**GroupPositionTargetGet** – When queried for a group, returns target position of the Tool coordinate system in the Work coordinate system with X, Y, Z, U, V, W values. Otherwise, can return the current position for one positioners or coordinate of the selected group.

#### Input tests

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 gets by the "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 gets by the "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

## 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 gets by the "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

## Matlab

### Prototype

[Error, TargetPosition] **GroupPositionTargetGet** (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 |
| – TargetPosition | doublePtr | Target position array |

## Python

### Prototype

[Error, TargetPosition] **GroupPositionTargetGet** (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 |
| – TargetPosition | doublePtr | Target position array |

### 3.3.3.15 GroupReadyAtPosition

#### Name

**GroupReadyAtPosition** – Go to READY state without home search at predefined position.

#### Input tests

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 allows going to READY state without performs a home search. The positions are initialized with the user predefined positions.

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 "READY".

#### 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

**GroupReadyAtPosition** $SocketID $GroupName $position1 $position2 $position3 $position4 $position5 $position6

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | GroupName | string | Group name (maximum size = 250) |
| – | Position1 | double | User position for positioner 1 |
| – | Position2 | double | User position for positioner 2 |
| – | Position3 | double | User position for positioner 3 |
| – | Position4 | double | User position for positioner 4 |
| – | Position5 | double | User position for positioner 5 |
| – | Position6 | double | User position for positioner 6 |

### Output parameters

– None

### Return

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

## C/C++

### Prototype

int **GroupReadyAtPosition** (int SocketID, char *GroupName, double position1, double position2, double position3, double position4, double position5, double position6)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | GroupName | char * | Group name |
| – | Position1 | double | User position for positioner 1 |
| – | Position2 | double | User position for positioner 2 |
| – | Position3 | double | User position for positioner 3 |
| – | Position4 | double | User position for positioner 4 |
| – | Position5 | double | User position for positioner 5 |
| – | Position6 | double | User position for positioner 6 |

### Output parameters

– None

### Return

– Function error code

## Visual Basic

### Prototype

Long **GroupReadyAtPosition** (ByVal SocketID As Long, ByVal GroupName As String, ByVal position1 As Double, ByVal position2 As Double, ByVal position3 As Double, ByVal position4 As Double, ByVal position5 As Double, ByVal position6 As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | Long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – GroupName | String | Group name |
| – Position1 | double | User position for positioner 1 |
| – Position2 | double | User position for positioner 2 |
| – Position3 | double | User position for positioner 3 |
| – Position4 | double | User position for positioner 4 |
| – Position5 | double | User position for positioner 5 |
| – Position6 | double | User position for positioner 6 |

### Output parameters

– None

### Return

– Function error code

## Matlab

### Prototype

int32 **GroupReadyAtPosition** (int32 SocketID, cstring GroupName, double position1, double position2, double position3, double position4, double position5, double position6)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" function |
| – GroupName | cstring | Group name |
| – Position1 | double | User position for positioner 1 |
| – Position2 | double | User position for positioner 2 |
| – Position3 | double | User position for positioner 3 |
| – Position4 | double | User position for positioner 4 |
| – Position5 | double | User position for positioner 5 |
| – Position6 | double | User position for positioner 6 |

### Return

– Function error code

## Python

### Prototype

integer **GroupReadyAtPosition** (integer SocketID, string GroupName, double position1, double position2, double position3, double position4, double position5, double position6)

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" function |
| – GroupName | string | Group name |
| – Position1 | double | User position for positioner 1 |
| – Position2 | double | User position for positioner 2 |
| – Position3 | double | User position for positioner 3 |
| – Position4 | double | User position for positioner 4 |
| – Position5 | double | User position for positioner 5 |
| – Position6 | double | User position for positioner 6 |

### Return

– Function error code

### 3.3.3.16 GroupStatusGet

#### Name

**GroupStatusGet** – Returns the group status code.

#### Input tests

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 State List" §3.14.

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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer"function |
| – | GroupName | string | Group name |

### Return

| | | | |
|---|---|---|---|
| – | Error | integer | Function error code |
| – | GroupStatus | integer | Status of the group |

### 3.3.3.17 GroupReferencingActionExecute

**NOTE**

**This function is disabled for Hexapod group.**

#### Name

**GroupReferencingActionExecute** – Initiates the given action, with the given sensor and parameters.

#### Input tests

- Controller initialization failed: 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 ≤ MaximumVelocity
- Parameter ≠ 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 the XPS User's manual referencing section.

| **Action list** | Sensor **to be defined** |
|---|---|
| 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 occurs during the referencing and motion is in progress, an emergency brake is applied and ERR_FOLLOWING_ERROR (-25) is returned. The group state becomes "NOTINIT".

If the home search time out is reached, ERR_GROUP_HOME_SEARCH_TIMEOUT (-28) is returned. The group state becomes "NOTINIT".

When referencing is done, you can exit the "REFERENCING" state to go in "READY" state with the "GroupReferencingStop" function.

**CAUTION**

**This function must be 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 | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | PositionerName | string | Positioner name (maximum size = 250) |
| – | ActionString | Referencing action name | |
| – | SensorString | 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 gets by the "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 gets by the "TCP_ConnectToServer" function |
| – PositionerName | string | Positioner name |
| – ActionString | Referencing action name | |
| – SensorString | 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 gets by the "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 | int | Socket identifier gets by the "TCP_ConnectToServer"function |
| – PositionerName | string | Positioner name |
| – ActionString | Referencing action name | |
| – SensorString | Referencing sensor name | |
| – Parameter | double | Referencing parameter (related to the referencing action) |

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

### 3.3.3.18 GroupReferencingStart

**NOTE**

**This function is disabled for Hexapod group.**

#### Name

**GroupReferencingStart** – Starts the referencing mode.

#### Input tests

- Controller initialization failed: 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 ERR_NOT_ALLOWED_ACTION (-22) is returned.

To stop the referencing mode and to go in the "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 | int | Socket identifier gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer"function |
| – | GroupName | cstring | Group name |

### Return

| | | | |
|---|---|---|---|
| – | Error | int32 | Function error code |

## Python

### Prototype

[Error] **GroupReferencingStart** (integer SocketID, string GroupName)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer"function |
| – | GroupName | string | Group name |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |

### 3.3.3.19 GroupReferencingStop

**NOTE**

**This function is disabled for Hexapod group.**

#### Name

**GroupReferencingStop** – Stops the referencing mode.

#### Input tests

- Controller initialization failed: 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 state to "READY".

To use this function, the selected group must be in "REFERENCING" state, else ERR_NOT_ALLOWED_ACTION (-22) is returned.

The travel limits are checked before stopping referencing mode. The ERR_TRAVEL_LIMITS (-35) is returned if the profiler position is out of range of the software travel limits and 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 | int | Socket identifier gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer"function |
| – | GroupName | cstring | Group name |

### Return

| | | | |
|---|---|---|---|
| – | Error | int32 | Function error code |

## Python

### Prototype

[Error] **GroupReferencingStop** (integer SocketID, string GroupName)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer"function |
| – | GroupName | string | Group name |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |

### 3.3.3.20 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)
- HXP controller 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 gets by the "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 gets by the "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

## 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 gets by the "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

## Matlab

### Prototype

[Error, CurrentVelocity] **GroupVelocityCurrentGet** (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 |
| – CurrentVelocity | doublePtr | Current Velocity array |

## Python

### Prototype

[Error, CurrentVelocity] **GroupVelocityCurrentGet** (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 |
| – CurrentVelocity | doublePtr | Current Velocity array |

## 3.4 SingleAxis Group

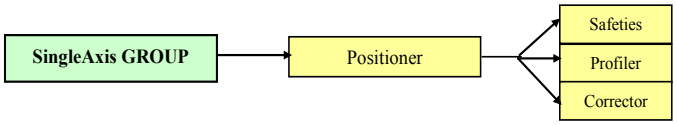
### 3.4.1 Description

The SingleAxis is composed of one single positioner for the execution of motion commands.

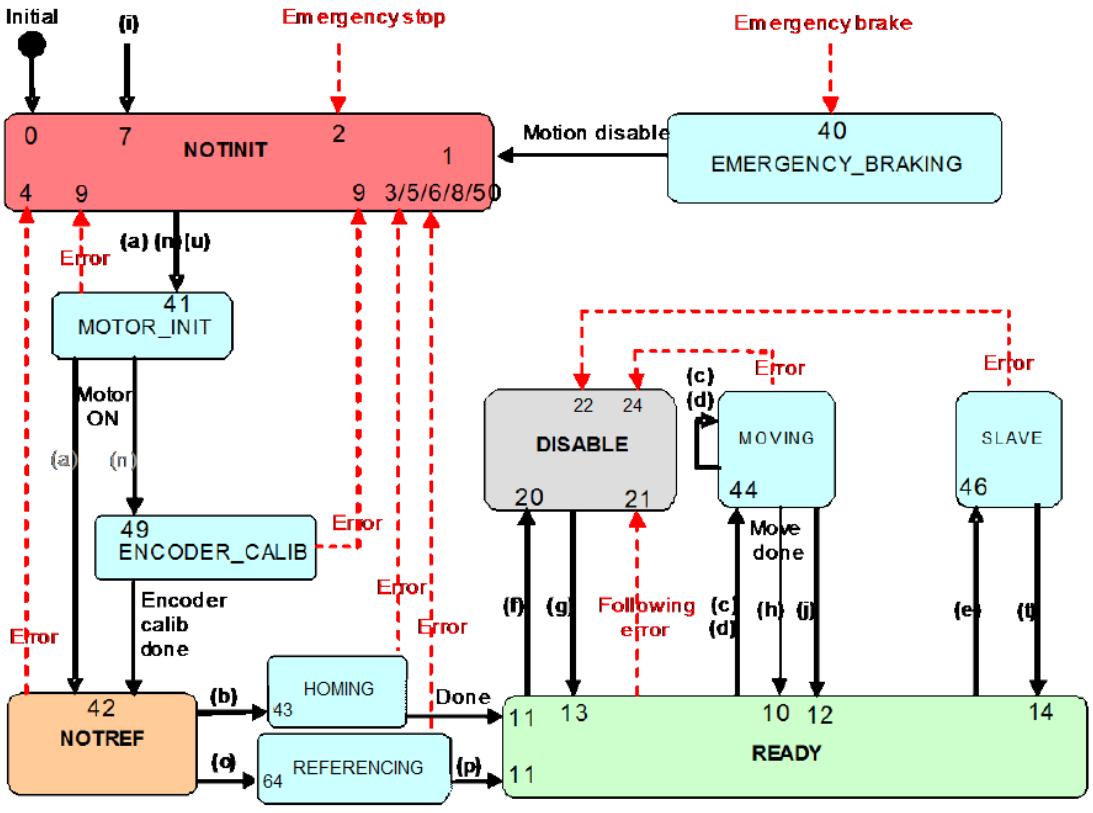
The XPS controller can handle several SingleAxis objects (1 to 2 max).

There is no relation between SingleAxis objects and other objects handled by the controller.

There is no relation between SingleAxis objects and other objects handled by the controller.




### 3.4.2 State Diagram




## Called functions:

| | | |
|---|---|---|
| **(a)** GroupInitialize | **(g)** GroupMotionEnable | **(t)** GroupSlaveModeDisable |
| **(b)** GroupHomeSearch | **(h)** GroupMoveAbort | **(u)** GroupInitializeNoEncoderReset |
| **(c)** GroupMoveAbsolute | **(i)** GroupKill or KillAll | |
| **(d)** GroupMoveRelative | **(n)** GroupInitializeWithEncoderCalibration | |
| **(e)** GroupSlaveModeEnable | **(o)** GroupReferencingStart | |
| **(f)** GroupMotionDisable | **(p)** GroupReferencingStop | |

### 3.4.3 Specific Function Description

#### 3.4.3.1 SingleAxisSlaveModeDisable

**Name**

**SingleAxisSlaveModeDisable** – Disables the slave-master mode.

**Input tests**

- Controller initialization failed: 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 disables 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 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

**SingleAxisSlaveModeDisable** $SocketID $GroupName

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer"function |
| – | GroupName | cstring | SingleAxis group name |

### Return

– Function error code

## Python

### Prototype

integer **SingleAxisSlaveModeDisable** (integer SocketID, string GroupName)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer"function |
| – | GroupName | string | SingleAxis group name |

### Return

– Function error code

### 3.4.3.2 SingleAxisSlaveModeEnable

#### Name

**SingleAxisSlaveModeEnable** – Enables the slave-master mode.

#### Input tests

- Controller initialization failed: 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" state. 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 ERR_NOT_ALLOWED_ACTION (-22) is returned.

To use this function, the master positioner and the slave ratio must be configured using the "SingleAxisSlaveParametersSet" function. If it's not the case then ERR_SLAVE_CONFIGURATION (-41) 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 | int | Socket identifier gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer"function |
| – | GroupName | cstring | SingleAxis group name |

### Return

– Function error code

## Python

### Prototype

integer **SingleAxisSlaveModeEnable** (integer SocketID, string GroupName)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer"function |
| – | GroupName | string | SingleAxis group name |

### Return

– Function error code

### 3.4.3.3 SingleAxisSlaveParametersGet

#### Name

**SingleAxisSlaveParametersGet** – Returns the slave parameters.

#### Input tests

- Controller initialization failed: 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 | int | Socket identifier gets by the "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, int NbPositioners, char * MasterPositionerName , double * Ratio)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "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 gets by the "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 gets by the "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 | int | Socket identifier gets by the "TCP_ConnectToServer"function |
| – | GroupName | string | Group name |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |
| – | MasterPositionerName | string | Master positioner name from any group |
| – | Ratio | double | Gear ratio between the master and the slave |

### 3.4.3.4 SingleAxisSlaveParametersSet

#### Name

**SingleAxisSlaveParametersSet** – Sets the slave parameters.

#### Input tests

- Controller initialization failed: 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 copy of the master 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 a spindle group. If the master group is a spindle then ERR_NOT_ALLOWED_ACTION (-22) is returned. The master positioner must be different from the slave positioner else ERR_WRONG_OBJECT_TYPE (-8) is returned.

**NOTE**

**After an emergency stop, the master group and the slave group are in "NOTINIT" status. To restart a master-slave relation, the slave group(s) must be reinitialized 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 | int | Socket identifier gets by the "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, int NbPositioners, char * MasterPositionerName , double Ratio)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "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 gets by the "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 gets by the "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 | int | Socket identifier gets by the "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 | int | Function error code |

### 3.4.4 Configuration Files

An example of a SingleAxis group (named "MySingleAxis") is in the system.ini file below. The "MySingleAxis" group is comprised of a positioner named "MyPositioner".

The positioner "MyPositioner" uses the parameters of "MYSTAGE" from the stages.ini file and is connected to the plug 7 of the XPS controller.

**System.ini file:**

```
[GROUPS]
SingleAxisInUse = MySingleAxis

[MySingleAxis]    ; Configuration of "MySingleAxis" SingleAxis grp
PositionerInUse = MyPositioner

[MySingleAxis.MyPositioner]
PlugNumber = 7    ; 7 or 8
StageName = MYSTAGE

; --- Time flasher
TimeFlasherBaseFrequency =    ; default value 40e6, must be between 39.5e6
                              and 40.5e6 Hz
```

**Stages.ini file:**

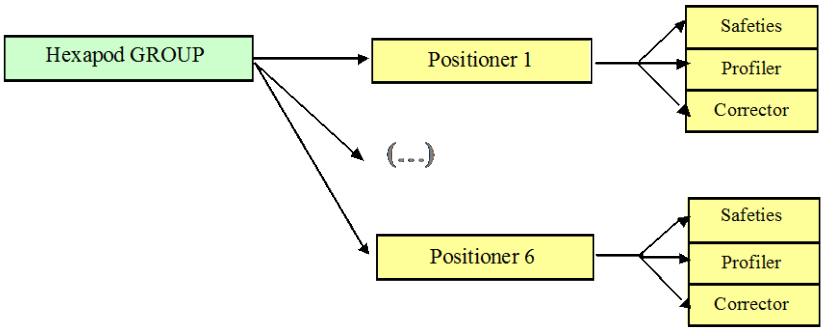
```
[MYSTAGE]

MYSTAGE configuration => See § "Positioner: Configuration files"
```

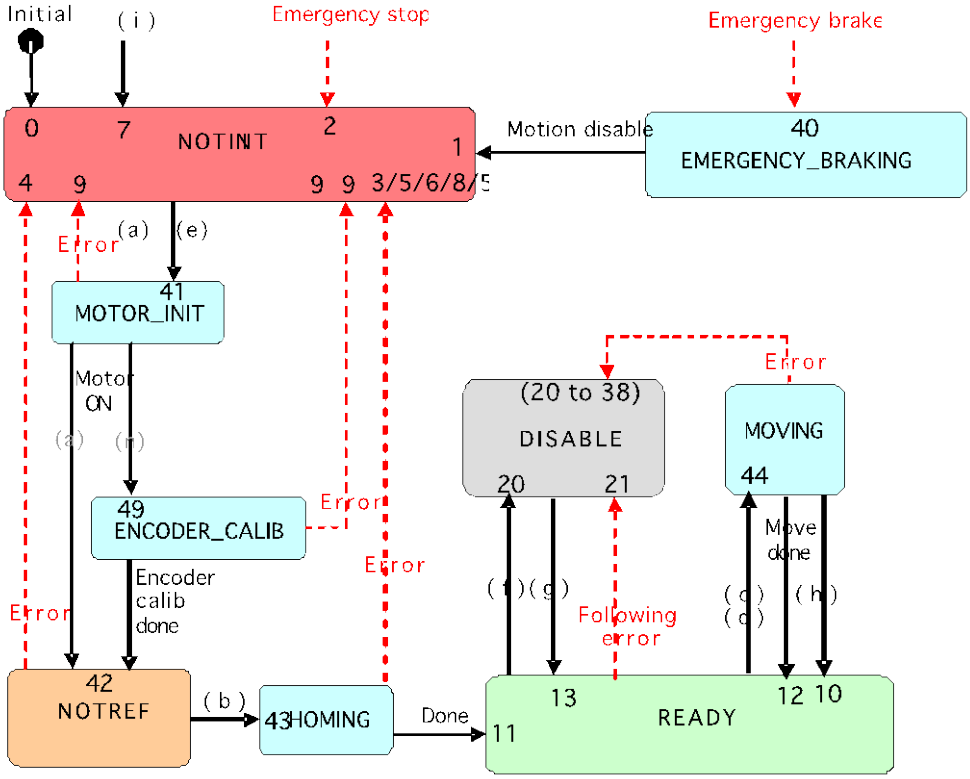
## 3.5 Hexapod Group

### 3.5.1 Description

A Hexapod is a group of 6 positioners




### 3.5.2 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)** GroupInitializeWithEncoderCalibration

**(f)** GroupMotionDisable

**(g)** GroupMotionEnable

**(h)** GroupMoveAbort

**(i)** GroupKill or KillAll

### 3.5.3 Specific Function Description

#### 3.5.3.1 HexapodCoordinatesGet

**Name**

**HexapodCoordinatesGet** – Takes a point X, Y, Z, U, V, W as input in a specified coordinate system (Work, Tool, Base) and give its position in another coordinate system (Work, Tool, Base)

**Input tests**

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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: ERR_POSITIONER_NAME (-18)
- Check the input parameter type: ERR_WRONG_TYPE_INT (-15)
- Check the input value (Number of executions > 0): ERR_PARAMETER_OUT_OF_RANGE (-17)

**Description**

This function transform a position (specified in one coordinate system – Work, Tool or Base) into its position in one other coordinate system (Work, Tool or Base).

