

XPS Controller

Universal High-Performance Motion Controller/Driver



Programmer's Manual V1.1.1



Newport®
Experience | Solutions

Confidentiality & Proprietary Rights

Reservation of Title:

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

Preservation of Secrecy and Confidentiality and Restrictions to Access:

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

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

Table of Contents

1.	Introduction	10
1.1	Scope of the Manual	10
1.1.1	XPS Controller Documentation	10
2.	Firmware	12
2.1	TCP/IP communication	12
2.1.1	Description.....	12
2.1.2	Function description	13
2.1.2.1	<i>OpenConnection.....</i>	<i>13</i>
2.1.2.2	<i>TCP_ConnectToServer</i>	<i>14</i>
2.1.2.3	<i>TCP_SetTimeout</i>	<i>15</i>
2.1.2.4	<i>TCP_CloseSocket.....</i>	<i>16</i>
2.1.2.5	<i>TCP_GetError</i>	<i>17</i>
2.2	Process examples (TCL scripts)	18
2.2.1	Used configuration in examples	19
2.2.2	Gathering with motion.....	20
2.2.3	External gathering.....	21
2.2.4	Output compare	22
2.2.5	Slave-Master mode	23
2.2.6	Jogging.....	24
2.2.7	Tracking.....	25
2.2.8	Backlash.....	26
2.2.9	Timer event.....	27
2.3	Positioner	29
2.3.1	Description.....	29
2.3.1.1	<i>Group.....</i>	<i>29</i>
2.3.1.2	<i>Positioner.....</i>	<i>29</i>
2.3.2	Object structure.....	29
2.3.3	API description	30
2.3.3.1	<i>PositionerAnalogTrackingPositionParametersGet.....</i>	<i>30</i>
2.3.3.2	<i>PositionerAnalogTrackingPositionParametersSet.....</i>	<i>31</i>
2.3.3.3	<i>PositionerAnalogTrackingVelocityParametersGet</i>	<i>32</i>
2.3.3.4	<i>PositionerAnalogTrackingVelocityParametersSet.....</i>	<i>33</i>
2.3.3.5	<i>PositionerBacklashEnable</i>	<i>35</i>
2.3.3.6	<i>PositionerBacklashDisable</i>	<i>36</i>
2.3.3.7	<i>PositionerBacklashSet.....</i>	<i>37</i>
2.3.3.8	<i>PositionerBacklashGet.....</i>	<i>38</i>
2.3.3.9	<i>PositionerCorrectorPIDDualFFVoltageGet.....</i>	<i>39</i>
2.3.3.10	<i>PositionerCorrectorPIDDualFFVoltageSet.....</i>	<i>41</i>
2.3.3.11	<i>PositionerCorrectorPIDFFAccelerationGet.....</i>	<i>44</i>
2.3.3.12	<i>PositionerCorrectorPIDFFAccelerationSet.....</i>	<i>46</i>
2.3.3.13	<i>PositionerCorrectorPIDFFVelocityGet</i>	<i>49</i>
2.3.3.14	<i>PositionerCorrectorPIDFFVelocitySet.....</i>	<i>51</i>
2.3.3.15	<i>PositionerCorrectorPIPPositionGet</i>	<i>54</i>

Preface

2.3.3.16	<i>PositionerCorrectorPIPositionSet</i>	55
2.3.3.17	<i>PositionerCorrectorTypeGet</i>	56
2.3.3.18	<i>PositionerCorrectorNotchFiltersGet</i>	57
2.3.3.19	<i>PositionerCorrectorNotchFiltersSet</i>	58
2.3.3.20	<i>PositionerMotionDoneGet</i>	60
2.3.3.21	<i>PositionerMotionDoneSet</i>	61
2.3.3.22	<i>PositionerPreviousMotionTimesGet</i>	63
2.3.3.23	<i>PositionerSGammaParametersSet</i>	64
2.3.3.24	<i>PositionerSGammaParametersGet</i>	65
2.3.3.25	<i>PositionerSGammaExactVelocityAjustedDisplacementGet</i>	66
2.3.3.26	<i>PositionerErrorGet</i>	67
2.3.3.27	<i>PositionerErrorStringGet</i>	68
2.3.3.28	<i>PositionerHardwareStatusGet</i>	69
2.3.3.29	<i>PositionerHardwareStatusStringGet</i>	70
2.3.3.30	<i>PositionerUserTravelLimitsGet</i>	71
2.3.3.31	<i>PositionerUserTravelLimitsSet</i>	72
2.3.3.32	<i>PositionerPositionCompareSet</i>	73
2.3.3.33	<i>PositionerPositionCompareGet</i>	74
2.3.3.34	<i>PositionerPositionCompareEnable</i>	75
2.3.3.35	<i>PositionerPositionCompareDisable</i>	76
2.3.3.36	<i>PositionerHardInterpolatorFactorSet</i>	77
2.3.3.37	<i>PositionerHardInterpolatorFactorGet</i>	78
2.3.3.38	<i>PositionerEncoderAmplitudeValuesGet</i>	79
2.3.3.39	<i>PositionerEncoderCalibrationParametersGet</i>	79
2.3.3.40	<i>GroupJogCurrentGet</i>	80
2.3.3.41	<i>GroupJogParametersGet</i>	82
2.3.3.42	<i>GroupJogParametersSet</i>	83
2.3.3.43	<i>GroupMoveAbsolute</i>	84
2.3.3.44	<i>GroupMoveRelative</i>	85
2.3.3.45	<i>GroupPositionCurrentGet</i>	86
2.3.3.46	<i>GroupPositionSetpointGet</i>	87
2.3.3.47	<i>GroupPositionTargetGet</i>	88
2.3.4	Configuration parameters	89
2.3.4.1	<i>Travels</i>	89
2.3.4.2	<i>Profiler</i>	89
2.3.4.3	<i>Motion done</i>	90
2.3.4.4	<i>Positioner mapping</i>	91
2.3.4.5	<i>Configuration parameters</i>	92
2.4	SingleAxis group	98
2.4.1	Description.....	98
2.4.2	Structure.....	98
2.4.3	State diagram	98
2.4.4	API description	99
2.4.4.1	<i>GroupAnalogTrackingModeDisable</i>	99
2.4.4.2	<i>GroupAnalogTrackingModeEnable</i>	100
2.4.4.3	<i>GroupHomeSearch</i>	101
2.4.4.4	<i>GroupHomeSearchAndRelativeMove</i>	102
2.4.4.5	<i>GroupInitialize</i>	103
2.4.4.6	<i>GroupInitializeWithEncoderCalibration</i>	104
2.4.4.7	<i>GroupJogModeDisable</i>	105

Preface

2.4.4.8	<i>GroupJogModeEnable</i>	106
2.4.4.9	<i>GroupJogCurrentGet</i>	107
2.4.4.10	<i>GroupJogParametersGet</i>	108
2.4.4.11	<i>GroupJogParametersSet</i>	109
2.4.4.12	<i>GroupKill</i>	110
2.4.4.13	<i>GroupMotionDisable</i>	111
2.4.4.14	<i>GroupMotionEnable</i>	112
2.4.4.15	<i>GroupMoveAbort</i>	113
2.4.4.16	<i>GroupMoveAbsolute</i>	114
2.4.4.17	<i>GroupMoveRelative</i>	115
2.4.4.18	<i>GroupPositionCurrentGet</i>	116
2.4.4.19	<i>GroupPositionSetpointGet</i>	117
2.4.4.20	<i>GroupPositionTargetGet</i>	118
2.4.4.21	<i>GroupStatusGet</i>	119
2.4.4.22	<i>SingleAxisSlaveParametersSet</i>	120
2.4.4.23	<i>SingleAxisSlaveParametersGet</i>	121
2.4.4.24	<i>SingleAxisSlaveModeEnable</i>	122
2.4.4.25	<i>SingleAxisSlaveModeDisable</i>	123
2.4.5	Configuration file	124
2.5	XY group	125
2.5.1	Description	125
2.5.2	Structure	125
2.5.3	State diagram	125
2.5.4	Additional XY correction from other object	126
2.5.5	API description	127
2.5.5.1	<i>GroupAnalogTrackingModeDisable</i>	127
2.5.5.2	<i>GroupAnalogTrackingModeEnable</i>	128
2.5.5.3	<i>GroupHomeSearch</i>	129
2.5.5.4	<i>GroupHomeSearchAndRelativeMove</i>	130
2.5.5.5	<i>GroupInitialize</i>	131
2.5.5.6	<i>GroupInitializeWithEncoderCalibration</i>	132
2.5.5.7	<i>GroupJogModeDisable</i>	133
2.5.5.8	<i>GroupJogModeEnable</i>	134
2.5.5.9	<i>GroupJogCurrentGet</i>	135
2.5.5.10	<i>GroupJogParametersGet</i>	136
2.5.5.11	<i>GroupJogParametersSet</i>	137
2.5.5.12	<i>GroupKill</i>	138
2.5.5.13	<i>GroupMotionDisable</i>	139
2.5.5.14	<i>GroupMotionEnable</i>	140
2.5.5.15	<i>GroupMoveAbort</i>	141
2.5.5.16	<i>GroupMoveAbsolute</i>	142
2.5.5.17	<i>GroupMoveRelative</i>	143
2.5.5.18	<i>GroupPositionCurrentGet</i>	144
2.5.5.19	<i>GroupPositionSetpointGet</i>	145
2.5.5.20	<i>GroupPositionTargetGet</i>	146
2.5.5.21	<i>GroupStatusGet</i>	147
2.5.5.22	<i>XYLineArcVerification</i>	148
2.5.5.23	<i>XYLineArcVerificationResultGet</i>	149
2.5.5.24	<i>XYLineArcExecution</i>	150
2.5.5.25	<i>XYLineArcParametersGet</i>	151

2.5.6	LineArc trajectory file	152
2.5.6.1	Trajectory file description.....	152
2.5.6.2	Trajectory file example	152
2.3.7	Configuration file	153
2.6	XYZ group.....	154
2.6.1	Description.....	154
2.6.2	Structure.....	154
2.6.3	State diagram	154
2.6.4	XYZ Mapping.....	155
2.6.5	API description	156
2.6.5.1	GroupAnalogTrackingModeDisable	156
2.6.5.2	GroupAnalogTrackingModeEnable	157
2.6.5.3	GroupHomeSearch.....	158
2.6.5.4	GroupHomeSearchAndRelativeMove.....	159
2.6.5.5	GroupInitialize	160
2.6.5.6	GroupInitializeWithEncoderCalibration.....	161
2.6.5.7	GroupJogModeDisable	162
2.6.5.8	GroupJogModeEnable	163
2.6.5.9	GroupJogCurrentGet.....	164
2.6.5.10	GroupJogParametersGet	165
2.6.5.11	GroupJogParametersSet.....	166
2.6.5.12	GroupKill	167
2.6.5.13	GroupMotionDisable	168
2.6.5.14	GroupMotionEnable	169
2.6.5.15	GroupMoveAbort	170
2.6.5.16	GroupMoveAbsolute	171
2.6.5.17	GroupMoveRelative	172
2.6.5.18	GroupPositionCurrentGet.....	173
2.6.5.19	GroupPositionSetpointGet	174
2.6.5.20	GroupPositionTargetGet.....	175
2.6.5.21	GroupStatusGet.....	176
2.6.5.22	XYZSplineVerification.....	177
2.6.5.23	XYZSplineVerificationResultGet	178
2.6.5.24	XYZSplineExecution.....	179
2.6.5.25	XYZSplineParametersGet.....	180
2.6.6	Spline trajectory file	181
2.6.6.1	Trajectory file description.....	181
2.6.6.2	Spline trajectory data file example.....	182
2.6.7	Configuration file	183
2.7	MultipleAxes group.....	185
2.7.1	Description.....	185
2.7.2	Structure.....	185
2.7.3	State diagram	185
2.7.4	API description	186
2.7.4.1	GroupAnalogTrackingModeDisable	186
2.7.4.2	GroupAnalogTrackingModeEnable	187
2.7.4.3	GroupHomeSearch.....	188
2.7.4.4	GroupHomeSearchAndRelativeMove.....	189
2.7.4.5	GroupInitialize	190

Preface

2.7.4.6	<i>GroupInitializeWithEncoderCalibration</i>	191
2.7.4.7	<i>GroupJogModeDisable</i>	192
2.7.4.8	<i>GroupJogModeEnable</i>	193
2.7.4.9	<i>GroupJogCurrentGet</i>	194
2.7.4.10	<i>GroupJogParametersGet</i>	195
2.7.4.11	<i>GroupJogParametersSet</i>	196
2.7.4.12	<i>GroupKill</i>	197
2.7.4.13	<i>GroupMotionDisable</i>	198
2.7.4.14	<i>GroupMotionEnable</i>	199
2.7.4.15	<i>GroupMoveAbort</i>	200
2.7.4.16	<i>GroupMoveAbsolute</i>	201
2.7.4.17	<i>GroupMoveRelative</i>	202
2.7.4.18	<i>GroupPositionCurrentGet</i>	203
2.7.4.19	<i>GroupPositionSetpointGet</i>	204
2.7.4.20	<i>GroupPositionTargetGet</i>	205
2.7.4.21	<i>GroupStatusGet</i>	206
2.7.4.22	<i>MultipleAxesPVTVerification</i>	207
2.7.4.23	<i>MultipleAxesPVTVerificationResultGet</i>	208
2.7.4.24	<i>MultipleAxesPVTExecution</i>	209
2.7.4.25	<i>MultipleAxesPVTParametersGet</i>	210
2.7.5	PVT trajectory file	211
2.7.5.1	File description	211
2.7.5.2	File example.....	211
2.7.6	Configuration files	212
2.8	General features	213
2.8.1.1	<i>ErrorStringGet</i>	213
2.8.1.2	<i>ElapsedTimeGet</i>	214
2.8.1.3	<i>FirmwareVersionGet</i>	215
2.8.1.4	<i>GroupStatusStringGet</i>	216
2.8.1.5	<i>GlobalArrayGet</i>	217
2.8.1.6	<i>GlobalArraySet</i>	218
2.8.1.7	<i>KillAll</i>	219
2.8.1.8	<i>Reboot</i>	220
2.8.1.9	<i>TimerSet</i>	221
2.8.1.10	<i>TimerGet</i>	222
2.9	Analog and digital I/O	223
2.9.1	GPIO name list	223
2.9.2	API description	224
2.9.2.1	<i>GPIOAnalogGet</i>	224
2.9.2.2	<i>GPIOAnalogSet</i>	225
2.9.2.3	<i>GPIOAnalogGainGet</i>	226
2.9.2.4	<i>GPIOAnalogGainSet</i>	227
2.9.2.5	<i>GPIODigitalGet</i>	228
2.9.2.6	<i>GPIODigitalSet</i>	229
2.10	Gathering	230
2.10.1	Internal Gathering and External Gathering.....	230
2.10.2	Definition of different positions for one positioner	231
2.10.3	Sequence	232
2.10.4	Gathering file	232

2.10.5	API description	233
2.10.5.1	<i>GatheringConfigurationGet</i>	233
2.10.5.2	<i>GatheringConfigurationSet</i>	234
2.10.5.3	<i>GatheringCurrentNumberGet</i>	235
2.10.5.4	<i>GatheringStopAndSave</i>	236
2.10.5.5	<i>GatheringExternalConfigurationGet</i>	237
2.10.5.6	<i>GatheringExternalConfigurationSet</i>	238
2.10.5.7	<i>GatheringExternalCurrentNumberGet</i>	239
2.10.5.8	<i>GatheringExternalStopAndSave</i>	240
2.11	Events and actions	241
2.11.1	Sequence	241
2.11.2	Events	241
2.11.2.1	<i>Event categories</i>	241
2.11.2.2	<i>Event definitions</i>	243
2.11.3	Actions	250
2.11.3.1	<i>List of actions</i>	250
2.11.3.2	<i>DOToggle</i>	250
2.11.3.3	<i>DOPulse</i>	251
2.11.3.4	<i>DOSet</i>	251
2.11.3.5	<i>DACSet.SetpointPosition</i>	252
2.11.3.6	<i>DACSet.SetpointVelocity</i>	252
2.11.3.7	<i>DACSet.SetpointAcceleration</i>	253
2.11.3.8	<i>ExecuteTCLScript</i>	253
2.11.3.9	<i>GatheringRun</i>	254
2.11.3.10	<i>ExternalGatheringRun</i>	254
2.11.4	API Description	255
2.11.4.1	<i>EventAdd</i>	255
2.11.4.2	<i>EventRemove</i>	256
2.11.4.3	<i>EventGet</i>	257
2.11.4.4	<i>EventWait</i>	258
2.12	Positioner error list	259
2.13	Positioner hardware status list	259
2.14	Group state list	260
2.15	Error list	261
3.	Using the Controller	263
3.1	Directory access rights and privileges	263
3.1.1	Root directory	264
3.1.2	ADMIN directory	264
3.1.3	Firmware directory	264
3.1.4	Webfiles directory	264
3.1.5	CONFIG directory	264
3.1.6	Public directory	265
3.1.7	Scripts directory	265
3.1.8	Trajectories directory	265
3.1.9	Drivers	265
3.2	Configuration	266
3.2.1	System.ini file	266

3.2.2	Stages.ini file	266
3.3	TCL programming	267
3.3.1	TCL using	267
3.3.2	TCL arguments	267
3.3.3	TCL errors	267
3.3.4	Boot TCL script	267
3.3.5	TCL APIs driver	268
3.3.5.1	<i>TCLScriptExecute</i>	268
3.3.5.2	<i>TCLScriptKill</i>	269
3.4	Version	270
3.4.1.1	<i>GetLibraryVersion</i>	270

1. Introduction

1.1 Scope of the Manual

1.1.1 XPS Controller Documentation

To maximize the value of the XPS Controller/Driver system, it is important that users become thoroughly familiar with available documentation:

- The User's manual is the getting-started part of the system. It serves as an introduction and as a reference. It includes:
 - System Overview
 - Getting Started Guide
 - Hardware Details
 - Ethernet Connection
 - Troubleshooting
 - Maintenance and Service
- Additional manuals complement the User's Manual. These include:

Programmer's manual

(printed by the customer from the XPS Controller web-site to provide users access to the most actual documentation).

This book is the general programmer's guide for the experienced user. It provides a short description of all features, but can take references to the motion tutorial for more exhaustive definitions and descriptions. It contains:

- Definition/Description of general features
- Overview of all features and API's by groups, functions, and alphabetic
- State diagrams
- Detailed list of all API's with complete syntax, parameters, errors, and short descriptions

Software drivers manual

(printed by the customer from the XPS Controller web-site to provide users access to the most actual documentation).

This book provides all information about software tools and drivers shipped with the XPS:

- Visual C++ / Visual Basic documentation (deliverable with DLL)
- LabView VI documentation (deliverable with VI)

TCL manual

(printed by the customer from the XPS Controller web-site to provide users access to the most actual documentation).

This book provides a short introduction to TCL, covers all XPS specific topics and contains the public documentation.

Motion Tutorial

The purpose of this book is to provide an exhaustive description of the XPS architecture, and its features and capabilities. Different than the programmer's guide, this book is educational and is organized by features starting by the basics and getting to the more advanced. It provides a more complete list of specifications for the different features. It contains:

- Explanations of XPS architecture
- Objects, Definitions, Physics, Consequences
- State diagrams
- Groups of motion, purpose and benefits
- Securities
- Features of motion groups
- Why different motion groups?
- Overview of features per motion group
- Guideline for assignment of actuators to motion groups per application
- Control loops
- Architecture
- Stage tuning: When it is an issue?
- Benefits of XPS specific control loop architecture

2. Firmware

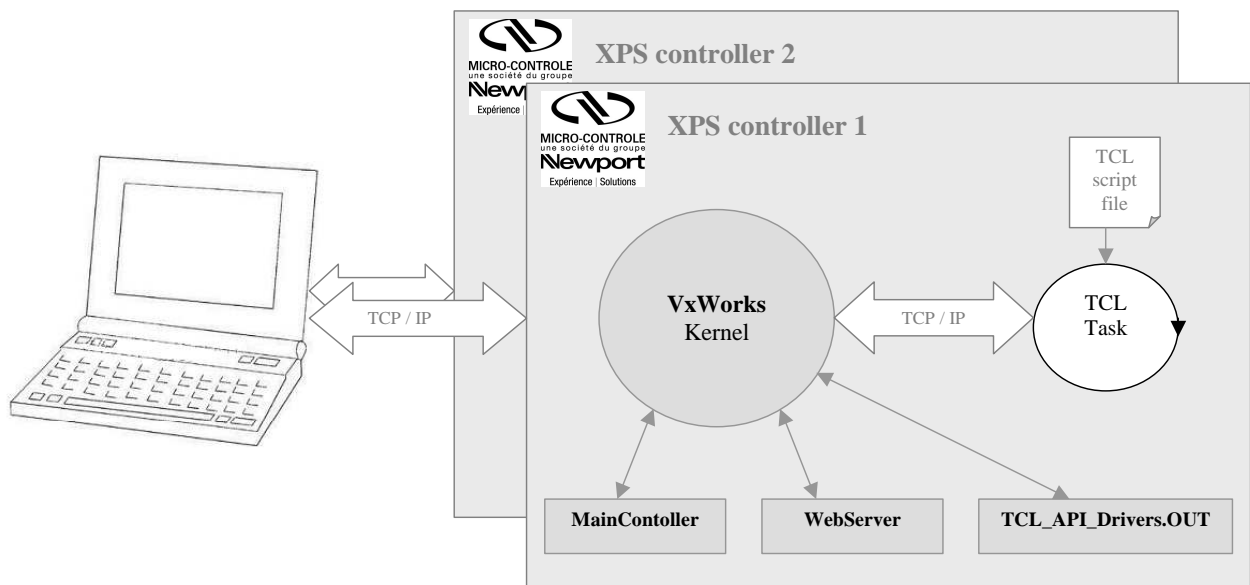
2.1 TCP/IP communication

2.1.1 Description

XPS is based on a high-performance 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 data transfer. This makes it almost 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, data transfer or diagnostics. The completely object oriented approach of the XPS firmware with powerful, multi-parameter API's (commands) is also much more self-consistent and intuitive to use than old-style mnemonic commands.

To connect to the XPS controller you must open a socket with the "TCP_ConnectToServer" API and gets the socket identifiant to use each API.



2.1.2 Function description

2.1.2.1 OpenConnection

TCL Prototype	int OpenConnection (double TimeOut, int * SocketID)
Input parameters	double : TimeOut (Timeout in seconds used for each API execution)
Output parameters	int * : SocketID (Socket identifiant used in each API function)
Return	TCL error (0 = success or 1 = syntax error)

Input tests	Verify the number of parameters. Parameters coherence test.
Description	<p>To use in a TCL script that is located in the “Scripts” directory of your controller.</p> <p>Configure the TCP/IP communication (Local Host Address = 127.0.0.1 and IP Port = 5001).</p> <p>Open a socket to connect TCP server.</p> <p>Gets a socket identifier to use for each API function call.</p> <p><u>NOTE :</u> If “SocketID” is –1 then the TCP/IP connection failed.</p>

2.1.2.2 TCP_ConnectToServer

TCL Prototype	int TCP_ConnectToServer (char * IP_Address, int IP_Port, double TimeOut, int * SocketID)
Input parameters	char [250] : IP_Address (TCP IP address : 195.168.33.xxx or another) int : PI_Port (TCP IP port : 5001 for XPS controller) double : TimeOut (Timeout in seconds used for each API execution)
Output parameters	int * : SocketID (Socket identifiant used in each API function)
Return	TCL error (0 = success or 1 = syntax error)

DLL Prototype	int TCP_ConnectToServer (char * IP_Address, int IP_Port, double TimeOut)
Input parameters	char [250] : IP_Address (TCP IP address : 195.168.33.xxx or another) int : PI_Port (TCP IP port : 5001 for XPS controller) double : TimeOut (Timeout in seconds used for each API execution)
Output parameters	None
Return	int SocketID : Socket identifiant used in each API function

Input tests	Verify the number of parameters. Parameters coherence test.
Description	Configure the TCP/IP communication. Open a socket to connect TCP server. Gets a socket identifier to use for each API function call. NOTE: If "SocketID" is -1 then the TCP/IP connection failed.

2.1.2.3 TCP_SetTimeout

TCL Prototype	int TCP_SetTimeout (int SocketID, double TimeOut)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) double : TimeOut (seconds)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error)

DLL Prototype	void TCP_SetTimeout (int SocketID, double TimeOut)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) double : TimeOut (seconds)
Output parameters	None
Return	None

Input tests	Verify the number of parameters. Parameters coherence test.
Description	Set a new timeout for TCP communication.

2.1.2.4 TCP_CloseSocket

TCL Prototype	int TCP_CloseSocket (int SocketID)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error)

DLL Prototype	void TCP_CloseSocket (int SocketID)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API)
Output parameters	None
Return	None

Input tests	Verify the number of parameters. Parameters coherence test.
Description	Close the socket selected by SocketID.

2.1.2.5 TCP_GetError

DLL Prototype	int TCP_GetError (int SocketID, char * ErrorString)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API)
Output parameters	char * : ErrorString
Return	TCL error (0 = success or 1 = syntax error)

TCL Prototype	char * TCP_GetError (int SocketID)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API)
Output parameters	None
Return	char * ErrorString : error description

Input tests	Verify the number of parameters. Parameters coherence test.
Description	Gets the last error from the socket selected by SocketID.

2.2 Process examples (TCL scripts)

To do a defensive code, the error must be read and tested.

```
# Initialization
set TimeOut 60
set ErrorCode 0

# Open TCP socket
set ErrorCode [catch "OpenConnection $TimeOut socketID"]
if {$ErrorCode == 0} {

    # Success => Read firmware version
    set ErrorCode [catch "FirmwareVersionGet $socketID FirmwareVersion"]
    if {$ErrorCode == 0} {

        # Success => Display firmware version
        puts stdout "Controller version is $FirmwareVersion"

    } else {

        # Error => Get error description
        set ErrorCode [catch "ErrorStringGet $socketID $ErrorCode ErrorString"]
        if {$ErrorCode == 0} {

            # Display error description
            puts stdout "$ErrorCode : $ErrorString"

        }
    }
}

# Close TCP socket
set ErrorCode [catch "TCP_CloseSocket $socketID"]
if {$ErrorCode == 0} {

    # Success
    puts stdout "The socket $socketID is closed "

} else {

    # Error
    puts stdout "TCP_CloseSocket failed => $ErrorCode"

}
} else {

    # Error => TCP socket not opened
    puts stdout "TCP_ConnectToServer failed !"

}
}
```

2.2.1 Used configuration in examples

The system configuration, that is used in the examples, is :

<i>Group type</i>	<i>Number</i>	<i>Group name</i>	<i>Positioner name</i>
SingleAxis	1	SINGLE_AXIS	SINGLE_AXIS.MY_STAGE
XY	1	XY	XY.X XY.Y
XYZ	0		
MultipleAxes	0		

2.2.2 Gathering with motion

Process to realize a gathering with a motion.

```
# Initialization
set TimeOut 60

set Group "SINGLE_AXIS"
set Positioner "SINGLE_AXIS.MY_STAGE"

set Type1 "SINGLE_AXIS.MY_STAGE.SetpointPosition"
set Type2 "SINGLE_AXIS.MY_STAGE.CurrentPosition"

set Event "SGamma.MotionStart"
set Action "GatheringRun"
set NbPoints 1000
set Div 1

set ErrCode 0

# Open TCP socket
set ErrCode [catch "OpenConnection $TimeOut socketID"]

# Initialize Group
set ErrCode [catch "GroupInitialize $socketID $Group"]

# Search home Group
set ErrCode [catch "GroupHomeSearch $socketID $Group"]

# Configure gathering
set ErrCode [catch "GatheringConfigurationSet $socketID $Type1 $Type2"]

# Add an event
set ErrCode [catch "EventAdd $socketID $Positioner $Event 0 $Action $NbPoints $Div 0"]

# Move positioner
set ErrCode [catch "GroupMoveRelative $socketID $Group $Displacement"]

# Stop gathering and save data
set ErrCode [catch "GatheringStopAndSave $socketID"]

# Close TCP socket
set ErrCode [catch "TCP_CloseSocket $socketID"]
```

2.2.3 External gathering

Process to realize a external gathering : the external positions are updated by an external trigger (TRIG IN connector).

```
# Initialization
set TimeOut 60

set Group "SINGLE_AXIS"
set Positioner "SINGLE_AXIS.MY_STAGE"

set Type1 "SINGLE_AXIS.MY_STAGE.ExternalLatchPosition"
set Type2 "GPIO2.ADC1"

set Event "Immediate"
set Action "ExternalGatheringRun"
set NbPoints 1000
set Div 1

set Current 0
set ErrCode 0

# Open TCP socket
set ErrCode [catch "OpenConnection $TimeOut socketID"]

# Initialize Group
set ErrCode [catch "GroupInitialize $socketID $Group"]

# Search home Group
set ErrCode [catch "GroupHomeSearch $socketID $Group"]

# Configure a gathering
set ErrCode [catch "GatheringExternalConfigurationSet $socketID $Type1 $Type2"]

# Add an event
set ErrCode [catch "EventAdd $socketID $Positioner $Event 0 $Action $NbPoints $Div 0"]

# Wait end of external gathering
while {$Current < $NbPoints} {

    # Get current number realized
    set ErrCode [catch "GatheringExternalCurrentNumberGet $socketID Current Max"]

}

# Stop external gathering and save data
set ErrCode [catch "GatheringExternalStopAndSave $socketID"]

# Close TCP socket
set ErrCode [catch "TCP_CloseSocket $socketID"]
```

2.2.4 Output compare

Process to realize a position compare (PCO connector)

```
# Initialization
set TimeOut 60

set Group "SINGLE_AXIS"
set Positioner "SINGLE_AXIS.MY_STAGE"

set StartPosition -15
set Displacement 25

set MinPos -10
set MaxPos 10
set StepPos 1

set ErrCode 0

# Open TCP socket
set ErrCode [catch "OpenConnection $TimeOut socketID"]

# Initialize Group
set ErrCode [catch "GroupInitialize $socketID $Group"]

# Search home Group
set ErrCode [catch "GroupHomeSearch $socketID $Group"]

# Move positioner to start position
set ErrCode [catch "GroupMoveAbsolute $socketID $Group $StartPosition"]

# Set position compare parameters
set ErrCode [catch "PositionerPositionCompareSet $socketID $Positioner $MinPos
$MaxPos $StepPos"]

# Enable position compare mode
set ErrCode [catch "PositionerPositionCompareEnable $socketID $Positioner"]

# Move positioner
set ErrCode [catch "GroupMoveRelative $socketID $Group $Displacement"]

# Disable position compare mode
set ErrCode [catch "PositionerPositionCompareDisable $socketID $Positioner"]

# Close TCP socket
set ErrCode [catch "TCP_CloseSocket $socketID"]
```

2.2.5 Slave-Master mode

Process to realize a slave-master mode.

```
# Initialization
set TimeOut 60

set SlaveGroup "SINGLE_AXIS"
set XYGroup "XY"
set MasterPositioner "XY.X"

set MasterRatio 1
set ErrCode 0

# Open TCP socket
set ErrCode [catch "OpenConnection $TimeOut socketID"]

# Initialize Single Axis Group
set ErrCode [catch "GroupInitialize $socketID $SlaveGroup"]

# Search home Single Axis Group
set ErrCode [catch "GroupHomeSearch $socketID $SlaveGroup"]

# Initialize XY Group
set ErrCode [catch "GroupInitialize $socketID $XYGroup"]

# Search home XY Group
set ErrCode [catch "GroupHomeSearch $socketID $XYGroup"]

# Set Slave (SingleAxis group) with its Master (positioner from any group : XY
here)
set ErrCode [catch "SingleAxisSlaveParametersSet $socketID $SlaveGroup
$MasterPositioner $MasterRatio"]

# Enable slave-master mode (group must be Ready)
set ErrCode [catch "SingleAxisSlaveModeEnable $socketID $SlaveGroup"]

# Move Master positioner (the slave must follow the master in relation to a
ratio)
set ErrCode [catch "GroupMoveRelative $socketID $MasterPositioner $Displacement"]

# Disable slave-master mode
set ErrCode [catch "SingleAxisSlaveModeDisable $socketID $SlaveGroup"]

# Close TCP socket
set ErrCode [catch "TCP_CloseSocket $socketID"]
```

2.2.6 Jogging

Process to realize a jogging.

```
# Initialization
set TimeOut 60

set Group "XY"
set Positioner "XY.X"

set Velocity 20
set Acceleration 80

set ErrCode 0

# Open TCP socket
set ErrCode [catch "OpenConnection $TimeOut socketID"]

# Initialize Group
set ErrCode [catch "GroupInitialize $socketID $Group"]

# Search home Group
set ErrCode [catch "GroupHomeSearch $socketID $Group"]

# Enable Jog mode (group must be Ready)
set ErrCode [catch "GroupJogModeEnable $socketID $Group"]

# Set Jog parameters to move a positioner => constant velocity is not null
set ErrCode [catch "GroupJogParametersSet $socketID $Positioner $Velocity $acceleration"]

# Set Jog parameters to stop a positioner => constant velocity is null
set ErrCode [catch "GroupJogParametersSet $socketID $Positioner 0 $acceleration"]

# Disable Jog mode (constant velocity must be null on all positioners from group)
set ErrCode [catch "GroupJogModeDisable $socketID $Group"]

# Close TCP socket
set ErrCode [catch "TCP_CloseSocket $socketID"]
```


2.2.7 Tracking

Process to realize an analog tracking.

```
# Initialization
set TimeOut 60

set Group "XY"
set Positioner "XY.X"

set AnalogInput "GPIO2.ADC1"
set Offset 0
set Scale 1
set Velocity 20
set Acceleration 80

set TrackingType "Position"

set ErrCode 0

# Open TCP socket
set ErrCode [catch "OpenConnection $TimeOut socketID"]

# Initialize Group
set ErrCode [catch "GroupInitialize $socketID $Group"]

# Search home Group
set ErrCode [catch "GroupHomeSearch $socketID $Group"]

# Set Tracking parameters
set ErrCode [catch "PositionerAnalogTrackingPositionParametersSet $socketID
$Positioner $AnalogInput $Offset $Scale $Velocity $Acceleration"]

# Enable Tracking mode (group must be Ready)
set ErrCode [catch "GroupAnalogTrackingModeEnable $socketID $Group
$TrackingType"]

# Disable Tracking mode
set ErrCode [catch "GroupAnalogTrackingModeDisable $socketID $Group"]

# Close TCP socket
set ErrCode [catch "TCP_CloseSocket $socketID"]
```

2.2.8 Backlash

Process to realize backlash compensation.