**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_STRING_TOO_LONG (-3)
- 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

**HexapodCoordinatesGet** $SocketID $GroupName $CoordinateSystemIn $CoordinateSystemOut $Xin $Yin $Zin $Uin $Vin $Win Xout yout Zout Uout Vout Wout

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | GroupName | string | Hexapod group name (maximum size = 250) |
| – | CoordinateSystemIn | string | Input coordinate system: has to be Work, Tool or Base |
| – | CoordinateSystemOut | string | Output coordinate system: has to be Work, Tool or Base |
| – | Xin, Yin, Zin, Uin, Vin, Win | floating point | Input positions |
| – | Xout, Yout, Zout, Uout, Vout, Wout | floating point | Output positions |

### Output parameters

– None

### Return

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

## C/C++

### Prototype

int **HexapodCoordinatesGet** (int SocketID, char *GroupName , char * CoordinateSystemIn, char * CoordinateSystemOut, double Xin, double Yin, …, double* Xout, double* Yout, …)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer"function |
| – | GroupName | char * | Hexapod group name |
| – | CoordinateSystemIn | char * | Input coordinate system: has to be Work, Tool or Base |
| – | CoordinateSystemOut | char * | Output coordinate system: has to be Work, Tool or Base |
| – | Xin, Yin, Zin, Uin, Vin, Win | double | Input positions |
| – | Xout, Yout, Zout, Uout, Vout, Wout | double * | Output positions |

### Output parameters

– None

### Return

– Function error code

## Visual Basic

### Prototype

Long **HexapodCoordinatesGet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal CoordinateSystemIn As String, ByVal CoordinateSystemOut As String, ByVal Xin As Double, ByVal Yin As Double, …, Xout As Double, Yout As Double, …)

### Input parameters

| | | | |
|---|---|---|---|
| – SocketID | Long | Socket identifier gets by the "TCP_ConnectToServer" function | |
| – GroupName | String | Hexapod group name | |
| – CoordinateSystemIn | String | Input coordinate system: has to be Work, Tool or Base | |
| – CoordinateSystemIn | String | Output coordinate system: has to be Work, Tool or Base | |
| – Xin, Yin, Zin, Uin, Vin, Win | | Double | Input positions |
| – Xout, Yout, Zout, Uout, Vout, Wout | | Double | Output positions |

### Output parameters

– None

### Return

– Function error code

## Matlab

### Prototype

int32 **HexapodCoordinatesGet** (int32 SocketID, cstring GroupName, cstring CoordinateSystemIn, cstring CoordinateSystemOut, doublePtr Xin, doublePtr Yin, …, doublePtr Xout, doublePtr Yout, …)

### Input parameters

| | | | |
|---|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" | |
| – GroupName | cstring | Hexapod group name | |
| – CoordinateSystemIn | cstring | Input coordinate system: has to be Work, Tool or Base | |
| – CoordinateSystemIn | cstring | Output coordinate system: has to be Work, Tool or Base | |
| – Xin, Yin, Zin, Uin, Vin, Win | | doublePtr | Input positions |
| – Xout, Yout, Zout, Uout, Vout, Wout | | doublePtr | Output positions |

### Return

– Function error code

## Python

### Prototype

integer **HexapodCoordinatesGet** (integer SocketID, string GroupName, string CoordinateSystemIn, string CoordinateSystemOut, doublePtr Xin, doublePtr Yin, …, doublePtr Xout, doublePtr Yout, …)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer"function |
| – | GroupName | string | Hexapod group name |
| – | CoordinateSystemIn | string | Input coordinate system: has to be Work, Tool or Base |
| – | CoordinateSystemIn | string | Output coordinate system: has to be Work, Tool or Base |
| – | Xin, Yin, Zin, Uin, Vin, Win | doublePtr | Input positions |
| – | Xout, Yout, Zout, Uout, Vout, Wout | doublePtr | Output positions |

### Return

– Function error code

### 3.5.3.2 HexapodCoordinateSystemGet

#### Name

**HexapodCoordinateSystemGet** – Get the position of a coordinate system.

#### Input tests

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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: ERR_POSITIONER_NAME (-18)
- Check the input parameter type: ERR_WRONG_TYPE_INT (-15)
- Check the input value (Number of executions > 0):
  ERR_PARAMETER_OUT_OF_RANGE (-17)

#### Description

This function allows you to get the position of a coordinate system in its predecessor.

- Work is defined in World
- Base is defined in World
- Tool is defined in Carriage

#### 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_STRING_TOO_LONG (-3)
- 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

**HexapodCoordinateSystemGet** $SocketID $GroupName $CoordinateSystem X Y Z U V W

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" function |
| – GroupName | string | Hexapod group name (maximum size = 250) |
| – CoordinateSystem | string | Has to be Work, Base or Tool |
| – X, Y, Z, U, V, W | floating point | Positions |

### Output parameters

– None

### Return

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

## C/C++

### Prototype

int **HexapodCoordinateSystemGet** (int SocketID, char *GroupName , char * CoordinateSystem, double* X, double * Y, double * Z, double * U, double * V, double * W)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer"function |
| – GroupName | char * | Hexapod group name |
| – CoordinateSystem | char * | Has to be Work or Tool |
| – X, Y, Z, U, V, W | double * | Positions |

### Output parameters

– None

### Return

– Function error code

## Visual Basic

### Prototype

Long **HexapodCoordinateSystemGet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal CoordinateSystem As String, ByVal X As Double, ByVal Y As Double, …)

### Input parameters

| | | |
|---|---|---|
| – SocketID | Long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – GroupName | String | Hexapod group name |
| – CoordinateSystem | String | Has to be Work or Tool |
| – X, Y, Z, U, V, W | Double | Positions |

### Output parameters

– None

### Return

– Function error code

## Matlab

### Prototype

int32 **HexapodCoordinateSystemGet** (int32 SocketID, cstring GroupName, cstring CoordinateSystem, doublePtr X, doublePtr Y, …)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer"function |
| – GroupName | cstring | Hexapod group name |
| – CoordinateSystem | cstring | Has to be Work or Tool |
| – X, Y, Z, U, V, W | doublePtr | Positions |

### Return

– Function error code

## Python

### Prototype

integer **HexapodCoordinateSystemGet** (integer SocketID, string GroupName, string CoordinateSystem, doublePtr X, doublePtr Y, …)

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer"function |
| – GroupName | string | Hexapod group name |
| – CoordinateSystem | string | Has to be Work or Tool |
| – X, Y, Z, U, V, W | doublePtr | Positions |

### Return

– Function error code

### 3.5.3.3 HexapodCoordinateSystemSet

#### Name

**HexapodCoordinateSystemSet** – Modify on-the-fly the position of a coordinate system.

#### Input tests

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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: ERR_POSITIONER_NAME (-18)
- Check the input parameter type: ERR_WRONG_TYPE_INT (-15)
- Check the input value (Number of executions > 0): ERR_PARAMETER_OUT_OF_RANGE (-17)

#### Description

This function allows you to set the position of a coordinate system in its predecessor.

- Work is defined in World
- Base is defined in World
- Tool is defined in Carriage

The modification is NOT recorded in the configuration files, thus will be forgotten at next reboot.

#### 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_STRING_TOO_LONG (-3)
- 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

**HexapodCoordinateSystemSet** $SocketID $GroupName $CoordinateSystem $X $Y $Z $U $V $W

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" function |
| – GroupName | string | Hexapod group name (maximum size = 250) |
| – CoordinateSystem | string | Has to be Work, Base or Tool |
| – X, Y, Z, U, V, W | floating point | Positions |

### Output parameters

– None

### Return

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

## C/C++

### Prototype

int **HexapodCoordinateSystemSet** (int SocketID, char *GroupName , char * CoordinateSystem, double X, double Y, double Z, double U, double V, double W)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer"function |
| – GroupName | char * | Hexapod group name |
| – CoordinateSystem | char * | Has to be Work or Tool |
| – X, Y, Z, U, V, W | double | Positions |

### Output parameters

– None

### Return

– Function error code

## Visual Basic

### Prototype

Long **HexapodCoordinateSystemSet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal CoordinateSystem As String, ByVal X As Double, ByVal Y As Double, …)

### Input parameters

| | | |
|---|---|---|
| – SocketID | Long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – GroupName | String | Hexapod group name |
| – CoordinateSystem | String | Has to be Work or Tool |
| – X, Y, Z, U, V, W | Double | Positions |

### Output parameters

– None

### Return

– Function error code

## Matlab

### Prototype

int32 **HexapodCoordinateSystemSet** (int32 SocketID, cstring GroupName, cstring CoordinateSystem, doublePtr X, doublePtr Y, …)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer"function |
| – GroupName | cstring | Hexapod group name |
| – CoordinateSystem | cstring | Has to be Work or Tool |
| – X, Y, Z, U, V, W | doublePtr | Positions |

### Return

– Function error code

## Python

### Prototype

integer **HexapodCoordinateSystemSet** (integer SocketID, string GroupName, string CoordinateSystem, doublePtr X, doublePtr Y, …)

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer"function |
| – GroupName | string | Hexapod group name |
| – CoordinateSystem | string | Has to be Work or Tool |
| – X, Y, Z, U, V, W | doublePtr | Positions |

### Return

– Function error code

### 3.5.3.4 HexapodMoveAbsolute

#### Name

**HexapodMoveAbsolute** – Do an absolute move of the Tool coordinate system in the Work coordinate system.

#### Input tests

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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: ERR_POSITIONER_NAME (-18)
- Check the input parameter type: ERR_WRONG_TYPE_INT (-15)
- Check the input value (Number of executions > 0):
  ERR_PARAMETER_OUT_OF_RANGE (-17)

#### Description

This function executes an absolute move.

#### 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_STRING_TOO_LONG (-3)
- 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

**HexapodMoveAbsolute** $SocketID $GroupName $CoordinateSystem $X $Y $Z $U $V $W

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" function |
| – GroupName | string | Hexapod group name (maximum size = 250) |
| – CoordinateSystem | string | Has to be Work |
| – X, Y, Z, U, V, W | floating point | Positions |

### Output parameters

– None

### Return

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

## C/C++

### Prototype

int **HexapodMoveAbsolute** (int SocketID, char *GroupName , char * CoordinateSystem, double X, double Y, double Z, double U, double V, double W)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer"function |
| – GroupName | char * | Hexapod group name |
| – CoordinateSystem | char * | Has to be Work |
| – X, Y, Z, U, V, W | double | Positions |

### Output parameters

– None

### Return

– Function error code

## Visual Basic

### Prototype

Long **HexapodMoveAbsolute** (ByVal SocketID As Long, ByVal GroupName As String, ByVal CoordinateSystem As String, ByVal X As Double, ByVal Y As Double, …)

### Input parameters

| | | |
|---|---|---|
| – SocketID | Long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – GroupName | String | Hexapod group name |
| – CoordinateSystem | String | Has to be Work |
| – X, Y, Z, U, V, W | Double | Positions |

### Output parameters

– None

### Return

– Function error code

## Matlab

### Prototype

int32 **HexapodMoveAbsolute** (int32 SocketID, cstring GroupName, cstring CoordinateSystem, doublePtr X, doublePtr Y, …)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer"function |
| – GroupName | cstring | Hexapod group name |
| – CoordinateSystem | cstring | Has to be Work |
| – X, Y, Z, U, V, W | doublePtr | Positions |

### Return

– Function error code

## Python

### Prototype

integer **HexapodMoveAbsolute** (integer SocketID, string GroupName, string CoordinateSystem, doublePtr X, doublePtr Y, …)

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer"function |
| – GroupName | string | Hexapod group name |
| – CoordinateSystem | string | Has to be Work |
| – X, Y, Z, U, V, W | doublePtr | Positions |

### Return

– Function error code

### 3.5.3.5 HexapodMoveIncremental

#### Name

**HexapodMoveIncremental** – Do an incremental move in the Tool coordinate system or in the Work coordinate system.

#### Input tests

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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: ERR_POSITIONER_NAME (-18)
- Check the input parameter type: ERR_WRONG_TYPE_INT (-15)
- Check the input value (Number of executions > 0):
  ERR_PARAMETER_OUT_OF_RANGE (-17)

#### Description

This function executes an incrmental move.

#### 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_STRING_TOO_LONG (-3)
- 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

**HexapodMoveIncremental** $SocketID $GroupName $CoordinateSystem $X $Y $Z $U $V $W

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" function |
| – GroupName | string | Hexapod group name (maximum size = 250) |
| – CoordinateSystem | string | Has to be Work or Tool |
| – X, Y, Z, U, V, W | floating point | Positions |

### Output parameters

– None

### Return

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

## C/C++

### Prototype

int **HexapodMoveIncremental** (int SocketID, char *GroupName , char * CoordinateSystem, double X, double Y, double Z, double U, double V, double W)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer"function |
| – GroupName | char * | Hexapod group name |
| – CoordinateSystem | char * | Has to be Work or Tool |
| – X, Y, Z, U, V, W | double | Positions |

### Output parameters

– None

### Return

– Function error code

## Visual Basic

### Prototype

Long **HexapodMoveIncremental** (ByVal SocketID As Long, ByVal GroupName As String, ByVal CoordinateSystem As String, ByVal X As Double, ByVal Y As Double, …)

### Input parameters

| | | |
|---|---|---|
| – SocketID | Long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – GroupName | String | Hexapod group name |
| – CoordinateSystem | String | Has to be Work or Tool |
| – X, Y, Z, U, V, W | Double | Positions |

### Output parameters

– None

### Return

– Function error code

## Matlab

### Prototype

int32 **HexapodMoveIncremental** (int32 SocketID, cstring GroupName, cstring CoordinateSystem, doublePtr X, doublePtr Y, …)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer"function |
| – GroupName | cstring | Hexapod group name |
| – CoordinateSystem | cstring | Has to be Work or Tool |
| – X, Y, Z, U, V, W | doublePtr | Positions |

### Return

– Function error code

## Python

### Prototype

integer **HexapodMoveIncremental** (integer SocketID, string GroupName, string CoordinateSystem, doublePtr X, doublePtr Y, …)

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer"function |
| – GroupName | string | Hexapod group name |
| – CoordinateSystem | string | Has to be Work or Tool |
| – X, Y, Z, U, V, W | doublePtr | Positions |

### Return

– Function error code

### 3.5.3.6 HexapodMoveIncrementalControl

#### Name

**HexapodMoveIncrementalControl** – Hexapod trajectory (Line, Arc or Rotation) execution.

#### *Input* tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 the coordinate system (must be "Work" or "Tool"): ERR_WRONG_COORDINATE_SYS (-116)
- Check the trajectory type (must be "Line", "Arc" or "Rotation"): ERR_WRONG_TRAJ_TYPE (-117)
- Check input parameter types: ERR_WRONG_TYPE_DOUBLE (-14)

#### Description

This function allows executing a hexapod trajectory (Line, Arc or Rotation) with the maximum velocity.

Line: the displacement value is not limited (only by a double format)

Rotation: the displacement value is between –90° and +90° (the limitation is due to Bryant representation)

Arc: the displacement value is between –90° and +90° (the limitation is due to Bryant representation) if the arc is defined with 3 axis. And if the rotation is defined with only one axis, a value greater 90° could be used.

#### 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_WRONG_COORDINATE_SYS (-116)
- ERR_WRONG_TRAJ_TYPE (-117)
- ERR_MOTION_EXECUTION_IMPOSSIBLE (-118)
- SUCCESS (0) : no error

## TCL

### Prototype

**HexapodMoveIncrementalControl** $SocketID $GroupName $CoordinateSystem $HexapodTrajectoryType $dX $dY $dZ

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | GroupName | string | Group name |
| – | CoordinateSystem | string | Coordinate System (Work or Tool) |
| – | HexapodTrajectoryType | string | Hexapod Trajectory Type (Line, Arc or Rotation) |
| – | dX | double | incremental displacement value (units) |
| – | dY | double | incremental displacement value (units) |
| – | dZ | double | incremental displacement 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 **HexapodMoveIncrementalControl** (int SocketID, char GroupName [250], char CoordinateSystem [250], char HexapodTrajectoryType[250], double dX, double dY, double dZ)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | GroupName | char * | Group name |
| – | CoordinateSystem | char * | Coordinate System (Work or Tool) |
| – | HexapodTrajectoryType | char * | Hexapod Trajectory Type (Line, Arc or Rotation) |
| – | dX | double | incremental displacement value (units) |
| – | dY | double | incremental displacement value (units) |
| – | dZ | double | incremental displacement value (units) |

### Output parameters

– None

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |

## Visual Basic

### Prototype

Long **HexapodMoveIncrementalControl** (ByVal SocketID As Long, ByVal GroupName As String, ByVal CoordinateSystem As String, ByVal HexapodTrajectoryType As String, ByVal dX As Double, ByVal dY As Double, ByVal dZ As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | Long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – GroupName | String | Group name |
| – CoordinateSystem | String | Coordinate System (Work or Tool) |
| – HexapodTrajectoryType | String | Hexapod Trajectory Type (Line, Arc or Rotation) |
| – dX | Double | incremental displacement value (units) |
| – dY | Double | incremental displacement value (units) |
| – dZ | Double | incremental displacement value (units) |

### Output parameters

– None

### Return

| | | |
|---|---|---|
| – Error | Long | Function error code |

## Matlab

### Prototype

[Error] **HexapodMoveIncrementalControl** (int32 SocketID, cstring GroupName, cstring CoordinateSystem, cstring HexapodTrajectoryType, double dX, double dY, double dZ)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" function |
| – GroupName | cstring | Group name |
| – CoordinateSystem | cstring | Coordinate System (Work or Tool) |
| – HexapodTrajectoryType | cstring | Hexapod Trajectory Type (Line, Arc or Rotation) |
| – dX | double | incremental displacement value (units) |
| – dY | double | incremental displacement value (units) |
| – dZ | double | incremental displacement value (units) |

### Return

| | | |
|---|---|---|
| – Error | int32 | Function error code |

## Python

### Prototype

[Error] **HexapodMoveIncrementalControl** (integer SocketID, string GroupName, string CoordinateSystem, string HexapodTrajectoryType, double dX, double dY, double dZ)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | GroupName | string | Group name |
| – | CoordinateSystem | string | Coordinate System (Work or Tool) |
| – | HexapodTrajectoryType | string | Hexapod Trajectory Type (Line, Arc or Rotation) |
| – | dX | double | incremental displacement value (units) |
| – | dY | double | incremental displacement value (units) |
| – | dZ | double | incremental displacement value (units) |

### Return

| | | | |
|---|---|---|---|
| – | Error | integer | Function error code |

### 3.5.3.7 HexapodMoveIncrementalControlWithTargetVelocity

#### Name

**HexapodMoveIncrementalControlWithTargetVelocity** – Hexapod trajectory (Line, Arc or Rotation) execution.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 the coordinate system (must be "Work" or "Tool"): ERR_WRONG_COORDINATE_SYS (-116)
- Check the trajectory type (must be "Line", "Arc" or "Rotation"): ERR_WRONG_TRAJ_TYPE (-117)
- Check input parameter types: ERR_WRONG_TYPE_DOUBLE (-14)

#### Description

This function allows executing a hexapod trajectory (Line, Arc or Rotation) with the specified velocity.