CAUTION :

- “HomeSearchSequenceType” must be different from “CurrentPositionAsHome” in configuration file (stages.ini).
- “PositionerMappingFileName” must not be defined in configuration file (stages.ini)
- “Backlash” must be greater than zero in configuration file (stages.ini)

```
# Initialization
set Timeout 60

set Group "SINGLE_AXIS"
set Positioner "SINGLE_AXIS.MY_STAGE"

set BacklashValue 0.1
set Displacement 10
set ErrCode 0

# Open TCP socket
set ErrCode [catch "OpenConnection $Timeout socketID"]

# Enable Backlash
# Caution : Group must be "Not Initialized" and Backlash > 0 in "stages.ini"
set ErrCode [catch "PositionerBacklashEnable $socketID $Positioner"]

# Initialize Group
set ErrCode [catch "GroupInitialize $socketID $Group"]

# Search home Group
set ErrCode [catch "GroupHomeSearch $socketID $Group"]

# Modify Backlash value
# Caution : Backlash > 0 in "stages.ini"
set ErrCode [catch "PositionerBacklashSet $socketID $Positioner $BacklashValue"]

# Move in positive direction
set ErrCode [catch "GroupMoveRelative $socketID $Group $Displacement"]

# Move in negative direction
set ErrCode [catch "GroupMoveRelative $socketID $Group -$Displacement"]

# Disable Backlash (if you want to do trajectory, jogging or tracking)
# Caution : to enable backlash, you must call "GroupKill" or "KillAll" to come back
# in "Not Initialized" status
set ErrCode [catch "PositionerBacklashDisable $socketID $Group"]

# Close TCP socket
set ErrCode [catch "TCP_CloseSocket $socketID"]
```

2.2.9 Timer event

“StartScript.tcl” allows to configure a timer and use this timer as an event. The action, in relation to this timer event, will execute a tcl script named “MyScript.tcl”.

StartScript.tcl

```
# Initialization
set TCPTimeOut 0.5
set ErrCode 0

# XPS initialization value
set ISRPeriodSec 0.0001

set Positioner "SINGLE_AXIS.MY_STAGE"

set Timer "Timer1"
set TimerPeriodSec 2

set EvtParam 0
set Action "ExecuteTCLScript"
set TCLFile "MyScript.tcl"
set TCLTask "MyTask"
set TCLArgs "0"

# open TCP socket
set ErrCode [catch "OpenConnection $TCPTimeOut socketID"]
if {$ErrCode == 0} {

    # Calculate divisor (Periods are in seconds)
    set Divisor [expr {$TimerPeriodSec / $ISRPeriodSec}]

    # Configure timer
    set ErrCode [catch "TimerSet $socketID $TimerName $Divisor"]

    # Add timer event with an action that allows to execute "MyScript.tcl"
    set ErrCode [catch "EventAdd $socketID $Positioner $Timer $EvtParam $Action
    $TCLFile $TCLTask $TCLArgs"]

    # close TCP socket
    set ErrCode [catch "TCP_CloseSocket $socketID"]

} else {

    # Erreur de connection TCP/IP
    puts stdout "TCP/IP Connection failed ! $ErrCode"

}
```

MyScript.tcl

```
set TCPTimeOut 0.5
set ErrCode 0

set GlobalVarNumber 1
set END 10

set Positioner "SINGLE_AXIS.MY_STAGE"

set EventName "Timer1"
set EventPara 0

# open TCP socket
set ErrCode [catch "OpenConnection $TCPTimeOut socketID"]
if {$ErrCode == 0} {

    # Read global variable
    set ErrCode [catch "GlobalArrayGet $socketID $GlobalVarNumber Value"]

    if { $Value < $END } {

        # Increment global variable
        set NewValue [expr {$Value + 1}]

        # Set global variable
        set ErrCode [catch "GlobalArraySet $socketID $GlobalVarNumber $NewValue"]

    } else {

        # Delete timer event
        set ErrCode [catch "EventRemove $socketID $Positioner $EventName $EventPara"]

    }

    # close TCP socket
    set ErrCode [catch "TCP_CloseSocket $socketID"]

} else {

    # Erreur de connection TCP/IP
    puts stdout "TCP/IP Connection failed ! $ErrCode"

}
```

2.3 Positioner

2.3.1 Description

2.3.1.1 Group

Group object is used to build a set of positioners (one or several positioners).

Several group types are available :

- SingleAxis
- XY
- XYZ
- MultipleAxes

2.3.1.2 Positioner

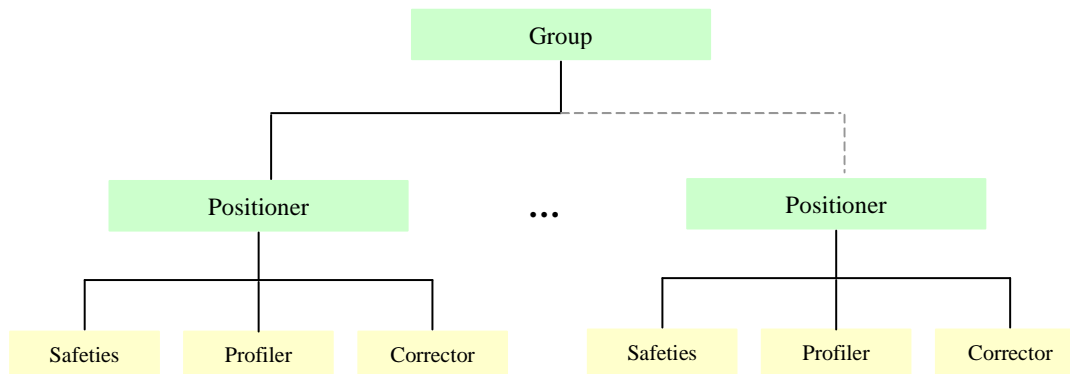
Basic core positioner object used to build user single or multi-actor objects.

In this object are defined all motion specific configuration parameters.

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

The positioner includes the SGamma profile.

2.3.2 Object structure



A **group** is built in relation to a group type (SingleAxis, XY, XYZ or MultipleAxes) and is composed one or several positioners. A group is defined by a **group name**.

Each **positioner** must belong to a group and is defined by the **full positioner name**. The full positioner name is composed of the group name and the positioner name, like as : **GroupName.PositionerName**

2.3.3 API description

2.3.3.1 PositionerAnalogTrackingPositionParametersGet

TCL Prototype	int PositionerAnalogTrackingPositionParametersGet (int SocketID, char FullPositionerName [250], char *GPIOName, double *Offset, double *Scale, double *velocity, double *acceleration)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	char * : GPIOName (Analog Input) double * : Offset (Volt) double * : Scale (Units) double * : velocity (Units / s) double * : acceleration (Units / s ²)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerAnalogTrackingPositionParametersGet (int SocketID, char FullPositionerName [250], char *GPIOName, double *Offset, double *Scale, double *velocity, double *acceleration)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	char * : GPIOName (Analog Input) double * : Offset (Volt) double * : Scale (Units) double * : velocity (Units / s) double * : acceleration (Units / s ²)
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner name. Parameters coherence test.
API Description	This API returns the current GPIO name, the current offset and the current scale used by analog tracking position mode.
API Errors	0 -7 -8 -9 -13 -14 -19

2.3.3.2 PositionerAnalogTrackingPositionParametersSet

TCL Prototype	int PositionerAnalogTrackingPositionParametersSet (int SocketID, char FullPositionerName [250], char GPIOName [250], double Offset, double Scale, double velocity, double acceleration)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName char [250] : GPIOName (Analog Input) double : Offset (Volt) double : Scale (Units) double : velocity (Units / s) double : acceleration (Units / s ²)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerAnalogTrackingPositionParametersSet (int SocketID, char FullPositionerName [250], char GPIOName [250], double Offset, double Scale, double velocity, double acceleration)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName char [250] : GPIOName (Analog Input) double : Offset (Volt) double : Scale (Units) double : velocity (Units / s) double : acceleration (Units / s ²)
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner name. Parameters coherence test
API Description	The analog tracking position mode must not be activated. This API allows to modify the GPIO name, the offset and the scale used by the analog tracking position mode. Position = InitialPosition + (AnalogValue - Offset) * Scale
API Errors	0 -7 -8 -9 -13 -14 -19 -48

2.3.3.3 PositionerAnalogTrackingVelocityParametersGet

TCL Prototype	int PositionerAnalogTrackingVelocityParametersGet (int SocketID, char FullPositionerName [250], char *GPIOName, double *Offset, double *Scale, double *DeadBandThreshold, int *Order, double *velocity, double *acceleration)
Input parameters	char [250] : FullPositionerName
Output parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char * : GPIOName (Analog Input) double * : Offset (Volt) double * : Scale (Units) double * : DeadBandThreshold (Volt) int * : Order (No unit) double * : velocity (Units / s) double * : acceleration (Units / s ²)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerAnalogTrackingVelocityParametersGet (int SocketID, char FullPositionerName [250], char *GPIOName, double *Offset, double *Scale, double *DeadBandThreshold, int *Order, double *velocity, double *acceleration)
Input parameters	char [250] : FullPositionerName
Output parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char * : GPIOName (Analog Input) double * : Offset (Volt) double * : Scale (Units) double * : DeadBandThreshold (Volt) int * : Order (No unit) double * : velocity (Units / s) double * : acceleration (Units / s ²)
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner name. Parameters coherence test.
API Description	This API returns the GPIO name, the offset, the scale, the deadband threshold and the order used by analog tracking velocity mode.
API Errors	0 -7 -8 -9 -13 -14 -15 -19

2.3.3.4 PositionerAnalogTrackingVelocityParametersSet

TCL Prototype	int PositionerAnalogTrackingVelocityParametersSet (int SocketID, char FullPositionerName [250], char GPIOName [250], double Offset, double Scale, double DeadBandThreshold, int Order, double velocity, double acceleration)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : FullPositionerName char [250] : GPIOName (Analog Input) double : Offset (Volt) double : Scale (Units) double : DeadBandThreshold (Volt) int : Order (No unit) double : velocity (Units / s) double : acceleration (Units / s ²)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerAnalogTrackingVelocityParametersSet (int SocketID, char FullPositionerName [250], char GPIOName [250], double Offset, double Scale, double DeadBandThreshold, int Order, double velocity, double acceleration)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : FullPositionerName char [250] : GPIOName (Analog Input) double : Offset (Volt) double : Scale (Units) double : DeadBandThreshold (Volt) int : Order (No unit) double : velocity (Units / s) double : acceleration (Units / s ²)
Output parameters	None
Return	API error

API Input tests	<p>Verify the number of parameters.</p> <p>Verify the positioner name.</p> <p>Parameters coherence test</p>
API Description	<p>The analog tracking position mode must not be activated.</p> <p>This API allows to modify the GPIO name, the offset, the scale, the deadband threshold and the order used by the analog tracking velocity mode.</p> <pre> E = 10 / AnalogGain InputValue = AnalogInput - Offset if (InputValue >= 0) then InputValue = InputValue - DeadBandThreshold if (InputValue < 0) then InputValue = 0 else InputValue = InputValue + DeadBandThreshold if (InputValue > 0) then InputValue = 0 OutputValue = (InputValue / E)^{Order} if (OutputValue > 1) then OutputValue = 1 Velocity = Sign(InputValue) * OutputValue * Scale </pre>
API Errors	0 -7 -8 -9 -13 -14 -15 -19 -48

2.3.3.5 PositionerBacklashEnable

TCL Prototype	int PositionerBacklashEnable(int SocketID, char FullPositionerName[250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerBacklashEnable(int SocketID, char FullPositionerName[250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner name.
API Description	This API enables the backlash mode. NOTES: This API can be used only if a backlash is enabled (Backlash > 0) in the stages.ini. The group must be into the NOT INITIALIZED status to use this API. <u>CAUTION:</u> In STAGES.INI, if the “HomeSearchSequenceType” is defined as “CurrentPositionAsHome” and “PositionerMappingFileName” is defined then the backlash can’t be enabled and the controller will not initialized correctly (Fatal error).
API Errors	0 -7 -8 -9 -18 -19 -22
Stage.ini	Backlash = 0 (0 → Disable backlash, >0 → Enable backlash)

2.3.3.6 PositionerBacklashDisable

TCL Prototype	int PositionerBacklashDisable(int SocketID, char FullPositionerName[250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerBacklashDisable(int SocketID, char FullPositionerName[250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner name.
API Description	This API disables the backlash mode. NOTES : To execute a trajectory, to use the jog mode or to use the analog tracking, the backlash must be deactivated. So, if a backlash is enabled, this API can be used before.
API Errors	0 -7 -8 -9 -18 -19
Stage.ini	Backlash = 0 (0 → Disable backlash, >0 → Enable backlash)

2.3.3.7 PositionerBacklashSet

TCL Prototype	int PositionerBacklashSet (int SocketID, char FullPositionerName[250], double BacklashValue)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : FullPositionerName double : BacklashValue (units)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerBacklashSet (int SocketID, char FullPositionerName[250], double BacklashValue)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : FullPositionerName double : BacklashValue (units)
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner name.
API Description	This API change the backlash value. NOTE : This API can be used only if a backlash is enabled (Backlash > 0) in the stages.ini.
API Errors	0 -7 -8 -9 -14 -18 -19 -22
API Stage.ini	Backlash = 0 (0 → Disable backlash, >0 → Enable backlash)

2.3.3.8 PositionerBacklashGet

TCL Prototype	int PositionerBacklashGet(int SocketID, char FullPositionerName[250], double* BacklashValue, char* status)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	double * : BacklashValue (units) char * : BacklashStatus (“Enable” or “Disable”)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerBacklashGet(int SocketID, char FullPositionerName[250], double* BacklashValue, char* status)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	double * : BacklashValue (units) char * : BacklashStatus (“Enable” or “Disable”)
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner name.
API Description	This API returns the backlash value and the backlash status (“Enable” or “Disable”)
API Errors	0 -7 -8 -9 -13 -14 -18 -19

2.3.3.9 PositionerCorrectorPIDDualFFVoltageGet

TCL 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	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : FullPositionerName
Output parameters	bool* : ClosedLoopStatus (true = closed, false = opened) double* : KP double* : KI double* : KD double* : KS double* : IntegrationTime (seconds) double* : DerivativeFilterCutOffFrequency double* : GKP double* : GKI double* : GKD double* : KForm double* : FeedForwardGainVelocity (units) double* : FeedForwardGainAcceleration (units) double* : Friction
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL 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	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	bool* : ClosedLoopStatus (true = closed, false = opened) double* : KP double* : KI double* : KD double* : KS double* : IntegrationTime (seconds) double* : DerivativeFilterCutOffFrequency double* : GKP double* : GKI double* : GKD double* : KForm double* : FeedForwardGainVelocity (units) double* : FeedForwardGainAcceleration (units) double* : Friction
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner group type. Verify the type of all output parameters. Parameters coherence test.
API Description	CorrectorType must be “PIDDualFFVoltage”. Returns the corrector filter parameters : UserClosedLoopStatus, UserKP, UserKI, UserKD, UserKS, UserIntegrationTime, UserDerivativeFilterDepth, UserGKP, UserGKI, UserGKD, UserKform, UserFeedForwardGainVelocity, FeedForwardGainAcceleration and Friction.
API Errors	0 -7 -8 -9 -12 -14 -19

2.3.3.10 PositionerCorrectorPIDDualFFVoltageSet

TCL Prototype	int PositionerCorrectorPIDDualFFVoltageSet (int SocketID, char FullPositionerName[250], bool NewClosedLoopStatus, double NewKP, double NewKI, double NewKD, double NewKS, double NewIntegrationTime, double NewDerivativeFilterCutOffFrequency, double NewGKP, double NewGKI, double NewGKD, double NewKForm, double NewFeedForwardGainVelocity, double NewFeedForwardGainAcceleration, double NewFriction)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char[250] : FullPositionerName bool : NewClosedLoopStatus (true = closed, false = opened) double : NewKP double : NewKI double : NewKD double : NewKS double : NewIntegrationTime (seconds) double : NewDerivativeFilterCutOffFrequency double : NewGKP double : NewGKI double : NewGKD double : NewKForm double : NewFeedForwardGainVelocity (units) double : NewFeedForwardGainAcceleration (units) double : NewFriction
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerCorrectorPIDDualFFVoltageSet (int SocketID, char FullPositionerName[250], bool NewClosedLoopStatus, double NewKP, double NewKI, double NewKD, double NewKS, double NewIntegrationTime, double NewDerivativeFilterCutOffFrequency, double NewGKP, double NewGKI, double NewGKD, double NewKForm, double NewFeedForwardGainVelocity, double NewFeedForwardGainAcceleration, double NewFriction)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char[250] : FullPositionerName bool : NewClosedLoopStatus (true = closed, false = opened) double : NewKP double : NewKI double : NewKD double : NewKS double : NewIntegrationTime (seconds) double : NewDerivativeFilterCutOffFrequency double : NewGKP double : NewGKI double : NewGKD double : NewKForm double : NewFeedForwardGainVelocity (units) double : NewFeedForwardGainAcceleration (units) double : NewFriction
Output parameters	None
Return	API error

API Input tests	<p>Verify the number of parameters. Verify the positioner group type. Verify the type for all input parameters and the parameters coherence test. Verify the limit for all input parameters</p>
API Description	<p>CorrectorType must be "PIDDualFFVoltage". Updates corrector filters' parameters. Saves the new corrector filters parameters :</p> <pre> UserClosedLoopStatus = NewClosedLoopStatus UserKP = NewKP UserKI = NewKI UserKD = NewKD UserKS = NewKS UserIntegrationTime = NewIntegrationTime UserDerivativeFilterCutOffFrequency = NewDerivativeFilterCutOffFrequency UserGKP = NewGKP UserGKI = NewGKI UserGKD = NewGKD UserKForm = NewKForm UserKFeedforwardGainVelocity = NewFeedForwardGainVelocity UserKFeedForwardGainAcceleration = NewFeedForwardGainAcceleration UserFriction = NewFriction </pre>
API Errors	0 -7 -8 -9 -12 -14 -17 -19
Stages.ini	<p>CorrectorType = PIDDualFFVoltage KP KI KD KS IntegrationTime (seconds) DerivativeFilterCutOffFrequency GKP GKD GKI KForm KFeedForwardVelocity KFeedForwardAcceleration KFeedForwardVelocityOpenLoop Friction ClosedLoopStatus (Closed or Opened) FatalFollowingError (units)</p>

2.3.3.11 PositionerCorrectorPIDFFAccelerationGet

TCL 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	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	bool* : ClosedLoopStatus (true = closed, false = opened) double* : KP double* : KI double* : KD double* : KS double* : IntegrationTime (seconds) double* : DerivativeFilterCutOffFrequency double* : GKP double* : GKI double* : GKD double* : KForm double* : FeedForwardGainAcceleration (units)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL 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	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	bool* : ClosedLoopStatus (true = closed, false = opened) double* : KP double* : KI double* : KD double* : KS double* : IntegrationTime (seconds) double* : DerivativeFilterCutOffFrequency double* : GKP double* : GKI double* : GKD double* : KForm double* : FeedForwardGainAcceleration (units)
Return	API error

API Input tests	<p>Verify the number of parameters.</p> <p>Verify the positioner group type.</p> <p>Verify the type of all output parameters.</p> <p>Parameters coherence test.</p>
API Description	<p>CorrectorType must be "PIDFFAcceleration".</p> <p>Returns the corrector filter parameters :</p> <p>UserClosedLoopStatus, UserKP, UserKI, UserKD, UserKS, UserIntegrationTime, UserDerivativeFilterDepth, UserGKP, UserGKI, UserGKD, UserKform and UserFeedForwardGainAcceleration.</p>
API Errors	0 -7 -8 -9 -12 -14 -19

2.3.3.12 PositionerCorrectorPIDFFAccelerationSet

TCL Prototype	int PositionerCorrectorPIDFFAccelerationSet (int SocketID, char FullPositionerName[250], bool NewClosedLoopStatus, double NewKP, double NewKI, double NewKD, double NewKS, double NewIntegrationTime, double NewDerivativeFilterCutOffFrequency, double NewGKP, double NewGKI, double NewGKD, double NewKForm, double NewFeedForwardGainAcceleration)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char[250] : FullPositionerName bool : NewClosedLoopStatus (true = closed, false = opened) double : NewKP double : NewKI double : NewKD double : NewKS double : NewIntegrationTime (seconds) double : NewDerivativeFilterCutOffFrequency double : NewGKP double : NewGKI double : NewGKD double : NewKForm double : NewFeedForwardGainAcceleration (units)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerCorrectorPIDFFAccelerationSet (int SocketID, char FullPositionerName[250], bool NewClosedLoopStatus, double NewKP, double NewKI, double NewKD, double NewKS, double NewIntegrationTime, double NewDerivativeFilterCutOffFrequency, double NewGKP, double NewGKI, double NewGKD, double NewKForm, double NewFeedForwardGainAcceleration)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char[250] : FullPositionerName bool : NewClosedLoopStatus (true = closed, false = opened) double : NewKP double : NewKI double : NewKD double : NewKS double : NewIntegrationTime (seconds) double : NewDerivativeFilterCutOffFrequency double : NewGKP double : NewGKI double : NewGKD double : NewKForm double : NewFeedForwardGainAcceleration (units)
Output parameters	None
Return	API error

API Input tests	<p>Verify the number of parameters. Verify the positioner group type. Verify the type for all input parameters. Parameters coherence test. Verify the limit for all input parameters</p>
API Description	<p>CorrectorType must be "PIDFFAcceleration". Updates corrector filters' parameters. Saves the new corrector filters parameters :</p> <pre> UserClosedLoopStatus = NewClosedLoopStatus UserKP = NewKP UserKI = NewKI UserKD = NewKD UserKS = NewKS UserIntegrationTime = NewIntegrationTime UserDerivativeFilterCutOffFrequency = NewDerivativeFilterCutOffFrequency UserGKP = NewGKP UserGKI = NewGKI UserGKD = NewGKD UserKForm = NewKForm UserKFeedforwardGainAcceleration = NewFeedForwardGainAcceleration </pre>
API Errors	0 -7 -8 -9 -12 -14 -17 -19
Stages.ini	<p>CorrectorType = PIDFFAcceleration KP KI KD KS IntegrationTime (seconds) DerivativeFilterDepth GKP GKD GKI KForm KFeedForwardAcceleration ClosedLoopStatus (Closed or Opened) FatalFollowingError (units) ThresholdPosition (units)</p>

2.3.3.13 PositionerCorrectorPIDFFVelocityGet

TCL 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	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	bool* : ClosedLoopStatus (true = closed, false = opened) double* : KP double* : KI double* : KD double* : KS double* : IntegrationTime (seconds) double* : DerivativeFilterCutOffFrequency double* : GKP double* : GKI double* : GKD double* : KForm double* : FeedForwardGainVelocity (units)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL 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	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	bool* : ClosedLoopStatus (true = closed, false = opened) double* : KP double* : KI double* : KD double* : KS double* : IntegrationTime (seconds) double* : DerivativeFilterCutOffFrequency double* : GKP double* : GKI double* : GKD double* : KForm double* : FeedForwardGainVelocity (units)
Return	API error

API Input tests	<p>Verify the number of parameters.</p> <p>Verify the positioner group type.</p> <p>Verify the type of all output parameters.</p> <p>Parameters coherence test.</p>
API Description	<p>CorrectorType must be "PIDFFVelocity".</p> <p>Returns the corrector filter parameters :</p> <p>UserClosedLoopStatus, UserKP, UserKI, UserKD, UserKS, UserIntegrationTime, UserDerivativeFilterDepth, UserGKP, UserGKI, UserGKD, UserKForm and UserFeedForwardGainVelocity.</p>
API Errors	0 -7 -8 -9 -12 -14 -19

2.3.3.14 PositionerCorrectorPIDFFVelocitySet

TCL Prototype	int PositionerCorrectorPIDFFVelocitySet (int SocketID, char FullPositionerName[250], bool NewClosedLoopStatus, double NewKP, double NewKI, double NewKD, double NewKS, double NewIntegrationTime, double NewDerivativeFilterCutOffFrequency, double NewGKP, double NewGKI, double NewGKD, double NewKForm, double NewFeedForwardGainVelocity)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char[250] : FullPositionerName bool : NewClosedLoopStatus (true = closed, false = opened) double : NewKP double : NewKI double : NewKD double : NewKS double : NewIntegrationTime (seconds) double : NewDerivativeFilterCutOffFrequency double : NewGKP double : NewGKI double : NewGKD double : NewKForm double : NewFeedForwardGainVelocity (units)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerCorrectorPIDFFVelocitySet (int SocketID, char FullPositionerName[250], bool NewClosedLoopStatus, double NewKP, double NewKI, double NewKD, double NewKS, double NewIntegrationTime, double NewDerivativeFilterCutOffFrequency, double NewGKP, double NewGKI, double NewGKD, double NewKForm, double NewFeedForwardGainVelocity)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char[250] : FullPositionerName bool : NewClosedLoopStatus (true = closed, false = opened) double : NewKP double : NewKI double : NewKD double : NewKS double : NewIntegrationTime (seconds) double : NewDerivativeFilterCutOffFrequency double : NewGKP double : NewGKI double : NewGKD double : NewKForm double : NewFeedForwardGainVelocity (units)
Output parameters	None
Return	API error

API Input tests	<p>Verify the number of parameters. Verify the positioner group type. Verify the type for all input parameters. Parameters coherence test. Verify the limit for all input parameters</p>
API Description	<p>CorrectorType must be "PIDFFVelocity". Updates corrector filters' parameters. Saves the new corrector filters parameters :</p> <pre> UserClosedLoopStatus = NewClosedLoopStatus UserKP = NewKP UserKI = NewKI UserKD = NewKD UserKS = NewKS UserIntegrationTime = NewIntegrationTime UserDerivativeFilterCutOffFrequency = NewDerivativeFilterCutOffFrequency UserGKP = NewGKP UserGKI = NewGKI UserGKD = NewGKD UserKForm = NewKForm UserKFeedforwardGainVelocity = NewFeedForwardGainVelocity </pre>
API Errors	0 -7 -8 -9 -12 -14 -17 -19
Stages.ini	<p>CorrectorType = PIDFFVelocity KP KI KD KS IntegrationTime (seconds) DerivativeFilterCutOffFrequency GKP GKD GKI KForm KFeedForwardVelocity ClosedLoopStatus (Closed or Opened) FatalFollowingError (units)</p>