Line: the displacement value is not limited (only by a double format)

Rotation: the displacement value is between –90° and +90° (the limitation is due to Bryant representation)

Arc: the displacement value is between –90° and +90° (the limitation is due to Bryant representation) if the arc is defined with 3 axis. And if the rotation is defined with only one axis, a value greater 90° could be used.

#### 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_WRONG_COORDINATE_SYS (-116)
- ERR_WRONG_TRAJ_TYPE (-117)
- ERR_MOTION_EXECUTION_IMPOSSIBLE (-118)
- SUCCESS (0) : no error

## TCL

### Prototype

**HexapodMoveIncrementalControlWithTargetVelocity** $SocketID $ GroupName $CoordinateSystem $HexapodTrajectoryType $dX $dY $dZ $Velocity

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" function |
| – GroupName | string | Group name |
| – CoordinateSystem | string | Coordinate System (Work or Tool) |
| – HexapodTrajectoryType | string | Hexapod Trajectory Type (Line, Arc or Rotation) |
| – dX | double | incremental displacement value (units) |
| – dY | double | incremental displacement value (units) |
| – dZ | double | incremental displacement value (units) |
| – Velocity | double | motion velocity (units / seconds) |

### Output parameters

– None

### Return

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

## C/C++

### Prototype

int **HexapodMoveIncrementalControlWithTargetVelocity** (int SocketID, char GroupName[250], char CoordinateSystem [250], char HexapodTrajectoryType[250], double dX, double dY, double dZ, double Velocity)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – GroupName | char * | Group name |
| – CoordinateSystem | char * | Coordinate System (Work or Tool) |
| – HexapodTrajectoryType | char * | Hexapod Trajectory Type (Line, Arc or Rotation) |
| – dX | double | incremental displacement value (units) |
| – dY | double | incremental displacement value (units) |
| – dZ | double | incremental displacement value (units) |
| – Velocity | double | motion velocity (units / seconds) |

### Output parameters

– None

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

## Visual Basic

### Prototype

Long **HexapodMoveIncrementalControlWithTargetVelocity** (ByVal SocketID As Long, ByVal GroupName As String, ByVal CoordinateSystem As String, ByVal HexapodTrajectoryType As String, ByVal dX As Double, ByVal dY As Double, ByVal dZ As Double, ByVal Velocity As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | Long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – GroupName | String | Group name |
| – CoordinateSystem | String | Coordinate System (Work or Tool) |
| – HexapodTrajectoryType | String | Hexapod Trajectory Type (Line, Arc or Rotation) |
| – dX | Double | incremental displacement value (units) |
| – dY | Double | incremental displacement value (units) |
| – dZ | Double | incremental displacement value (units) |
| – Velocity | Double | motion velocity (units / seconds) |

### Output parameters

– None

### Return

| | | |
|---|---|---|
| – Error | Long | Function error code |

## Matlab

### Prototype

[Error] **HexapodMoveIncrementalControlWithTargetVelocity** (int32 SocketID, cstring GroupName, cstring CoordinateSystem, cstring HexapodTrajectoryType, double dX, double dY, double dZ, double Velocity)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" function |
| – GroupName | cstring | Group name |
| – CoordinateSystem | cstring | Coordinate System (Work or Tool) |
| – HexapodTrajectoryType | cstring | Hexapod Trajectory Type (Line, Arc or Rotation) |
| – dX | double | incremental displacement value (units) |
| – dY | double | incremental displacement value (units) |
| – dZ | double | incremental displacement value (units) |
| – Velocity | double | motion velocity (units / seconds) |

### Return

| | | |
|---|---|---|
| – Error | int32 | Function error code |

## Python

### Prototype

[Error] **HexapodMoveIncrementalControlWithTargetVelocity** (integer SocketID, string GroupName, string CoordinateSystem, string HexapodTrajectoryType, double dX, double dY, double dZ, double Velocity)

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" function |
| – GroupName | string | Group name |
| – CoordinateSystem | string | Coordinate System (Work or Tool) |
| – HexapodTrajectoryType | string | Hexapod Trajectory Type (Line, Arc or Rotation) |
| – dX | double | incremental displacement value (units) |
| – dY | double | incremental displacement value (units) |
| – dZ | double | incremental displacement value (units) |
| – Velocity | double | motion velocity (units / seconds) |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error code |

### 3.5.3.8 HexapodMoveIncrementalControlLimitGet

#### Name

**HexapodMoveIncrementalControlLimitGet** – Returns the maximum velocity of carriage and the percent of the trajectory executable

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 the coordinate system (must be "Work" or "Tool"): ERR_WRONG_COORDINATE_SYS (-116)
- Check the trajectory type (must be "Line", "Arc" or "Rotation"): ERR_WRONG_TRAJ_TYPE (-117)
- Check input parameter types: ERR_WRONG_TYPE_DOUBLE (-14)

#### Description

This function allows getting the maximum velocity of carriage and the percent of the trajectory executable

#### 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_WRONG_COORDINATE_SYS (-116)
- ERR_WRONG_TRAJ_TYPE (-117)
- SUCCESS (0) : no error

## TCL

### Prototype

**HexapodMoveIncrementalControlLimitGet** $SocketID $ GroupName $CoordinateSystem $HexapodTrajectoryType $dX $dY $dZ $Velocity MaximumVelocityCarriage TrajectoryPercent

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | GroupName | string | Group name |
| – | CoordinateSystem | string | Coordinate System (Work or Tool) |
| – | HexapodTrajectoryType | string | Hexapod Trajectory Type (Line, Arc or Rotation) |
| – | dX | double | incremental displacement value (units) |
| – | dY | double | incremental displacement value (units) |
| – | dZ | double | incremental displacement value (units) |
| – | Velocity | double | motion velocity (units / seconds) |

### Output parameters

| | | | |
|---|---|---|---|
| – | MaximumVelocityCarriage | double | maximum velocity of carriage (units / seconds) |
| – | TrajectoryPercent | double | Percent of the trajectory that can be executed |

### Return

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

## C/C++

### Prototype

int **HexapodMoveIncrementalControlLimitGet** (int SocketID, char GroupName[250], char CoordinateSystem[250], char HexapodTrajectoryType[250], double dX, double dY, double dZ, double *MaximumVelocityCarriage, double *TrajectoryPercent)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | GroupName | char * | Group name |
| – | CoordinateSystem | char * | Coordinate System (Work or Tool) |
| – | HexapodTrajectoryType | char * | Hexapod Trajectory Type (Line, Arc or Rotation) |
| – | dX | double | incremental displacement value (units) |
| – | dY | double | incremental displacement value (units) |
| – | dZ | double | incremental displacement value (units) |

### Output parameters

| | | | |
|---|---|---|---|
| – | MaximumVelocityCarriage | double | maximum velocity of carriage (units / seconds) |
| – | TrajectoryPercent | double | Percent of the trajectory that can be executed |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |

## Visual Basic

### Prototype

Long **HexapodMoveIncrementalControl** (ByVal SocketID As Long, ByVal GroupName As String, ByVal CoordinateSystem As String, ByVal HexapodTrajectoryType As String, ByVal dX As Double, ByVal dY As Double, ByVal dZ As Double, MaximumVelocityCarriage As Double, TrajectoryPercent As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | Long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – GroupName | String | Group name |
| – CoordinateSystem | String | Coordinate System (Work or Tool) |
| – HexapodTrajectoryType | String | Hexapod Trajectory Type (Line, Arc or Rotation) |
| – dX | Double | incremental displacement value (units) |
| – dY | Double | incremental displacement value (units) |
| – dZ | Double | incremental displacement value (units) |

### Output parameters

| | | |
|---|---|---|
| – MaximumVelocityCarriage | Double | maximum velocity of carriage (units / seconds) |
| – TrajectoryPercent | Double | Percent of the trajectory that can be executed |

### Return

| | | |
|---|---|---|
| – Error | Long | Function error code |

## Matlab

### Prototype

[Error, MaximumVelocityCarriage, TrajectoryPercent]
**HexapodMoveIncrementalControlLimitGet** (int32 SocketID, cstring GroupName, cstring CoordinateSystem, cstring HexapodTrajectoryType, double dX, double dY, double dZ)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" function |
| – GroupName | cstring | Group name |
| – CoordinateSystem | cstring | Coordinate System (Work or Tool) |
| – HexapodTrajectoryType | cstring | Hexapod Trajectory Type (Line, Arc or Rotation) |
| – dX | double | incremental displacement value (units) |
| – dY | double | incremental displacement value (units) |
| – dZ | double | incremental displacement value (units) |

### Return

| | | |
|---|---|---|
| – Error | int32 | Function error code |
| – MaximumVelocityCarriage | double | maximum velocity of carriage (units / seconds) |
| – TrajectoryPercent | double | Percent of the trajectory that can be executed |

## Python

### Prototype

[Error, MaximumVelocityCarriage, TrajectoryPercent]
**HexapodMoveIncrementalControlLimitGet** (integer SocketID, string GroupName, string CoordinateSystem, string HexapodTrajectoryType, double dX, double dY, double dZ)

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" function |
| – GroupName | string | Group name |
| – CoordinateSystem | string | Coordinate System (Work or Tool) |
| – HexapodTrajectoryType | string | Hexapod Trajectory Type (Line, Arc or Rotation) |
| – dX | double | incremental displacement value (units) |
| – dY | double | incremental displacement value (units) |
| – dZ | double | incremental displacement value (units) |
| – Velocity | double | motion velocity (units / seconds) |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error code |
| – MaximumVelocityCarriage | double | maximum velocity of carriage (units / seconds) |
| – TrajectoryPercent | double | Percent of the trajectory that can be executed |

### 3.5.3.9 HexapodMoveIncrementalControlPulseAndGatheringSet

#### Name

**HexapodMoveIncrementalControlPulseAndGatheringSet** – Configure gathering with pulses

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP initialization in progress: ERR_IN_INITIALIZATION (-21)
- Check the divisor value (must be positive):
  ERR_PARAMETER_OUT_OF_RANGE (-17)
- Check the number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)

#### Description

This function allows configuring gathering with pulses: gathered data are X, Y, Z, U, V, W and pulses will be generated during only constant velocity.

The gathered data are the Tool/Work position:

Hexapod.X, Hexapod.Y, Hexapod.Z,Hexapod.U, Hexapod.V, Hexapod.W

The pulses are generated during only SGamma Constant Velocity and at the same time the data'll be gathered.

The interval between 2 pulses:

Divisor * 0.0001 sec

So at the end of the displacement, the customer could send the StopAndSaveGathering API, and see the gathered data and know exactly at which position a pulse has been generated.

#### Error codes

- ERR_FATAL_INIT (-20)
- ERR_IN_INITIALIZATION (-21)
- ERR_WRONG_FORMAT (-7)
- ERR_WRONG_PARAMETERS_NUMBER (-9)
- ERR_PARAMETER_OUT_OF_RANGE (-17)
- SUCCESS (0) : no error

## TCL

### Prototype

HexapodMoveIncrementalControlPulseAndGatheringSet $SocketID $ GroupName $Divisor

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" function |
| – GroupName | string | Group name |
| – Divisor | int | Divisor * 0.0001 sec = interval between 2 pulses |

### Output parameters

– None

### Return

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

## C/C++

### Prototype

int **HexapodMoveIncrementalControlPulseAndGatheringSet** (int SocketID, char GroupName[250], int Divisor)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – GroupName | char * | Group name |
| – Divisor | int | Divisor * 0.0001 sec = interval between 2 pulses |

### Output parameters

– None

### Return

| | | |
|---|---|---|
| – Error | int | Function error code |

## Visual Basic

### Prototype

Long **HexapodMoveIncrementalControlPulseAndGatheringSet** (ByVal SocketID As Long, ByVal GroupName As String, ByVal Divisor As Long)

### Input parameters

| | | |
|---|---|---|
| – SocketID | Long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – GroupName | String | Group name |
| – Divisor | Long | Divisor * 0.0001 sec = interval between 2 pulses |

### Output parameters

– None

### Return

| | | |
|---|---|---|
| – Error | Long | Function error code |

## Matlab

### Prototype

[Error] **HexapodMoveIncrementalControlPulseAndGatheringSet** (int32 SocketID, cstring GroupName, int32 Divisor)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" function |
| – GroupName | cstring | Group name |
| – Divisor | int32 | Divisor * 0.0001 sec = interval between 2 pulses |

### Return

| | | |
|---|---|---|
| – Error | int32 | Function error code |

## Python

### Prototype

[Error] **HexapodMoveIncrementalControlPulseAndGatheringSet** (integer SocketID, string GroupName, integer Divisor)

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" function |
| – GroupName | string | Group name |
| – Divisor | integer | Divisor * 0.0001 sec = interval between 2 pulses |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error code |

### 3.5.3.10 HexapodSGammaParametersDistanceGet

#### Name

**HexapodSGammaParametersDistanceGet** – Returns distance during acceleration phase and distance during constant velocity phase in the vitual SGamma profiler used to define a hexapod trajectory.

#### Input tests

- Check command format: ERR_WRONG_FORMAT (-7)
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 allows getting the distance during acceleration phase and the distance during constant velocity phase of the SGamma motion profile used to define a hexapod trajectory.

#### 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

**HexapodSGammaParametersDistanceGet** $SocketID $FullPositionerName $Displacement, $Velocity $Acceleration $MinJerkTime $MaxJerkTime DisplacementDuringAcc DisplacementDuringVel

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | string | Positioner name |
| – | Displacement | double | displacement value (units) |
| – | 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

| | | | |
|---|---|---|---|
| – | DisplacementDuringAcc | double | displacement value during acceleration phase (units) |
| – | DisplacementDuringVel | double | displacement value during constant velocity phase (units) |

### Return

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

## C/C++

### Prototype

int **HexapodSGammaParametersDistanceGet** (int SocketID, char FullPositionerName[250], double Displacement, double Velocity, double Acceleration, double MinJerkTime, double MaxJerkTime, double *DisplacementDuringAcc, double *DisplacementDuringVel)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | FullPositionerName | char * | Positioner name |
| – | Displacement | double | displacement value (units) |
| – | 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

| | | | |
|---|---|---|---|
| – | DisplacementDuringAcc | double * | displacement value during acceleration phase (units) |
| – | DisplacementDuringVel | double * | displacement value during constant velocity phase (units) |

### Return

| | | | |
|---|---|---|---|
| – | Error | int | Function error code |

## Visual Basic

### Prototype

Long **HexapodSGammaParametersDistanceGet** (ByVal SocketID As Long, ByVal FullPositionerName As String, ByVal Displacement As Double, ByVal Velocity As Double, ByVal Acceleration As Double, ByVal MinimumJerkTime As Double, ByVal MaximumJerkTime As Double, MinimumJerkTime As Double, MaximumJerkTime As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | Long | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | String | Positioner name |
| – Displacement | Double | displacement value (units) |
| – 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

| | | |
|---|---|---|
| – DisplacementDuringAcc | Double | displacement value during acceleration phase (units) |
| – DisplacementDuringVel | Double | displacement value during constant velocity phase (units) |

### Return

| | | |
|---|---|---|
| – Error | Long | Function error code |

## Matlab

### Prototype

[Error, DisplacementDuringAcc, DisplacementDuringVel]
**HexapodSGammaParametersDistanceGet** (int32 SocketID, cstring FullPositionerName, double Displacement, double Velocity, double Acceleration, double MinimumJerkTime, double MaximumJerkTime )

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" function |
| – FullPositionerName | cstring | Positioner name |
| – Displacement | double | displacement value (units) |
| – 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 |
| – DisplacementDuringAcc | double | displacement value during acceleration phase (units) |
| – DisplacementDuringVel | double | displacement value during constant velocity phase (units) |

## Python

### Prototype

[Error, DisplacementDuringAcc, DisplacementDuringVel]
**HexapodSGammaParametersDistanceGet** (integer SocketID, string FullPositionerName, double Displacement, double Velocity, double Acceleration, double MinimumJerkTime, double MaximumJerkTime)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | integer | Socket identifier gets by the "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 |
| – | DisplacementDuringAcc | double | displacement value during acceleration phase (units) |
| – | DisplacementDuringVel | double | displacement value during constant velocity phase (units) |

### 3.5.4 Configuration Files

- **System.ini file:**

```
[HEXAPOD]
GeometryFileName = Geometry_hexapod.ini
PositionerInUse = 1,2,3,4,5,6
InitializationAndHomeSearchSequence = Together        ; Together or OneAfterAnother
[HEXAPOD.1]
PLugNumber = 1
StageName = MYSTAGE1
STAGE configuration => See § "Positioner : Configuration files"
[HEXAPOD.1]
PLugNumber = 2
StageName = MYSTAGE2
STAGE configuration => See § "Positioner : Configuration files"
```

- **Stages.ini file:**

```
[HEXAPOD.1]
MYSTAGE1 configuration => See § "Positioner : Configuration files"
[HEXAPOD.2]
MYSTAGE2 configuration => See § "Positioner : Configuration files"
```

## 3.6 Analog and Digital I/O

### 3.6.1 GPIO Name List

#### 3.6.1.1 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) |

#### 3.6.1.2 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) |

#### 3.6.1.3 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 |

#### 3.6.1.4 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 |

## 3.6.2 Function Description

### 3.6.2.1 GPIOAnalogGainGet

#### Name

**GPIOAnalogGainGet** – Gets the gain for one or several analog inputs (ADC)

#### Input tests

- HXP configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 HXP 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 HXP 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 HXP 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 HXP 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 gets by the "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

## C/C++

### Prototype

int **GPIOAnalogGainGet** (int SocketID, int NbElements, char* GPIONameList, double* AnalogGainValueArray)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – | NbElements | int | number of analog GPIO to read. |
| – | GPIONameList | char * | List of analog input names – separator is comma |

### Output parameters

| | | | |
|---|---|---|---|
| – | AnalogGainValueArray | int * | value of analog input gain |

### Return

– Function error code

## 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 gets by the "TCP_ConnectToServer" Function |
| – NbElements | Long | number of analog GPIO to read. |
| – GPIONameList | String | List of analog input names – separator is comma |

### Output parameters

| | | |
|---|---|---|
| – AnalogGainValueArray | Long | value of analog input gain |

### Return

– Function error code

## Matlab

### Prototype

[Error, AnalogGainValueArray] **GPIOAnalogGainGet** (int32 SocketID, cstring GPIONameArray)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – GPIONameArray | cstring | Analog input name array (maximum size = 250) |

### Return

| | | |
|---|---|---|
| – Error | int32 | Function error code |
| – AnalogGainValueArray | int32 | value of analog input gain |

## Python

### Prototype

[Error, AnalogGainValueArray] **GPIOAnalogGainGet** (integer SocketID, string GPIONameArray)

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – GPIONameArray | string | Analog input name array (maximum size = 250) |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error code |
| – AnalogGainValueArray | integer | value of analog input gain |

### 3.6.2.2 GPIOAnalogGainSet

#### Name

**GPIOAnalogGainSet** – Sets a gain for one or several analog inputs (ADC)

#### Input tests

- HXP configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 HXP 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 HXP 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 HXP 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 gets by the "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

## C/C++

### Prototype

int **GPIOAnalogGainSet** (int SocketID, int NbElements, char* GPIONameList, double* AnalogGainValueArray)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – NbElements | int | number of analog GPIO to read. |
| – GPIONameList | char * | List of analog input names – separator is comma |
| – AnalogGainValueArray | int * | value of analog input gain |

### Output parameters

– None

### Return

– Function error code

## Visual Basic

### Prototype

Long **GPIOAnalogGainSet** (ByVal SocketID As Long, ByVal NbElements As Long, GPIONameList As String, AnalogGainValueArray As Long)

### Input parameters

| | | |
|---|---|---|
| – SocketID | Long | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – NbElements | Long | number of analog GPIO to read. |
| – GPIONameList | String | List of analog input names (maximum size = 250) |
| – AnalogGainValueArray | Long | value of analog input gain |

### Output parameters

– None

### Return

– Function error code

## Matlab

### Prototype

[Error] **GPIOAnalogGainSet** (int32 SocketID, cstring GPIONameArray, int32 AnalogGainValueArray)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – GPIONameArray | cstring | Analog input name array (maximum size = 250) |
| – AnalogGainValueArray | int32 | value of analog input gain |

### Return

| | | |
|---|---|---|
| – Error | int32 | Function error code |

## Python

### Prototype

[Error] **GPIOAnalogGainSet** (integer SocketID, string GPIONameArray, integer AnalogGainValueArray)

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – GPIONameArray | string | Analog input name array (maximum size = 250) |
| – AnalogGainValueArray | integer | value of analog input gain |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error code |

### 3.6.2.3 GPIOAnalogGet

#### Name

**GPIOAnalogGet** – Read one or several analog GPIO (DAC or ADC)

#### Input tests

- HXP configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 HXP 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 Inputs list §3.6.1.3 and Analog Outputs list §3.6.1.4.