2.3.3.15 PositionerCorrectorPIPositionGet

TCL Prototype	int PositionerCorrectorPIPositionGet (int SocketID, char FullPositionerName[250], bool *ClosedLoopStatus, double *KP, double *KI, double *IntegrationTime)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	bool* : ClosedLoopStatus (true = closed, false = opened) double* : KP double* : KI double* : IntegrationTime (seconds)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerCorrectorPIPositionGet (int SocketID, char FullPositionerName[250], bool *ClosedLoopStatus, double *KP, double *KI, double *IntegrationTime)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	bool* : ClosedLoopStatus (true = closed, false = opened) double* : KP double* : KI double* : IntegrationTime (seconds)
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner group type. Verify the type of all output parameters. Parameters coherence test.
API Description	CorrectorType must be “PIPosition”. Returns the corrector filter parameters : UserClosedLoopStatus, UserKP, UserKI and UserIntegrationTime.
API Errors	0 -7 -8 -9 -12 -14 -19

2.3.3.16 PositionerCorrectorPIPositionSet

TCL Prototype	int PositionerCorrectorPIPositionSet (int SocketID, char FullPositionerName[250], bool ClosedLoopStatus, double KP, double KI, double IntegrationTime)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char[250] : FullPositionerName bool : NewClosedLoopStatus (true = closed, false = opened) double : NewKP double : NewKI double : NewIntegrationTime (seconds)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerCorrectorPIPositionSet (int SocketID, char FullPositionerName[250], bool ClosedLoopStatus, double KP, double KI, double IntegrationTime)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char[250] : FullPositionerName bool : NewClosedLoopStatus (true = closed, false = opened) double : NewKP double : NewKI double : NewIntegrationTime (seconds)
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner group type. Verify the type for all input parameters. Parameters coherence test. Verify the limit for all input parameters
API Description	CorrectorType must be "PIPosition". Updates corrector filters' parameters. Saves the new corrector filters parameters : UserClosedLoopStatus = NewClosedLoopStatus UserKP = NewKP UserKI = NewKI UserIntegrationTime = NewIntegrationTime
API Errors	0 -7 -8 -9 -12 -14 -17 -19
Stages.ini	CorrectorType = PIPosition KP KI IntegrationTime (seconds)

2.3.3.17 PositionerCorrectorTypeGet

TCL Prototype	int PositionerCorrectorTypeGet (int SocketID, char FullPositionerName[250], char CorrectorType [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char[250] : FullPositionerName
Output parameters	char[250] : CorrectorType
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerCorrectorTypeGet (int SocketID, char FullPositionerName[250], char CorrectorType [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char[250] : FullPositionerName
Output parameters	char[250] : CorrectorType
Return	API error

API Input tests	Verify the number of parameters. Verify the type for all input parameters.
API Description	Return the corrector type used by the positioner : <ul style="list-style-type: none"> • PositionerCorrectorPIDFFAcceleration • PositionerCorrectorPIDFFVelocity • PositionerCorrectorPIDDualFFVltage • PositionerCorrectorPIPosition • NoCorrector
API Errors	0 -7 -8 -9 -12 -14 -17 -19

2.3.3.18 PositionerCorrectorNotchFiltersGet

TCL Prototype	int PositionerCorrectorNotchFiltersGet (int SocketID, char FullPositionerName[250], double* NotchFrequency1, double* NotchBandwidth1, double* NotchGain1, double* NotchFrequency2, double* NotchBandwidth2, double* NotchGain2)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	double* : NotchFrequency1 (Herz) double* : NotchBandwidth1 (Herz) double* : NotchGain1 double* : NotchFrequency2 (Herz) double* : NotchBandwidth2 (Herz) double* : NotchGain2
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerCorrectorNotchFiltersGet (int SocketID, char FullPositionerName[250], double* NotchFrequency1, double* NotchBandwidth1, double* NotchGain1, double* NotchFrequency2, double* NotchBandwidth2, double* NotchGain2)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	double* : NotchFrequency1 (Herz) double* : NotchBandwidth1 (Herz) double* : NotchGain1 double* : NotchFrequency2 (Herz) double* : NotchBandwidth2 (Herz) double* : NotchGain2
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner group type. Verify the type of all output parameters. Parameters coherence test.
API Description	Returns parameters of the two notch filters : UserNotchFrequency1, UserNotchBandwidth1, UserNotchGain1, UserNotchFrequency2, UserNotchBandwidth2 and UserNotchGain2.
API Errors	0 -7 -8 -9 -14 -19

2.3.3.19 PositionerCorrectorNotchFiltersSet

TCL Prototype	int PositionerCorrectorNotchFiltersSet (int SocketID, char FullPositionerName[250], double NewNotchFrequency1, double NewNotchBandwidth1, double NewNotchGain1, double NewNotchFrequency2, double NewNotchBandwidth2, double NewNotchGain2)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : FullPositionerName double : NewNotchFrequency1 (Herz) double : NewNotchBandwidth1 (Herz) double : NewNotchGain1 double : NewNotchFrequency2 (Herz) double : NewNotchBandwidth2 (Herz) double : NewNotchGain2
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerCorrectorNotchFiltersSet (int SocketID, char FullPositionerName[250], double NewNotchFrequency1, double NewNotchBandwidth1, double NewNotchGain1, double NewNotchFrequency2, double NewNotchBandwidth2, double NewNotchGain2)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : FullPositionerName double : NewNotchFrequency1 (Herz) double : NewNotchBandwidth1 (Herz) double : NewNotchGain1 double : NewNotchFrequency2 (Herz) double : NewNotchBandwidth2 (Herz) double : NewNotchGain2
Output parameters	None
Return	API error

API Input tests	<p>Verify the number of parameters. Verify the positioner group type. Verify the type for all input parameters. Parameters coherence test. Verify the limit for all input parameters : NewNotchFrequency1 ≥ 0 NewNotchBandwidth1 ≥ 0 NewNotchGain1 ≥ 0 NewNotchFrequency2 ≥ 0 NewNotchBandwidth2 ≥ 0 NewNotchGain2 ≥ 0</p>
API Description	<p>Updates parameters of the two notch filters :</p> <pre>UserNotchFrequency1 = NewNotchFrequency1 UserNotchBandwidth1 = NewNotchBandwidth1 UserNotchGain1 = NewNotchGain1 UserNotchFrequency2 = NewNotchFrequency2 UserNotchBandwidth2 = NewNotchBandwidth2 UserNotchGain2 = NewNotchGain2</pre> <p>If NewNotchFrequency is NULL or NewNotchGain is NULL, then the notch filter is not activated.</p>
API Errors	0 -7 -8 -9 -14 -17 -19
Stages.ini	<p>NotchFrequency1 (Hertz) NotchBandwidth1 (Hertz) NotchGain1 NotchFrequency2 (Hertz) NotchBandwidth2 (Hertz) NotchGain2</p>

2.3.3.20 PositionerMotionDoneGet

TCL Prototype	int PositionerMotionDoneGet (int SocketID, char FullPositionerName[250], double* PositionWindow, double* VelocityWindow, double* CheckingTime, double* MeanPeriod, double* Timeout)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	double* : PositionWindow (units) double* : VelocityWindow (units / seconds) double* : CheckingTime (seconds) double* : MeanPeriod (seconds) double* : Timeout (seconds)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerMotionDoneGet (int SocketID, char FullPositionerName[250], double* PositionWindow, double* VelocityWindow, double* CheckingTime, double* MeanPeriod, double* Timeout)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	double* : PositionWindow (units) double* : VelocityWindow (units / seconds) double* : CheckingTime (seconds) double* : MeanPeriod (seconds) double* : Timeout (seconds)
Return	API error

API Input tests	Verify the number of parameters. Parameters coherence test. Verify the positioner group type. Verify the type of all output parameters.
API Description	Returns motion done values in case “VelocityAndPositionWindow” : UserPositionWindow, UserVelocityWindow, UserCheckingTime, UserMeanPeriod and UserTimeout.
API Errors	0 -7 -8 -9 -14 -19

2.3.3.21 PositionerMotionDoneSet

TCL Prototype	int PositionerMotionDoneSet (int SocketID, char FullPositionerName[250], double NewPositionWindow, double NewVelocityWindow, double NewCheckingTime, double NewMeanPeriod, double NewTimeout)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName double : PositionWindow (units) double : VelocityWindow (units / seconds) double : CheckingTime (seconds) double : MeanPeriod (seconds) double : Timeout (seconds)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerMotionDoneSet (int SocketID, char FullPositionerName[250], double NewPositionWindow, double NewVelocityWindow, double NewCheckingTime, double NewMeanPeriod, double NewTimeout)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName double : PositionWindow (units) double : VelocityWindow (units / seconds) double : CheckingTime (seconds) double : MeanPeriod (seconds) double : Timeout (seconds)
Output parameters	None
Return	API error

API Input tests	<p>Verify the number of parameters.</p> <p>Verify the positioner group type.</p> <p>Verify the type for all input parameters and the parameters coherence test.</p>
API Description	<p>Updates motion done parameters (only if MotionDoneMode is “VelocityAndPositionWindow”) :</p> <pre>UserPositionWindow = NewPositionWindow UserVelocityWindow = NewVelocityWindow UserCheckingTime = NewCheckingTime UserMeanPeriod = NewMeanPeriod UserTimeout = NewTimeout</pre> <p>If the MotionDoneMode parameter is “Theoretical”, these parameters are not took into consideration.</p>
API Errors	0 -7 -8 -9 -14 -17 -19
Stages.ini	<p>MotionDoneMode = <i>VelocityAndPositionWindow</i> or <i>Theoretical</i></p> <p>MotionDonePositionThreshold (units)</p> <p>MotionDoneVelocityThreshold (units / second)</p> <p>MotionDoneCheckingTime (seconds)</p> <p>MotionDoneMeanPeriod (seconds)</p> <p>MotionDoneTimeout (seconds)</p>

2.3.3.22 PositionerPreviousMotionTimesGet

TCL Prototype	int PositionerPreviousMotionTimesGet (int SocketID, char FullPositionerName[250], double* SettingTime, double* SettlingTime)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	double* : SettingTime (seconds) double* : SettlingTime (seconds)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerPreviousMotionTimesGet (int SocketID, char FullPositionerName[250], double* SettingTime, double* SettlingTime)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	double* : SettingTime (seconds) double* : SettlingTime (seconds)
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner group type. Verify the type of all output parameters. Parameters coherence test.
API Description	Returns the setting and settling times from the motion done.
API Errors	0 -7 -8 -9 -14 -19

2.3.3.23 PositionerSGammaParametersSet

TCL Prototype	int PositionerSGammaParametersSet (int SocketID, char FullPositionerName[250], double NewVelocity, double NewAcceleration, double NewMinimumJerkTime, double NewMaximumJerkTime)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : FullPositionerName double : NewVelocity (units / seconds) double : NewAcceleration (units / seconds ²) double : NewMinimumJerkTime (seconds) double : NewMaximumJerkTime (seconds)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerSGammaParametersSet (int SocketID, char FullPositionerName[250], double NewVelocity, double NewAcceleration, double NewMinimumJerkTime, double NewMaximumJerkTime)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : FullPositionerName double : NewVelocity (units / seconds) double : NewAcceleration (units / seconds ²) double : NewMinimumJerkTime (seconds) double : NewMaximumJerkTime (seconds)
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner group type. Verify the type for all input parameters. Parameters coherence test. Verify the limit for all input parameters : <div style="display: flex; justify-content: space-between;"> <div>NewVelocity > 0</div> <div>NewVelocity <= MaximumVelocity</div> </div> <div style="display: flex; justify-content: space-between;"> <div>NewAcceleration > 0</div> <div>NewAcceleration <= MaximumAcceleration</div> </div> <div style="display: flex; justify-content: space-between;"> <div>NewMinimumJerkTime >= 0</div> <div>NewMinimumJerkTime <= NewMaximumJerkTime</div> </div>
API Description	Updates positioner dynamic parameters for a future displacement : UserVelocity = NewVelocity UserAcceleration = NewAcceleration UserMinimumJerkTime = NewMinimumJerkTime UserMaximumJerkTime = NewMaximumJerkTime
API Errors	0 -7 -8 -9 -14 -17 -19
Stages.ini	MaximumVelocity (units/second) MaximumAcceleration (units/second ²) MinimumJerkTime (seconds) MaximumJerkTime (seconds)

2.3.3.24 PositionerSGammaParametersGet

TCL Prototype	int PositionerSGammaParametersGet (int SocketID, char FullPositionerName[250], double* Velocity, double* Acceleration, double* MinimumJerkTime, double* MaximumJerkTime)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	double* Velocity (units / seconds) double* Acceleration (units / seconds ²) double* MinimumJerkTime (seconds) double* MaximumJerkTime (seconds)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerSGammaParametersGet (int SocketID, char FullPositionerName[250], double* Velocity, double* Acceleration, double* MinimumJerkTime, double* MaximumJerkTime)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	double* Velocity (units / seconds) double* Acceleration (units / seconds ²) double* MinimumJerkTime (seconds) double* MaximumJerkTime (seconds)
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner group type. Verify the type of all output parameters. Parameters coherence test.
API Description	Returns the dynamic parameters of the positioner for a future motion : UserVelocity, UserAcceleration, UserMinimumJerkTime and UserMaximumJerkTime.
API Errors	0 -7 -8 -9 -14 -19

2.3.3.25 PositionerSGammaExactVelocityAjustedDisplacementGet

TCL Prototype	int PositionerSGammaExactVelocityAjustedDisplacementGet (int SocketID, char FullPositionerName[250], double DesiredDisplacement, double * AdjustedDisplacement)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName double : DesiredDisplacement
Output parameters	double * : AdjustedDisplacement
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerSGammaExactVelocityAjustedDisplacementGet (int SocketID, char FullPositionerName[250], double DesiredDisplacement, double * AdjustedDisplacement)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName double : DesiredDisplacement
Output parameters	double * : AdjustedDisplacement
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner group type. Parameters coherence test.
API Description	Return adjusted displacement to get exact velocity in the SGamma profile case
API Errors	0 -7 -8 -9 -14 -19

2.3.3.26 PositionerErrorGet

TCL Prototype	int PositionerErrorGet (int SocketID, char FullPositionerName [250], int * PositionerError)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	int * : PositionerError (see § Positioner errors list)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerErrorGet (int SocketID, char FullPositionerName [250], int * PositionerError)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	int * : PositionerError (see § Positioner errors list)
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner name. Parameters coherence test.
API Description	Returns the positioner error code and clears the positioner error. To get the error code description, call the “PositionerErrorStringGet” API. The positioner error is composed of the corrector error, the profile generator error and the servitudes error.
API Errors	0 -7 -8 -9 -15 -18 -19

2.3.3.27 PositionerErrorStringGet

TCL Prototype	int PositionerErrorStringGet (int SocketID, int PositionerErrorCode, char * PositionerErrorString)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) int : PositionerErrorCode
Output parameters	char * : PositionerErrorString
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerErrorStringGet (int SocketID, int PositionerErrorCode, char * PositionerErrorString)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) int : PositionerErrorCode
Output parameters	char * : PositionerErrorString
Return	API error

API Input tests	Verify the number of parameters. Verify the first parameter type and the second parameter type. Parameters coherence test.
API Description	Returns the positioner error description in relation to a positioner error code (see § Positioner errors list).
API Errors	0 -7 -9 -13

2.3.3.28 PositionerHardwareStatusGet

TCL Prototype	int PositionerHardwareStatusGet (int SocketID, char FullPositionerName [250], int * PositionerHardwareStatus)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	int * : PositionerHardwareStatus (see § Positioner hardware status list)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerHardwareStatusGet (int SocketID, char FullPositionerName [250], int * PositionerHardwareStatus)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	int * : PositionerHardwareStatus (see § Positioner hardware status list)
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner name. Parameters coherence test.
API Description	Returns the positioner hardware status Positioner hardware status is composed of corrector and servitudes hardware status: PositionerHardwareStatus = <i>CorrectorHardwareStatus</i> or <i>ServitudesHardwareStatus</i> or <i>INTGlobalServitudesHardwareStatus</i>
API Errors	0 -7 -8 -9 -15 -18 -19

2.3.3.29 PositionerHardwareStatusStringGet

TCL Prototype	int PositionerHardwareStatusStringGet (int SocketID, int PositionerHardwareStatusCode, char * PositionerHardwareStatusString)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) int : PositionerHardwareStatusCode
Output parameters	char * : PositionerHardwareStatusString
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerHardwareStatusStringGet (int SocketID, int PositionerHardwareStatusCode, char * PositionerHardwareStatusString)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) int : PositionerHardwareStatusCode
Output parameters	char * : PositionerHardwareStatusString
Return	API error

API Input tests	Verify the number of parameters. Verify the first parameter type and the second parameter type. Parameters coherence test.
API Description	Returns the positioner hardware status description in relation to a positioner hardware status (see § Positioner hardware status list).
API Errors	0 -7 -9 -13

2.3.3.30 PositionerUserTravelLimitsGet

TCL Prototype	int PositionerUserTravelLimitsGet (int SocketID, char FullPositionerName[250], double* UserMinimumTarget, double* UserMaximumTarget)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	double* : UserMinimumTarget (units) double* : UserMaximumTarget (units)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerUserTravelLimitsGet (int SocketID, char FullPositionerName[250], double* UserMinimumTarget, double* UserMaximumTarget)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	double* : UserMinimumTarget (units) double* : UserMaximumTarget (units)
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner group type. Verify the type of all output parameters. Parameters coherence test.
API Description	Read user travel limits : UserMinimumTargetPosition and UserMinimumTargetPosition.
API Errors	0 -7 -8 -9 -14 -18

2.3.3.31 PositionerUserTravelLimitsSet

TCL Prototype	int PositionerUserTravelLimitsSet (int SocketID, char FullPositionerName[250], double NewUserMinimumTargetPosition, double NewUserMaximumTargetPosition)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : FullPositionerName double : NewUserMinimumTargetPosition (units) double : NewUserMaximumTargetPosition (units)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerUserTravelLimitsSet (int SocketID, char FullPositionerName[250], double NewUserMinimumTargetPosition, double NewUserMaximumTargetPosition)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : FullPositionerName double : NewUserMinimumTargetPosition (units) double : NewUserMaximumTargetPosition (units)
Output parameters	None
Return	API error

API Input tests	<p>Verify the number of parameters. Verify the positioner group type. Verify the type for all input parameters. Parameters coherence test. Verify the limits for all input parameters :</p> <pre> NewUserMinimumTargetPosition >= MinimumTargetPosition NewUserMinimumTargetPosition <= MaximumTargetPosition NewUserMaximumTargetPosition >= MinimumTargetPosition NewUserMaximumTargetPosition <= MaximumTargetPosition NewUserMinimumTargetPosition < NewUserMaximumTargetPosition NewUserMinimumTargetPosition <= ProfilerPosition NewUserMaximumTargetPosition >= ProfilerPosition </pre>
API Description	<p>Updates user travel limits' parameters. Saves travel limits parameters :</p> <pre> UserMinimumTargetPosition = NewUserMinimumTargetPosition; UserMaximumTargetPosition = NewUserMaximumTargetPosition; </pre>
API Errors	0 -7 -8 -9 -14 -17 -18
Stages.ini	<p>MinimumTargetPosition (units) MaximumTargetPosition (units)</p>

2.3.3.32 PositionerPositionCompareSet

TCL Prototype	int PositionerPositionCompareSet (int SocketID, char FullPositionerName[250], double MinimumPosition, double MaximumPosition, double PositionStep)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName double : MinimumPosition (units) double : MaximumPosition (units) double : PositionStep (units)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerPositionCompareSet (int SocketID, char FullPositionerName[250], double MinimumPosition, double MaximumPosition, double PositionStep)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName double : MinimumPosition (units) double : MaximumPosition (units) double : PositionStep (units)
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner name and the parameters coherence test.
API Description	This API set the position compare parameters. These parameters are used as soon as the position compare mode is enabled. The PCO connector is used.
API Errors	0 -7 -8 -9 -14 -19 -22 -24

2.3.3.33 PositionerPositionCompareGet

TCL Prototype	int PositionerPositionCompareGet (int SocketID, char FullPositionerName[250], double* MinimumPosition, double* MaximumPosition, double* PositionStep, bool * EnableState)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	double * : MinimumPosition (units) double * : MaximumPosition (units) double * : PositionStep (units) bool * : EnableState
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerPositionCompareGet (int SocketID, char FullPositionerName[250], double* MinimumPosition, double* MaximumPosition, double* PositionStep, bool * EnableState)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	double * : MinimumPosition (units) double * : MaximumPosition (units) double * : PositionStep (units) bool * : EnableState
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner name and the parameters coherence test.
API Description	This API returns the position compare parameters. These parameters are used when the position compare mode is enabled. The PCO connector is used.
API Errors	0 -7 -8 -9 -12 -14 -19 -23

2.3.3.34 PositionerPositionCompareEnable

TCL Prototype	int PositionerPositionCompareEnable (int SocketID, char FullPositionerName[250])
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : FullPositionerName
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerPositionCompareEnable (int SocketID, char FullPositionerName[250])
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : FullPositionerName
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner name.
API Description	This API enables the position compare mode. The interpolation is managed by the interpolation board and the pulses are generated on the interpolation board output (PCO connector). The group must be READY to use this API.
API Errors	0 -7 -8 -9 -18 -19 -22

2.3.3.35 PositionerPositionCompareDisable

TCL Prototype	int PositionerPositionCompareDisable(int SocketID, char FullPositionerName[250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerPositionCompareDisable(int SocketID, char FullPositionerName[250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner name.
API Description	This API disables the position compare mode. The interpolation is managed by the controller and the pulses generation on the interpolation board output (PCO connector) is stopped.
API Errors	0 -7 -8 -9 -19

2.3.3.36 PositionerHardInterpolatorFactorSet

TCL Prototype	int PositionerHardInterpolatorFactorSet (int SocketID, char FullPositionerName[250], int InterpolationFactor)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName int : InterpolationFactor
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerHardInterpolatorFactorSet (int SocketID, char FullPositionerName[250], int InterpolationFactor)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName int : InterpolationFactor
Output parameters	None
Return	API error

API Input tests	<p>Verify the number of parameters. Verify the positioner name. Parameters coherence test. The “InterpolationFactor“ must be a value from this list :</p> <ul style="list-style-type: none"> 20 25 40 50 80 100 160 200
API Description	<p>This API sets the hard interpolator factor used in the “PositionCompare” mode. The group must be NOT INITIALIZED to use this API.</p> <p><u>CAUTION</u> : the Encoder Type must be “AnalogInterpolated”</p>
API Errors	0 -7 -8 -9 -15 -18 -19 -22

2.3.3.37 PositionerHardInterpolatorFactorGet

TCL Prototype	int PositionerHardInterpolatorFactorGet(int SocketID, char FullPositionerName[250], int * InterpolationFactor)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	int * : InterpolationFactor
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerHardInterpolatorFactorGet(int SocketID, char FullPositionerName[250], int * InterpolationFactor)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	int * : InterpolationFactor
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner name. Parameters coherence test.
API Description	This API returns the hard interpolator factor. <u>CAUTION :</u> the Encoder Type must be “AnalogInterpolated”
API Errors	0 -7 -8 -9 -15 -19

2.3.3.38 PositionerEncoderAmplitudeValuesGet

TCL Prototype	int PositionerEncoderAmplitudeValuesGet(int SocketID, char FullPositionerName[250], double * MaxSinusAmplitude, double * CurrentSinusAmplitude, double * MaxCosinusAmplitude, double * CurrentCosinusAmplitude)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	double * : Encoder sinus signal maximum amplitude value double * : Encoder sinus signal current amplitude value double * : Encoder cosinus signal maximum amplitude value double * : Encoder cosinus signal current amplitude value
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerEncoderAmplitudeValuesGet(int SocketID, char FullPositionerName[250], double * MaxSinusAmplitude, double * CurrentSinusAmplitude, double * MaxCosinusAmplitude, double * CurrentCosinusAmplitude)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	double * : Encoder sinus signal maximum amplitude value double * : Encoder sinus signal current amplitude value double * : Encoder cosinus signal maximum amplitude value double * : Encoder cosinus signal current amplitude value
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner name. Parameters coherence test.
API Description	This API returns the encoder amplitude values (in Volts), it is useful to the user to know how big is the encoder output signals at any moment. <u>CAUTION</u> : the Encoder Type must be “AnalogInterpolated”
API Errors	0 -7 -8 -9 -14 -19

2.3.3.39 PositionerEncoderCalibrationParametersGet

TCL Prototype	int PositionerEncoderCalibrationParametersGet(int SocketID, char FullPositionerName[250], double * SinusOffset, double * CosinusOffset, double * DifferentialGain, double * PhaseCompensation)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	double * : Encoder sinus signal offset double * : Encoder cosinus signal offset double * : Encoder differential gain double * : Encoder phase compensation
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int PositionerEncoderCalibrationParametersGet(int SocketID, char FullPositionerName[250], double * SinusOffset, double * CosinusOffset, double * DifferentialGain, double * PhaseCompensation)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	double * : Encoder sinus signal offset double * : Encoder cosinus signal offset double * : Encoder differential gain double * : Encoder phase compensation
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner name. Parameters coherence test.
API Description	After an encoder calibration (by GroupInitializeWithEncoderCalibration API), the user sends this API to get the encoder calibration parameter values. These values are to be put by the user to the stage.ini file corresponding stage section in order to have good encoder imperfections compensation at the controller next booting. <u>CAUTION :</u> the Encoder Type must be “AnalogInterpolated”
API Errors	0 -7 -8 -9 -14 -19

2.3.3.40 GroupJogCurrentGet

TCL Prototype	int GroupJogCurrentGet (int SocketID, char FullPositionerName [250], double *Velocity, double *Acceleration)
----------------------	--

Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	double * : Velocity (units / s) double * : Acceleration (units / s ²)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupJogCurrentGet (int SocketID, char FullPositionerName [250], int NbPositioners, double Velocity[], double Acceleration[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName int : NbPositioners (Must be 1 in case of a positioner)
Output parameters	double [1] : Velocity array (units / s) double [1] : Acceleration array (units / s ²)
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner name. Parameters coherence test.
API Description	This API returns the current velocity and the current acceleration used by JOG mode.
API Errors	0 -7 -8 -9 -14 -19

2.3.3.41 GroupJogParametersGet

TCL Prototype	int GroupJogParametersGet (int SocketID, char FullPositionerName [250], double *Velocity, double *Acceleration)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	double * : Velocity (units / s) double * : Acceleration (units / s ²)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupJogParametersGet (int SocketID, char FullPositionerName [250], int NbPositioners, double *Velocity, double *Acceleration)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName int : NbPositioners (Must be 1 in case of a positioner)
Output parameters	double [1] : Velocity array (units / s) double [1] : Acceleration array (units / s ²)
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner name. Parameters coherence test.
API Description	This API returns the user velocity and the user acceleration changed with the “GroupJogParametersSet” API.
API Errors	0 -7 -8 -9 -14 -19

2.3.3.42 GroupJogParametersSet

TCL Prototype	int GroupJogParametersSet (int SocketID, char FullPositionerName [250], double Velocity, double Acceleration)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : FullPositionerName double : Velocity (units / s) double : Acceleration (units / s ²)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupJogParametersSet (int SocketID, char FullPositionerName [250], int NbPositioners, double Velocity[], double Acceleration[])
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : FullPositionerName int : NbPositioners (Must be 1 in case of a positioner) double[1] : Velocity array (units / s) double[1] : Acceleration array (units / s ²)
Output parameters	None
Return	API error

API Input tests	<p>Verify the number of parameters. Verify the positioner name. Parameters coherence test :</p> <p>Velocity > MaximumVelocity => Velocity = MaximumVelocity Velocity < - MaximumVelocity => Velocity = - MaximumVelocity Acceleration ≤ 0 => ERROR and stop motion Acceleration > MaximumAcceleration => Acceleration = MaximumAcceleration</p>
API Description	<p>The JOG mode must be activated (after the call of "GroupJogModeEnable" API) This API allows to change on fly the velocity and the acceleration used by the JOG mode. If an error is occurred, the positioner and all axes from the same group are stopped with a velocity NULL.</p> <p><u>NOTE</u> : This API returns the result once the constant velocity is reached</p>
API Errors	0 -7 -8 -9 -14 -19 -42 -22

2.3.3.43 GroupMoveAbsolute

TCL Prototype	int GroupMoveAbsolute (int SocketID, char FullPositionerName [250], double TargetPosition)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : Positioner name double : TargetPosition (units)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupMoveAbsolute (int SocketID, char FullPositionerName [250], int NbPositioners, double TargetPosition[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : Positioner name int : Number of Positioners (Must be 1 in case of a positioner) double [1] : TargetPosition array (units)
Output parameters	None
Return	API error

Input tests	Verify number of parameters. Verify positioner name. Parameters coherence test. Verify target position in relation with the travel limits : $\text{TargetPosition} \geq \text{MinimumTargetPosition}$ $\text{TargetPosition} \leq \text{MaximumTargetPosition}$
Description	The selected positioner moves to the target position. The absolute move refers to the acceleration, velocity, minimumTjerkTime and maximumTjerkTime predefined in the “Stages.ini” or redefined with the “PositionerSGammaParametersSet” API.
Errors	0 -7 -8 -9 -14 -17 -18 -19 -22 -25 -33 -44

2.3.3.44 GroupMoveRelative

TCL Prototype	int GroupMoveRelative (int SocketID, char FullPositionerName [250], double TargetDisplacement)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : Positioner name double : TargetDisplacement (units)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupMoveRelative (int SocketID, char FullPositionerName [250], int NbPositioners, double TargetDisplacement [])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : Positioner name int : Number of Positioners (Must be 1 in case of a positioner) double [1] : TargetDisplacement array (units)
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner name. Parameters coherence test. Test the recalculated target position in relation with the travel limits : $\text{TargetPosition} \geq \text{MinimumTargetPosition}$ $\text{TargetPosition} \leq \text{MaximumTargetPosition}$
API Description	The selected positioner moves to the target position : $\text{TargetPosition} = \text{CurrentPosition} + \text{TargetDisplacement}$ The relative move refers to the acceleration, velocity, minimumTjerkTime and maximumTjerkTime predefined in the “Stages.ini” or redefined with the “PositionerSGammaParametersSet” API.
API Errors	0 -7 -8 -9 -14 -17 -18 -19 -22 -25 -33 -44

2.3.3.45 GroupPositionCurrentGet

TCL Prototype	int GroupPositionCurrentGet (int SocketID, char FullPositionerName [250], double * CurrentPosition)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : Positioner name
Output parameters	double * : CurrentPosition (units)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupPositionCurrentGet (int SocketID, char FullPositionerName [250], int NbPositioner, double CurrentPosition[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : Positioner name int : Number of Positioners (Must be 1 in case of a positioner)
Output parameters	double [1] : CurrentPosition array (units)
Return	API error

API Input tests	Verify number of parameters. Verify positioner name. Parameters coherence test.
API Description	Reads the selected positioner current position (SetpointPosition - PositionError).
API Errors	0 -7 -8 -9 -14 -18 -19

2.3.3.46 GroupPositionSetpointGet

TCL Prototype	int GroupPositionSetpointGet (int SocketID, char FullPositionerName [250], double * SetPointPosition)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : Positioner name
Output parameters	double * : SetPointPosition (units)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupPositionSetpointGet (int SocketID, char FullPositionerName [250], int NbPositioner, double SetPointPosition [])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : Positioner name int : Number of Positioners (Must be 1 in case of a positioner)
Output parameters	double [1] : SetPointPosition array (units)
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner name. Parameters coherence test.
API Description	Read the selected positioner Setpoint position (profiler position).
API Errors	0 -7 -8 -9 -14 -18 -19

2.3.3.47 GroupPositionTargetGet

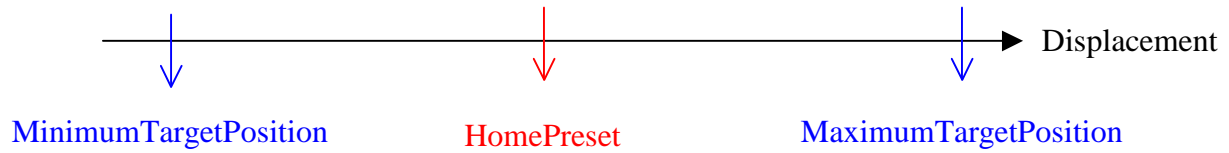
TCL Prototype	int GroupPositionTargetGet (int SocketID, char FullPositionerName [250], double * TargetPosition)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : Positioner name
Output parameters	double * : TargetPosition (units)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupPositionTargetGet (int SocketID, char FullPositionerName [250], int NbPositioner, double TargetPosition [])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : Positioner name int : Number of Positioners (Must be 1 in case of a positioner)
Output parameters	double [1] : TargetPosition array (units)
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner name. Parameters coherence test.
API Description	Read the selected positioner Target position (user position).
API Errors	0 -7 -8 -9 -14 -18 -19

2.3.4 Configuration parameters

2.3.4.1 Travels



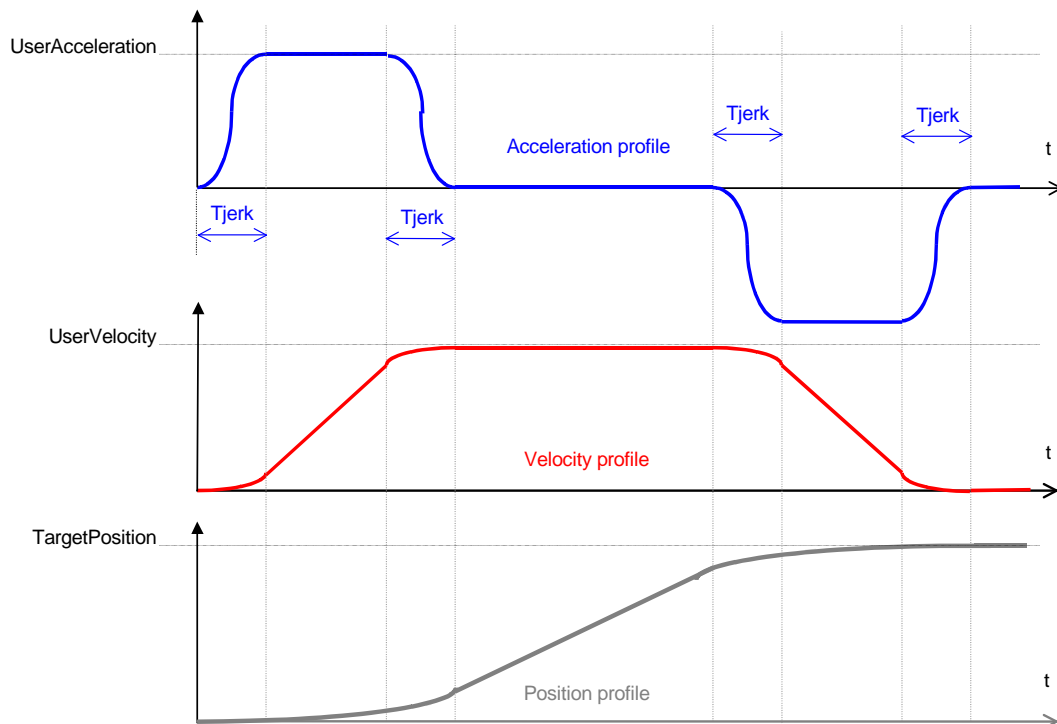
Conditions:

$$\text{MinimumTargetPosition} \leq \text{HomePreset} \leq \text{MaximumTargetPosition}$$

MinimumTargetPosition and MaximumTargetPosition represent the limits of run.
HomePreset

2.3.4.2 Profiler

MaximumVelocity
MaximumAcceleration
MinimumJerkTime
MaximumJerkTime



2.3.4.3 Motion done

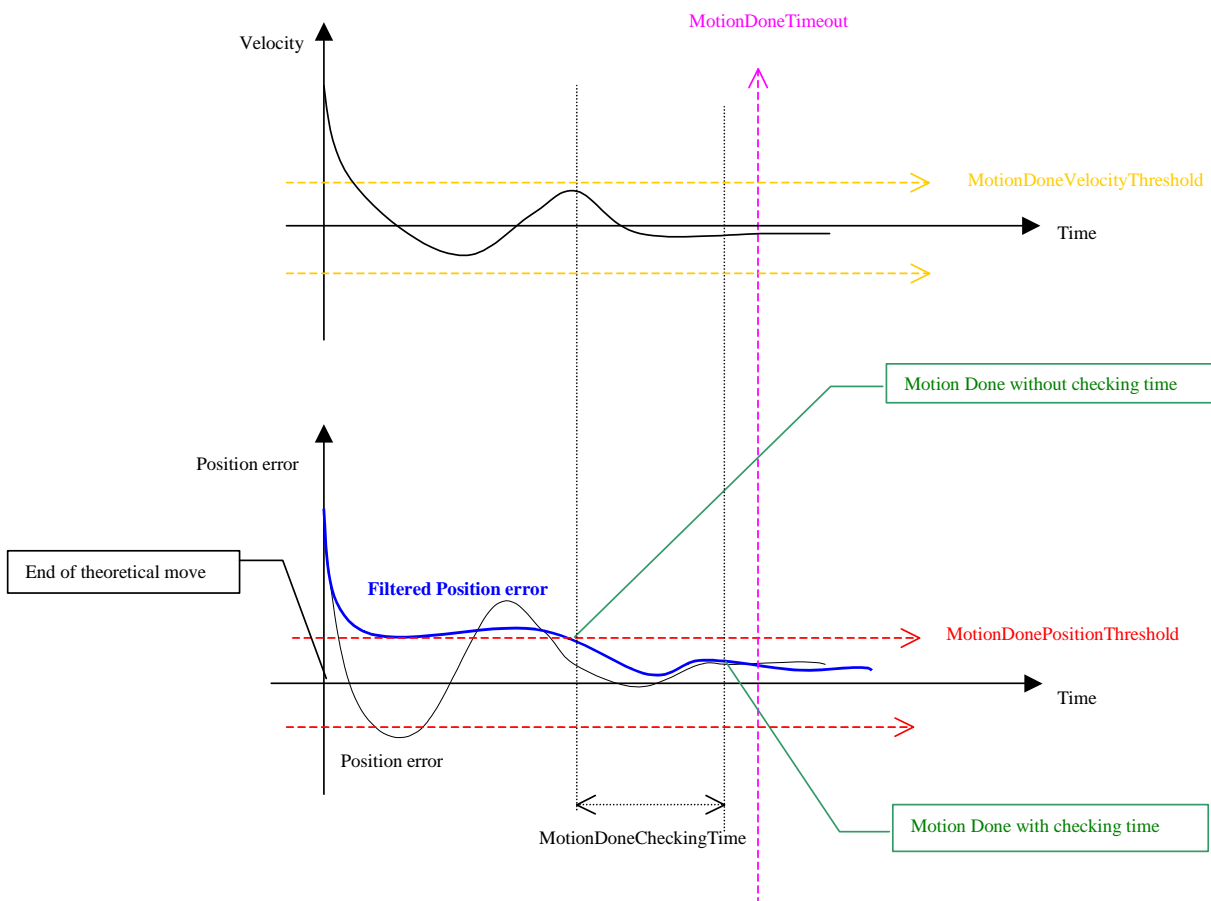
* MotionDone **without** checking time is activated as soon as:

$$\begin{aligned} & | \text{PositionErrorMeanValue} | < \text{MotionDonePositionThreshold} \\ \&\& \\ & | \text{VelocityMeanValue} | < \text{MotionDoneVelocityThreshold} \end{aligned}$$

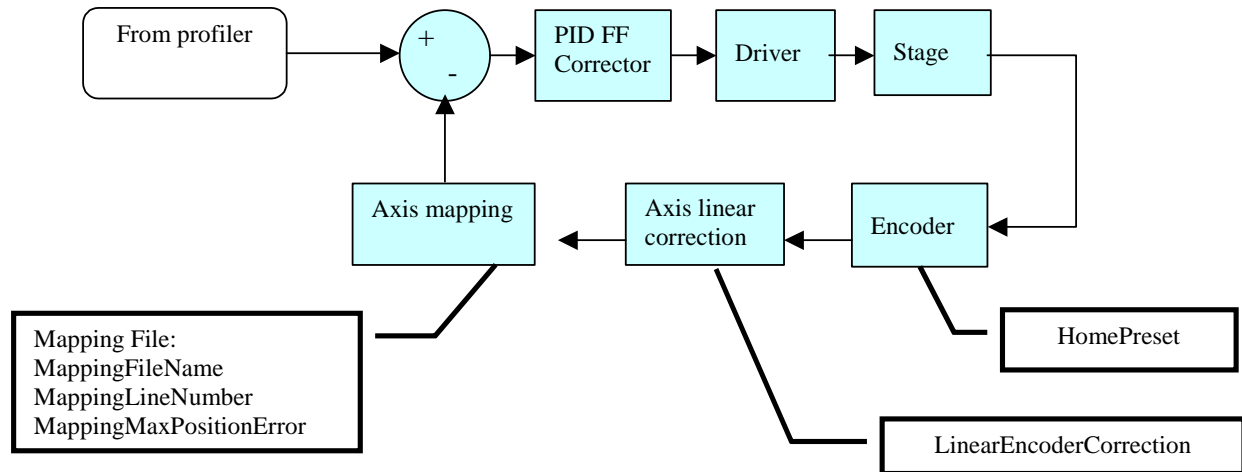
* MotionDone **with** checking time is activated at the end of MotionDoneCheckingTime, if ...

$$\begin{aligned} & | \text{PositionErrorMeanValue} | < \text{MotionDonePositionThreshold} \\ \&\& \\ & | \text{VelocityMeanValue} | < \text{MotionDoneVelocityThreshold} \end{aligned}$$

... is verified during the MotionDoneCheckingTime period.



2.3.4.4 Positioner mapping



HomePreset = Encoder position value at the home position.

LinearEncoderCorrection = value in ppm . Correction is given by

$\text{Correction} = \text{EncoderPosition} (1 + \text{LinearCorrection} / 10^6) - \text{HomePreset} (\text{LinearCorrection} / 10^6)$

Mapping declaration

Mapping will be activated if the file specify in the parameter **MappingFileName** is found and coherent.

MappingFileName

The mapping parameters are used to verify the coherence of the file :

MappingLineNumber

MaxPositionError

Mapping File

The mapping files must be in the “\ADMIN\CONFIG” directory.

The “MappingFileName” file is composed of 2 columns : Position and ΔPosition

PosMin	Error 0
Pos 1	Error 1
Pos 2	Error 2
...	...
0	0
...	...
PosMax	Error LineNumber-1

First row = Positioner positions. Must contain the value 0. This value 0 in the file correspond to the HomePreset position in the positioner referential. PosMin and PosMax must at least cover the entire travel.

Second row = Position errors. Error is equal to 0 at the position 0.

2.3.4.5 Configuration parameters

Two configuration files are used during the controller initialization : “System.ini” and “Stages.ini”

The system and groups configuration parameters from System.ini file.

The positioner configuration parameters from Stages.ini file (units = mm).

2.3.4.5.1 System.ini file

[GENERAL]

```
FirmwareName = MainController
ExternalModuleNames = TCL_API_drivers.out
CorrectorISRPeriod = 100e-6 ; seconds
IRQDelay = 10e-6 ; seconds
ProfileGeneratorISRRatio = 4
ServitudesISRRatio = 10
BootScriptFileName =
BootScriptArguments = ; the separator is the comma
```

[GROUPS]

```
SingleAxisInUse = MyGROUP ; the separator is the comma
XYInUse =
XYZInUse =
MultipleAxesInUse =
```

[MyGROUP]

```
PositionerInUse = MyPOSITIONER ; the separator is the comma
```

[MyGROUP. MyPOSITIONER]

```
PlugNumber =
PositionerName = MySTAGE ; see “ stages.ini” file
```

2.3.4.5.2 Stages.ini file

```
[MySTAGE]

;--- MOTOR DRIVER INTERFACE
MotorDriverInterface = ; AnalogVelocity
                        ; AnalogAcceleration
                        ; AnalogVoltage
                        ; AnalogPosition
                        ; AnalogStepperPosition
                        ; AnalogSin60Acceleration
                        ; AnalogSin90Acceleration
                        ; AnalogSin120Acceleration
                        ; AnalogDualSin60Acceleration
                        ; AnalogDualSin90Acceleration
                        ; AnalogDualSin120Acceleration

; 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
InitializationCycleDuration = ; seconds

; 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 = ; CurrentPositionAsHome
                        ; IndexHomeSearch
                        ; MechanicalZeroHomeSearch
                        ; MechanicalZeroAndIndexHomeSearch
                        ; MinusEndOfRunAndIndexHomeSearch
                        ; MinusEndOfRunHomeSearch
```

```
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)
NotchBandwidth1 = ; Hertz
NotchGain1 =
NotchFrequency2 = ; Hertz (0 = not activated)
NotchBandwidth2 = ; Hertz
NotchGain2 =

```

;--- MOTION DONE

```

MotionDoneMode = ; Theoretical
; VelocityAndPositionWindow

```

; If MotionDoneMode = VelocityAndPositionWindow

```

MotionDonePositionThreshold = ; units
MotionDoneVelocityThreshold = ; units / second
MotionDoneCheckingTime = ; seconds
MotionDoneMeanPeriod = ; seconds
MotionDoneTimeout = ; seconds

```



```

;--- SERVITUDES
ServitudesType = ; StandardEORDriverPlug
                ; StandardEOREncoderPlug

;--- Stage
SmartStageName = ; Smart stage

;--- DRIVER
DriverName = ; XPS-DRV00 (pass through board)
             ; XPS-DRV01 (eq. ESP300 driver)
             ; XPS-DRV02 (eq. MM56CC)

; If DriverName = XPS-DRV01 driver
;----- If MotorDriverInterface = AnalogVelocity
DriverPWMPFrequency =
DriverErrorAmplifierGain =

;----- If MotorDriverInterface = AnalogVoltage
PWMPFrequency =

;----- If MotorDriverInterface = AnalogStepper
PWMPFrequency =
HalfWinding =

; If DriverName = XPS-DRV02 driver
DriverBridgeFreeWheel =
DriverStepperWinding =

```

2.4 SingleAxis group

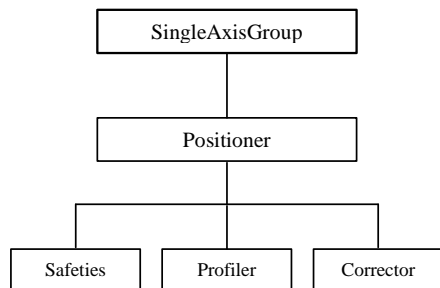
2.4.1 Description

Single positioner object that allows execution of motion commands.

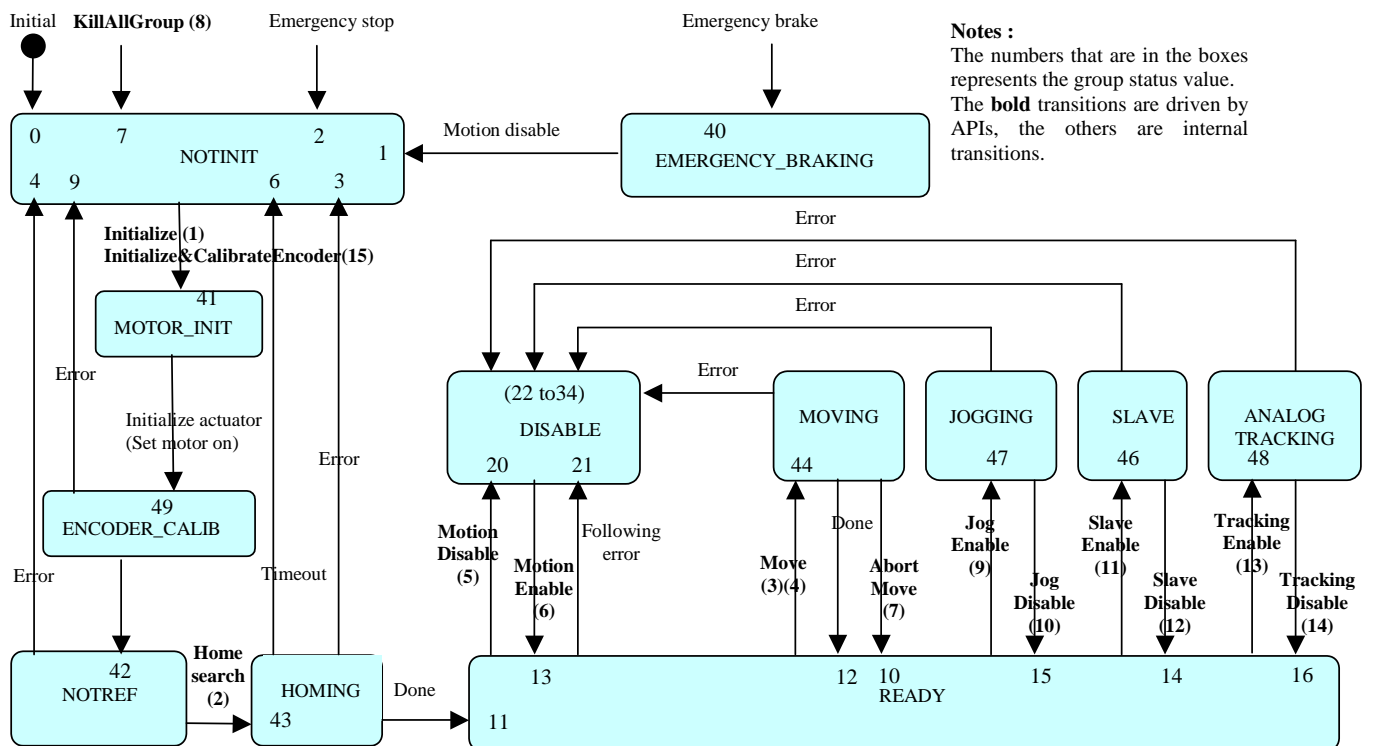
The controller can handle several SingleAxis objects.

There is no relations between SingleAxis objects and other objects handled by the controller.

2.4.2 Structure



2.4.3 State diagram



Called API:

- | | | | |
|-----------------------|--------------------------|----------------------------|--|
| (1) GroupInitialize | (5) GroupMotionDisable | (9) GroupJogModeEnable | (13) GroupAnalogTrackingModeEnable |
| (2) GroupHomeSearch | (6) GroupMotionEnable | (10) GroupJogModeDisable | (14) GroupAnalogTrackingModeDisable |
| (3) GroupMoveAbsolute | (7) GroupMoveAbort | (11) GroupSlaveModeEnable | (15) GroupInitializeWithEncoderCalibration |
| (4) GroupMoveRelative | (8) GroupKill or KillAll | (12) GroupSlaveModeDisable | |

2.4.4 API description

2.4.4.1 GroupAnalogTrackingModeDisable

TCL Prototype	int GroupAnalogTrackingModeDisable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupAnalogTrackingModeDisable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	The group must be in ANALOG TRACKING status.
API Errors	0 -7 -8 -9 -19 -22

2.4.4.2 GroupAnalogTrackingModeEnable

TCL Prototype	int GroupAnalogTrackingModeEnable (int SocketID, char GroupName [250], char Type [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name char [250] : Type (“Position” or “Velocity”)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupAnalogTrackingModeEnable (int SocketID, char GroupName [250], char Type [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name char [250] : Type (“Position” or “Velocity”)
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name.
API Description	The group must be in ready status (after initialization and home search) and if OK the group is setting in ANALOG TRACKING mode. If type is “Position”, the analog input is interpreted as an axis position command and the parameters are settled by the “AnalogTrackingPositionParametersSet” API and can be read by the “AnalogTrackingPositionParametersGet” API. If type is “valocity”, the analog input is interpreted as axis velocity command and the parameters are settled by the “AnalogTrackingVelocityParametersSet” API and can be read by the “AnalogTrackingVelocityParametersSet” API.
API Errors	0 -7 -8 -9 -19 -22

2.4.4.3 GroupHomeSearch

TCL Prototype	int GroupHomeSearch (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupHomeSearch (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name and the parameters coherence test.
API Description	Activates the selected home search, configured in the Stages.ini, for the selected single axis. HomeSearchProfileGeneratorType = <ul style="list-style-type: none"> ✓ IndexHomeSearch ✓ CurrentPositionAsHome ✓ MechanicalZeroAndIndexHomeSearch ✓ MechanicalZeroHomeSearch ✓ MinusEndOfRunAndIndexHomeSearch ✓ MinusEndOfRunHomeSearch
API Errors	0 -7 -8 -9 -19 -21 -22 -25 -27 -28 -31 -33 -45

2.4.4.4 GroupHomeSearchAndRelativeMove

TCL Prototype	int GroupHomeSearchAndRelativeMove (int SocketID, char GroupName [250], double displacement)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name double : displacement
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupHomeSearchAndRelativeMove (int SocketID, char GroupName [250], int NbPositioners, double displacement[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name int : Number of Positioners in the group (must be 1 for a single axis group) double [] : displacement array
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name and the parameters coherence test.
API Description	Activates the selected home search, configured in the Stages.ini, for the selected single axis and execute a relative motion at end of the home search. HomeSearchProfileGeneratorType = <ul style="list-style-type: none"> ✓ IndexHomeSearch ✓ CurrentPositionAsHome ✓ MechanicalZeroAndIndexHomeSearch ✓ MechanicalZeroHomeSearch ✓ MinusEndOfRunAndIndexHomeSearch ✓ MinusEndOfRunHomeSearch
API Errors	0 -7 -8 -9 -19 -21 -22 -25 -27 -28 -31 -33 -45

2.4.4.5 GroupInitialize

TCL Prototype	int GroupInitialize (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupInitialize (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name.
API Description	Initialize the motors and activate the servo loop of the selected single axis group. NOTE : In Master-Slave case, after an emergency signal, the master group and the slave group are in “Not Initialized” status. To reinitialise these groups, the process is the following : the slave(s) group must be reinitialised before the master group.
API Errors	0 -7 -8 -9 -19 -22

2.4.4.6 GroupInitializeWithEncoderCalibration

TCL Prototype	int GroupInitializeWithEncoderCalibration (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : SingleAxis group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupInitializeWithEncoderCalibration (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : SingleAxis group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Initialize the motors, calibrate the encoder and activate the servo loop of the selected single axis group. To get the calibration results, use the PositionerEncoderCalibrationParametersGet API.
API Errors	0 -7 -8 -9 -19 -22

2.4.4.7 GroupJogModeDisable

TCL Prototype	int GroupJogModeDisable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupJogModeDisable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	The group must be in JOGGING status and the positioner must be idle (velocity must be NULL)
API Errors	0 -7 -8 -9 -19 -22

2.4.4.8 GroupJogModeEnable

TCL Prototype	int GroupJogModeEnable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupJogModeEnable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name.
API Description	The group must be in ready status (after initialization and home search) and if OK the group is setting in JOG mode (JOGGING status).
API Errors	0 -7 -8 -9 -19 -22

2.4.4.9 GroupJogCurrentGet

TCL Prototype	int GroupJogCurrentGet (int SocketID, char GroupName [250], double *Velocity, double *Acceleration)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name
Output parameters	double * : Velocity (units / s) double * : Acceleration (units / s ²)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupJogCurrentGet (int SocketID, char GroupName [250], int NbPositioners, double Velocity[], double Acceleration[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name int : Number of Positioners in the group (must be 1 for a single axis group)
Output parameters	double [] : Velocity array (units / s) double [] : Acceleration array (units / s ²)
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner name. Parameters coherence test.
API Description	This API returns the current velocity and the current acceleration used by JOG mode.
API Errors	0 -7 -8 -9 -14 -19

2.4.4.10 GroupJogParametersGet

TCL Prototype	int GroupJogParametersGet (int SocketID, char GroupName [250], double *Velocity, double *Acceleration)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name (can be positioner name : see § positioner API)
Output parameters	double * : Velocity (units / s) double * : Acceleration (units / s ²)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupJogParametersGet (int SocketID, char GroupName [250], int NbPositioners, double Velocity[], double Acceleration[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name (can be positioner name : see § positioner API) int : Number of Positioners in the group (must be 1 for a single axis group)
Output parameters	double [] : Velocity array (units / s) double [] : Acceleration array (units / s ²)
Return	API error

API Input tests	Verify the number of parameters. Verify the group name or the positioner name. Parameters coherence test.
API Description	This API returns the user velocity and the user acceleration changed by GroupJogParametersSet.
API Errors	0 -7 -8 -9 -14 -19

2.4.4.11 GroupJogParametersSet

TCL Prototype	int GroupJogParametersSet (int SocketID, char GroupName [250], double Velocity, double Acceleration)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char[250] : SingleAxis group name (can be positioner name : see § positioner API) double : Velocity (units / s) double : Acceleration (units / s ²)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupJogParametersSet (int SocketID, char GroupName [250], int NbPositioners, double Velocity[], double Acceleration[])
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char[250] : SingleAxis group name (can be positioner name : see § positioner API) int : Number of Positioners in the group (must be 1 for a single axis group) double [] : Velocity (units / s) double [] : Acceleration (units / s ²)
Output parameters	None
Return	API error

API Input tests	<p>Verify the number of parameters. Verify the group name or the positioner name. Parameters coherence test :</p> <p>Velocity > MaximumVelocity => Velocity = MaximumVelocity Velocity < - MaximumVelocity => Velocity = - MaximumVelocity Acceleration ≤ 0 => ERROR and stop motion Acceleration > MaximumAcceleration => Acceleration = MaximumAcc.</p>
API Description	<p>The JOG mode must be activated (after the call of "GroupJogModeEnable" API) This API allows to change on fly the velocity and the acceleration used by the JOG mode. If an error is occurred, the positioner is stopped with a velocity NULL.</p>
API Errors	0 -7 -8 -9 -14 -17 -19 -22 -42

2.4.4.12 GroupKill

TCL Prototype	int GroupKill (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupKill (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Kills the SingleAxis group. The SingleAxis groups comes back to the “not initialised” status
API Errors	0 -7 -8 -9 -26

2.4.4.13 GroupMotionDisable

TCL Prototype	int GroupMotionDisable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupMotionDisable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Turn motor OFF and set the “MotionEnable” status to FALSE for the selected SingleAxis.
API Errors	0 -7 -8 -9 -19 -22

2.4.4.14 GroupMotionEnable

TCL Prototype	int GroupMotionEnable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupMotionEnable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Initializes positions and corrector before to turn motor ON and to set the “MotionEnable” status to TRUE for the selected SingleAxis.
API Errors	0 -7 -8 -9 -19 -22

2.4.4.15 GroupMoveAbort

TCL Prototype	int GroupMoveAbort (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupMoveAbort (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name
Output parameters	None
Return	API error

API Input tests	Verify number of parameters. Verify group name. Parameters coherence test.
API Description	Aborts the motion : stops the motion in progress on the selected SingleAxis.
API Errors	0 -7 -8 -9 -19 -22

2.4.4.16 GroupMoveAbsolute

TCL Prototype	int GroupMoveAbsolute (int SocketID, char GroupName [250], double TargetPosition)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char[250] : SingleAxis group name (can be an positioner name : see § positioner API) double : TargetPosition (units)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupMoveAbsolute (int SocketID, char GroupName [250], int NbPositioners, double TargetPosition[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char[250] : SingleAxis group name (can be an positioner name : see § positioner API) int : Number of Positioners in the group (must be 1 for a single axis group) double [] : TargetPosition array (units)
Output parameters	None
Return	API error

API Input tests	Verify number of parameters. Verify group name. Parameters coherence test. Verify target position in relation with the travel limits : $\text{TargetPosition} \geq \text{MinimumTargetPosition}$ $\text{TargetPosition} \leq \text{MaximumTargetPosition}$
API Description	The selected single axis moves to the target position. The absolute move refers to the acceleration, velocity, minimumJerkTime and maximumJerkTime predefined in the “Stages.ini” or redefined with the “PositionerSGammaParametersSet” API.
API Errors	0 -7 -8 -9 -14 -17 -19 -22 -25 -26 -27 -33 -44

2.4.4.17 GroupMoveRelative

TCL Prototype	int GroupMoveRelative (int SocketID, char GroupName [250], double TargetDisplacement)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name (can be positioner name : see § positioner API) double : TargetDisplacement (units)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupMoveRelative (int SocketID, char GroupName [250], int NbPositioners, double TargetDisplacement[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char[250] : SingleAxis group name (can be positioner name : see § positioner API) int : Number of Positioners in the group (must be 1 for a single axis group) double [] : TargetDisplacement array (units)
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test. Test the recalculated target position in relation with the travel limits : $\text{TargetPosition} \geq \text{MinimumTargetPosition}$ $\text{TargetPosition} \leq \text{MaximumTargetPosition}$
API Description	The SingleAxis moves to the target position : $\text{TargetPosition} = \text{CurrentPosition} + \text{TargetDisplacement}$ <p>The relative move refers to the acceleration, velocity, minimumJerkTime and maximumJerkTime predefined in the “Stages.ini” or redefined with the “PositionerSGammaParametersSet” API.</p>
API Errors	0 -7 -8 -9 -14 -17 -19 -22 -25 -26 -27 -33 -44