---

**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 HXP 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 gets by the "TCP_ConnectToServer" Function |
| – GPIOName | string | Analog GPIO name (maximum size = 250) |

### Output parameters

| | | |
|---|---|---|
| – AnalogValue | floating point | value of analog GPIO (DAC or ADC) |

### Return

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

## C/C++

### Prototype

int **GPIOAnalogGet** (int SocketID, int NbElements, char* GPIONameList, double* AnalogValueArray)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – NbElements | int | number of analog GPIO to read. |
| – GPIONameList | char * | List of analog GPIO names – separator is comma |

### Output parameters

| | | |
|---|---|---|
| – AnalogValueArray | double * | Analog GPIO value array (DAC or ADC) |

### Return

– Function error

## Visual Basic

### Prototype

Long **GPIOAnalogGet** (ByVal SocketID As Long, ByVal NbElements As Long, GPIONameList As String, AnalogValueArray As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | Long | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – NbElements | Long | number of analog GPIO to read. |
| – GPIONameList | String | List of analog GPIO names (maximum size = 250) |

### Output parameters

| | | |
|---|---|---|
| – AnalogValueArray | Double | Analog GPIO value array (DAC or ADC) |

### Return

| | |
|---|---|
| – | Function error |

## Matlab

### Prototype

[Error, AnalogValueArray]**GPIOAnalogGet** (int32 SocketID, cstring GPIONameArray)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – GPIONameArray | cstring | Analog GPIO name array (maximum size = 250) |

### Return

| | | |
|---|---|---|
| – Function error | | |
| – AnalogValueArray | double | Analog GPIO value array (DAC or ADC) |

## Python

### Prototype

[Error, AnalogValueArray]**GPIOAnalogGet** (integer SocketID, string GPIONameArray)

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – GPIONameArray | string | Analog GPIO name array (maximum size = 250) |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error |
| – AnalogValueArray | double | Analog GPIO value array (DAC or ADC) |

### 3.6.2.4 GPIOAnalogSet

#### Name

**GPIOAnalogSet** – Sets one or several analog output (DAC )

#### Input tests

- HXP configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 HXP 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 Outputs list §3.6.1.4.

**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 HXP 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 gets by the "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

## C/C++

### Prototype

int **GPIOAnalogSet** (int SocketID, int NbElements, char* GPIONameList, double* AnalogValueArray)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – NbElements | int | number of analog GPIO to read. |
| – GPIONameList | char * | List of analog GPIO names – separator is comma |
| – AnalogValueArray | double * | Analog GPIO value array (DAC ) |

### Output parameters

– None

### Return

– Function error

## Visual Basic

### Prototype

Long **GPIOAnalogSet** (ByVal SocketID As Long, ByVal NbElements As Long, GPIONameList As String, AnalogValueArray As Double)

### Input parameters

| | | |
|---|---|---|
| – SocketID | Long | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – NbElements | Long | number of analog GPIO to read. |
| – GPIONameList | String | List of analog GPIO names (maximum size = 250) |
| – AnalogValueArray | Double | Analog GPIO value array (DAC ) |

### Output parameters

– None

### Return

– Function error

## Matlab

### Prototype

[Error] **GPIOAnalogSet** (int32 SocketID, cstring GPIONameArray, double AnalogValueArray)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – GPIONameArray | cstring | Analog GPIO name array (maximum size = 250) |
| – AnalogValueArray | double | Analog GPIO value array (DAC ) |

### Return

– Function error

## Python

### Prototype

[Error] **GPIOAnalogSet** (integer SocketID, string GPIONameArray, double AnalogValueArray)

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – GPIONameArray | string | Analog GPIO name array (maximum size = 250) |
| – AnalogValueArray | double | Analog GPIO value array (DAC) |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error |

### 3.6.2.5 GPIODigitalGet

#### Name

**GPIODigitalGet** – Read one digital input or output.

#### Input tests

- HXP configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 HXP 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 Outputs list §3.6.1.2 and Digital Inputs list §3.6.1.1.

#### 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" Function |
| – | GPIOName | string | Digital GPIO name (maximum size = 250) |

### Return

| | | | |
|---|---|---|---|
| – | Error | integer | Function error |
| – | DigitalValue | unsigned short * | Digital value (DI or DO) |

### 3.6.2.6 GPIODigitalSet

#### Name

**GPIODigitalSet** – Sets one digital output.

#### Input tests

- HXP configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 HXP initialization in progress: ERR_NOT_ALLOWED_ACTION (-22)

#### Description

Sets the value of the selected digital output (DO).

See Digital Outputs list §3.6.1.2.

#### 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 |

## 3.7 Gathering

### 3.7.1 Function Description

#### 3.7.1.1 GatheringConfigurationGet

**Name**

**GatheringConfigurationGet** – Returns the current configuration of the internal triggered data gathering.

**Input tests**

- HXP configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" Function |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error |
| – TypeList | string | List of configured gathering types (separator is semicolon) |

### 3.7.1.2 GatheringConfigurationSet

#### Name

**GatheringConfigurationSet** – Configures a gathering.

#### Input tests

- HXP configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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.

The positions (Setpoint, Current) from the Tool coordinate system in the Work coordinate system can also be gathered.

Gathering data types:

*PositionerName*.CorrectorOutput
*PositionerName*.CurrentAcceleration
*PositionerName*.CurrentPosition
*PositionerName*.CurrentVelocity
*PositionerName*.FollowingError
*PositionerName*.SetpointAcceleration
*PositionerName*.SetpointPosition
*PositionerName*.SetpointVelocity
*HEXAPOD*.X.CurrentPosition (also for Y, Z, U, V, W)
*HEXAPOD*.X.SetpointPosition (also for Y, Z, U, V, W)
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

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 HXP 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 $TypeList

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – | TypeList | string | List of configured gathering types |

### Output parameters

– None

### Return

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

## C/C++

### Prototype

int **GatheringConfigurationSet** (int SocketID, int NbElements, char * TypeArray)

### Input parameters

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – | NbElements | int | Number of types |
| – | TypeArray | char * | Array of configured gathering types |

### Output parameters

– None

### Return

– Function error

## Visual Basic

### Prototype

Long **GatheringConfigurationSet** (ByVal SocketID As Long, ByVal NbElements As Long, ByVal TypeNameArray As String)

### Input parameters

| | | |
|---|---|---|
| – SocketID | Long | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – NbElements | Long | Number of types |
| – TypeNameArray | String | Array of configured gathering types |

### Output parameters

– None

### Return

– Function error

## Matlab

### Prototype

[Error] **GatheringConfigurationSet** (int32 SocketID, cstring TypeArray)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – TypeArray | cstring | Array of configured gathering types |

### Return

| | | |
|---|---|---|
| – Error | int32 | Function error |

## Python

### Prototype

[Error] **GatheringConfigurationSet** (integer SocketID, string TypeArray)

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – TypeArray | string | Array of configured gathering types |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error |

### 3.7.1.3 GatheringCurrentNumberGet

#### Name

**GatheringCurrentNumberGet** – Returns the current and maximum number of gathered data.

#### Input tests

- HXP configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" Function |

### Return

| | | | |
|---|---|---|---|
| – | Error | integer | Function error |
| – | CurrentNumber | integer | Current number during acquisition |
| – | MaxSamplesNumber | integer | Maximum number of samples |

### 3.7.1.4 GatheringDataAcquire

#### Name

**GatheringDataAcquire** – Acquires manually only one data.

#### Input tests

- HXP configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 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 **GatheringDataAcquire** (int SocketID)

**Input parameters**

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |

**Output parameters**

– None

**Return**

– Function error

## Visual Basic

**Prototype**

Long **GatheringDataAcquire** (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] **GatheringDataAcquire** (int32 SocketID)

**Input parameters**

| | | | |
|---|---|---|---|
| – | SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" Function |

**Return**

| | | | |
|---|---|---|---|
| – | Error | int32 | Function error |

## Python

**Prototype**

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

**Input parameters**

| | | | |
|---|---|---|---|
| – | SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" Function |

**Return**

| | | | |
|---|---|---|---|
| – | Error | integer | Function error |

### 3.7.1.5 GatheringDataGet

#### Name

**GatheringDataGet** – Reads a data line from the current gathering buffer.

#### Input tests

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 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 **GatheringDataGet** (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 **GatheringDataGet** (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] **GatheringDataGet** (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] **GatheringDataGet** (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. |

### 3.7.1.6 GatheringDataMultipleLinesGet

#### Name

**GatheringDataMultipleLinesGet** – Reads several data lines from the current gathering buffer in memory.

#### Input tests

- Check command format: (-7)
- Verify the number of parameters: (-9)
- Check output parameter type: (-13)
- Check input parameter type: (-15)
- Check index number: (-17)
  - *IndexPoint ≥ 0 (Note: index #0 = line #1).*
  - *IndexPoint <currently gathered data number.*
- Controller initialization failed: (-20)
- XPS initialization in progress: (-21)
- Check gathering state: (-32)

#### Description

This function reads 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 (-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 (32767 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

**Prototype**

int **GatheringDataMultipleLinesGet**(

int SocketID,

int IndexPoint,

int NbLines,

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 values from the current buffer at the selected index. |

**Return**

- 0: No error.
- -17: Parameter out of range or incorrect.
- -32: Gathering not configured.

### 3.7.1.7 GatheringExternalConfigurationGet

#### Name

**GatheringExternalConfigurationGet** – Returns the current configuration of the external triggered data gathering.

#### Input tests

- HXP configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" Function |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error |
| – TypeList | string | List of configured gathering types (separator is semicolon) |

### 3.7.1.8 GatheringExternalConfigurationSet

#### Name

**GatheringExternalConfigurationSet** – Configures an external gathering.

#### Input tests

- HXP configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 HXP 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 $TypeList

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – TypeList | string | List of configured gathering types |

### Output parameters

– None

### Return

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

## C/C++

### Prototype

int **GatheringExternalConfigurationSet** (int SocketID, int NbElements, char * TypeArray)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |
| – NbElements | int | Number of types |
| – TypeArray | char * | Array of configured gathering types |

### Output parameters

– None

### Return

– Function error

## Visual Basic

### Prototype

Long **GatheringExternalConfigurationSet** (ByVal SocketID As Long, ByVal NbElements As Long, ByVal TypeNameArray As String)

### Input parameters

| | | |
|---|---|---|
| – SocketID | Long | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – NbElements | Long | Number of types |
| – TypeNameArray | String | Array of configured gathering types |

### Output parameters

– None

### Return

– Function error

## Matlab

**Prototype**

[Error] **GatheringExternalConfigurationSet** (int32 SocketID, cstring TypeArray)

**Input parameters**

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – TypeArray | cstring | Array of configured gathering types |

**Return**

| | | |
|---|---|---|
| – Error | int32 | Function error |

## Python

**Prototype**

[Error] **GatheringExternalConfigurationSet** (integer SocketID, string TypeArray)

**Input parameters**

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – TypeArray | string | Array of configured gathering types |

**Return**

| | | |
|---|---|---|
| – Error | integer | Function error |

### 3.7.1.9 GatheringExternalCurrentNumberGet

#### Name

**GatheringExternalCurrentNumberGet** – Returns the current and maximum number of external gathered data.

#### Input tests

- HXP configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" Function |

### Return

| | | | |
|---|---|---|---|
| – | Error | integer | Function error |
| – | CurrentNumber | integer | Current number during acquisition |
| – | MaxSamplesNumber | integer | Maximum number of samples |

### 3.7.1.10 GatheringExternalStopAndSave

#### Name

**GatheringExternalStopAndSave** – Stops external triggered data gathering and saves data to the HXP controller.

#### Input tests

- HXP configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 HXP controller. Gathered data are stored in the file "GatheringExternal.dat" in the "..\PUBLIC" folder of the HXP controller.

For a more thorough description of the external data gathering capability, please refer to the HXP 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 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 **GatheringExternalStopAndSave** (int SocketID)

**Input parameters**

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |

**Output parameters**

– None

**Return**

– Function error

## Visual Basic

**Prototype**

Long **GatheringExternalStopAndSave** (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] **GatheringExternalStopAndSave** (int32 SocketID)

**Input parameters**

| | | | |
|---|---|---|---|
| – | SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" Function |

**Return**

| | | | |
|---|---|---|---|
| – | Error | int32 | Function error |

## Python

**Prototype**

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

**Input parameters**

| | | | |
|---|---|---|---|
| – | SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" Function |

**Return**

| | | | |
|---|---|---|---|
| – | Error | integer | Function error |

### 3.7.1.11 GatheringReset

#### Name

**GatheringReset** – Resets gathered data to start new gathering from scratch.

#### Input tests

- HXP configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 HXP 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 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 **GatheringReset** (int SocketID)

**Input parameters**

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |

**Output parameters**

– None

**Return**

| | | |
|---|---|---|
| – | | Function error |

## Visual Basic

**Prototype**

Long **GatheringReset** (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] **GatheringReset** (int32 SocketID)

**Input parameters**

| | | | |
|---|---|---|---|
| – | SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" Function |

**Return**

| | | | |
|---|---|---|---|
| – | Error | int32 | Function error |

## Python

**Prototype**

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

**Input parameters**

| | | | |
|---|---|---|---|
| – | SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" Function |

**Return**

| | | | |
|---|---|---|---|
| – | Error | integer | Function error |

### 3.7.1.12 GatheringRun

#### Name

**GatheringRun** – Starts to gather data.

#### Input tests

- HXP configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 |

### 3.7.1.13 GatheringStop

#### Name

**GatheringStopAndSave** – Stops internal and external triggered data gathering.

#### Input tests

- HXP configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 HXP 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 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 **GatheringStop** (int SocketID)

**Input parameters**

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |

**Output parameters**

– None

**Return**

– Function error

## Visual Basic

**Prototype**

Long **GatheringStop** (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] **GatheringStop** (int32 SocketID)

**Input parameters**

| | | | |
|---|---|---|---|
| – | SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" Function |

**Return**

| | | | |
|---|---|---|---|
| – | Error | int32 | Function error |

## Python

**Prototype**

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

**Input parameters**

| | | | |
|---|---|---|---|
| – | SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" Function |

**Return**

| | | | |
|---|---|---|---|
| – | Error | integer | Function error |

### 3.7.1.14 GatheringStopAndSave

#### Name

**GatheringStopAndSave** – Stops internal triggered data gathering and saves data to the HXP controller.

#### Input tests

- HXP configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 HXP controller. Data is stored in the file GATHERING.DAT in the "..\PUBLIC" folder of the HXP controller.

For a more thorough description of the internal data gathering capability, please refer to the HXP 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 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 **GatheringStopAndSave** (int SocketID)

**Input parameters**

| | | | |
|---|---|---|---|
| – | SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" function |

**Output parameters**

– None

**Return**

| | | |
|---|---|---|
| – | | Function error |

## Visual Basic

**Prototype**

Long **GatheringStopAndSave** (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] **GatheringStopAndSave** (int32 SocketID)

**Input parameters**

| | | | |
|---|---|---|---|
| – | SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" Function |

**Return**

| | | | |
|---|---|---|---|
| – | Error | int32 | Function error |

## Python

**Prototype**

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

**Input parameters**

| | | | |
|---|---|---|---|
| – | SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" Function |

**Return**

| | | | |
|---|---|---|---|
| – | Error | integer | Function error |

## 3.8 Events and Actions

### 3.8.1 Functions Description

#### 3.8.1.1 EventExtendedAllGet

**Name**

**EventExtendedAllGet** – Return all "event and action" identifiers in progress.

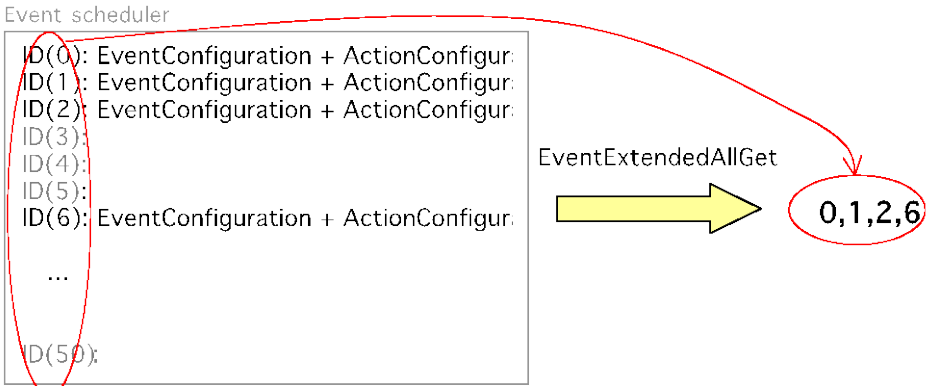
**Input tests**

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 |

### 3.8.1.2 EventExtendedConfigurationActionGet

#### Name

**EventExtendedConfigurationActionGet** – Return the action combination defined in buffer.

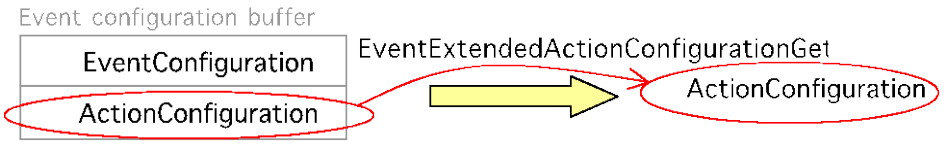
#### Input tests

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" Function |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error code |
| – ActionConfiguration | string | Action combination configured in buffer |

### 3.8.1.3 EventExtendedConfigurationActionSet

#### Name

**EventExtendedConfigurationActionSet** – Define a combination of one or several actions in buffer.

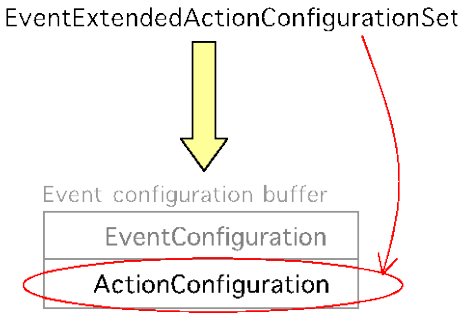
#### Input tests

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 HXP 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. ExecuteTCLScript
10. KillTCLScript

11. ExternalGatheringRun
12. GatheringRun
13. GatheringRunAppend
14. GatheringOneData
15. GatheringStop
16. GroupName.MoveAbort

**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** $SocketID **{$**ExtendedActionName $ActionParameter1 $ActionParameter2 $ActionParameter3 $ActionParameter4**} …**

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – ExtendedActionName | string | event full name (maximum size = 250) - see § Events - |
| – ActionParameter1 | string | optional action's parameter #1 (maximum size = 250) |
| – ActionParameter2 | string | optional action's parameter #2 (maximum size = 250) |
| – ActionParameter3 | string | optional action's parameter #3 (maximum size = 250) |
| – ActionParameter4 | string | optional action's parameter #4 (maximum size = 250) |

### Output parameters

– None

### Return

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

## C/C++

### Prototype

int **EventExtendedConfigurationActionSet** (int SocketID, int NbElements, char* ExtendedActionName, char* ActionParameter1, char* ActionParameter2, char* ActionParameter3, char* ActionParameter4)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – NbElements | int | number of events in configuration. |
| – ExtendedActionName | char* | event full name (maximum size = 250) - see § Events - |
| – ActionParameter1 | char* | optional action's parameter #1 (maximum size = 250) |
| – ActionParameter2 | char* | optional action's parameter #2 (maximum size = 250) |
| – ActionParameter3 | char* | optional action's parameter #3 (maximum size = 250) |
| – ActionParameter4 | char* | optional action's parameter #4 (maximum size = 250) |

### Output parameters

– None

### Return

– Function error code

## Visual Basic

### Prototype

Long **EventExtendedConfigurationActionSet** (ByVal SocketID As Long, ByVal NbElements As Long, ByVal ExtendedActionName As String, ByVal ActionParameter1 As String, ByVal ActionParameter2 As String, ByVal ActionParameter3 As String, ByVal ActionParameter4 As String)

### Input parameters

| | | |
|---|---|---|
| – SocketID | Long | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – NbElements | Long | number of events in configuration. |
| – ExtendedActionName | String | event full name (maximum size = 250) - see § Events - |
| – ActionParameter1 | String | optional action's parameter #1 (maximum size = 250) |
| – ActionParameter2 | String | optional action's parameter #2 (maximum size = 250) |
| – ActionParameter3 | String | optional action's parameter #3 (maximum size = 250) |
| – ActionParameter4 | String | optional action's parameter #4 (maximum size = 250) |

### Output parameters

– None

### Return

– Function error code

## Matlab

### Prototype

int32 **EventExtendedConfigurationActionSet** (int32 SocketID, cstring ExtendedActionName, cstring ActionParameter1, cstring ActionParameter2, cstring ActionParameter3, cstring ActionParameter4)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – ExtendedActionName | cstring | event full name (maximum size = 250) - see § Events - |
| – ActionParameter1 | cstring | optional action's parameter #1 (maximum size = 250) |
| – ActionParameter2 | cstring | optional action's parameter #2 (maximum size = 250) |
| – ActionParameter3 | cstring | optional action's parameter #3 (maximum size = 250) |
| – ActionParameter4 | cstring | optional action's parameter #4 (maximum size = 250) |

### Return

– Function error code

## Python

### Prototype

integer **EventExtendedConfigurationActionSet** (integer SocketID, string ExtendedActionName, string ActionParameter1, string ActionParameter2, string ActionParameter3, string ActionParameter4)

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – ExtendedActionName | string | event full name (maximum size = 250) - see § Events - |
| – ActionParameter1 | string | optional action's parameter #1 (maximum size = 250) |
| – ActionParameter2 | string | optional action's parameter #2 (maximum size = 250) |
| – ActionParameter3 | string | optional action's parameter #3 (maximum size = 250) |
| – ActionParameter4 | string | optional action's parameter #4 (maximum size = 250) |

### Return

– Function error code

### 3.8.1.4 EventExtendedConfigurationTriggerGet

#### Name

**EventExtendedConfigurationTriggerGet** – Return the trigger defined in buffer.

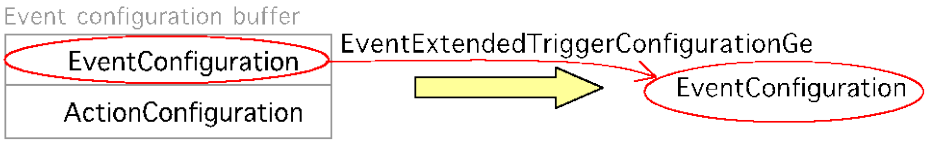
#### Input tests

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" Function |

### Return

| | | | |
|---|---|---|---|
| – | Error | integer | Function error code |
| – | EventTriggerConfiguration | string | Event combination configured in buffer |

### 3.8.1.5 EventExtendedConfigurationTriggerSet

#### Name

**EventExtendedConfigurationTriggerSet** - Define a combination of one or several events in buffer.

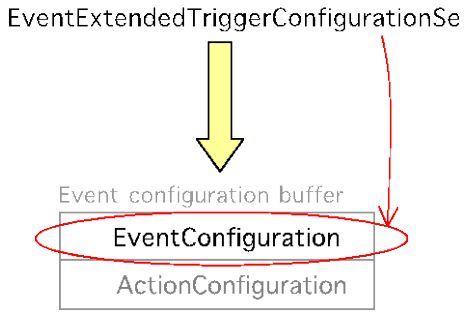
#### Input tests

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 HXP 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***

3. SGamma
4. Slave
5. Spin
6. Jog
7. TrackingPosition
8. TrackingVelocity

***Category list for "trajectory" group events***

1. XYLineArc
2. Spline
3. PVT

**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** $SocketID {**$**FullEventName $EventParameter1 $EventParameter2 $EventParameter3 $EventParameter4} **…**

**Input parameters**

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – ExtendedEventName | string | event full name (maximum size = 250) - see § Events - |
| – EventParameter1 | string | optional event's parameter #1 (maximum size = 250) |
| – EventParameter2 | string | optional event's parameter #2 (maximum size = 250) |
| – EventParameter3 | string | optional event's parameter #3 (maximum size = 250) |
| – EventParameter4 | string | optional event's parameter #4 (maximum size = 250) |

**Output parameters**

– None

**Return**

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

## C/C++

### Prototype

int **EventExtendedConfigurationTriggerSet** (int SocketID, int NbElements, char* ExtendedEventName, char* EventParameter1, char* EventParameter2 , char* EventParameter3, char* EventParameter4 )

### Input parameters

| | | |
|---|---|---|
| – SocketID | int | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – NbElements | int | number of events in configuration. |
| – ExtendedEventName | char* | list of event full names (maximum size = 250) – separator is ';' |
| – EventParameter1 | char* | list of optional event's parameter #1 (maximum size = 250) |
| – EventParameter2 | char* | list of optional event's parameter #2 (maximum size = 250) |
| – EventParameter3 | char* | list of optional event's parameter #3 (maximum size = 250) |
| – EventParameter4 | char* | list of optional event's parameter #4 (maximum size = 250) |

### Output parameters

– None

### Return

– Function error code

## Visual Basic

### Prototype

Long **EventExtendedConfigurationTriggerSet** (ByVal SocketID As Long, ByVal NbElements As Long, ByVal ExtendedEventName As String, ByVal EventParameter1 As String, ByVal EventParameter2 As String, ByVal EventParameter3 As String, ByVal EventParameter4 As String)

### Input parameters

| | | |
|---|---|---|
| – SocketID | Long | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – NbElements | Long | number of events in configuration. |
| – ExtendedEventName | String | array of event full names (maximum size = 250) - see § Events |
| – EventParameter1 | String | array of optional event's parameter #1 (maximum size = 250) |
| – EventParameter2 | String | array of optional event's parameter #2 (maximum size = 250) |
| – EventParameter3 | String | array of optional event's parameter #3 (maximum size = 250) |
| – EventParameter4 | String | array of optional event's parameter #4 (maximum size = 250) |

### Output parameters

– None

### Return

– Function error code

## Matlab

### Prototype

int32 **EventExtendedConfigurationTriggerSet** (int32 SocketID, cstring ExtendedEventName, cstring EventParameter1, cstring EventParameter2, cstring EventParameter3, cstring EventParameter4)

### Input parameters

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – ExtendedEventName | cstring | array of event full names (maximum size = 250) - see § Events |
| – EventParameter1 | cstring | array of optional event's parameter #1 (maximum size = 250) |
| – EventParameter2 | cstring | array of optional event's parameter #2 (maximum size = 250) |
| – EventParameter3 | cstring | array of optional event's parameter #3 (maximum size = 250) |
| – EventParameter4 | cstring | array of optional event's parameter #4 (maximum size = 250) |

### Return

– Function error code

## Python

### Prototype

integer **EventExtendedConfigurationTriggerSet** (integer SocketID, string ExtendedEventName, string EventParameter1, string EventParameter2, string EventParameter3, string EventParameter4)

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" Function |
| – ExtendedEventName | string | array of event full names (maximum size = 250) - see § Events |
| – EventParameter1 | string | array of optional event's parameter #1 (maximum size = 250) |
| – EventParameter2 | string | array of optional event's parameter #2 (maximum size = 250) |
| – EventParameter3 | string | array of optional event's parameter #3 (maximum size = 250) |
| – EventParameter4 | string | array of optional event's parameter #4 (maximum size = 250) |

### Return

– Function error code

### 3.8.1.6 EventExtendedGet

#### Name

**EventExtendedGet** – Return the detail of "event and action" combination in scheduler defined by an identifier.

#### Input tests

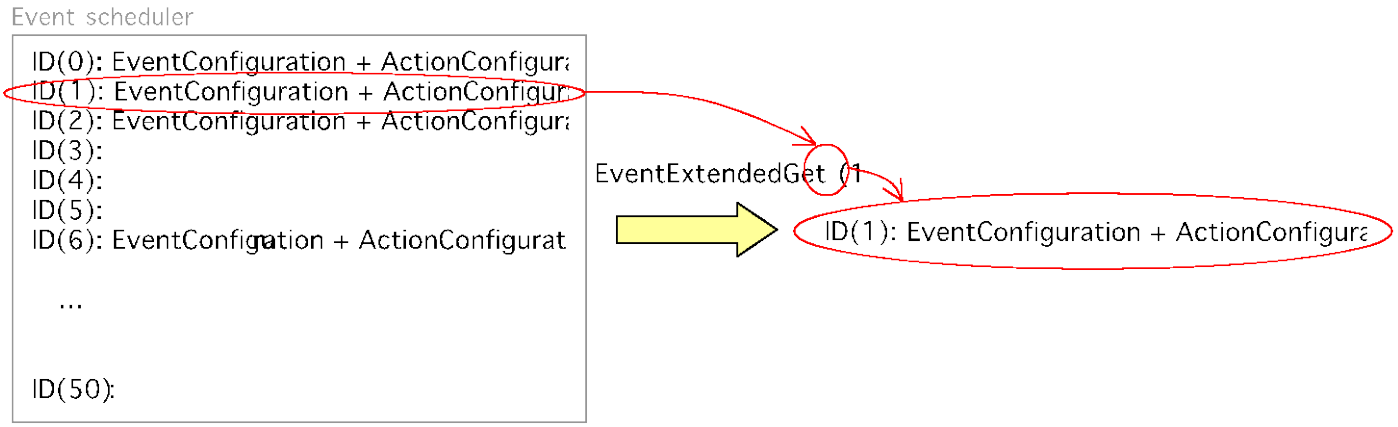
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of parameters: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Parameter type: ERR_WRONG_TYPE_INT (-15), ERR_WRONG_TYPE_CHAR (-13)
- Event identifier [0:50]: 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 50, 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 |

### 3.8.1.7 EventExtendedRemove

#### Name

**EventExtendedRemove** – Remove an "event and action" combination in scheduler defined by an identifier.

#### Input tests

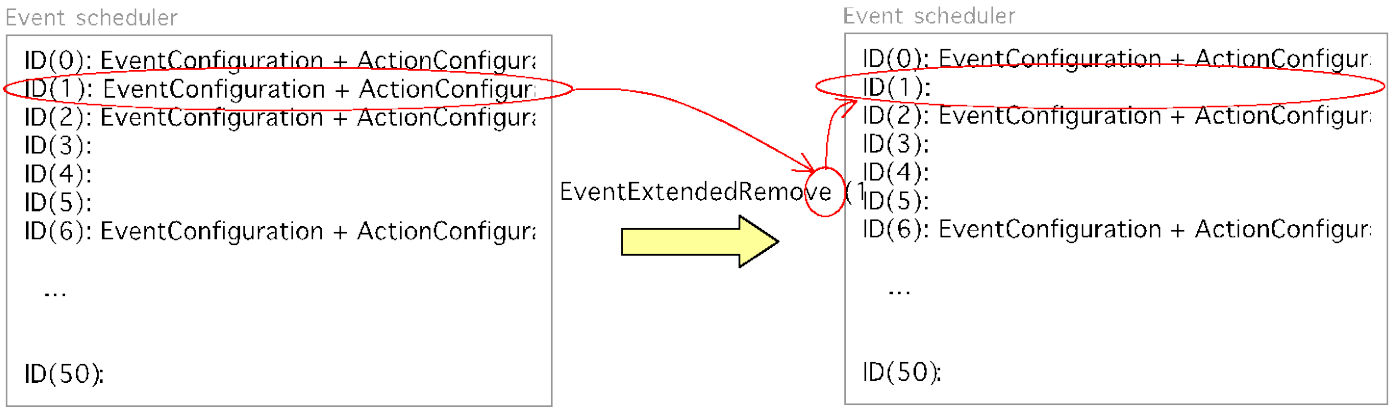
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of parameters [1]: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Parameter type: ERR_WRONG_TYPE_INT (-15)
- Event identifier [0:50]: ERR_EVENT_ID_UNDEFINED (-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 50, 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" Function |
| – EventID | integer | "Event and action" identifier |

**Return**

| | | |
|---|---|---|
| – Error | integer | Function error code |

### 3.8.1.8 EventExtendedStart

#### Name

**EventExtendedStart** – Activate the "event and action" defined in buffer.

#### Input tests

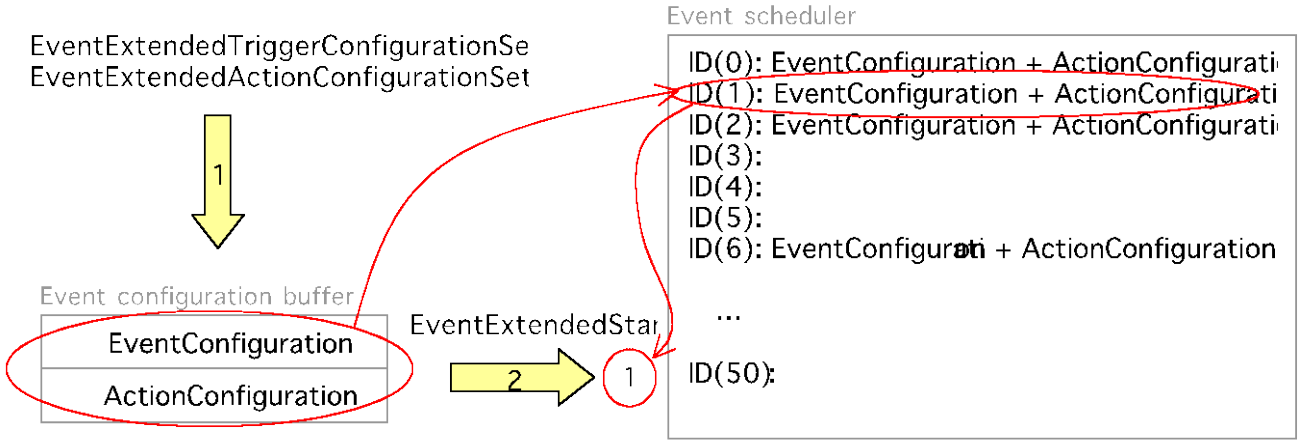
- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP initialization in progress: ERR_IN_INITIALIZATION (-21)
- Valid command format: ERR_WRONG_FORMAT (-7)
- Number of arguments: ERR_WRONG_PARAMETERS_NUMBER (-9)
- Output parameter type: ERR_WRONG_TYPE_INT (-15)
- Number of compositions in execution: 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 50, 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 gets by the "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 gets by the "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 | Long | Socket identifier gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" Function |

**Return**

| | | |
|---|---|---|
| – Error | integer | Function error code |
| – EventID | integer | "Event and action" identifier |

### 3.8.1.9 EventExtendedWait

#### Name

**EventExtendedWait** – Activate the last "event" configuration in memory and wait it occurs.

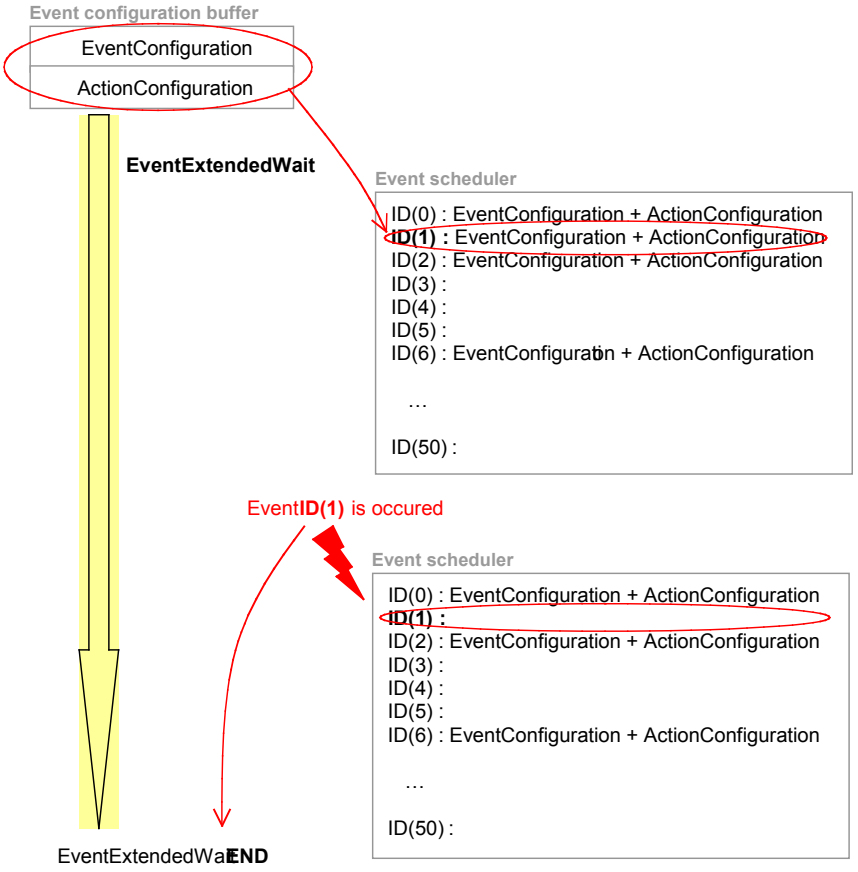
#### Input tests

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" Function |

**Output parameters**

– None

**Return**

– Function error code

## Matlab

**Prototype**

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

**Input parameters**

| | | |
|---|---|---|
| – SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" Function |