2.4.4.18 GroupPositionCurrentGet

TCL Prototype	int GroupPositionCurrentGet (int SocketID, char GroupName [250], double * CurrentPosition)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char[250] : SingleAxis group name (can be positioner name : see § positioner API)
Output parameters	double * : CurrentPosition (units)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupPositionCurrentGet (int SocketID, char GroupName [250], int NbPositioners, double CurrentPosition[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char[250] : SingleAxis group name (can be positioner name : see § positioner API) int : Number of Positioners in the group (must be 1 for a single axis group)
Output parameters	double [] : CurrentPosition array (units)
Return	API error

API Input tests	Verify number of parameters. Verify group name. Parameters coherence test.
API Description	Reads the SingleAxis current position (SetpointPosition - PositionError).
API Errors	0 -7 -8 -9 -14 -19

2.4.4.19 GroupPositionSetpointGet

TCL Prototype	int GroupPositionSetpointGet (int SocketID, char GroupName [250], double * SetPointPosition)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char[250] : SingleAxis group name (can be positioner name : see § positioner API)
Output parameters	double * : SetPointPosition (units)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupPositionSetpointGet (int SocketID, char GroupName [250], int NbPositioners, double SetPointPosition[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char[250] : SingleAxis group name (can be positioner name : see § positioner API) int : Number of Positioners in the group (must be 1 for a single axis group)
Output parameters	double [] : SetPointPosition array (units)
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Read the SetpointPosition (profiler position) of the SingleAxis.
API Errors	0 -7 -8 -9 -14 -19

2.4.4.20 GroupPositionTargetGet

TCL Prototype	int GroupPositionTargetGet (int SocketID, char GroupName [250], double * TargetPosition)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char[250] : SingleAxis group name (can be positioner name : see § positioner API)
Output parameters	double * : TargetPosition (units)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupPositionTargetGet (int SocketID, char GroupName [250], int NbPositioners, double TargetPosition[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char[250] : SingleAxis group name (can be positioner name : see § positioner API) int : Number of Positioners in the group (must be 1 for a single axis group)
Output parameters	double [] : TargetPosition array (units)
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Read the TargetPosition (user position) of the SingleAxis.
API Errors	0 -7 -8 -9 -14 -19

2.4.4.21 GroupStatusGet

TCL Prototype	int GroupStatusGet (int SocketID, char GroupName [250], int * Status)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name
Output parameters	int * : Group status (see § “Group state list”)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupStatusGet (int SocketID, char GroupName [250], int * Status)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name
Output parameters	int * : Group status (see § “Group state list”)
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Reads SingleAxis group status (See § “Group state list”) and returns the group status code. Call the “GroupStatusStringGet” API to get the group status description from the group status code.
API Errors	0 -7 -8 -9 -15 -19

2.4.4.22 SingleAxisSlaveParametersSet

TCL Prototype	int SingleAxisSlaveParametersSet (int SocketID, char GroupName [250], char FullPositionerName[250], double Ratio)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name (slave must be a SingleAxis group) char [250] : FullPositionerName (master must be Positioner from any group) double : Ratio
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int SingleAxisSlaveParametersSet (int SocketID, char GroupName [250], char FullPositionerName[250], double Ratio)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name (slave must be a SingleAxis group) char [250] : FullPositionerName (master must be Positioner from any group) double : Ratio
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test. The ratio must different from zero
API Description	Set the slave parameters : the slave must be a “SingleAxis” group and the master must be an positioner from any group. The slave is a master copy : a ratio can be apply (Slave = ratio * Master). The slave-master mode is activated only after the call of “SingleAxisSlaveModeEnable” API. NOTE : After an emergency signal, the master group and the slave group are in “Not Initialized” status. To reinitialise these groups, the process is the following : the slave(s) group must be reinitialised before the master group.
API Errors	0 -7 -8 -9 -13 -14 -19 -22 -48

2.4.4.23 SingleAxisSlaveParametersGet

TCL Prototype	int SingleAxisSlaveParametersGet (int SocketID, char GroupName [250], char *FullPositionerName, double *Ratio)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name (slave must be a SingleAxis group)
Output parameters	char * : FullPositionerName (master must be Positioner from any group) double * : Ratio
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int SingleAxisSlaveParametersGet (int SocketID, char GroupName [250], char *FullPositionerName, double *Ratio)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name (slave must be SingleAxis group)
Output parameters	char * : FullPositionerName (master must be Positioner from any group) double * : Ratio
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Returns the slave parameters : the master positioner (from any group) and the ratio.
API Errors	0 -7 -8 -9 -13 -14 -19

2.4.4.24 SingleAxisSlaveModeEnable

TCL Prototype	int SingleAxisSlaveModeEnable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name (slave must be SingleAxis group)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int SingleAxisSlaveModeEnable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name (slave must be SingleAxis group)
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Enable the slave mode only if the “SingleAxis” group is READY. The “SingleAxisSlaveParametersSet” API must be called before to enable the slave-master mode.
API Errors	0 -7 -8 -9 -19 -22 -41

2.4.4.25 SingleAxisSlaveModeDisable

TCL Prototype	int SingleAxisSlaveModeDisable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int SingleAxisSlaveModeDisable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : SingleAxis group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Disable the slave mode only if the “SingleAxis” group status is SLAVE. The motion profile is setting to the previous motion profile.
API Errors	0 -7 -8 -9 -15 -19 -22

2.4.5 Configuration file

Example of configuration file for a SingleAxis group named “AXIS” composed of an positioner named “STAGE”. System.ini file :

System.ini file :

```
[GROUPS]
SingleAxisInUse = AXIS

[AXIS]                                ; AXIS SingleAxis group configuration
PositionerInUse = STAGE

[AXIS. STAGE]
PlugNumber =
StageName = MYSTAGE
```

Stages.ini file :

```
[MYSTAGE]
MYSTAGE configuration => See § “Positioner : Configuration parameters”
```

2.5 XY group

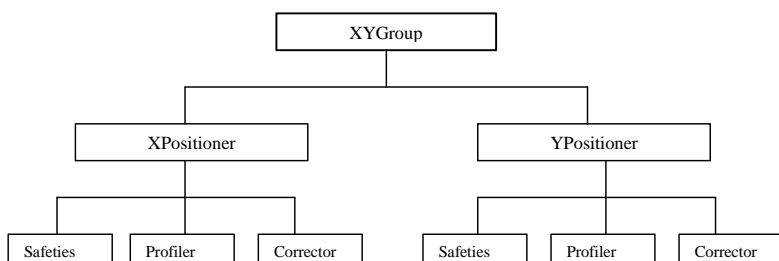
2.5.1 Description

This is a XY combination of two positioners objects.

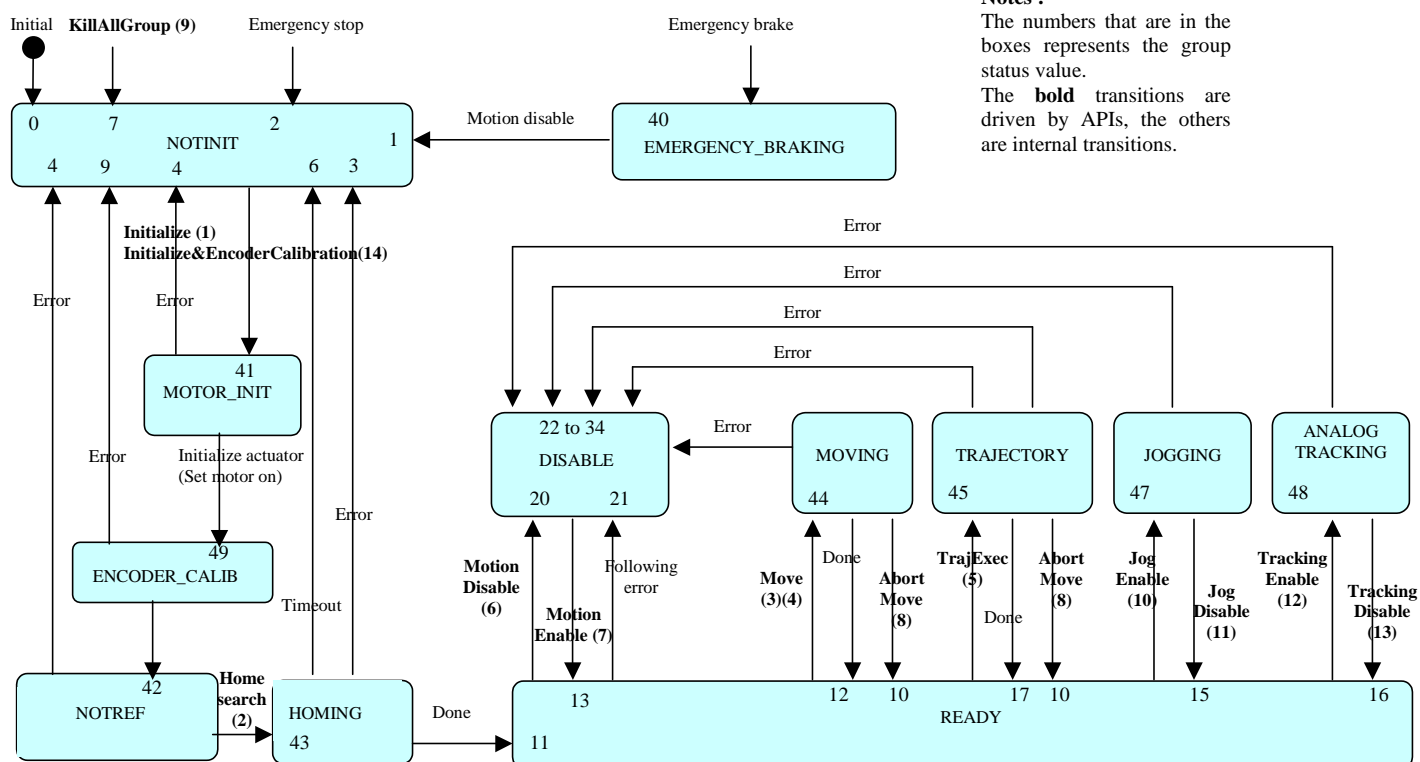
It includes an XY mapping feature : $XY = f(XY)$

It includes the XY trajectory (2D) : the trajectory is defined in a two dimensional X-Y plane.

2.5.2 Structure



2.5.3 State diagram



Notes :

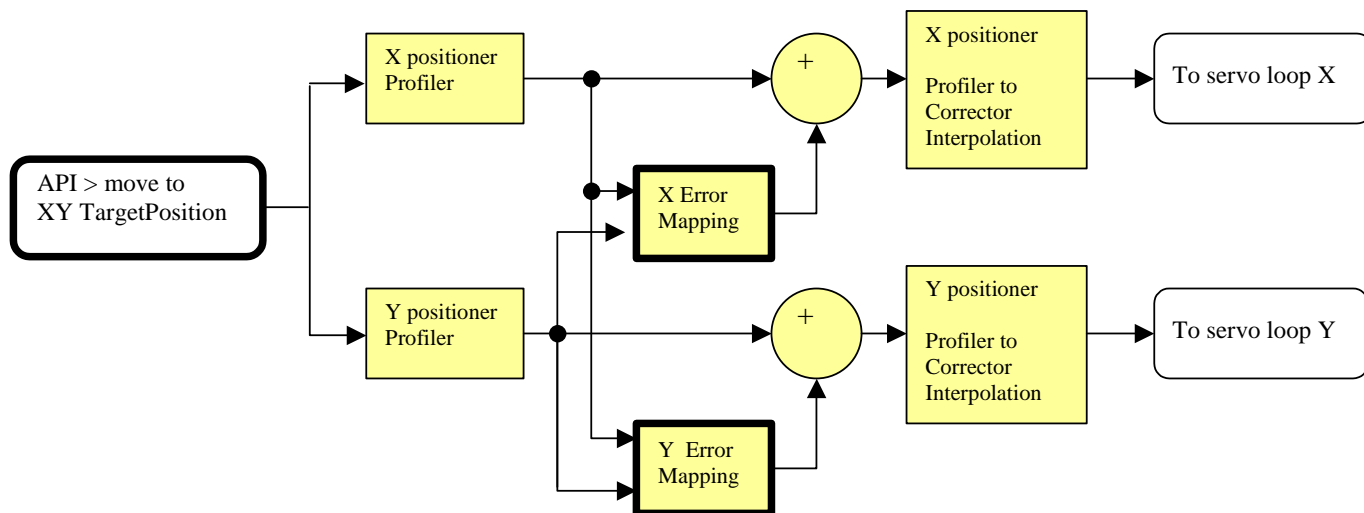
The numbers that are in the boxes represents the group status value.

The **bold** transitions are driven by APIs, the others are internal transitions.

Called API :

- | | | | |
|-----------------------|------------------------|------------------------------------|--|
| (1) GroupInitialize | (5) XYLineArcExecution | (9) GroupKill or KillAll | (13) GroupAnalogTrackingModeDisable |
| (2) GroupHomeSearch | (6) GroupMotionDisable | (10) GroupJogModeEnable | (14) GroupInitializeWithEncoderCalibration |
| (3) GroupMoveAbsolute | (7) GroupMotionEnable | (11) GroupJogModeDisable | |
| (4) GroupMoveRelative | (8) GroupMoveAbort | (12) GroupAnalogTrackingModeEnable | |

2.5.4 Additionnal XY correction from other object



Mapping will be activated if the files specified in the parameters **X(Y)MappingFileName** are found and coherent.

XMappingFileName
YMappingFileName

Each parameter is used to verify the coherence of the file :

XMappingLineNumber
XMappingColumnNumber
XMappingMaxPositionError
YMappingLineNumber
YMappingColumnNumber
YMappingMaxPositionError

The mapping files must be in the “\ADMIN\CONFIG” directory and the format is as following :

First cell must be 0.
First column = X Positions
First row = Y Positions
Each cell = corresponding error

NOTES :

Error in X = Y = 0 must be 0. This value in the file correspond to the HomePreset positions in the XY group referential. **X and Y positions must at least cover the entire travel of the XY group.**

X Mapping file

0	YMin	Y1	...	0	...	YMax
X Min	X err 0 0
X 1
...
0	0
...
XMax

XMapping
LineNumber

XMappingColumnNumber

Y Mapping file

0	YMin	Y1	...	0	...	YMax
X Min	Y err 0 0
X 1
...
0	0
...
XMax

YMapping
LineNumber

YMappingColumnNumber

2.5.5 API description

2.5.5.1 GroupAnalogTrackingModeDisable

TCL Prototype	int GroupAnalogTrackingModeDisable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupAnalogTrackingModeDisable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Disable the tracking mode : Exit ANALOG TRACKING status to come back in READY status. The group must be in ANALOG TRACKING status.
API Errors	0 -7 -8 -9 -19 -22

2.5.5.2 GroupAnalogTrackingModeEnable

TCL Prototype	int GroupAnalogTrackingModeEnable (int SocketID, char GroupName [250], char Type [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name char [250] : Type (“Position” or “Velocity”)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupAnalogTrackingModeEnable (int SocketID, char GroupName [250], char Type [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name char [250] : Type (“Position” or “Velocity”)
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name.
API Description	The group must be in READY status (after initialization and home search) and if OK the group is setting in ANALOG TRACKING mode. If type is “Position”, the analog input is interpreted as an axis position command and the parameters are settled by the “AnalogTrackingPositionParametersSet” API and can be read by the “AnalogTrackingPositionParametersGet” API. If type is “valocity”, the analog input is interpreted as axis velocity command and the parameters are settled by the “AnalogTrackingVelocityParametersSet” API and can be read by the “AnalogTrackingVelocityParametersSet” API.
API Errors	0 -7 -8 -9 -13 -19 -22

2.5.5.3 GroupHomeSearch

TCL Prototype	int GroupHomeSearch (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupHomeSearch (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Activates the selected home search, configured in the Stages.ini, for the selected XY group. The XY home search sequence can be Together, X then Y. HomeSearchProfileGeneratorType = <ul style="list-style-type: none"> ✓ IndexHomeSearch ✓ CurrentPositionAsHome ✓ MechanicalZeroAndIndexHomeSearch ✓ MechanicalZeroHomeSearch ✓ MinusEndOfRunAndIndexHomeSearch ✓ MinusEndOfRunHomeSearch
API Errors	0 -7 -8 -9 -19 -22 -25 -28 -31 -33 -45
System.ini	InitializationAndHomeSearchSequence = <i>Together</i> or <i>XthenY</i>

2.5.5.4 GroupHomeSearchAndRelativeMove

TCL Prototype	int GroupHomeSearchAndRelativeMove (int SocketID, char GroupName [250], double displacementX, double displacementY)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : XY group name double : displacementX (units) double : displacementY (units)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupHomeSearchAndRelativeMove (int SocketID, char GroupName [250], int NbPositioners, double displacement [])
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : XY group name int : Number of Positioners in the group (must be 2 for an XY group) double [2] : displacement array (units) with array[0] => X, array[1] => Y
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name and the parameters coherence test.
API Description	Activates the selected home search, configured in the Stages.ini, for the selected single axis and execute a relative motion at end of the home search. HomeSearchProfileGeneratorType = <ul style="list-style-type: none"> ✓ IndexHomeSearch ✓ CurrentPositionAsHome ✓ MechanicalZeroAndIndexHomeSearch ✓ MechanicalZeroHomeSearch ✓ MinusEndOfRunAndIndexHomeSearch ✓ MinusEndOfRunHomeSearch
API Errors	0 -7 -8 -9 -19 -21 -22 -25 -27 -28 -31 -33 -45

2.5.5.5 GroupInitialize

TCL Prototype	int GroupInitialize (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupInitialize (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name.
API Description	Initialize the motors and activate the servo loop of each positioner of the selected XY group. NOTE : In Master-Slave case, after an emergency signal, the master group and the slave group are in “Not Initialized” status. To reinitialise these groups, the process is the following : the slave(s) group must be reinitialised before the master group.
API Errors	0 -7 -8 -9 -19 -22

2.5.5.6 GroupInitializeWithEncoderCalibration

TCL Prototype	int GroupInitializeWithEncoderCalibration (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupInitializeWithEncoderCalibration (int SocketID, char GroupName [250],)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name.
API Description	Initialize the motors, calibrate the encoder and activate the servo loop of each positioner of the selected XY group. To get the calibration results for each positioner, use the PositionerEncoderCalibrationParametersGet API.
API Errors	0 -7 -8 -9 -19 -22

2.5.5.7 GroupJogModeDisable

TCL Prototype	int GroupJogModeDisable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupJogModeDisable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Disable the Jog mode : Exit JOGGING status to come back in READY status. The group must be in JOGGING status and the positioner must be idle (velocity must be NULL).
API Errors	0 -7 -8 -9 -19 -22

2.5.5.8 GroupJogModeEnable

TCL Prototype	int GroupJogModeEnable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupJogModeEnable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name.
API Description	The group must be in READY status (after initialization and home search) and if OK the group is setting in JOG mode (JOGGING status).
API Errors	0 -7 -8 -9 -19 -22

2.5.5.9 GroupJogCurrentGet

TCL Prototype	int GroupJogCurrentGet (int SocketID, char GroupName [250], double *VelocityX, double *AccelerationX, double *VelocityY, double *AccelerationY)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name
Output parameters	double * : VelocityX (units / s) double * : AccelerationX (units / s ²) double * : VelocityY (units / s) double * : AccelerationY (units / s ²)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupJogCurrentGet (int SocketID, char GroupName [250], int NbPositioners, double Velocity[], double Acceleration[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name int : Number of Positioners in the group (must be 2 for an XY group)
Output parameters	double [2] : Velocity array (units / s) with [0] => X, [1] => Y double [2] : Acceleration array (units / s ²) with [0] => X, [1] => Y
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner name. Parameters coherence test.
API Description	This API returns the current velocity and the current acceleration used by JOG mode.
API Errors	0 -7 -8 -9 -14 -19

2.5.5.10 GroupJogParametersGet

TCL Prototype	int GroupJogParametersGet (int SocketID, char GroupName [250], double *VelocityX, double *AccelerationX, double *VelocityY, double *AccelerationY)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name (can be positioner name : see § positioner API)
Output parameters	double * : VelocityX (units / s) double * : AccelerationX (units / s ²) double * : VelocityY (units / s) double * : AccelerationY (units / s ²)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupJogParametersGet (int SocketID, char GroupName [250], int NbPositioners, double Velocity[], double Acceleration[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name (can be positioner name : see § positioner API) int : Number of Positioners in the group (must be 2 for an XY group)
Output parameters	double [2] : Velocity array (units / s) with [0] => X, [1] => Y double [2] : Acceleration array (units / s ²) with [0] => X, [1] => Y
Return	API error

API Input tests	Verify the number of parameters. Verify the group name or the positioner name. Parameters coherence test.
API Description	This API returns the user velocity and the user acceleration changed with “GroupJogParametersSet”.
API Errors	0 -7 -8 -9 -14 -19

2.5.5.11 GroupJogParametersSet

TCL Prototype	int GroupJogParametersSet (int SocketID, char GroupName [250], double VelocityX, double AccelerationX, double VelocityY, double AccelerationY)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : XY group name (can be positioner name : see § positioner API) double : VelocityX (units / s) double : AccelerationX (units / s ²) double : VelocityY (units / s) double : AccelerationY (units / s ²)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupJogParametersSet (int SocketID, char GroupName [250], int NbPositioners, double Velocity[], double Acceleration[])
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : XY group name (can be positioner name : see § positioner API) int : Number of Positioners in the group (must be 2 for an XY group) double [2] : Velocity array (units / s) with [0] => X, [1] => Y double [2] : Acceleration array (units / s ²) with [0] => X, [1] => Y
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name or the positioner name. Parameters coherence test for each positioner (X and Y): Velocity > MaximumVelocity => Velocity = MaximumVelocity Velocity < - MaximumVelocity => Velocity = - MaximumVelocity Acceleration ≤ 0 => ERROR and stop motion Acceleration > MaximumAcceleration => Acceleration = MaximumAcc.
API Description	The JOG mode must be activated (after the call of "GroupJogModeEnable" API) This API allows to change on fly the velocity and the acceleration used by the JOG mode on each positioner. If an error is occurred, each positioner is stopped with a velocity NULL.
API Errors	0 -7 -8 -9 -14 -19 -22 -42

2.5.5.12 GroupKill

TCL Prototype	int GroupKill (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupKill (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Kills and resets the XY group. The XY group comes back to the “not initialized” status.
API Errors	0 -7 -8 -9 -19 -26

2.5.5.13 GroupMotionDisable

TCL Prototype	int GroupMotionDisable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupMotionDisable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Turns X motor OFF and turns Y motor OFF. Set the “MotionEnable” status to FALSE for the selected XY group.
API Errors	0 -7 -8 -9 -19 -22

2.5.5.14 GroupMotionEnable

TCL Prototype	int GroupMotionEnable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupMotionEnable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Initializes positions and corrector before to turn motor ON. Set the “MotionEnable” status to TRUE for the selected XY group.
API Errors	0 -7 -8 -9 -19 -22

2.5.5.15 GroupMoveAbort

TCL Prototype	int GroupMoveAbort (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupMoveAbort (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name
Output parameters	None
Return	API error

API Input tests	Verify number of parameters. Verify group name. Parameters coherence test.
API Description	Stops motion in progress on the X and Y axes.
API Errors	0 -7 -8 -9 -19 -22

2.5.5.16 GroupMoveAbsolute

TCL Prototype	int GroupMoveAbsolute (int SocketID, char GroupName [250], double TargetPositionX, double TargetPositionY)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name (can be positioner name : see § positioner API) double : TargetPositionX (units) double : TargetPositionY (units)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupMoveAbsolute (int SocketID, char GroupName [250], int NbPositioners, double TargetPosition[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name (can be positioner name : see § positioner API) int : Number of Positioners in the group (must be 2 for an XY group) double [2] : TargetPosition array (units) with [0] => X, [1] => Y
Output parameters	None
Return	API error

API Input tests	Verify number of parameters. Verify group name. Parameters coherence test. Verify target position in relation with the travel limits : <div style="margin-left: 40px;"> TargetPositionX ≥ MinimumTargetPosition (Positioner X) TargetPositionX ≤ MaximumTargetPosition (Positioner X) TargetPositionY ≥ MinimumTargetPosition (Positioner Y) TargetPositionY ≤ MaximumTargetPosition (Positioner Y) </div>
API Description	The selected XY group moves to the X and Y target positions. The absolute move refers to the acceleration, velocity, minimumTjerkTime and maximumTjerkTime predefined in the “Stages.ini” or redefined with the “PositionerSGammaParametersSet” API.
API Errors	0 -7 -8 -9 -14 -17 -19 -22 -25 -27 -33 -44

2.5.5.17 GroupMoveRelative

TCL Prototype	int GroupMoveRelative (int SocketID, char GroupName [250], double TargetDisplacementX, double TargetDisplacementY)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name (can be positioner name : see § positioner API) double : TargetDisplacementX (units) double : TargetDisplacementY (units)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupMoveRelative (int SocketID, char GroupName [250], int NbPositioners, double TargetDisplacement[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name (can be positioner name : see § positioner API) int : Number of Positioners in the group (must be 2 for an XY group) double [2] : TargetDisplacement array (units) with [0] => X, [1] => Y
Output parameters	None
Return	API error

API Input tests	<p>Verify number of parameters. Verify group name. Parameters coherence test. Verify target position in relation with the travel limits :</p> <p style="margin-left: 40px;"> TargetPositionX ≥ MinimumTargetPosition (Positioner X) TargetPositionX ≤ MaximumTargetPosition (Positioner X) TargetPositionY ≥ MinimumTargetPosition (Positioner Y) TargetPositionY ≤ MaximumTargetPosition (Positioner Y) </p>
API Description	<p>The selected XY group moves to the X and Y target positions :</p> <p>TargetPositionX = CurrentPositionX + TargetDisplacementX TargetPositionY = CurrentPositionY + TargetDisplacementY</p> <p>The relative move refers to the acceleration, velocity, minimumTjerkTime and maximumTjerkTime predefined in the “Stages.ini” or redefined with the “PositionerSGammaParametersSet” API.</p>
API Errors	0 -7 -8 -9 -14 -17 -19 -22 -25 -27 -33 -44

2.5.5.18 GroupPositionCurrentGet

TCL Prototype	int GroupPositionCurrentGet (int SocketID, char GroupName [250] , double * CurrentPositionX, double * CurrentPositionY)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name (can be positioner name : see § positioner API)
Output parameters	double * : CurrentPositionX (units) double * : CurrentPositionY (units)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupPositionCurrentGet (int SocketID, char GroupName [250] , int NbPositioners, double CurrentPosition[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name (can be positioner name : see § positioner API) int : Number of Positioners in the group (must be 2 for an XY group)
Output parameters	double [2] : CurrentPosition array (units) with [0] => X, [1] => Y
Return	API error

API Input tests	Verify number of parameters. Verify group name. Parameters coherence test.
API Description	Reads the X and Y current positions (SetpointPosition - PositionError).
API Errors	0 -7 -8 -9 -14 -19

2.5.5.19 GroupPositionSetpointGet

TCL Prototype	int GroupPositionSetpointGet (int SocketID, char GroupName [250] , double * SetPointPositionX, double * SetPointPositionY)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name (can be positioner name : see § positioner API)
Output parameters	double * : SetPointPositionX (units) double * : SetPointPositionY (units)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupPositionSetpointGet (int SocketID, char GroupName [250] , int NbPositioners, double SetPointPosition[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name (can be positioner name : see § positioner API) int : Number of Positioners in the group (must be 2 for an XY group)
Output parameters	double [2] : SetPointPosition array (units) with [0] => X, [1] => Y
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Read X and Y SetpointPositions (profiler position).
API Errors	0 -7 -8 -9 -14 -19

2.5.5.20 GroupPositionTargetGet

TCL Prototype	int GroupPositionTargetGet (int SocketID, char GroupName [250] , double * TargetPositionX, double * TargetPositionY)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name (can be positioner name : see § positioner API)
Output parameters	double * : TargetPositionX (units) double * : TargetPositionY (units)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupPositionTargetGet (int SocketID, char GroupName [250], int NbPositioners, double TargetPosition[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name (can be positioner name : see § positioner API) int : Number of Positioners in the group (must be 2 for an XY group)
Output parameters	double [2] : TargetPosition array (units) with [0] => X, [1] => Y
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Read X and Y TargetPositions (user position).
API Errors	0 -7 -8 -9 -14 -19

2.5.5.21 GroupStatusGet

TCL Prototype	int GroupStatusGet (int SocketID, char GroupName [250], int * Status)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name
Output parameters	int* : Status
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupStatusGet (int SocketID, char GroupName [250], int * Status)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name
Output parameters	int* : Status
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Reads the XY group status (See § “Group state list”)
API Errors	0 -7 -8 -9 -19

2.5.5.22 XYLineArcVerification

TCL Prototype	int XYLineArcVerification (int SocketID, char GroupName[250], char *FileName)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name char [250] : Trajectory file name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int XYLineArcVerification (int SocketID, char GroupName[250], char *FileName)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name char [250] : Trajectory file name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	XY LineArc trajectory file verification. The result is to be read with the “XYLineArcVerificationResultGet” API. This API is independent from the XYLineArcExecution API, so the user needs not execute this API before executing the XYLineArcExecution. The trajectory file must be stocked in the “\ADMIN\Public\Trajectory” controller directory. <u>NOTE :</u> This API is recommended before to execute a trajectory to know the velocity and the acceleration.
API Errors	0 -3 -7 -8 -9 -14 -19 -22 -61 -62 -63 -64 -65 -66

2.5.5.23 XYLineArcVerificationResultGet

TCL Prototype	int XYLineArcVerificationResultGet (int SocketID, char FullPositionerName[250], char *FileName, double *MinimumPosition, double *MaximumPosition, double *MaximumVelocity, double *MaximumAcceleration)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : Positioner name from an XY group name
Output parameters	char * : Examined trajectory file name double * : Minimum position (units) double * : Maximum position (units) double * : Maximum trajectory velocity (units / seconds) double * : Maximum trajectory acceleration (units / seconds ²)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int XYLineArcVerificationResultGet (int SocketID, char FullPositionerName[250], char *FileName, double *MinimumPosition, double *MaximumPosition, double *MaximumVelocity, double *MaximumAcceleration)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : Positioner name from an XY group name
Output parameters	char * : Examined trajectory file name double * : Minimum position (units) double * : Maximum position (units) double * : Maximum trajectory velocity (units / seconds) double * : Maximum trajectory acceleration (units / seconds ²)
Return	API error

API Input tests	Verify the positioner name.
API Description	This API used to get results previously calculated by “XYLineArcVerification” API, positioner by positioner. The results is the position limits (min and max values), max trajectory curved velocity, max trajectory curved acceleration. This API doesn't work if not any trajectory file has not previously been examined.
API Errors	0 -3 -7 -8 -9 -13 -14 -18 -22

2.5.5.24 XYLineArcExecution

TCL Prototype	int XYLineArcExecution (int SocketID, char GroupName[250], char *FileName , double Velocity, double Acceleration, int ExecutionNumber)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name char [250] : Trajectory file name double : Trajectory velocity (units / seconds) double : Trajectory acceleration (units / seconds ²) int : Number of times of execution
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int XYLineArcExecution (int SocketID, char GroupName[250], char *FileName , double Velocity, double Acceleration, int ExecutionNumber)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name char [250] : Trajectory file name double : Trajectory velocity (units / seconds) double : Trajectory acceleration (units / seconds ²) int : Number of times of execution
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Execute an XY LineArc trajectory from a trajectory data file. The trajectory file must be stocked in the “\ADMIN\Public\Trajectory” controller directory. Before to execute a trajectory, verify if the velocity and the acceleration with the “XYLineArcVerification” and “XYLineArcVerificationResultGet” APIs.
API Errors	0 -3 -7 -8 -9 -14 -15 -17 -19 -22 -68 -69 -71 -72

2.5.5.25 XYLineArcParametersGet

TCL Prototype	int XYLineArcParametersGet (int SocketID, char GroupName[250], char *FileName, double *Velocity, double *Acceleration, int *CurrentElementNumber)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name
Output parameters	char * : Executing trajectory file name double * : Trajectory velocity (units / seconds) double * : Trajectory acceleration (units / seconds ²) int * : Current executing element number
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int XYLineArcParametersGet (int SocketID, char GroupName[250], char *FileName, double *Velocity, double *Acceleration, int *CurrentElementNumber)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XY group name
Output parameters	char * : Executing trajectory file name double * : Trajectory velocity (units / seconds) double * : Trajectory acceleration (units / seconds ²) int * : Current executing element number
Return	API error

API Input tests	Verify the number of parameters. Verify the group name.
API Description	Get the XY LineArc trajectory parameters (trajectory name, trajectory velocity, trajectory acceleration, current executing element number). This API works only during the period of trajectory execution.
API Errors	0 -7 -8 -9 -13 -14 -15 -19 -22

2.5.6 LineArc trajectory file

2.5.6.1 Trajectory file description

A segment of a trajectory that can be defined by a simple geometric shape, in our case a line or an arc of circle.

The **first** line allows to set the “FirstTangent” : Define the tangent angle for the first point

The **Second** line allows to set the “DiscontinuityAngle” : Define the maximum allowed angle of discontinuity

The **third** line is empty

The **other** lines allow to define the XY LineArc trajectory : Each line defines an element of the trajectory.

An element can be “Line” or “Arc” :

Line: Define X and Y position for a line segment = $f(X,Y)$

Arc: Define radius and sweep angle to build an arc of circle = $f(Radius,SweepAngle)$

2.5.6.2 Trajectory file example

The XY trajectory file must be from the “\ADMIN\Public\Trajectory” controller directory. File example :

```
FirstTangent = 0      ; Degrees
DiscontinuityAngle = 1 ; Degrees

Line = 10,0
Arc = 10,90
Line = 20,20
Arc = 10,90
Line = 0,30
Arc = 10,90
Line = -10,10
Arc = 10,90
```

With this file, the control points if the group axes position at the moment of trajectory execution is (0, 0) are:

X	Y	
0		(starting point)
10	0	(first element)
20	10	(2 nd element)
20	20	(3 rd element)
10	30	(4 th element)
0	30	(5 th element)
-10	20	(6 th element)
-10	10	(7 th element)
0	0	(8 th element, ending point)

2.3.7 Configuration file

Example of configuration file for a XY group named “XY” composed of two positioners named “X” and “Y”.
System.ini file :

System.ini file :

```
[GROUPS]
XYInUse = XY

[XY] ; XY group configuration
PositionerInUse = X, Y
InitializationAndHomeSearchSequence = Together ; Together, XThenY

;--- Mapping XY
XMappingFileName =
XMappingLineNumber =
XMappingColumnNumber =
XMappingMaxPositionError= } If XMappingFileName is defined
                           ; must be same unit as positioner
YMappingFileName =
YMappingLineNumber =
YMappingColumnNumber =
YMappingMaxPositionError= } If YMappingFileName is defined
                           ; must be same unit as positioner

[XY.X]
PlugNumber = 1
StageName = MYSTAGE1

[XY.X]
PlugNumber = 2
StageName = MYSTAGE2
```

Stages.ini file :

```
[MYSTAGE1]
MYSTAGE positioner configuration => See § “Positioner : Configuration parameters”

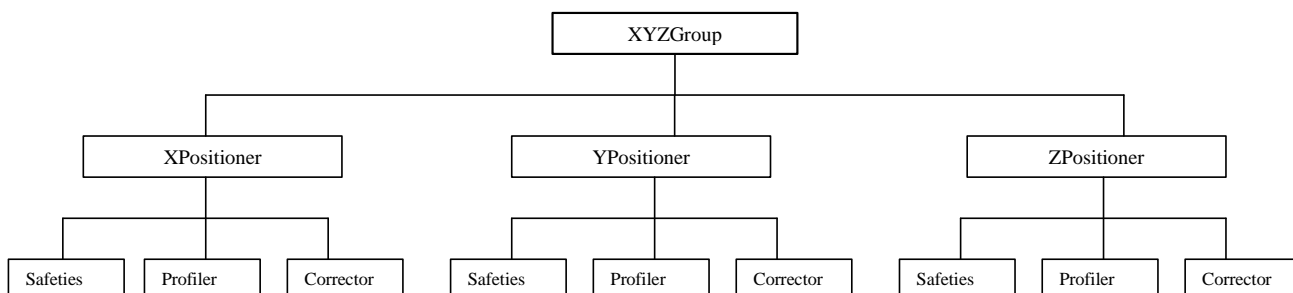
[MYSTAGE2]
MYSTAGE positioner configuration => See § “Positioner : Configuration parameters”
```

2.6 XYZ group

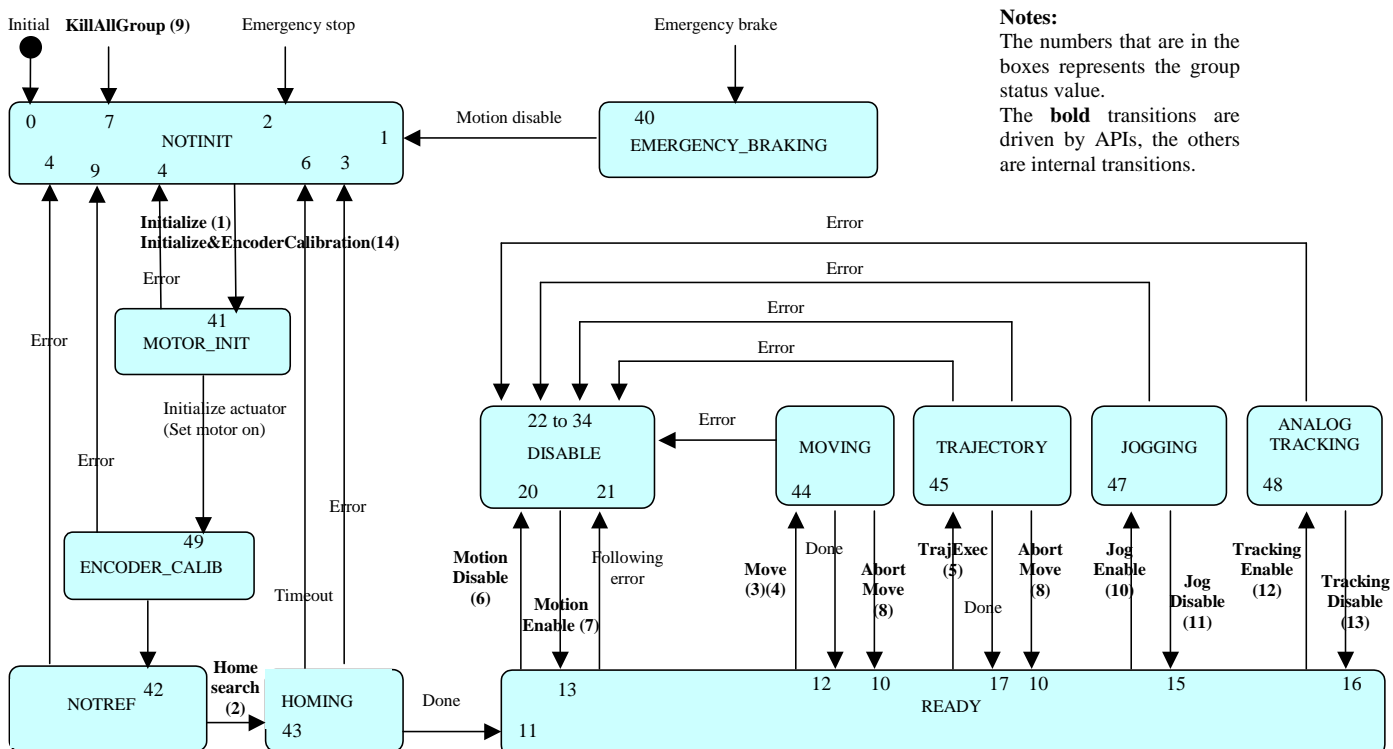
2.6.1 Description

XYZ is a combination of three positioner objects. It includes the spline trajectory (3D).

2.6.2 Structure



2.6.3 State diagram



Called API:

- | | | | |
|-----------------------|------------------------|------------------------------------|--|
| (1) GroupInitialize | (5) XYZSplineExecution | (9) GroupKill or KillAll | (13) GroupAnalogTrackingModeDisable |
| (2) GroupHomeSearch | (6) GroupMotionDisable | (10) GroupJogModeEnable | (14) GroupInitializeWithEncoderCalibration |
| (3) GroupMoveAbsolute | (7) GroupMotionEnable | (11) GroupJogModeDisable | |
| (4) GroupMoveRelative | (8) GroupMoveAbort | (12) GroupAnalogTrackingModeEnable | |

2.6.4 XYZ Mapping

XYZ mapping will be activated if the files specified in the parameters **X(Y or Z)MappingFileName** are found and coherent.

```
XMappingFileName
YMappingFileName
ZMappingFileName
```

Mapping parameters are used to verify the coherence of the file :

```
XMappingXLineNumber
XMappingYColumnNumber
XMappingZDimNumber
XMappingMaxPositionError
YMappingXLineNumber
YMappingYColumnNumber
YMappingZDimNumber
YMappingMaxPositionError
ZMappingXLineNumber
ZMappingYColumnNumber
ZMappingZDimNumber
ZMappingMaxPositionError
```

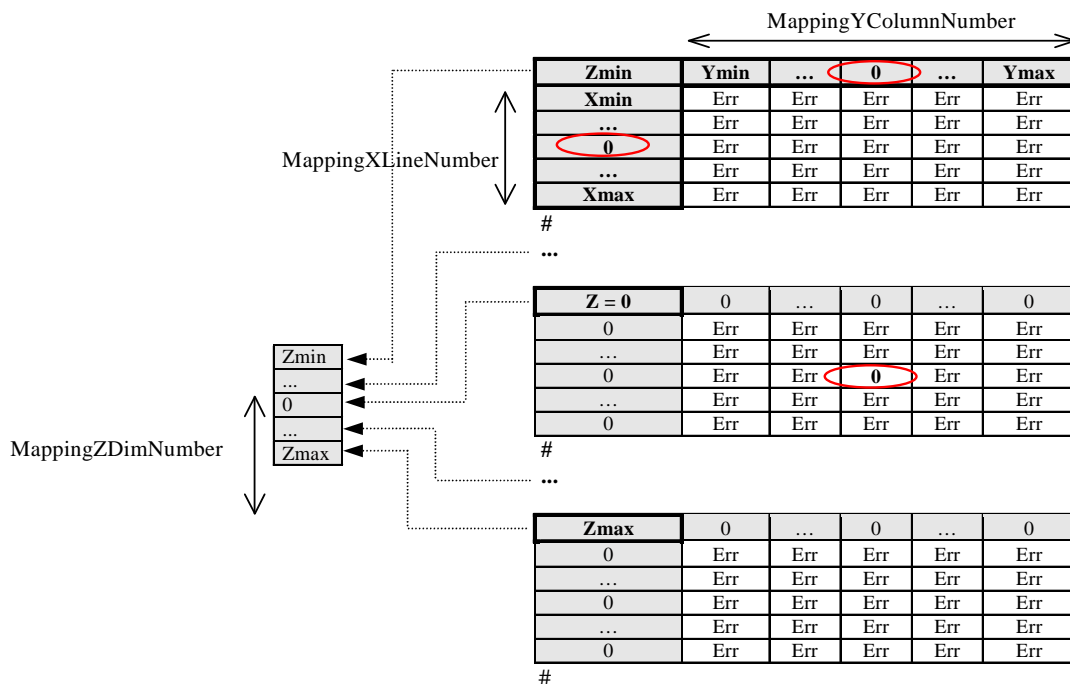
The mapping files must be in the “\ADMIN\CONFIG” directory.

For each X, Y and Z positioner, a mapping file is built as following :

First cell of each table = Z Position
First column of the first table = X Position
First row of the first table = Y Position
Each other cell = corresponding error

NOTES :

Error in X = Y = Z = 0 must be 0. This value in the file correspond to the HomePreset positions in the XY group referential. **X, Y and Z positions must at least cover the entire travel of the XYZ group.** A terminator (#) must be added at end of each matrix.



2.6.5 API description

2.6.5.1 GroupAnalogTrackingModeDisable

TCL Prototype	int GroupAnalogTrackingModeDisable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupAnalogTrackingModeDisable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name and the parameters coherence test.
API Description	Disable the tracking mode : Exit ANALOG TRACKING status to come back in READY status. The group must be in ANALOG TRACKING status.
API Errors	0 -7 -8 -9 -19 -22

2.6.5.2 GroupAnalogTrackingModeEnable

TCL Prototype	int GroupAnalogTrackingModeEnable (int SocketID, char GroupName [250], char Type [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name char [250] : Type (“Position” or “Velocity”)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupAnalogTrackingModeEnable (int SocketID, char GroupName [250], char Type [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name char [250] : Type (“Position” or “Velocity”)
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name.
API Description	The group must be in READY status (after initialization and home search) and if OK the group is setting in ANALOG TRACKING mode. If type is “Position”, the analog input is interpreted as an axis position command and the parameters are settled by the “AnalogTrackingPositionParametersSet” API and can be read by the “AnalogTrackingPositionParametersGet” API. If type is “velocity”, the analog input is interpreted as axis velocity command and the parameters are settled by the “AnalogTrackingVelocityParametersSet” API and can be read by the “AnalogTrackingVelocityParametersSet” API.
API Errors	0 -7 -8 -9 -13 -19 -22

2.6.5.3 GroupHomeSearch

TCL Prototype	int GroupHomeSearch (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupHomeSearch (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name and the parameters coherence test.
API Description	Activates the selected home search, configured in the Stages.ini, for the selected XYZ group. HomeSearchProfileGeneratorType = <ul style="list-style-type: none"> ✓ IndexHomeSearch ✓ CurrentPositionAsHome ✓ MechanicalZeroAndIndexHomeSearch ✓ MechanicalZeroHomeSearch ✓ MinusEndOfRunAndIndexHomeSearch ✓ MinusEndOfRunHomeSearch
API Errors	0 -7 -8 -9 -19 -22 -25 -28 -31 -33 -45
System.ini	InitializationAndHomeSearchSequence = <i>Together</i> or <i>XthenYThenZ</i>

2.6.5.4 GroupHomeSearchAndRelativeMove

TCL Prototype	int GroupHomeSearchAndRelativeMove (int SocketID, char GroupName [250], double displacementX, double displacementY, double displacementZ)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name double : displacementX double : displacementY double : displacementZ
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupHomeSearchAndRelativeMove (int SocketID, char GroupName [250], int NbPositioners, double displacement[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name int : Number of Positioners in the group (must be 3 for an XYZ group) double[3] : displacement array (units) with array[0] = X, array[1] = Y, array[2] = Z
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name and the parameters coherence test.
API Description	Activates the selected home search, configured in the Stages.ini, for the selected single axis and execute a relative motion at end of the home search. HomeSearchProfileGeneratorType = <ul style="list-style-type: none"> ✓ IndexHomeSearch ✓ CurrentPositionAsHome ✓ MechanicalZeroAndIndexHomeSearch ✓ MechanicalZeroHomeSearch ✓ MinusEndOfRunAndIndexHomeSearch ✓ MinusEndOfRunHomeSearch
API Errors	0 -7 -8 -9 -19 -21 -22 -25 -27 -28 -31 -33 -45

2.6.5.5 GroupInitialize

TCL Prototype	int GroupInitialize (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupInitialize (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name.
API Description	Initialize the motors and activate the servo loop of each positioner of the selected XYZ group. NOTE : In Master-Slave case, after an emergency signal, the master group and the slave group are in “Not Initialized” status. To reinitialise these groups, the process is the following : the slave(s) group must be reinitialised before the master group.
API Errors	0 -7 -8 -9 -19 -22 -24

2.6.5.6 GroupInitializeWithEncoderCalibration

TCL Prototype	int GroupInitializeWithEncoderCalibration (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupInitializeWithEncoderCalibration (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name.
API Description	Initialize the motors, calibrate the encoders and activate the servo loop of each positioner of the selected XYZ group. To get the calibration results for each positioner, use the PositionerEncoderCalibrationParametersGet API.
API Errors	0 -7 -8 -9 -19 -22

2.6.5.7 GroupJogModeDisable

TCL Prototype	int GroupJogModeDisable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupJogModeDisable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name.
API Description	The group must be in JOGGING status and the positioner must be idle (velocity NULL)
API Errors	0 -7 -8 -9 -19 -22

2.6.5.8 GroupJogModeEnable

TCL Prototype	int GroupJogModeEnable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

TCL Prototype	int GroupJogModeEnable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name.
API Description	The group must be in ready status (after initialization and home search) and if OK the group is setting in JOG mode (JOGGING status).
API Errors	0 -7 -8 -9 -19 -22

2.6.5.9 GroupJogCurrentGet

TCL Prototype	int GroupJogCurrentGet (int SocketID, char GroupName [250], double *VelocityX, double *AccelerationX, double *VelocityY, double *AccelerationY, double *VelocityZ, double *AccelerationZ)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name
Output parameters	double * : Velocity (units / s) double * : Acceleration (units / s ²) double * : VelocityY (units / s) double * : AccelerationY (units / s ²) double * : VelocityZ (units / s) double * : AccelerationZ (units / s ²)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupJogCurrentGet (int SocketID, char GroupName [250], int NbPositioners, double Velocity[], double Acceleration[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name int : Number of Positioners in the group (must be 3 for an XYZ group)
Output parameters	double [3] : Velocity array (units / s) with [0] = X, [1] = Y, [2] = Z double [3] : Acceleration array (units / s ²) with [0] = X, [1] = Y, [2] = Z
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner name. Parameters coherence test.
API Description	This API returns the current velocity and the current acceleration used by JOG mode.
API Errors	0 -7 -8 -9 -14 -19

2.6.5.10 GroupJogParametersGet

TCL Prototype	int GroupJogParametersGet (int SocketID, char GroupName [250], double *VelocityX, double *AccelerationX, double *VelocityY, double *AccelerationY, double *VelocityZ, double *AccelerationZ)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name (can be positioner name : see § positioner API)
Output parameters	double * : VelocityX (units / s) double * : AccelerationX (units / s ²) double * : VelocityY (units / s) double * : AccelerationY (units / s ²) double * : VelocityZ (units / s) double * : AccelerationZ (units / s ²)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupJogParametersGet (int SocketID, char GroupName [250], int NbPositioners, double Velocity[], double Acceleration[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name (can be positioner name : see § positioner API) int : Number of Positioners in the group (must be 3 for an XYZ group)
Output parameters	double [3] : Velocity array (units / s) with [0] => X, [1] => Y, [2] => Z double [3] : Acceleration array (units / s ²) with [0] => X, [1] => Y, [2] => Z
Return	API error

API Input tests	Verify the number of parameters. Verify the group name or the positioner name. Parameters coherence test.
API Description	This API returns the user velocity and the user acceleration changed by the “GroupJogParametersSet”.
API Errors	0 -7 -9 -14 -19

2.6.5.11 GroupJogParametersSet

TCL Prototype	int GroupJogParametersSet (int SocketID, char GroupName [250], double VelocityX, double AccelerationX, double VelocityY, double AccelerationY, double VelocityZ, double AccelerationZ)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : XYZ group name (can be positioner name : see § positioner API) double : VelocityX (units / s) double : AccelerationX (units / s ²) double : VelocityY (units / s) double : AccelerationY (units / s ²) double : VelocityZ (units / s) double : AccelerationZ (units / s ²)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupJogParametersSet (int SocketID, char GroupName [250], int NbPositioners, double Velocity[], double Acceleration[])
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : XYZ group name (can be positioner name : see § positioner API) int : Number of Positioners in the group (must be 3 for an XYZ group) double [3] : Velocity array (units / s) with [0] => X, [1] => Y, [2] => Z double [3] : Acceleration array (units / s ²) with [0] => X, [1] => Y, [2] => Z
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name or the positioner name. Parameters coherence test for each positioner (X, Y and Z) : Velocity > MaximumVelocity => Velocity = MaximumVelocity Velocity < - MaximumVelocity => Velocity = - MaximumVelocity Acceleration ≤ 0 => ERROR and stop motion Acceleration > MaximumAcceleration => Acceleration = MaximumAcc.
API Description	The JOG mode must be activated (after the call of "GroupJogModeEnable" API) This API allows to change on fly the velocity and the acceleration used by the JOG mode on each positioner. If an error is occurred, each positioner is stopped with a velocity NULL.
API Errors	0 -7 -8 -9 -14 -19 -22 -42

2.6.5.12 GroupKill

TCL Prototype	int GroupKill (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupKill (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Kill and reset XYZ group. The XYZ group comes back to “not initialised” status.
API Errors	0 -7 -8 -9 -26

2.6.5.13 GroupMotionDisable

TCL Prototype	int GroupMotionDisable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupMotionDisable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Turn motor OFF and set the “MotionEnable” status to FALSE for the selected XYZ group.
API Errors	0 -7 -8 -9 -19 -22

2.6.5.14 GroupMotionEnable

TCL Prototype	int GroupMotionEnable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupMotionEnable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Initializes positions and corrector before to turn motor ON. Set the “MotionEnable” status to TRUE for the selected XYZ group.
API Errors	0 -7 -8 -9 -19 -22

2.6.5.15 GroupMoveAbort

TCL Prototype	int GroupMoveAbort (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupMoveAbort (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name
Output parameters	None
Return	API error

API Input tests	Verify number of parameters. Verify group name. Parameters coherence test.
API Description	Aborts motion : stops the motion in progress on the selected XYZ group.
API Errors	0 -7 -8 -9 -19 -22 -27

2.6.5.16 GroupMoveAbsolute

TCL Prototype	int GroupMoveAbsolute (int SocketID, char GroupName [250], double TargetPositionX, double TargetPositionY, double TargetPositionZ)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name (can be positioner name : see § positioner API) double : TargetPositionX (units) double : TargetPositionY (units) double : TargetPositionZ (units)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupMoveAbsolute (int SocketID, char GroupName [250], int NbPositioners, double TargetPosition[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name (can be positioner name : see § positioner API) int : Number of Positioners in the group (must be 3 for an XYZ group) double [3] : TargetPosition array (units) with [0] => X, [1] => Y, [2] => Z
Output parameters	None
Return	API error

API Input tests	Verify number of parameters. Verify group name. Parameters coherence test. Verify target position in relation with the travel limits : <div style="margin-left: 40px;"> TargetPositionX ≥ MinimumTargetPositionX TargetPositionX ≤ MaximumTargetPositionX TargetPositionY ≥ MinimumTargetPositionY TargetPositionY ≤ MaximumTargetPositionY TargetPositionZ ≥ MinimumTargetPositionZ TargetPositionZ ≤ MaximumTargetPositionZ </div>
API Description	The three axes from the selected XYZ group moves to their target position. The absolute move refers to the acceleration, velocity, minimumTjerkTime and maximumTjerkTime predefined in the “Stages.ini” or redefined with the “PositionerSGammaParametersSet” API.
API Errors	0 -7 -8 -9 -14 -17 -19 -22 -25 -33 -44

2.6.5.17 GroupMoveRelative

TCL Prototype	int GroupMoveRelative (int SocketID, char GroupName [250] , double TargetDisplacementX, double TargetDisplacementY, double TargetDisplacementZ)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name (can be positioner name : see § positioner API) double : TargetDisplacementX (units) double : TargetDisplacementY (units) double : TargetDisplacementZ (units)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupMoveRelative (int SocketID, char GroupName [250] , int NbPositioners, double TargetDisplacement[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name (can be positioner name : see § positioner API) int : Number of Positioners in the group (must be 3 for an XYZ group) double [3] : TargetDisplacement array (units) with [0] => X, [1] => Y, [2] => Z
Output parameters	None
Return	API error

API Input tests	<p>Verify the number of parameters. Verify the group name. Parameters coherence test. Test the recalculated target position in relation with the travel limits :</p> <div style="margin-left: 40px;"> TargetPositionX ≥ MinimumTargetPositionX TargetPositionX ≤ MaximumTargetPositionX TargetPositionY ≥ MinimumTargetPositionY TargetPositionY ≤ MaximumTargetPositionY TargetPositionZ ≥ MinimumTargetPositionZ TargetPositionZ ≤ MaximumTargetPositionZ </div>
API Description	<p>The three positioner from XYZ group moves to their target position :</p> <p>TargetPositionX = CurrentPositionX + TargetDisplacementX TargetPositionY = CurrentPositionY + TargetDisplacementY TargetPositionZ = CurrentPositionZ + TargetDisplacementZ</p> <p>The relative move refers to the acceleration, velocity, minimumTjerkTime and maximumTjerkTime predefined in the “Stages.ini” or redefined with the “PositionerSGammaParametersSet” API.</p>
API Errors	0 -7 -8 -9 -14 -17 -19 -22 -25 -33 -44

2.6.5.18 GroupPositionCurrentGet

TCL Prototype	int GroupPositionCurrentGet (int SocketID, char GroupName [250] , double * CurrentPositionX, double * CurrentPositionY, double * CurrentPositionZ)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name (can be positioner name : see § positioner API)
Output parameters	double * : CurrentPositionX (units) double * : CurrentPositionY (units) double * : CurrentPositionZ (units)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupPositionCurrentGet (int SocketID, char GroupName [250] , int NbPositioners, double CurrentPosition[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name (can be positioner name : see § positioner API) int : Number of Positioners in the group (must be 3 for an XYZ group)
Output parameters	double [3] : CurrentPosition array (units) with [0] => X, [1] => Y, [2] => Z
Return	API error

API Input tests	Verify number of parameters. Verify group name. Parameters coherence test.
API Description	Reads the X, Y, Z current positions (SetpointPosition - PositionError).
API Errors	0 -7 -8 -9 -14 -19

2.6.5.19 GroupPositionSetpointGet

TCL Prototype	int GroupPositionSetpointGet (int SocketID, char GroupName [250], double * SetPointPositionX, double * SetPointPositionY, double * SetPointPositionZ)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name (can be positioner name : see § positioner API)
Output parameters	double * : SetPointPositionX (units) double * : SetPointPositionY (units) double * : SetPointPositionZ (units)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupPositionSetpointGet (int SocketID, char GroupName [250], int NbPositioners, double SetPointPosition[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name (can be positioner name : see § positioner API) int : Number of Positioners in the group (must be 3 for an XYZ group)
Output parameters	double [3] : SetPointPosition array (units) with [0] => X, [1] => Y, [2] => Z
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Read the three positioner Setpoint positions (profiler position) from XYZ group.
API Errors	0 -7 -8 -9 -14 -19

2.6.5.20 GroupPositionTargetGet

TCL Prototype	int GroupPositionTargetGet (int SocketID, char GroupName [250], double * TargetPositionX, double * TargetPositionY, double * TargetPositionZ)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name (can be positioner name : see § positioner API)
Output parameters	double * : TargetPositionX (units) double * : TargetPositionY (units) double * : TargetPositionZ (units)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupPositionTargetGet (int SocketID, char GroupName [250], int NbPositioners, double TargetPosition[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name (can be positioner name : see § positioner API) int : Number of Positioners in the group (must be 3 for an XYZ group)
Output parameters	double [3] : TargetPosition array (units) with [0] => X, [1] => Y, [2] => Z
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Read the three positioner Target positions (user position) from XYZ group.
API Errors	0 -7 -8 -9 -14 -19

2.6.5.21 GroupStatusGet

TCL Prototype	int GroupStatusGet (int SocketID, char GroupName [250], int * Status)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name
Output parameters	int * : Status
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupStatusGet (int SocketID, char GroupName [250], int * Status)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name
Output parameters	int * : Status
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Reads the XYZ group status (See § “Group state list”)
API Errors	0 -7 -8 -9 -15 -19

2.6.5.22 XYZSplineVerification

TCL Prototype	int XYZSplineVerification (int SocketID, char GroupName[250], char *FileName)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name char [250] : Trajectory file name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int XYZSplineVerification (int SocketID, char GroupName[250], char *FileName)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name char [250] : Trajectory file name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	<p>XYZ Spline trajectory file verification. The result is to be read with the API “XYZSplineVerificationResultGet”.</p> <p>This API is independent from the XYZSplineExecution API, so the user need not execute this API before executing the XYZSplineExecution.</p> <p>The trajectory file must be stocked in the “\ADMIN\Public\Trajectory” controller directory.</p> <p><u>NOTE :</u> This API is recommended before to execute a trajectory to know the velocity and the acceleration.</p>
API Errors	0 -3 -7 -8 -9 -14 -19 -22 -61 -66

2.6.5.23 XYZSplineVerificationResultGet

TCL Prototype	int XYZSplineVerificationResultGet (int SocketID, char FullPositionerName[250], char *FileName, double *MinimumPosition, double *MaximumPosition, double *MaximumVelocity, double *MaximumAcceleration)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName from an XYZ group name
Output parameters	char * : Examined trajectory file name double * : Minimum position (units) double * : Maximum position (units) double * : Maximum trajectory velocity (units / seconds) double * : Maximum trajectory acceleration (units / seconds ²)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int XYZSplineVerificationResultGet (int SocketID, char FullPositionerName[250], char *FileName, double *MinimumPosition, double *MaximumPosition, double *MaximumVelocity, double *MaximumAcceleration)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName from an XYZ group name
Output parameters	char * : Examined trajectory file name double * : Minimum position (units) double * : Maximum position (units) double * : Maximum trajectory velocity (units / seconds) double * : Maximum trajectory acceleration (units / seconds ²)
Return	API error

API Input tests	Verify the positioner name.
API Description	This API used to get results previously calculated by XYZSplineVerification API, positioner by positioner. The result is the position limits (min and max values), max trajectory curved velocity, max trajectory curved acceleration. This API doesn't work if not any trajectory file has not previously been examined.
API Errors	0 -3 -7 -8 -9 -13 -14 -18 -22

2.6.5.24 XYZSplineExecution

TCL Prototype	int XYZSplineExecution (int SocketID, char GroupName[250], char *FileName, double Velocity, double Acceleration)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name char [250] : Trajectory file name double : Trajectory velocity (units / seconds) double : Trajectory acceleration (units / seconds ²)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int XYZSplineExecution (int SocketID, char GroupName[250], char *FileName, double Velocity, double Acceleration)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name char [250] : Trajectory file name double : Trajectory velocity (units / seconds) double : Trajectory acceleration (units / seconds ²)
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Execute an XYZ Spline trajectory from a trajectory data file. The trajectory file must be stocked in the “\ADMIN\Public\Trajectory” controller directory. Before to execute a trajectory, verify if the velocity and the acceleration with the “XYZSplineVerification” and “XYZSplineVerificationResultGet” APIs.
API Errors	0 -3 -7 -8 -9 -17 -19 -22 -68 -69 -71 -72

2.6.5.25 XYZSplineParametersGet

TCL Prototype	int XYZSplineParametersGet (int SocketID, char GroupName[250], char *FileName, double *Velocity, double *Acceleration, int *CurrentElementNumber)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name
Output parameters	char * : Executing trajectory file name double * : Trajectory velocity (units / seconds) double * : Trajectory acceleration (units / seconds ²) int * : Current executing element number
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int XYZSplineParametersGet (int SocketID, char GroupName[250], char *FileName, double *Velocity, double *Acceleration, int *CurrentElementNumber)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : XYZ group name
Output parameters	char * : Executing trajectory file name double * : Trajectory velocity (units / seconds) double * : Trajectory acceleration (units / seconds ²) int * : Current executing element number
Return	API error

API Input tests	Verify the number of parameters. Verify the group name.
API Description	Get the XYZ Spline trajectory parameters (trajectory name, trajectory velocity, trajectory acceleration, current executing element number). This API works only during the period of trajectory execution.
API Errors	0 -7 -8 -9 -13 -14 -15 -19 -22

2.6.6 Spline trajectory file

2.6.6.1 Trajectory file description

The Spline trajectory file must be read from the “\Public\Trajectory” directory.