**Return**

| | | |
|---|---|---|
| – Error | int32 | Function error code |

## Python

**Prototype**

– [Error] **EventExtendedWait** (integer SocketID)

**Input parameters**

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" Function |

**Return**

| | | |
|---|---|---|
| – Error | integer | Function error code |

## 3.9 TCL Programming

### 3.9.1 Function Description

#### 3.9.1.1 TCLScriptExecute

**Name**

**TCLScriptExecute** – Executes a TCL script.

**Input tests**

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 |

### 3.9.1.2 TCLScriptExecuteAndWait

#### Name

**TCLScriptExecuteAndWait** – Executes a TCL script and waits the end of execution.

#### Input tests

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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] **TCLScriptExecuteAndWait** (integer SocketID, string TCLFileName, string TaskName, string InputArguments, string OutputArguments)

### Input parameters

| | | |
|---|---|---|
| – SocketID | integer | Socket identifier gets by the "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) |

### 3.9.1.3 TCLScriptKill

#### Name

**TCLScriptKill** – Kills a TCL script.

#### Input tests

- Configuration files reading: ERR_FATAL_INIT (-20)
- HXP 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 gets by the "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 gets by the "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 gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" Function |
| – TaskName | string | Task name to kill |

### Return

| | | |
|---|---|---|
| – Error | integer | Function error code |

## 3.10 Version

### 3.10.1 Function Description

#### 3.10.1.1 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.

**Error**

- None

**TCL**

**Prototype**

**GetLibraryVersion** $SocketID LibVersion

**Input parameters**

| | | | |
|---|---|---|---|
| – | SocketID | integer | Socket identifier gets by the "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 gets by the "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 gets by the "TCP_ConnectToServer" function |

**Output parameters**

– None

**Return**

| | | | |
|---|---|---|---|
| – | LibVersion | String | DLL library version |

## Matlab

**Prototype**

[LibVersion] **GetLibraryVersion** (int32 SocketID)

**Input parameters**

| | | | |
|---|---|---|---|
| – | SocketID | int32 | Socket identifier gets by the "TCP_ConnectToServer" function |

**Return**

| | | | |
|---|---|---|---|
| – | LibVersion | cstring | DLL library version |

## Python

**Prototype**

[LibVersion] **GetLibraryVersion** (integer SocketID)

**Input parameters**

| | | | |
|---|---|---|---|
| – | SocketID | integer | Socket identifier gets by the "TCP_ConnectToServer" Function |

**Return**

| | | | |
|---|---|---|---|
| – | LibVersion | string | DLL library version |

## 3.11 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.**

## 3.12 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 |

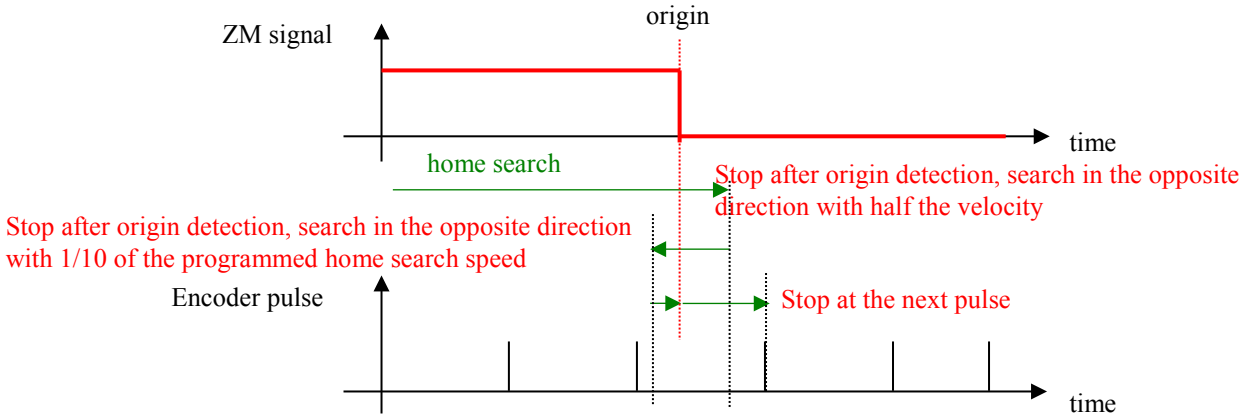
**NOTE**

**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 HXP controller. The General Inhibiton connector is a safety feature 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 which case, inhibition is detected.

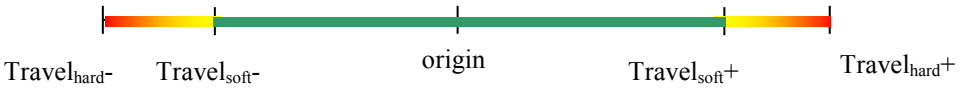
*ZM high level*: This refers to the mechanical zero signal used with some stages. The ZM signal is high during one part of the travel and low during the other part of the travel. The detection of the ZM high/low transition in combination with an encoder index pulse signal allows a fast and repeatable 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 should never be activated and any motion will be stopped by the detection of the minus software limit.

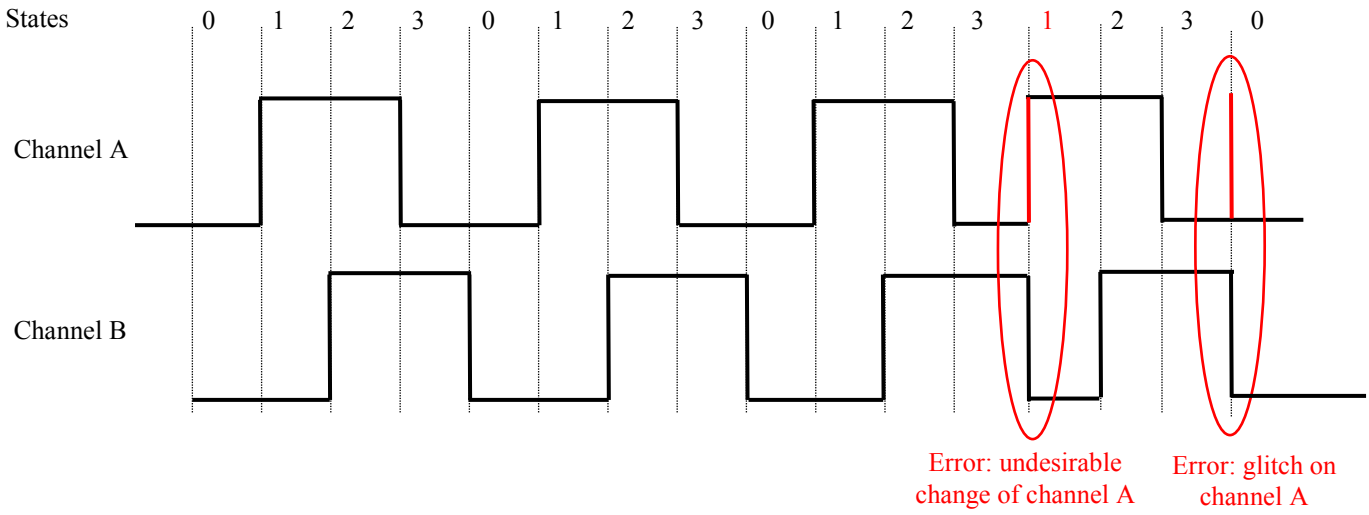
*Plus end of run activated*: Refers to the hardware positive end of run limit switch. During normal operation, this end of run switch should never be activated and any motion will be stopped by the detection of the positive software limit.

*Minus end of run glitch*: Undesirable, momentary instability of the hardware minus end of run signal, for instance can be generated by ripple or noise.



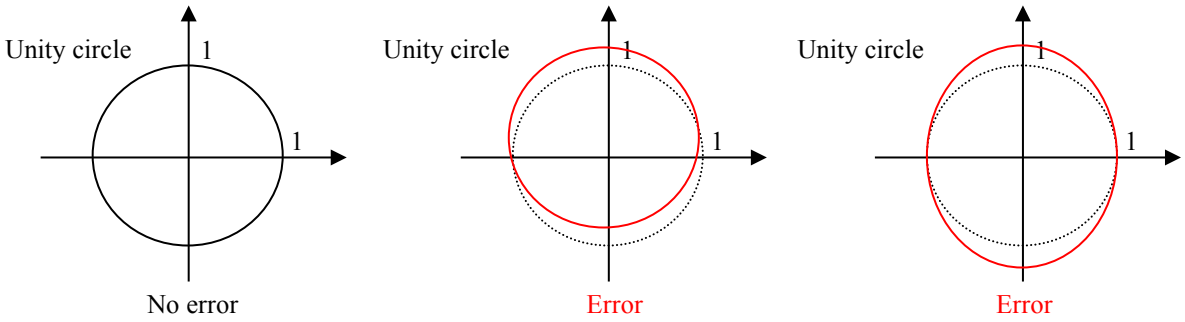

*Plus end of run glitch*: Undesirable, momentary instability of the hardware positive end of run signal, for instance can be generated by ripple or noise.

*Encoder quadrature error*: Error generated when the signals of both encoder channels simultaneously change. In normal operation, only one quadrature signal changes state at a time. This error can occur due to an undesirable level change or a glitch as illustrated below.




*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 change. 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 with motor ON after initialization.

*Second driver powered on*: Second driver with motor ON after initialization, in case two drivers are connected to one axis.

## 3.13 Positioner Driver Status List

| Bit | code | Error description | DRV01 | DRV02 | DRV03 | DRVM |
|---|---|---|---|---|---|---|
| **0** | 0x0001 | Short-circuit | | ✓ | ✓ | |
| **1** | 0x0002 | Broken fuse | | ✓ | ✓ | |
| **2** | 0x0004 | Thermistor fault | | ✓ | | |
| **3** | 0x0008 | Initialization error | | ✓ | ✓ | |
| **4** | 0x0010 | $I^2T$ | | ✓ | ✓ | |
| **5** | 0x0020 | Current limit | | ✓ | | |
| **6** | 0x0040 | TG is opened | | | ✓ | |
| **7** | 0x0080 | Inhibition input | | ✓ | ✓ | |
| **8** | 0x8000 | Driver in fault | ✓ | ✓ | ✓ | ✓ |

## 3.14 Group State 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 |

## 3.15 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_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 |

## 3.16 Function List Classed in Categories

**General function**

1. DoubleGlobalArrayGet
2. DoubleGlobalArraySet
3. CloseAllOtherSockets
4. ElapsedTimeGet
5. ErrorStringGet
6. EventAdd
7. EventGet
8. EventRemove
9. EventWait
10. EventExtendedConfigurationTriggerSet
11. EventExtendedConfigurationTriggerGet
12. EventExtendedConfigurationActionSet
13. EventExtendedConfigurationActionGet
14. EventExtendedStart
15. EventExtendedAllGet
16. EventExtendedGet
17. EventExtendedRemove
18. EventExtendedWait
19. FirmwareVersionGet
20. GatheringConfigurationGet
21. GatheringConfigurationSet
22. GatheringCurrentNumberGet
23. GatheringExternalConfigurationSet
24. GatheringExternalConfigurationGet
25. GatheringExternalCurrentNumberGet
26. GatheringExternalStopAndSave
27. GatheringDataAcquire
28. GatheringDataGet
29. GatheringReset
30. GatheringRun
31. GatheringStop
32. GatheringStopAndSave
33. GlobalArrayGet
34. GlobalArraySet
35. GPIOAnalogGet
36. GPIOAnalogSet
37. GPIOAnalogGainGet
38. GPIOAnalogGainSet
39. GPIODigitalGet

40. GPIODigitalSet
41. KillAll
42. Login
43. Reboot
44. TCLScriptExecute
45. TCLScriptExecuteAndWait
46. TCLScriptKill
47. TimerGet
48. TimerSet

**Positioner functions**

1. PositionerBacklashGet
2. PositionerBacklashSet
3. PositionerBacklashEnable
4. PositionerBacklashDisable
5. PositionerCorrectorNotchFiltersSet
6. PositionerCorrectorNotchFiltersGet
7. PositionerCorrectorPIDFFAccelerationSet
8. PositionerCorrectorPIDFFAccelerationGet
9. PositionerCorrectorPIDFFVelocitySet
10. PositionerCorrectorPIDFFVelocityGet
11. PositionerCorrectorPIDDualFFVoltageSet
12. PositionerCorrectorPIDDualFFVoltageGet
13. PositionerCorrectorPIPositionSet
14. PositionerCorrectorPIPositionGet
15. PositionerCorrectorTypeGet
16. PositionerCurrentVelocityAccelerationFiltersSet
17. PositionerCurrentVelocityAccelerationFiltersGet
18. PositionerDriverStatusGet
19. PositionerDriverStatusStringGet
20. PositionerEncoderAmplitudeValuesGet
21. PositionerEncoderCalibrationParametersGet
22. PositionerErrorGet
23. PositionerErrorRead
24. PositionerErrorStringGet
25. PositionerHardwareStatusGet
26. PositionerHardwareStatusStringGet
27. PositionerHardInterpolatorFactorGet
28. PositionerHardInterpolatorFactorSet
29. PositionerMaximumVelocityAndAccelerationGet
30. PositionerMotionDoneGet
31. PositionerMotionDoneSet
32. ementGet
33. PositionerSGammaParametersGet
34. PositionerSGammaParametersSet
35. PositionerSGammaPreviousMotionTimesGet
36. PositionerStageParameterGet
37. PositionerStageParameterSet
38. PositionerUserTravelLimitsGet
39. PositionerUserTravelLimitsSet

# 4.0 Process Examples

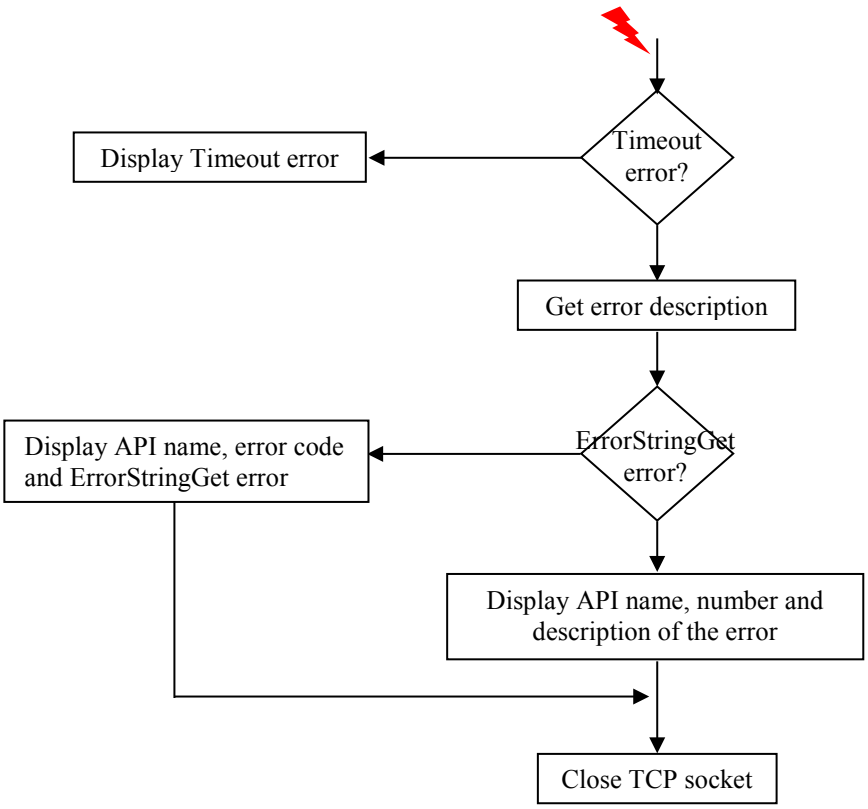
This part provides examples of programming sequences. Graph7 diagrams are presented to show you the order of use of the different Functions. To see the programming code examples, please refer to:

- The TCL Manual for TCL scripts (part 7. Examples of TCL programs with HXP).
- The Software Drivers Manual for C++ sequences (part 1.3 Example of C++ programs using the HXP DLL).

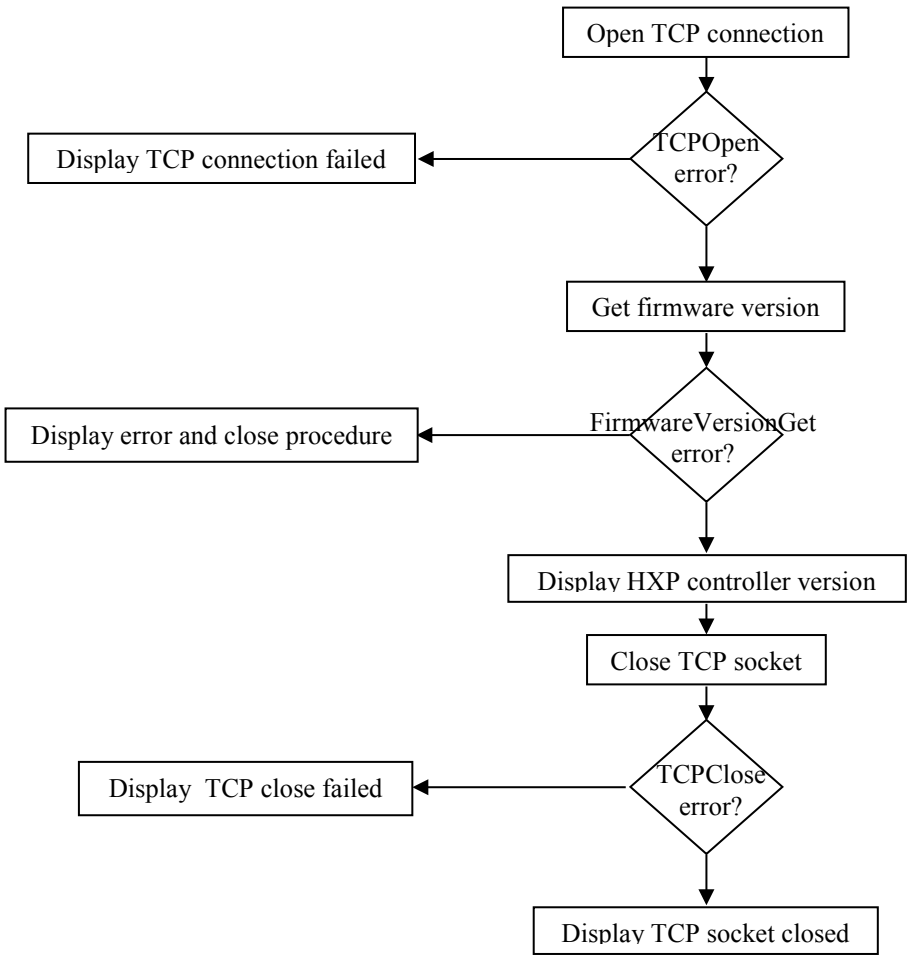
## 4.1 Management of the Errors

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.