The Spline trajectory is defined for a XYZ group and is described in a file. Each line of this file represents a point of the spline trajectory. Two consecutive points forms a spline element.

The line format is described as a set of 3 columns:

Column #1: Element X => X-axis position (units)
 Column #2: Element Y => Y-axis position (units)
 Column #3: Element Z => Z-axis position (units)

The separator is the comma.

For the XPS controller, the first and last lines are used only for the starting and ending elements (segments) spline interpolation, and not used to move. So the trajectory start at the data file second line and ends at the data file before-ending line (line N –1 if the file has N lines).

The position values in the data file are absolute values, but relative to the controlled axes position at the moment of trajectory execution. In addition, if the file second line (starting point) position values are not equal to zeros, the real trajectory is shifted of these values.

Example: We have a spline trajectory data file that forms the following data matrix:

x0	y0	z0
x1	y1	z1
x2	y2	z2
x3	y3	z3
x4	y4	z4
...

At the moment of trajectory execution, the controlled axes position is X_c , Y_c , Z_c .

The real interpolated matrix will be:

$X_c + x_0 - x_1$	$Y_c + y_0 - y_1$	$Z_c + z_0 - z_1$
X_c	Y_c	Z_c
$X_c + x_2 - x_1$	$Y_c + y_2 - y_1$	$Z_c + z_2 - z_1$
$X_c + x_3 - x_1$	$Y_c + y_3 - y_1$	$Z_c + z_3 - z_1$
$X_c + x_4 - x_1$	$Y_c + y_4 - y_1$	$Z_c + z_4 - z_1$
...

2.6.6.2 Spline trajectory data file example

This trajectory presents a circle from (0,1,0) starting point to (0,1,0) ending point. The Catmull-Rom spline algorithm uses four consecutive points (ex. ABCD) for the middle (BC) segment spline calculation. So the first (-0.5,0.866,0) and last (0.5,0.866,0) points are used only for the starting and ending segments spline calculation, and not used to move) :

The file original data are :

```
-0.5,0.866,0
0,1.0,0
0.5,0.866,0
0.866,0.5,0
1.0,0,0
0.866,-0.5,0
0.5,-0.866,0
0,-1.0,0
-0.5,-0.866,0
-0.866,-0.5,0
-1.0,0,0
-0.866,0.5,0
-0.5,0.866,0
0,1.0,0
0.5,0.866,0
```

With this data file, the real control points if the group axes position at the moment of trajectory execution is (0, 0, 0) are :

-0.5	-0.134	0
0	0	0
0.5	-0.134	0
0.866	-0.5	0
1.0	-1.0	0
0.866	-1.5	0
0.5	-1.866	0
0	-2.0	0
-0.5	-1.866	0
-0.866	-1.5	0
-1.0	-1.0	0
-0.866	-0.5	0
-0.5	-0.134	0
0	0	0
0.5	-0.134	0

2.6.7 Configuration file

Example of configuration file for a XYZ group named “XYZ” composed of three positioners named “X”, “Y” and “Z”.

System.ini file :

```
[GROUPS]
XYZInUse = XYZ

[XYZ]           ; XYZ group configuration
PositionerInUse = X, Y, Z
InitializationAndHomeSearchSequence = Together ; Together or XThenYThenZ

;--- Mapping XYZ
XMappingFileName = MatriceX.txt
XMappingXLineNumber = 5
XMappingYColumnNumber = 3
XMappingZDimNumber = 3
XMappingMaxPositionError = 2 ; must be same unit as positioner
YMappingFileName = MatriceY.txt
YMappingXLineNumber = 5
YMappingYColumnNumber = 3
YMappingZDimNumber = 3
YMappingMaxPositionError = 2 ; must be same unit as positioner
ZMappingFileName = MatriceZ.txt
ZMappingXLineNumber = 5
ZMappingYColumnNumber = 3
ZMappingZDimNumber = 3
ZMappingMaxPositionError = 2 ; must be same unit as positioner

[XYZ.X]
StageName = MYSTAGE1
X positioner configuration => See § “Positioner : Configuration parameters”

[XYZ.Y]
StageName = MYSTAGE2
X positioner configuration => See § “Positioner : Configuration parameters”

[XYZ.Z]
StageName = MYSTAGE3
X positioner configuration => See § “Positioner : Configuration parameters”
```

Stages.ini file :

[MYSTAGE1]

MYSTAGE1 configuration => See § “Positioner : Configuration parameters”

[MYSTAGE2]

MYSTAGE2 configuration => See § “Positioner : Configuration parameters”

[MYSTAGE3]

MYSTAGE3 configuration => See § “Positioner : Configuration parameters”

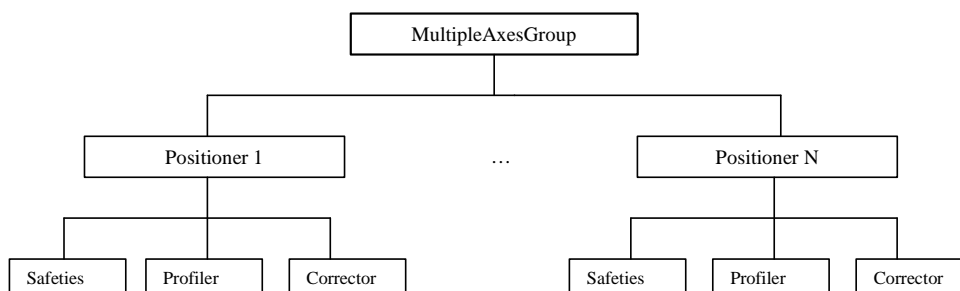
2.7 MultipleAxes group

2.7.1 Description

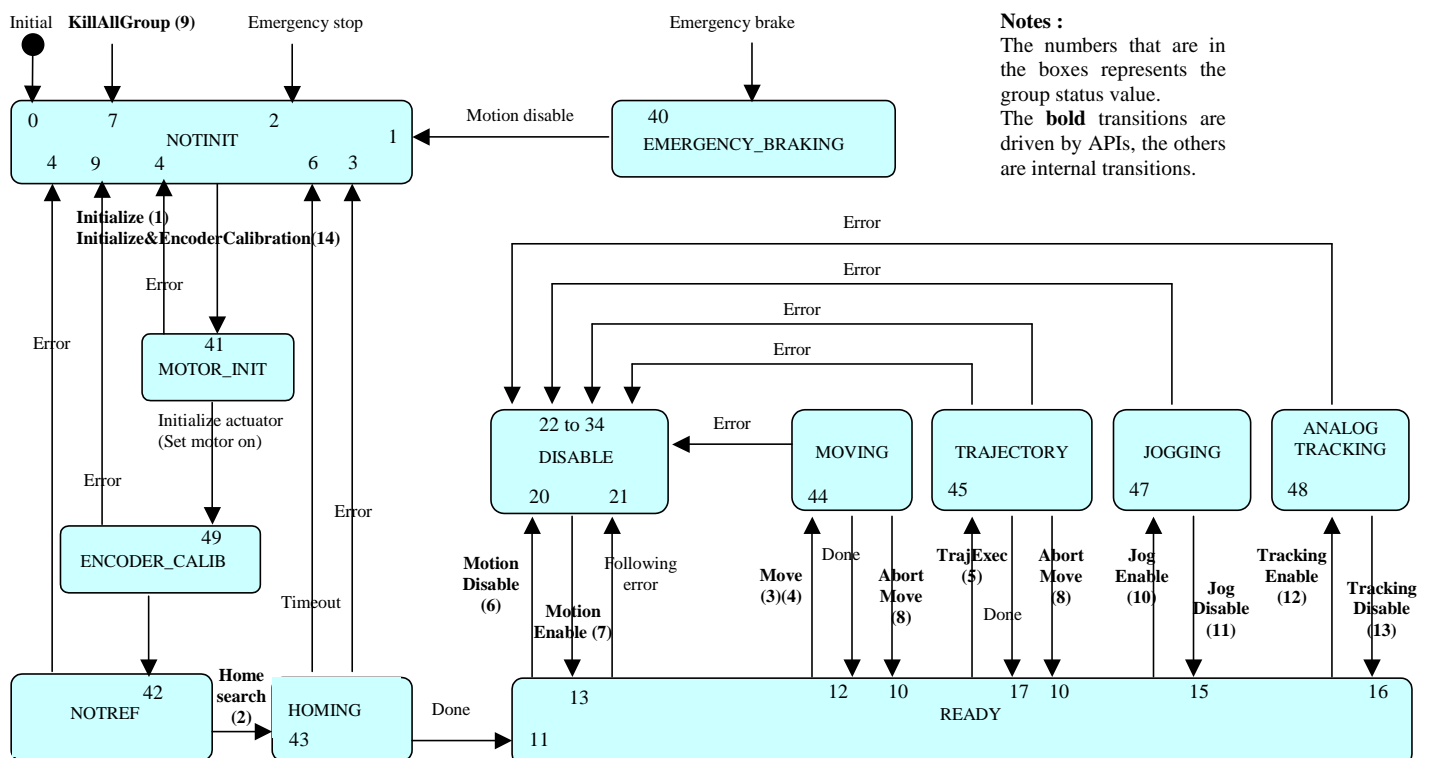
MultipleAxes is a combination of several positioner objects (positioner objects number \leq controller axes number)

It includes the PVT trajectory.

2.7.2 Structure



2.7.3 State diagram



Notes :

The numbers that are in the boxes represents the group status value. The **bold** transitions are driven by APIs, the others are internal transitions.

Called API:

- | | | | |
|-----------------------|-------------------------------|------------------------------------|--|
| (1) GroupInitialize | (5) MultipleAxesPVTEExecution | (9) GroupKill or KillAll | (13) GroupAnalogTrackingModeDisable |
| (2) GroupHomeSearch | (6) GroupMotionDisable | (10) GroupJogModeEnable | (14) GroupInitializeWithEncoderCalibration |
| (3) GroupMoveAbsolute | (7) GroupMotionEnable | (11) GroupJogModeDisable | |
| (4) GroupMoveRelative | (8) GroupMoveAbort | (12) GroupAnalogTrackingModeEnable | |

2.7.4 API description

2.7.4.1 GroupAnalogTrackingModeDisable

TCL Prototype	int GroupAnalogTrackingModeDisable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupAnalogTrackingModeDisable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Disable tracking mode. The group must be in ANALOG TRACKING status. If OK, the group status becomes READY.
API Errors	0 -7 -8 -9 -19 -22

2.7.4.2 GroupAnalogTrackingModeEnable

TCL Prototype	int GroupAnalogTrackingModeEnable (int SocketID, char GroupName [250], char Type [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name char [250] : Type (“Position” or “Velocity”)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupAnalogTrackingModeEnable (int SocketID, char GroupName [250], char Type [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name char [250] : Type (“Position” or “Velocity”)
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name.
API Description	The group must be in READY status (after initialization and home search) and if OK the group status becomes ANALOG TRACKING. If type is “Position”, the analog input is interpreted as an axis position command and the parameters are settled by the “AnalogTrackingPositionParametersSet” API and can be read by the “AnalogTrackingPositionParametersGet” API. If type is “velocity”, the analog input is interpreted as axis velocity command and the parameters are settled by the “AnalogTrackingVelocityParametersSet” API and can be read by the “AnalogTrackingVelocityParametersGet” API.
API Errors	0 -7 -8 -9 -13 -19 -22

2.7.4.3 GroupHomeSearch

TCL Prototype	int GroupHomeSearch (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : MultipleAxes group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupHomeSearch (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : MultipleAxes group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Activates the selected home search, configured in the Stages.ini, for the selected MultipleAxes group. HomeSearchProfileGeneratorType = <ul style="list-style-type: none"> ✓ IndexHomeSearch ✓ CurrentPositionAsHome ✓ MechanicalZeroAndIndexHomeSearch ✓ MechanicalZeroHomeSearch ✓ MinusEndOfRunAndIndexHomeSearch ✓ MinusEndOfRunHomeSearch
API Errors	0 -7 -8 -9 -19 -22 -25 -28 -31 -33 -45
System.ini	InitializationAndHomeSearchSequence = <i>Together</i> or <i>OneAfterAnother</i>

2.7.4.4 GroupHomeSearchAndRelativeMove

TCL Prototype	int GroupHomeSearchAndRelativeMove (int SocketID, char GroupName [250], {double displacementX})
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : MultipleAxes group name double : displacement => this parameter must be repeated for each positioner of the selected MultipleAxes group
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupHomeSearchAndRelativeMove (int SocketID, char GroupName [250], int NbPositioners, double displacement[])
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : MultipleAxes group name int : Number of Positioners in the MultipleAxes group double [] : displacement array => size array defined by the number of positioners from the selected MultipleAxes group
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name and the parameters coherence test.
API Description	Activates the selected home search, configured in the Stages.ini, for the selected single axis and execute a relative motion at end of the home search. HomeSearchProfileGeneratorType = <ul style="list-style-type: none"> ✓ IndexHomeSearch ✓ CurrentPositionAsHome ✓ MechanicalZeroAndIndexHomeSearch ✓ MechanicalZeroHomeSearch ✓ MinusEndOfRunAndIndexHomeSearch ✓ MinusEndOfRunHomeSearch
API Errors	0 -7 -8 -9 -19 -21 -22 -25 -27 -28 -31 -33 -45

2.7.4.5 GroupInitialize

TCL Prototype	int GroupInitialize (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupInitialize (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name.
API Description	Initialize the motors and activate the servo loop of each positioner of the selected multiple axes group. NOTE : In Master-Slave case, after an emergency signal, the master group and the slave group are in “Not Initialized” status. To reinitialise these groups, the process is the following : the slave(s) group must be reinitialised before the master group.
API Errors	0 -7 -8 -9 -19 -22 -24

2.7.4.6 GroupInitializeWithEncoderCalibration

TCL Prototype	int GroupInitializeWithEncoderCalibration (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupInitializeWithEncoderCalibration (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name.
API Description	Initialize the motors, calibrate the encoders and activate the servo loop of each positioner of the selected multiple axes group. To get the calibration results for each positioner, use the PositionerEncoderCalibrationParametersGet API.
API Errors	0 -7 -8 -9 -19 -22

2.7.4.7 GroupJogModeDisable

TCL Prototype	int GroupJogModeDisable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupJogModeDisable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name.
API Description	Disable the Jog mode. The group must be in JOGGING status and the positioner must be idle (velocity NULL)
API Errors	0 -7 -8 -9 -19 -22

2.7.4.8 GroupJogModeEnable

TCL Prototype	int GroupJogModeEnable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupJogModeEnable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name.
API Description	The group must be in ready status (after initialization and home search) and if OK the group is setting in JOG mode (JOGGING status).
API Errors	0 -7 -8 -9 -19 -22

2.7.4.9 GroupJogCurrentGet

TCL Prototype	int GroupJogCurrentGet (int SocketID, char GroupName [250], double *Velocity, double *Acceleration, ...)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name
Output parameters	double * : Velocity (units / s) double * : Acceleration (units / s ²) } these parameters must be repeated for each positioner of the selected MultipleAxes group
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupJogCurrentGet (int SocketID, char GroupName [250], int NbPositioners, double Velocity[], double Acceleration[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name int : Number of Positioners in the MultipleAxes group
Output parameters	double [] : Velocity array (units / s) double [] : Acceleration array (units / s ²) } size array defined by the number of positioners from the selected MultipleAxes group
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner name. Parameters coherence test.
API Description	This API returns the current velocity and the current acceleration used by JOG mode.
API Errors	0 -7 -8 -9 -14 -19

2.7.4.10 GroupJogParametersGet

TCL Prototype	int GroupJogParametersGet (int SocketID, char GroupName [250], double *Velocity, double *Acceleration, ...)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char[250] : MultipleAxes group name (can be positioner name : see § positioner API)
Output parameters	double * : Velocity (units / s) double * : Acceleration (units / s ²) } these parameters must be repeated for each positioner of the selected MultipleAxes group
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupJogParametersGet (int SocketID, char GroupName [250], int NbPositioners, double Velocity[], double *Acceleration[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char[250] : MultipleAxes group name (can be positioner name : see § positioner API) int : Number of Positioners in the MultipleAxes group
Output parameters	double [] : Velocity array (units / s) double [] : Acceleration array (units / s ²) } size array defined by the number of positioners from the selected MultipleAxes group
Return	API error

API Input tests	Verify the number of parameters. Verify the group name or the positioner name. Parameters coherence test.
API Description	This API returns the user velocity and the user acceleration changed with the “GroupJogParametersSet” API.
API Errors	0 -7 -8 -9 -14 -19

2.7.4.11 GroupJogParametersSet

TCL Prototype	int GroupJogParametersSet (int SocketID, char GroupName [250], double Velocity, double Acceleration, ...)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : MultipleAxes group name (can be positioner name : see § positioner API) double : Velocity (units / s) double : Acceleration (units / s ²)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupJogParametersSet (int SocketID, char GroupName [250], int NbPositioners, double Velocity[], double Acceleration[])
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : MultipleAxes group name (can be positioner name : see § positioner API) int : Number of Positioners in the MultipleAxes group double [] : Velocity array (units / s) double [] : Acceleration array (units / s ²)
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name or the positioner name. Parameters coherence test for each positioner : Velocity > MaximumVelocity => Velocity = MaximumVelocity Velocity < - MaximumVelocity => Velocity = - MaximumVelocity Acceleration ≤ 0 => ERROR and stop motion Acceleration > MaximumAcceleration => Acceleration = MaximumAcc.
API Description	The JOG mode must be activated (after the call of "GroupJogModeEnable" API) This API allows to change on fly the velocity and the acceleration used by the JOG mode on each positioner. If an error is occurred, each positioner is stopped with a velocity NULL.
API Errors	0 -7 -8 -9 -19 -22 -42

2.7.4.12 GroupKill

TCL Prototype	int GroupKill (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupKill (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Kills and resets the MultipleAxes group. The MultipleAxes comes back to “not initialised” status.
API Errors	0 -7 -8 -9 -26

2.7.4.13 GroupMotionDisable

TCL Prototype	int GroupMotionDisable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupMotionDisable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Turn motor OFF and set the “MotionEnable” status to FALSE for the selected MultipleAxes group.
API Errors	0 -7 -8 -9 -19 -22

2.7.4.14 GroupMotionEnable

TCL Prototype	int GroupMotionEnable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupMotionEnable (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Initializes positions and corrector before to turn motor ON and to set the “MotionEnable” status to TRUE for the selected MultipleAxes group.
API Errors	0 -7 -8 -9 -19 -22

2.7.4.15 GroupMoveAbort

TCL Prototype	int GroupMoveAbort (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupMoveAbort (int SocketID, char GroupName [250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name
Output parameters	None
Return	API error

API Input tests	Verify number of parameters. Verify group name. Parameters coherence test.
API Description	Aborts motion : stops the motion in progress on the selected MultipleAxes group.
API Errors	0 -7 -8 -9 -19 -22 -27

2.7.4.16 GroupMoveAbsolute

TCL Prototype	int GroupMoveAbsolute (int SocketID, char GroupName [250], double TargetPosition, ...)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name (can be positioner name : see § positioner API) double : TargetPosition (units) => this parameter must be repeated for each positioner of the selected MultipleAxes group.
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupMoveAbsolute (int SocketID, char GroupName [250], int NbPositioners, double TargetPosition[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name (can be positioner name : see § positioner API) int : Number of Positioners in the MultipleAxes group double[] : TargetPosition array (units) size array defined by the number of positioners
Output parameters	None
Return	API error

API Input tests	Verify number of parameters. Verify group name. Parameters coherence test. Verify target position in relation with the travel limits : $\text{TargetPosition} \geq \text{MinimumTargetPosition}$ $\text{TargetPosition} \leq \text{MaximumTargetPosition}$
API Description	The axes from the selected MultipleAxes group move to their target position. The absolute move refers to the acceleration, velocity, minimumTjerkTime and maximumTjerkTime predefined in the “Stages.ini” or redefined with the “PositionerSGammaParametersSet” API.
API Errors	0 -7 -8 -9 -14 -17 -19 -22 -25 -27 -33 -44

2.7.4.17 GroupMoveRelative

TCL Prototype	int GroupMoveRelative (int SocketID, char GroupName [250], double TargetDisplacement, ...)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name (can be positioner name : see § positioner API) double : TargetDisplacement (units) => this parameter must be repeated for each positioner of the selected MultipleAxes group
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupMoveRelative (int SocketID, char GroupName [250], int NbPositioners, double TargetDisplacement [])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name (can be positioner name : see § positioner API) int : Number of Positioners in the MultipleAxes group double [] : TargetDisplacement array (units) size array defined by the number of positioners
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test. Test the recalculated target position in relation with the travel limits : $\text{TargetPosition} \geq \text{MinimumTargetPosition}$ $\text{TargetPosition} \leq \text{MaximumTargetPosition}$
API Description	The axes from the selected MultipleAxes group moves to the target position : $\text{TargetPosition} = \text{CurrentPosition} + \text{TargetDisplacement}$ <p>The relative move refers to the acceleration, velocity, minimumTjerkTime and maximumTjerkTime predefined in the “Stages.ini” or redefined with the “PositionerSGammaParametersSet” API.</p>
API Errors	0 -7 -8 -9 -14 -17 -19 -22 -25 -27 -33 -44

2.7.4.18 GroupPositionCurrentGet

TCL Prototype	int GroupPositionCurrentGet (int SocketID, char GroupName [250], double * CurrentPosition, ...)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name (can be positioner name : see § positioner API)
Output parameters	double * : CurrentPosition (units) => this parameter must be repeated for each positioner of the selected MultipleAxes group
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupPositionCurrentGet (int SocketID, char GroupName [250], int NbPositioners, double CurrentPosition[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name (can be positioner name : see § positioner API) int : Number of Positioners in the MultipleAxes group
Output parameters	double [] : CurrentPosition (units) size array defined by the number of positioners
Return	API error

API Input tests	Verify number of parameters. Verify group name. Parameters coherence test.
API Description	Reads the MultipleAxes group current position (SetpointPosition - PositionError).
API Errors	0 -7 -8 -9 -14 -19

2.7.4.19 GroupPositionSetpointGet

TCL Prototype	int GroupPositionSetpointGet (int SocketID, char GroupName [250], double * SetPointPosition, ...)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name (can be positioner name : see § positioner API)
Output parameters	double * : SetPointPosition (units) => this parameter must be repeated for each positioner of the selected MultipleAxes group
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupPositionSetpointGet (int SocketID, char GroupName [250], int NbPositioners, double SetPointPosition[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name (can be positioner name : see § positioner API) int : Number of Positioners in the MultipleAxes group
Output parameters	double [] : SetPointPosition (units) size array defined by the number of positioners
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Read MultipleAxes group SetpointPosition (profiler position).
API Errors	0 -7 -8 -9 -14 -19

2.7.4.20 GroupPositionTargetGet

TCL Prototype	int GroupPositionTargetGet (int SocketID, char GroupName [250], double * TargetPosition, ...)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name (can be positioner name : see § positioner API)
Output parameters	double * : TargetPosition (units) => this parameter must be repeated for each positioner of the selected MultipleAxes group
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupPositionTargetGet (int SocketID, char GroupName [250], int NbPositioners, double TargetPosition[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name (can be positioner name : see § positioner API) int : Number of Positioners in the MultipleAxes group
Output parameters	double [] : TargetPosition (units) size array defined by the number of positioners
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Read MultipleAxes group TargetPosition (user position).
API Errors	0 -7 -8 -9 -14 -19

2.7.4.21 GroupStatusGet

TCL Prototype	int GroupStatusGet (int SocketID, char GroupName [250], int * Status)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name
Output parameters	int * : Status
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupStatusGet (int SocketID, char GroupName [250], int * Status)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name
Output parameters	int * : Status
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Reads MultipleAxes group status (See § “Group state list”)
API Errors	0 -7 -8 -9 -15 -19

2.7.4.22 MultipleAxesPVTVerification

TCL Prototype	int MultipleAxesPVTVerification (int SocketID, char GroupName[250], char *FileName)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : MultipleAxes group name char [50] : Trajectory file name
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int MultipleAxesPVTVerification (int SocketID, char GroupName[250], char *FileName)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : MultipleAxes group name char [50] : Trajectory file name
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Multiple axes PVT trajectory verification. The result is to be read with the API MultipleAxesPVTVerificationResultGet. This API is independent from the MultipleAxesPVTExecution API, so the user need not execute this API before executing the MultipleAxesPVTExecution. The trajectory file must be stocked in the "\\ADMIN\\Public\\Trajectory" controller directory.
API Errors	0 -3 -7 -8 -9 -14 -19 -22 -61 -66 -68 -69 -70

2.7.4.23 MultipleAxesPVTVerificationResultGet

TCL Prototype	int MultipleAxesPVTVerificationResultGet (int SocketID, char FullPositionerName[250], char *FileName, double *MinimumPosition, double *MaximumPosition, double *MaximumVelocity, double *MaximumAcceleration)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : FullPositionerName from a MultipleAxes group
Output parameters	char * : Examined trajectory file name double * : Minimum position (units) double * : Maximum position (units) double * : Positioner maximum velocity (units / seconds) double * : Positioner maximum acceleration (units / seconds ²)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int MultipleAxesPVTVerificationResultGet (int SocketID, char FullPositionerName[250], char *FileName, double *MinimumPosition, double *MaximumPosition, double *MaximumVelocity, double *MaximumAcceleration)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : FullPositionerName from a MultipleAxes group
Output parameters	char * : Examined trajectory file name double * : Minimum position (units) double * : Maximum position (units) double * : Positioner maximum velocity (units / seconds) double * : Positioner maximum acceleration (units / seconds ²)
Return	API error

API Input tests	Verify the positioner name.
API Description	<p>This API used to get results previously calculated by MultipleAxesPVTVerification API, positioner by positioner. The results is the position limits (min and max values), positioner maximum velocity, positioner maximum acceleration. This API doesn't work if not any trajectory file has not previously been examined.</p> <p><u>NOTE :</u> This API is recommended before to execute a trajectory to know the velocity and the acceleration.</p>
API Errors	0 -3 -7 -8 -9 -13 -14 -18 -22

2.7.4.24 MultipleAxesPVTExecution

TCL Prototype	int MultipleAxesPVTExecution (int SocketID, char GroupName[250], char *FileName , int ExecutionNumber)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name char [50] : Trajectory file name int : ExecutionNumber
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int MultipleAxesPVTExecution (int SocketID, char GroupName[250], char *FileName , int ExecutionNumber)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : MultipleAxes group name char [50] : Trajectory file name int : ExecutionNumber
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the group name. Parameters coherence test.
API Description	Execute a PVT trajectory from a trajectory data file. The trajectory file must be stocked in the “\ADMIN\Public\Trajectory” controller directory. Before to execute a trajectory, verify if the velocity and the acceleration with the “MultipleAxesPVTVerification” and “MultipleAxesPVTVerificationResultGet” APIs.
API Errors	0 -3 -7 -8 -9 -17 -19 -22 -71 -72

2.7.4.25 MultipleAxesPVTPParametersGet

TCL Prototype	int MultipleAxesPVTPParametersGet (int SocketID, char GroupName[250], char *FileName, int *CurrentElementNumber)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : MultipleAxes group name
Output parameters	char * : Executing trajectory file name int * : Current executing element number
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int MultipleAxesPVTPParametersGet (int SocketID, char GroupName[250], char *FileName, int *CurrentElementNumber)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : MultipleAxes group name
Output parameters	char * : Executing trajectory file name int * : Current executing element number
Return	API error

API Input tests	Verify the number of parameters. Verify the group name.
API Description	Get the MultipleAxes PVT trajectory parameters (trajectory name, current executing element number). This API works only during the period of trajectory execution.
API Errors	0 -7 -8 -9 -13 -15 -19 -22

2.7.5 PVT trajectory file

2.7.5.1 File description

The PVT trajectory is defined for a MultipleAxes group and is described in a file. Each line of this file represents an element of the PVT trajectory.

A line is composed of several columns and the number of columns depends on the total number of positioners:

The first column is the time period to execute this element.

The next columns are grouped in pairs of two columns representing the displacement and the output velocity for each positioner of the group.

The line format is described as following as :

Column #1 : Element time period (seconds)

Column #2 : Element displacement (units) *=> Element for the first positioner*

Column #3 : Element output velocity (units / second)

Column #4 : Element displacement (units) *=> Element for the second positioner*

Column #5 : Element output velocity (units / second)

(And so on ...)