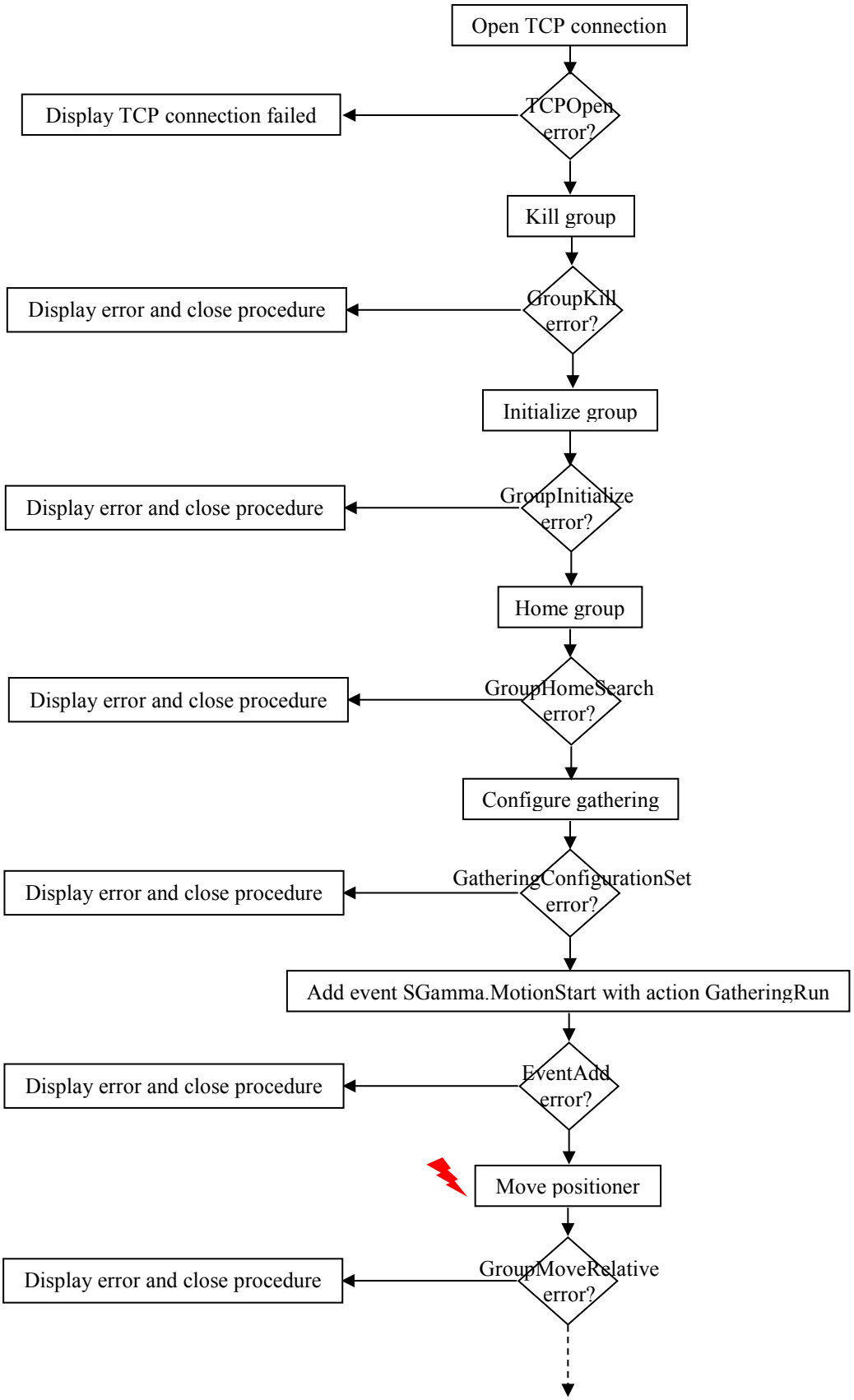
***Display error and close procedure:***

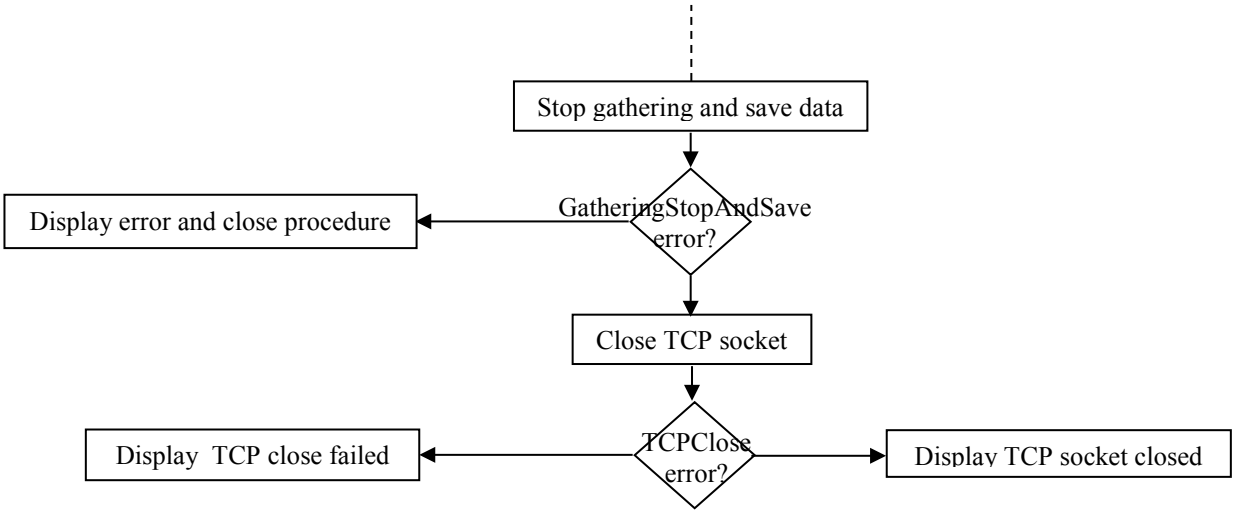


## 4.2 Firmware Version

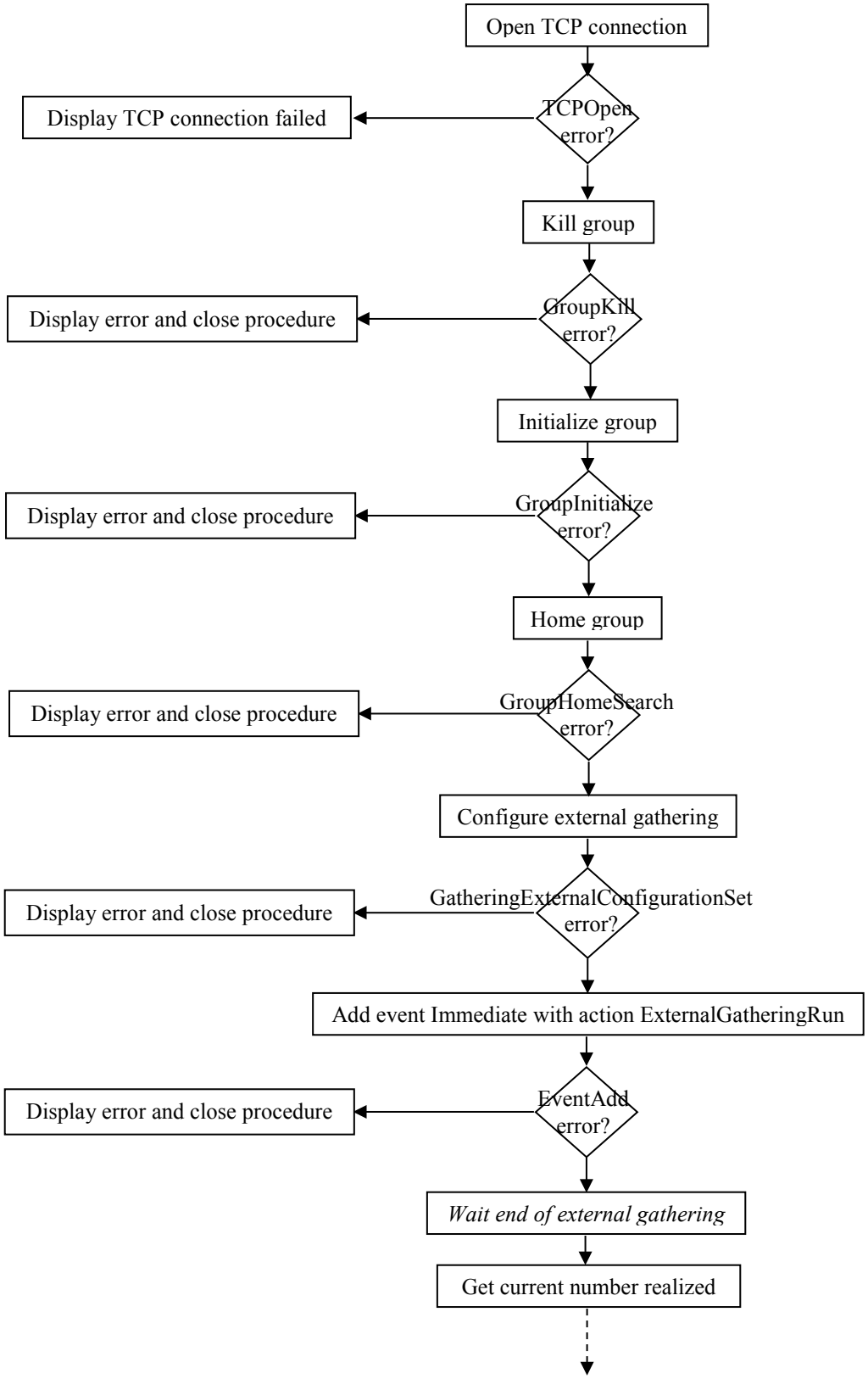


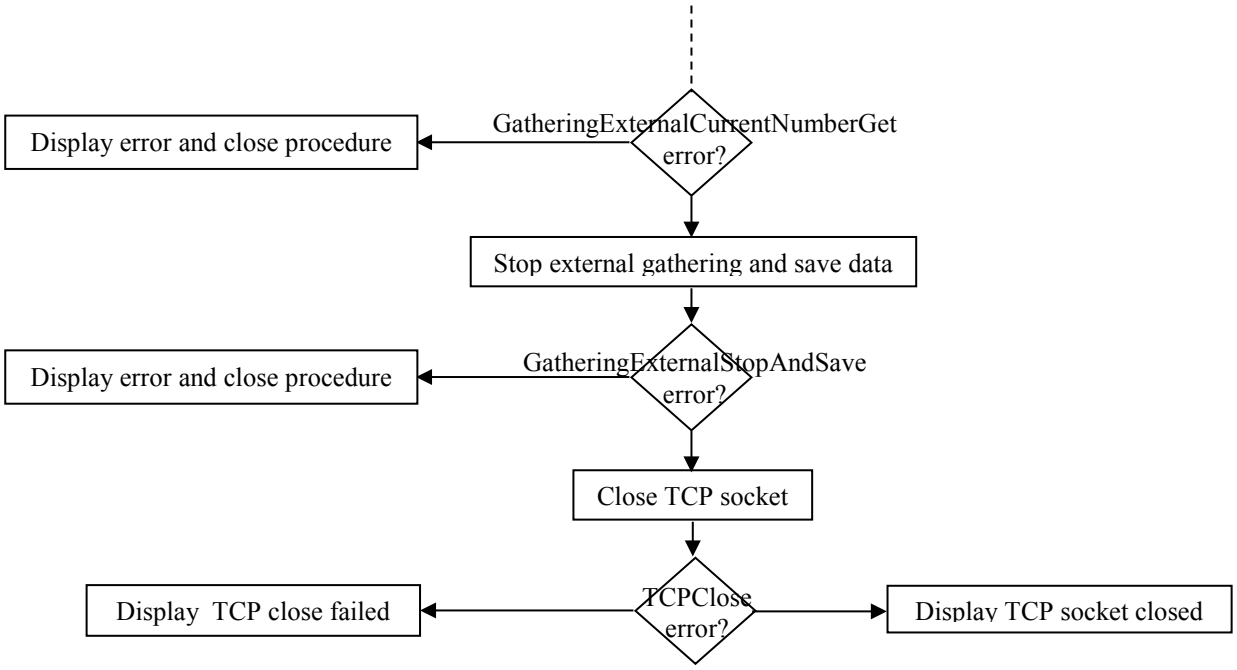
## 4.3 Gathering with Motion



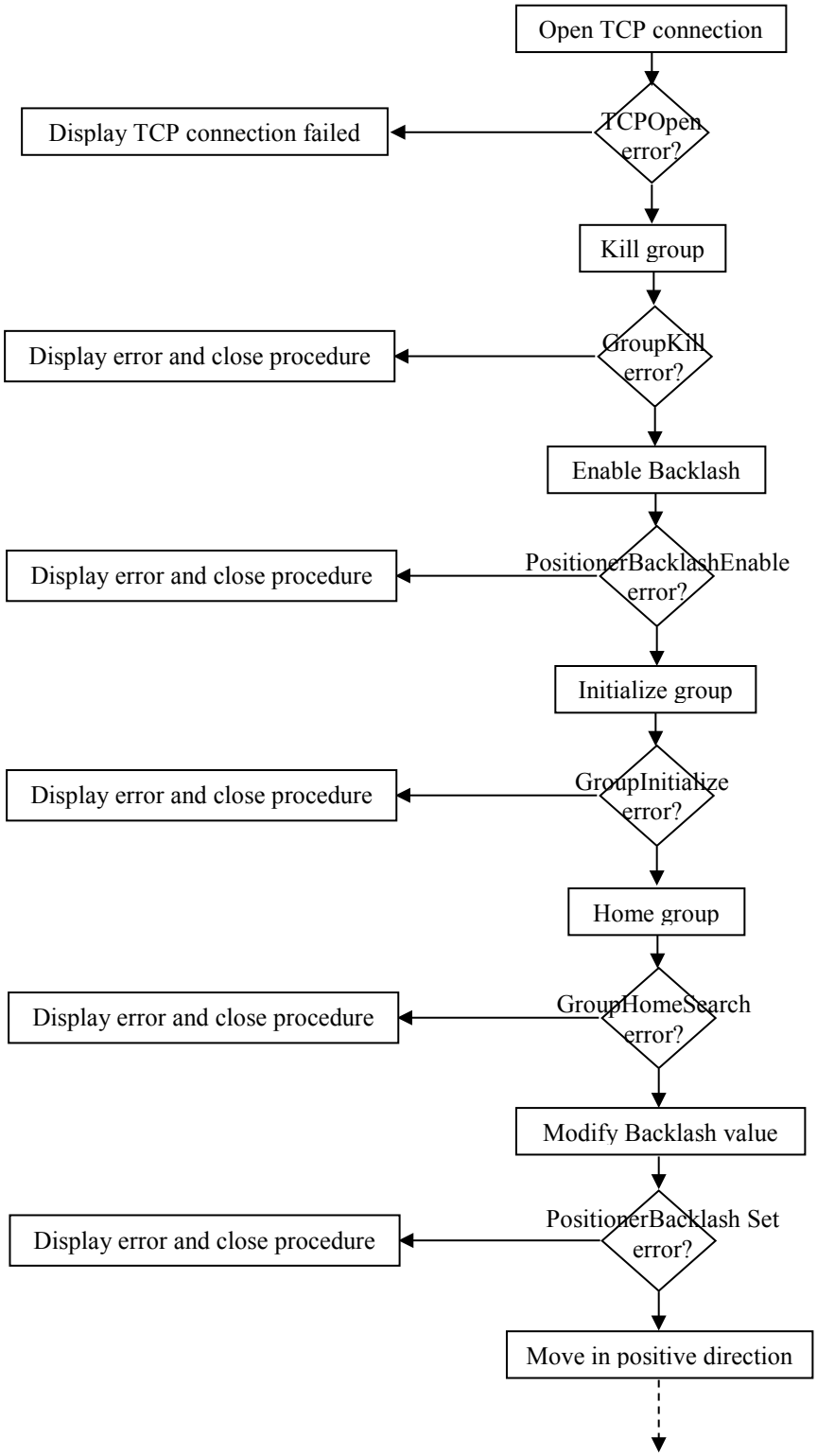

Stop gathering and save data
GatheringStopAndSave error?
Display error and close procedure
Close TCP socket
TCPClose error?
Display TCP close failed
Display TCP socket closed

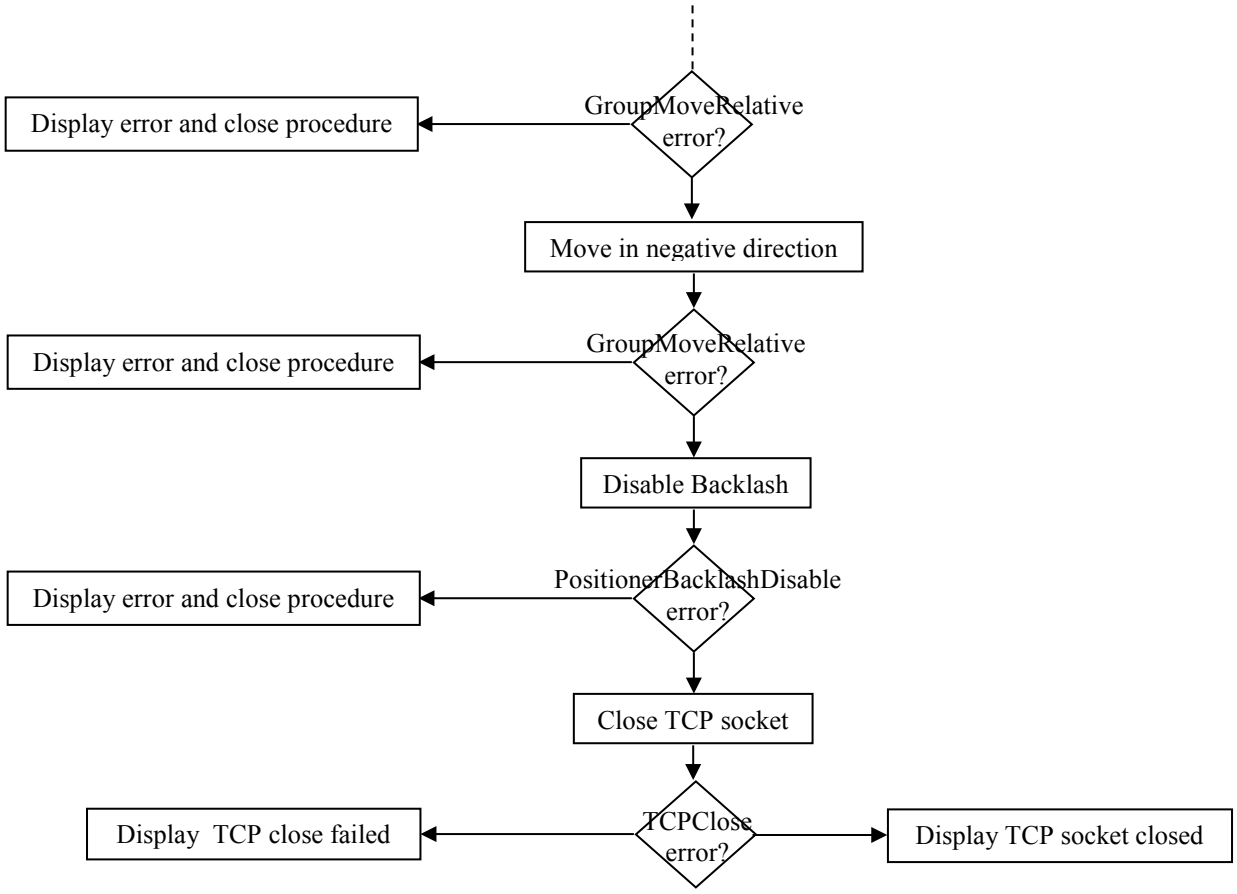
## 4.4 External Gathering



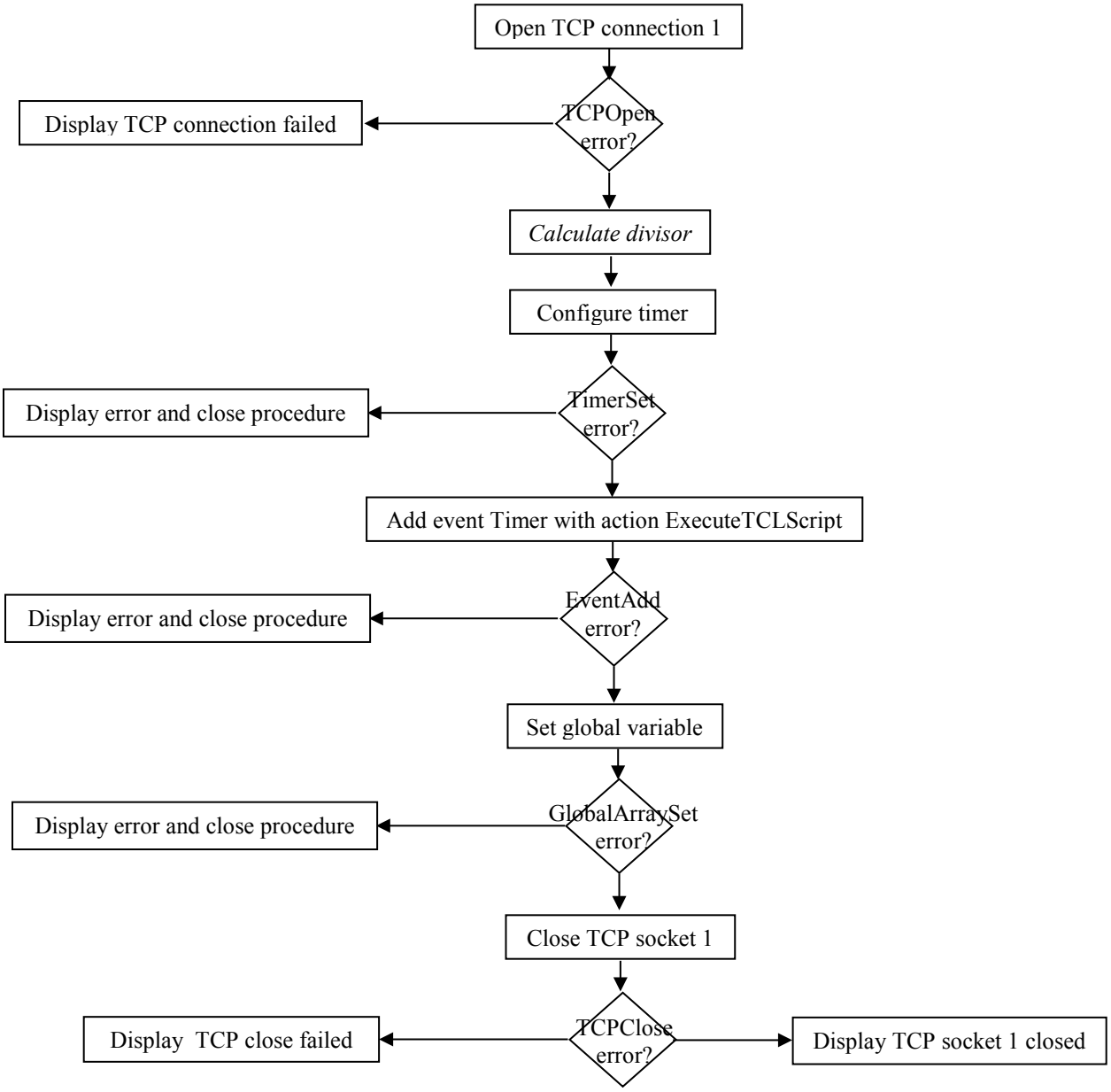

GatheringExternalCurrentNumberGet error?
Display error and close procedure
Stop external gathering and save data
GatheringExternalStopAndSave error?
Display error and close procedure
Close TCP socket
TCPClose error?
Display TCP close failed
Display TCP socket closed

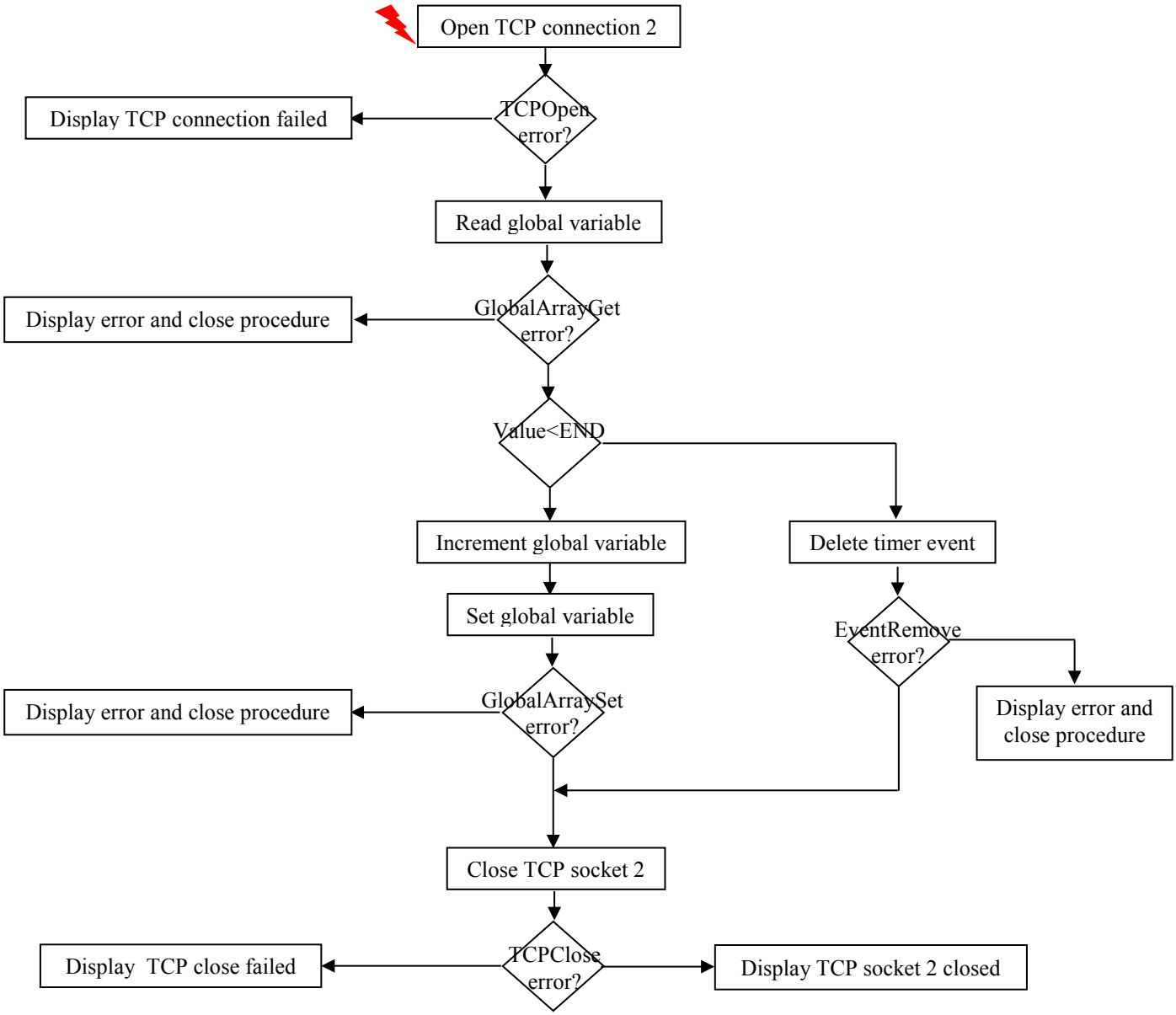
## 4.5 Backlash


Open TCP connection
TCPOpen error?
Display TCP connection failed
Kill group
GroupKill error?
Display error and close procedure
Enable Backlash
PositionerBacklashEnable error?
Display error and close procedure
Initialize group
GroupInitialize error?
Display error and close procedure
Home group
GroupHomeSearch error?
Display error and close procedure
Modify Backlash value
PositionerBacklash Set error?
Display error and close procedure
Move in positive direction


GroupMoveRelative error?
Display error and close procedure
Move in negative direction
GroupMoveRelative error?
Display error and close procedure
Disable Backlash
PositionerBacklashDisable error?
Display error and close procedure
Close TCP socket
TCPClose error?
Display TCP close failed
Display TCP socket closed

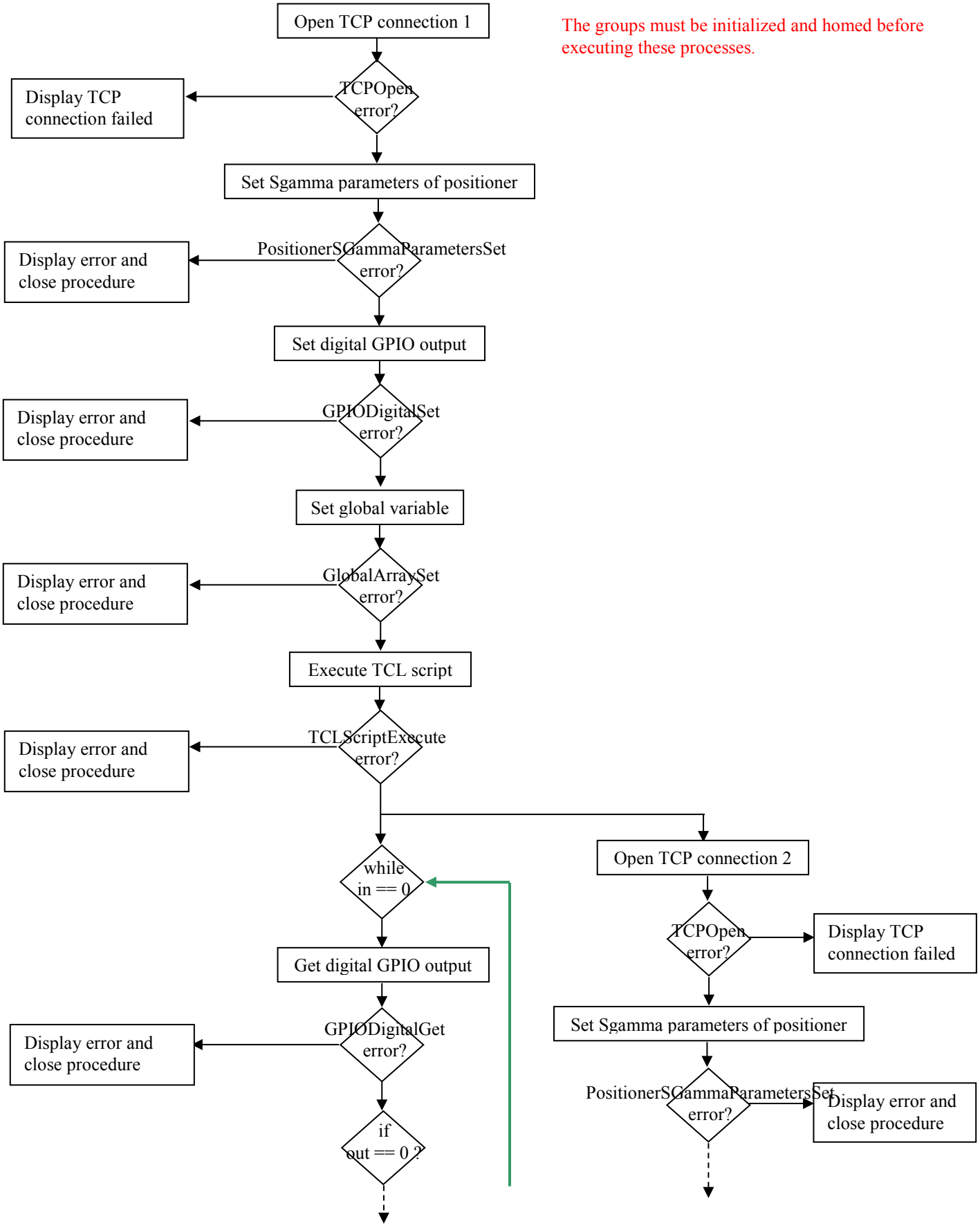
## 4.6 Timer Event and Global Variables

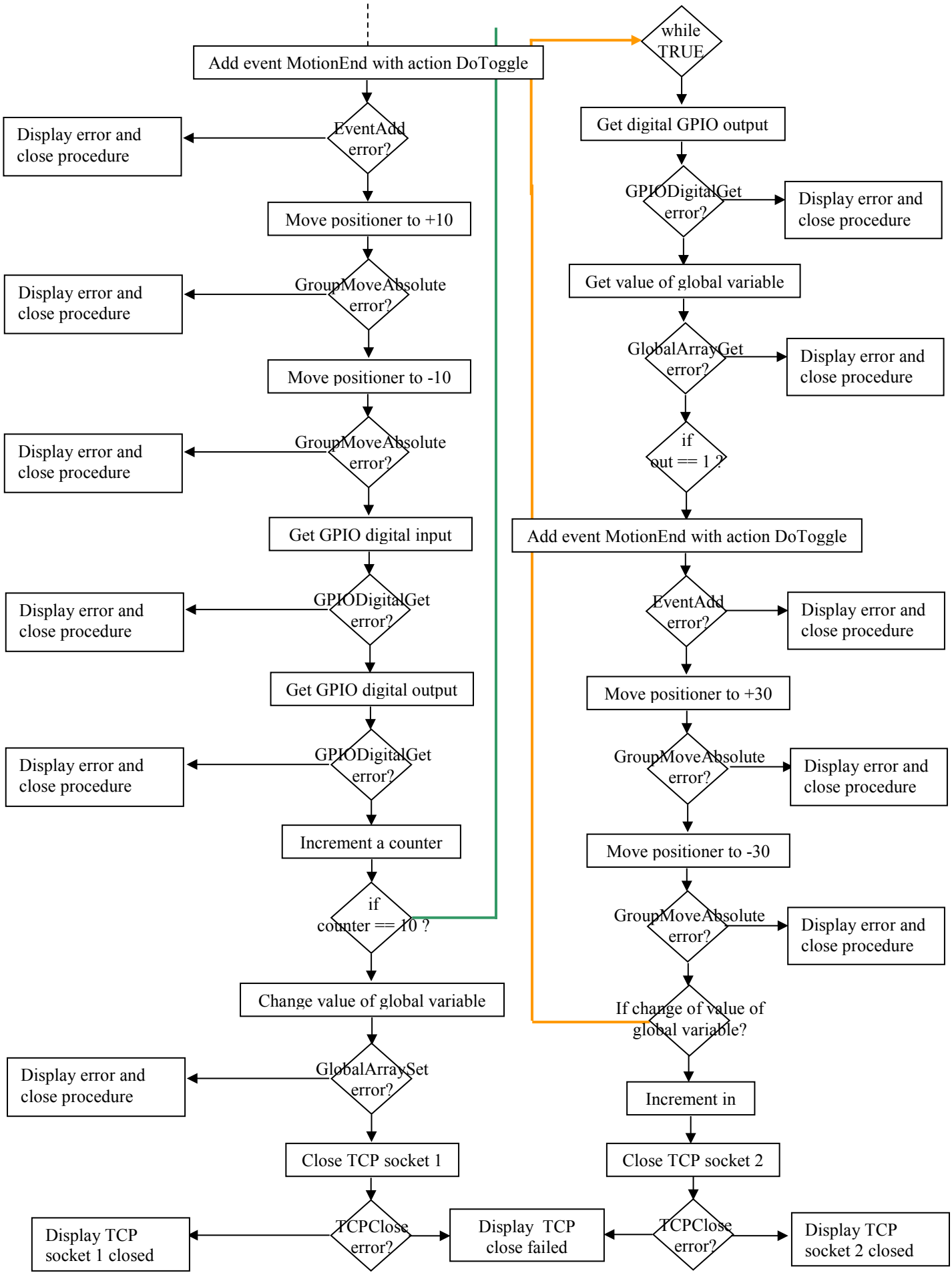



Open TCP connection 2
TCPOpen error?
Display TCP connection failed
Read global variable
GlobalArrayGet error?
Display error and close procedure
Value<END
Increment global variable
Delete timer event
Set global variable
EventRemove error?
GlobalArraySet error?
Display error and close procedure
Display error and close procedure
Close TCP socket 2
TCPClose error?
Display TCP close failed
Display TCP socket 2 closed

## 4.7 Running Simultaneously Several Motion Processes

The groups must be initialized and homed before executing these processes.




Add event MotionEnd with action DoToggle
EventAdd error?
Display error and close procedure
Move positioner to +10
GroupMoveAbsolute error?
Display error and close procedure
Move positioner to -10
GroupMoveAbsolute error?
Display error and close procedure
Get GPIO digital input
GPIODigitalGet error?
Display error and close procedure
Get GPIO digital output
GPIODigitalGet error?
Display error and close procedure
Increment a counter
if counter == 10 ?
Change value of global variable
GlobalArraySet error?
Display error and close procedure
Close TCP socket 1
TCPClose error?
Display TCP socket 1 closed
Display TCP close failed
while TRUE
Get digital GPIO output
GPIODigitalGet error?
Display error and close procedure
Get value of global variable
GlobalArrayGet error?
Display error and close procedure
if out == 1 ?
Add event MotionEnd with action DoToggle
EventAdd error?
Display error and close procedure
Move positioner to +30
GroupMoveAbsolute error?
Display error and close procedure
Move positioner to -30
GroupMoveAbsolute error?
Display error and close procedure
If change of value of global variable?
Increment in
Close TCP socket 2
TCPClose error?
Display TCP socket 2 closed

# Service Form

**Your Local Representative**

Tel.: _____

Fax: _____

Name: _____

Company: _____

Address: _____

Country: _____

P.O. Number: _____

Item(s) Being Returned: _____

Model#: _____

Return authorization #: _____
*(Please obtain prior to return of item)*

Date: _____

Phone Number: _____

Fax Number: _____

Serial #: _____

Description: _____

Reasons of return of goods (please list any specific problems): _____

Newport®

Visit Newport Online at:
www.newport.com


**North America & Asia**
Newport Corporation
1791 Deere Ave.
Irvine, CA 92606, USA

**Sales**
Tel.: (800) 222-6440
e-mail: sales@newport.com

**Technical Support**
Tel.: (800) 222-6440
e-mail: tech@newport.com

**Service, RMAs & Returns**
Tel.: (800) 222-6440
e-mail: service@newport.com

**Europe**
MICRO-CONTROLE Spectra-Physics S.A.S
9, rue du Bois Sauvage
91055 Évry CEDEX
France

**Sales**
Tel.: +33 (0)1.60.91.68.68
e-mail: france@newport.com

**Technical Support**
e-mail: tech_europe@newport.com

**Service & Returns**
Tel.: +33 (0)2.38.40.51.55


mks

Newport® Ophir® Spectra-Physics®