Note :

The first positioner is the one that is defined in first in the group section in "PositionerInUse".

The second positioner is the one defined in second in the group section in "PositionerInUse".

(And so on ...)

2.7.5.2 File example

The PVT trajectory file must be read from the "\\ADMIN\\Public\\Trajectory" directory. File example:

```
1,0.416666667,1.25,0,0
1,2.916666667,5,0,0
1,7.083333333,8.75,0,0
1,9.583333333,10,0,0
1,10,10,0.416666667,1.25
1,10,10,2.916666667,5
1,10,10,7.083333333,8.75
1,10,10,9.583333333,10
1,9.583333333,8.75,10,10
1,7.083333333,5,10,10
1,2.916666667,1.25,10,10
1,0.416666667,0,10,10
1,0,0,9.583333333,8.75
1,0,0,7.083333333,5
1,0,0,2.916666667,1.25
1,0,0,0.416666667,0
```

2.7.6 Configuration files

Example of configuration file for a MultipleAxes group named “MULTI” composed of two positioners named “ONE” and “TWO”.

System.ini file :

```
[GROUPS]
MultipleAxesInUse = MULTI

[MULTI]                                     ; AXIS MultipleAxes group configuration
PositionerInUse = ONE, TWO
PositionerNumber = 2
InitializationAndHomeSearchSequence = Together ; Together or OneAfterAnother

[MULTI.ONE]
StageName = MYSTAGE1
STAGE configuration => See § “Positioner : Configuration parameters”

[MULTI.TWO]
StageName = MYSTAGE2
STAGE configuration => See § “Positioner : Configuration parameters”
```

Stages.ini file :

```
[MYSTAGE1]
MYSTAGE1 configuration => See § “Positioner : Configuration parameters”

[MYSTAGE2]
MYSTAGE2 configuration => See § “Positioner : Configuration parameters”
```

2.8 General features

2.8.1.1 **ErrorStringGet**

TCL Prototype	int ErrorStringGet (int SocketID, int ErrorCode, char * ErrorString)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) int : ErrorCode
Output parameters	char * : ErrorString
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int ErrorStringGet (int SocketID, int ErrorCode, char * ErrorString)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) int : ErrorCode
Output parameters	char * : ErrorString
Return	API error

API Input tests	Verify the number of parameters.
API Description	Returns the error description corresponding to an error code (see § Error list).
API Errors	0 -7 -9 -13 -15

2.8.1.2 ElapsedTimeGet

TCL Prototype	int ElapsedTimeGet (int SocketID, double* ElapsedTime)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API)
Output parameters	Double * : ElapsedTime (seconds)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int ElapsedTimeGet (int SocketID, double* ElapsedTime)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API)
Output parameters	Double * : ElapsedTime (seconds)
Return	API error

API Input tests	Verify the number of parameters.
API Description	Returns the elapsed time in seconds from controller power on.
API Errors	0 -7 -9 -14

2.8.1.3 FirmwareVersionGet

TCL Prototype	int FirmwareVersionGet (int SocketID, char Version[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API)
Output parameters	char [] : Version
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int FirmwareVersionGet (int SocketID, char Version[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API)
Output parameters	char [] : Version
Return	API error

API Input tests	Verify the number of parameters.
API Description	Gets the firmware name and the version number. Firmware version string : “XPS-C8 Firmware V1.0.0”
API Errors	0 -7 -9 -13

2.8.1.4 GroupStatusStringGet

TCL Prototype	int GroupStatusStringGet (int SocketID, int GroupStatusCode, char GroupStatusString [])
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) int : GroupStatusCode
Output parameters	char [] : GroupStatusString
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GroupStatusStringGet (int SocketID, int GroupStatusCode, char GroupStatusString [])
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) int : GroupStatusCode
Output parameters	char [] : GroupStatusString
Return	API error

API Input tests	Verify the number of parameters. Parameters coherence test.
API Description	Returns the group status string corresponding to the group status code (see § group state list).
API Errors	0 -7 -9 -13 -15

2.8.1.5 GlobalArrayGet

TCL Prototype	int GlobalArrayGet (int SocketID, int Number, char * StringValue)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) int : Number
Output parameters	char * : StringValue
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GlobalArrayGet (int SocketID, int Number, char * StringValue)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) int : Number
Output parameters	char * : StringValue
Return	API error

API Input tests	Verify the number of parameters. Parameters coherence test.
API Description	Gets the global variable value from the global array at the “Number” index.
API Errors	0 -7 -9 -13 -15

2.8.1.6 GlobalArraySet

TCL Prototype	int GlobalArraySet (int SocketID, int Number, char StringValue[250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) int : Number char [250] : StringValue
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GlobalArraySet (int SocketID, int Number, char StringValue[250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) int : Number char [250] : StringValue
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Parameters coherence test.
API Description	Sets the global variable value in the global array at the “Number” index
API Errors	0 -7 -9 -13 -15

2.8.1.7 KillAll

TCL Prototype	int KillAll (int SocketID)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int KillAll (int SocketID)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API)
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters.
API Description	Kills and resets all groups. All groups come back to the “not initialised” status.
API Errors	0 -7 -9 -26

2.8.1.8 Reboot

TCL Prototype	int Reboot (int SocketID)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int Reboot (int SocketID)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API)
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters.
API Description	Reboots the controller.
API Errors	0 -7 -9

2.8.1.9 TimerSet

TCL Prototype	int TimerSet (char TimerName[250], int FrequencyTicks)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : TimerName ("Timer1", "Timer2", "Timer3", "Timer4", "Timer5") int : FrequencyTicks
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int TimerSet (char TimerName[250], int FrequencyTicks)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : TimerName ("Timer1", "Timer2", "Timer3", "Timer4", "Timer5") int : FrequencyTicks
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Parameters coherence test.
API Description	Set the FrequencyTicks to activate a timer. If FrequencyTicks = 0, the timer is disabled.
API Errors	0 -7 -9 -10 -13 -15

2.8.1.10 TimerGet

TCL Prototype	int TimerGet (char TimerName[250], int* FrequencyTicks)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : TimerName ("Timer1", "Timer2", "Timer3", "Timer4", "Timer5")
Output parameters	int * : FrequencyTicks
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int TimerGet (char TimerName[250], int* FrequencyTicks)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : TimerName ("Timer1", "Timer2", "Timer3", "Timer4", "Timer5")
Output parameters	int * : FrequencyTicks
Return	API error

API Input tests	Verify the number of parameters. Parameters coherence test.
API Description	Retruns the FrequenyTicks. If FrequencyTicks = 0, the timer is disabled.
API Errors	0 -7 -9 -13 -15

2.9 Analog and digital I/O

2.9.1 GPIO name list

GPIO name	Description
GPIO1.DI	Digital Input of the INT card connector # 1 (8 bits)
GPIO2.DI	Digital Input of the INT card connector # 2 (6 bits)
GPIO3.DI	Digital Input of the INT card connector # 3 (6 bits)
GPIO4.DI	Digital Input of the INT card connector # 4 (16 bits)
GPIO1.DO	Digital Output of the INT card connector # 1 (8 bits)
GPIO3.DO	Digital Output of the INT card connector # 3 (6 bits)
GPIO4.DO	Digital Output of the INT card connector # 4 (16 bits)
GPIO2.ADC1	Analog Input # 1 of the INT card connector # 2
GPIO2.ADC2	Analog Input # 2 of the INT card connector # 2
GPIO2.ADC3	Analog Input # 3 of the INT card connector # 2
GPIO2.ADC4	Analog Input # 4 of the INT card connector # 2
GPIO2.DAC1	Analog Output # 1 of the INT card connector # 2
GPIO2.DAC2	Analog Output # 2 of the INT card connector # 2
GPIO2.DAC3	Analog Output # 3 of the INT card connector # 2
GPIO2.DAC4	Analog Output # 4 of the INT card connector # 2

2.9.2 API description

2.9.2.1 GPIOAnalogGet

TCL Prototype	int GPIOAnalogGet (int SocketID, char GPIOName[250], double * AnalogValue, ...)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : GPIOName (ADC or DAC)
Output parameters	double* : AnalogValue
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GPIOAnalogGet (int SocketID, int NbAnalogIO, char * GPIOName[], double AnalogValue[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) int : Number of analog I/O char *[] : GPIOName array (ADC or DAC)
Output parameters	double [] : AnalogValue array
Return	API error

API Input tests	Verify the number of parameters. Parameters coherence test.
API Description	Reads analog value for one or several analog inputs or analog outputs.
API Errors	0 -7 -8 -9 -14 -17

2.9.2.2 GPIOAnalogSet

TCL Prototype	int GPIOAnalogSet (int SocketID, char GPIOName[250], double AnalogValue, ...)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : GPIOName (DAC) double : Analog Output Value
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GPIOAnalogSet (int SocketID, int NbAnalogOutputs, char * GPIOName[], double AnalogValue[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) int : Number of analog Outputs char *[] : GPIOName array (DAC) double : Analog Output Value
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Parameters coherence test.
API Description	Set analog value for one or several analog outputs.
API Errors	0 -7 -8 -9 -14 -17

2.9.2.3 GPIOAnalogGainGet

TCL Prototype	int GPIOAnalogGainGet (int SocketID, char GPIOName[250], int* AnalogInputGainValue, ...)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : GPIOName (ADC)
Output parameters	int * : AnalogInputGainValue
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GPIOAnalogGainGet (int SocketID, int NbAnalogInputs, char * GPIOName[], int* AnalogInputGainValue[])
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) int : Number of analog Inputs char *[] : GPIO Name array (ADC)
Output parameters	int [] : AnalogInputGainValue
Return	API error

API Input tests	Verify the number of parameters. Parameters coherence test.
API Description	Reads gain (1, 2, 4 or 8) for one or several analog inputs.
API Errors	0 -7 -8 -9 -15 -17

2.9.2.4 GPIOAnalogGainSet

TCL Prototype	int GPIOAnalogGainSet (int SocketID, char GPIOName[250], int AnalogInputGainValue, ...)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : GPIOName (ADC) int : AnalogInputGainValue
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GPIOAnalogGainSet (int SocketID, int NbAnalogInputs, char * GPIOName[], int AnalogInputGainValue[])
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) int : Number of analog Inputs char *[] : GPIOName array (ADC) int [] : AnalogInputGainValue array
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Parameters coherence test. ADC gains must be 1, 2, 4 or 8
API Description	Set gain (1, 2, 4 or 8) for one or several analog inputs.
API Errors	0 -7 -8 -9 -15 -17

2.9.2.5 GPIODigitalGet

TCL Prototype	int GPIODigitalGet (int SocketID, char GPIOName[250], unsigned short* DigitalValue)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : GPIOName (DI or DO)
Output parameters	unsigned short * DigitalValue
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GPIODigitalGet (int SocketID, char GPIOName[250], unsigned short* DigitalValue)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : GPIOName (DI or DO)
Output parameters	unsigned short * DigitalValue
Return	API error

API Input tests	Verify the number of parameters. Parameters coherence test.
API Description	Reads digital input or digital output.
API Errors	0 -7 -8 -9 -15 -17

2.9.2.6 GPIODigitalSet

TCL Prototype	int GPIODigitalSet (int SocketID, char GPIOName[250], unsigned short Mask, unsigned short DigitalOutputValue)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : GPIOName (DO) unsigned short : Mask unsigned short : DigitalOutputValue
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GPIODigitalSet (int SocketID, char GPIOName[250], unsigned short Mask, unsigned short DigitalOutputValue)
Input parameters	int : SocketID (Socket identifiant gets by the "TCP_ConnectToServer" API) char [250] : GPIOName (DO) unsigned short : Mask unsigned short : DigitalOutputValue
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Parameters coherence test.
API Description	Set digital value for one or several digital outputs.
API Errors	0 -7 -8 -9 -15 -17

2.10 Gathering

2.10.1 Internal Gathering and External Gathering

To collect data, two methods can be used:

1. *Internal Gathering*

The data from the standard gathering are updated by an internal interrupt (10 KHz).

Data types:

- FullPositionerName.CurrentPosition
- FullPositionerName.SetpointPosition
- FullPositionerName.FollowingError
- FullPositionerName.CurrentVelocity
- FullPositionerName.SetpointVelocity
- FullPositionerName.CurrentAcceleration
- FullPositionerName.SetpointAcceleration
- GPIO (ADC,DAC,DI,DO)

APIs to use internal gathering :

- GatheringConfigurationSet
- GatheringConfigurationGet
- GatheringStopAndSave
- GatheringCurrentNumberGet

2. *External Gathering*

The external positions from the external gathering are updated by an external trigger (TRIG IN connector).

The jitter is strictly inferior to 50 ns.

The ADC are updated by an internal interrupt (10 KHz) thus their jitter is strictly inferior to 100 µs.

CAUTION: The frequency from TRIG IN connector must be strictly inferior to 10 KHz.

Data types:

- FullPositionerName.ExternalLatchPosition
- GPIO (ADC)

APIs to use external gathering:

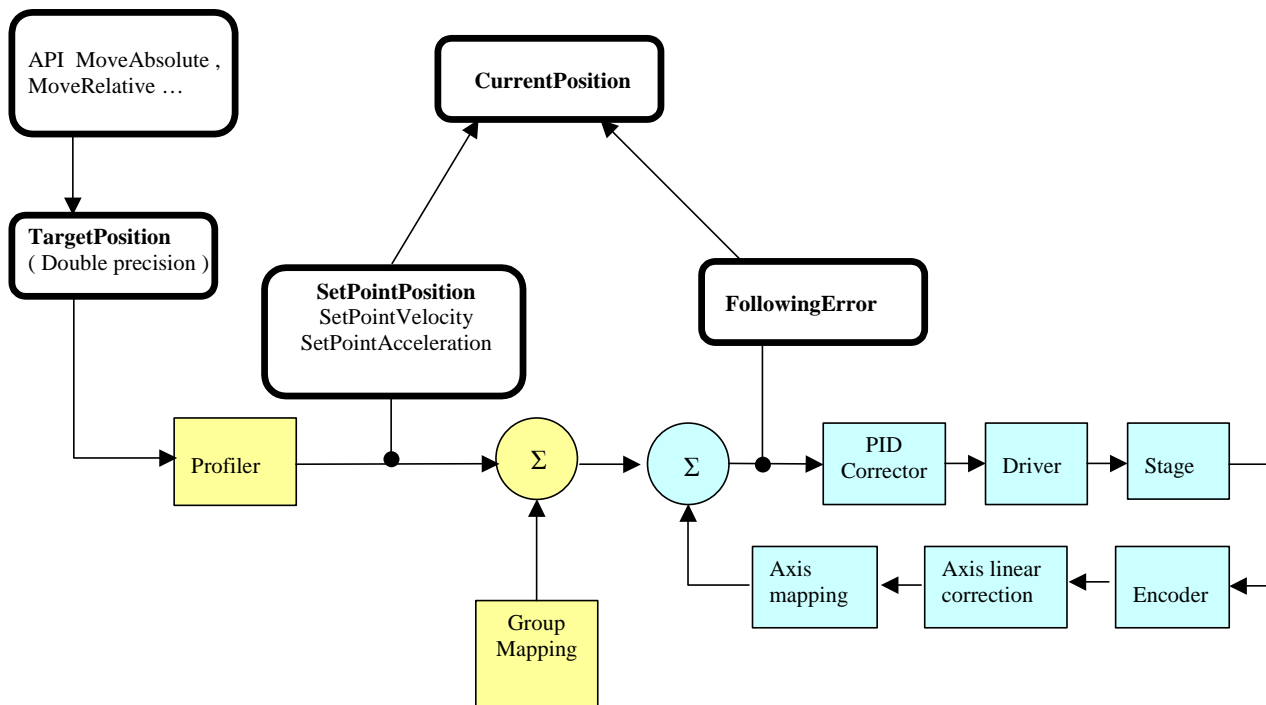
- GatheringExternalConfigurationSet
- GatheringExternalConfigurationGet
- GatheringExternalStopAndSave
- GatheringExternalCurrentNumberGet

NOTE :

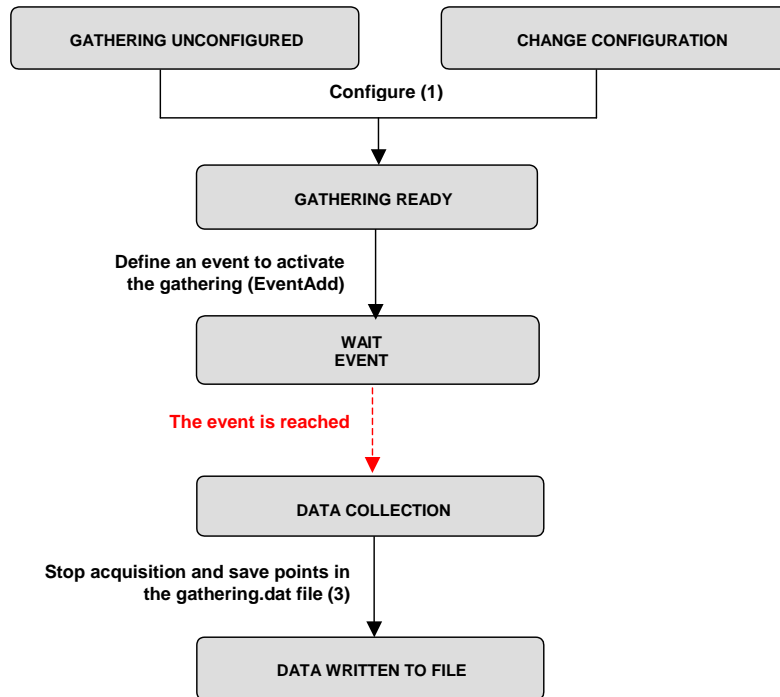
To launch a internal gathering or an external gathering, you must program an event (see § Events and Actions)

2.10.2 Definition of different positions for one positioner

SetPointPosition & **CurrentPosition** are accessible via API and Gathering
SetPointSpeed & **SetPointAcceleration** are accessible via **Extended Gathering**



2.10.3 Sequence



Called API :

- (1) GatheringConfigurationSet or GatheringExternalConfigurationSet
- (2) EventAdd (FullPositionerName, EventName, EventParameter, **GatheringRun, NbPoints, Divisor, 0**)
or EventAdd (FullPositionerName, EventName, EventParameter, **ExternalGatheringRun, NbPoints, Divisor, 0**)
- (3) GatheringStopAndSave or GatheringExternalConfigurationSet

2.10.4 Gathering file

The “Gathering.dat” file or the “ExternalGathering.dat” file is saved in the PUBLIC directory.

1000000 data maximum are possibles to acquire.

25 rows maximum are possibles to configure.

The separator is the tabulation.

The first line contains the **sample period**

The second line contains the **gathering types**

The others lines contain the **points**

Gathering.dat

SamplePeriod	0	0
GatheringTypeA	GatheringTypeB	GatheringTypeC
ValueA1	ValueB1	ValueC1
ValueA2	ValueB2	ValueC2
...
ValueAN	ValueBN	ValueCN

2.10.5 API description

2.10.5.1 GatheringConfigurationGet

TCL Prototype	int GatheringConfigurationGet (int SocketID, char TypeList[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API)
Output parameters	char [] : TypeList (separator is semicolon)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GatheringConfigurationGet (int SocketID, char TypeList[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) int : Number of types
Output parameters	char * : Type list (separator is semicolon)
Return	API error

API Input tests	Verify the number of parameters. Parameters coherence test. Verify the gathering status.
API Description	Gets the current Gathering configuration. Use the “GatheringListGet” API to get the list of gathering types.
API Errors	0 -7 -9 -13 -30

2.10.5.2 GatheringConfigurationSet

TCL Prototype	int GatheringConfigurationSet (int SocketID, char Type[500], ...)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [500] : Type
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GatheringConfigurationSet (int SocketID, int NbTypes, char * Type[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) int : Number of types char *[] : Type array
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the mnemonic gathering type. Parameters coherence test.
API Description	Configures the next acquisition. Use the “GatheringListGet” API to get the list of gathering types.
API Errors	0 -7 -9 -29

2.10.5.3 GatheringCurrentNumberGet

TCL Prototype	int GatheringCurrentNumberGet (int SocketID, int* CurrentNumber, int* MaximumSamplesNumber)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API)
Output parameters	int * : CurrentNumber int* : MaximumSamplesNumber
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GatheringCurrentNumberGet (int SocketID, int* CurrentNumber, int* MaximumSamplesNumber)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API)
Output parameters	int * : CurrentNumber int* : MaximumSamplesNumber
Return	API error

API Input tests	Verify the number of parameters.
API Description	Returns the maximum number of possible samples and the current number during acquisition.
API Errors	0 -7 -9 -15 -32

2.10.5.4 GatheringStopAndSave

TCL Prototype	int GatheringStopAndSave (int SocketID)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GatheringStopAndSave (int SocketID)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API)
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the gathering data.
API Description	Stop acquisition and save data in the GATHERING.DAT file (in the “\PUBLIC” controller folder).
API Errors	0 -7 -9 -30 -60

2.10.5.5 GatheringExternalConfigurationGet

TCL Prototype	int GatheringExternalConfigurationGet (int SocketID, char TypeList[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API)
Output parameters	char [] : TypeList (separator is semicolon)
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GatheringExternalConfigurationGet (int SocketID, char TypeList[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API)
Output parameters	char [] : TypeList (separator is semicolon)
Return	API error

API Input tests	Verify the number of parameters. Parameters coherence test. Verify the gathering status.
API Description	Gets the current External Gathering configuration. Use the “GatheringExternalListGet” API to get the list of external gathering types.
API Errors	0 -7 -9 -13 -30

2.10.5.6 GatheringExternalConfigurationSet

TCL Prototype	int GatheringExternalConfigurationSet (int SocketID, char Type[500], ...)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [500] : Type
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GatheringExternalConfigurationSet (int SocketID, int NbTypes, char * Type[])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) int : Number of types char * [] : Type array
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the mnemonic gathering type. Parameters coherence test.
API Description	Configures the next acquisition. Use the “GatheringExternalListGet” API to get the list of external gathering types.
API Errors	0 -7 -9 -29

2.10.5.7 GatheringExternalCurrentNumberGet

TCL Prototype	int GatheringExternalCurrentNumberGet (int SocketID, int* CurrentNumber, int* MaximumSamplesNumber)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API)
Output parameters	int * : CurrentNumber int* : MaximumSamplesNumber
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GatheringExternalCurrentNumberGet (int SocketID, int* CurrentNumber, int* MaximumSamplesNumber)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API)
Output parameters	int * : CurrentNumber int* : MaximumSamplesNumber
Return	API error

API Input tests	Verify the number of parameters.
API Description	Returns the maximum number of possible samples and the current number during acquisition.
API Errors	0 -7 -9 -15 -32

2.10.5.8 GatheringExternalStopAndSave

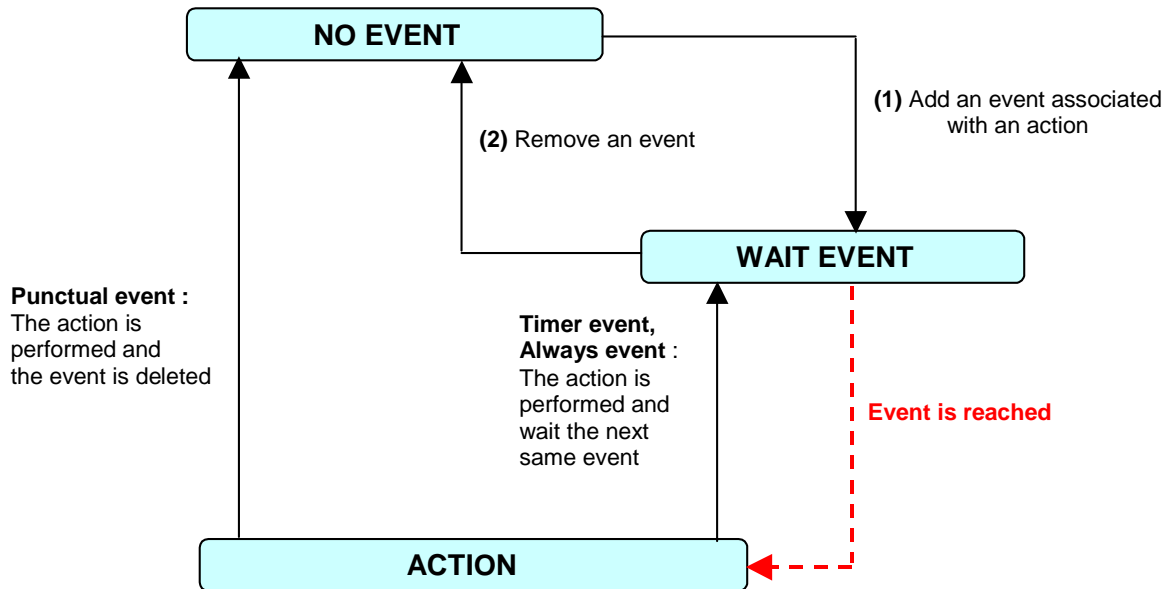
TCL Prototype	int GatheringExternalStopAndSave (int SocketID)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API)
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int GatheringExternalStopAndSave (int SocketID)
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API)
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the gathering data.
API Description	Stop acquisition and save data in the ExternalGathering.dat file (in the “\PUBLIC” controller folder).
API Errors	0 -7 -9 -30 -60

2.11 Events and actions

2.11.1 Sequence



Called API :

- (1) EventAdd
- (2) EventRemove

2.11.2 Events

2.11.2.1 Event categories

SGamma	Select an “SGamma” event for an positioner equipped of an SGamma profile
Jog	Select an “Jog” event for an group equipped of an Jog profile
XYLineArc	Select an “XYLineArc” event for an group equipped of an XYLineArc profile
Spline	Select an “Spline” event for an group equipped of an Spline profile
PVT	Select an “PVT” event for an group equipped of an PVT profile

Each category can be used with an event. The point is the separator, for examples :

SGamma.ConstantVelocityStart
XYLineArc.TrajectorySate

List of events

Event name	Event Parameter	SGamma	Jog	XYLineArc	Spline	PVT
• Always	0					
• Immediate	0					
• Timer	0					
• MotionDone	0					
• ConstantVelocityStart	0	•	•			
• ConstantVelocityEnd	0	•				
• ConstantVelocityState	0	•	•			
• ConstantAccelerationStart	0	•				
• ConstantAccelerationEnd	0	•				
• ConstantAccelerationState	0	•				
• ConstantDecelerationStart	0	•				
• ConstantDecelerationEnd	0	•				
• ConstantDecelerationState	0	•				
• MotionStart	0	•	•			
• MotionEnd	0	•	•			
• MotionState	0	•	•			
• TrajectoryStart	0			•	•	•
• TrajectoryEnd	0			•	•	•
• TrajectoryState	0			•	•	•
• ElementNumberStart	Element number			•	•	•
• ElementNumberState	Element number			•	•	•

2.11.2.2 Event definitions

The following events are defined for one positioner

2.11.2.2.1 Always

Event name : Always

Event parameter : 0

Event Description : This event triggers an action ALWAYS

NOTE : This event is **PERMANENT**. Call the “EventRemove” API to remove it.

2.11.2.2.2 Immediate

Event name : Immediate

Event parameter : 0

Event Description : This event triggers an action IMMEDIATELY

NOTE : This event is **EPHEMERAL**.

2.11.2.2.3 Timer (1 to 5)

Event name : Timer1, Timer2, Timer3, Timer4 or Timer5

Event parameter : 0

Event Description : This event triggers an action on a timer defined by “TimerSet” API.

NOTE : This event is **PERMANENT**. Call the “EventRemove” API to remove it.

2.11.2.2.4 MotionDone

Event name : MotionDone

Event parameter : 0

Event Description : This event triggers an action when the motion done is reached

2.11.2.2.5 ConstantVelocityStart

Event name : ConstantVelocityStart

Event parameter : 0

Event Description : This event triggers an action when the constant velocity is reached

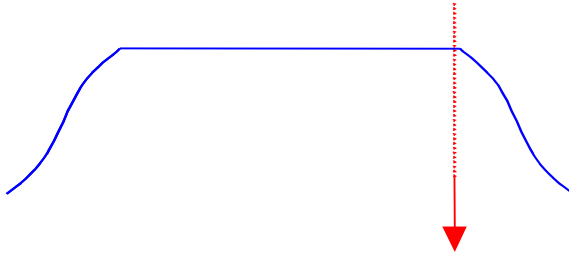


2.11.2.2.6 ConstantVelocityEnd

Event name : ConstantVelocityEnd

Event parameter : 0

Event Description : This event triggers an action when the constant velocity is finished

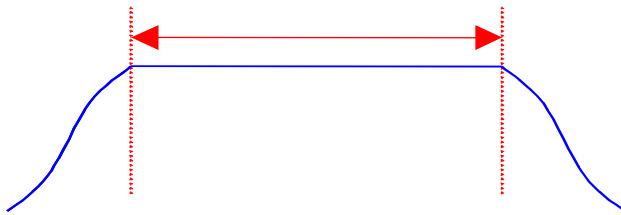


2.11.2.2.7 ConstantVelocityState

Event name : ConstantVelocityState

Event parameter : 0

Event Description : This event triggers an action during the constant velocity

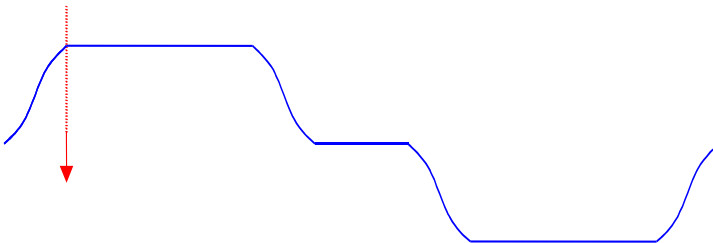


2.11.2.2.8 ConstantAccelerationStart

Event name : ConstantAccelerationStart

Event parameter : 0

Event Description : This event triggers an action when the constant acceleration is reached

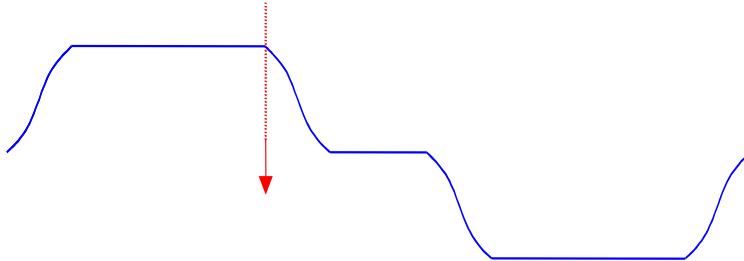


2.11.2.2.9 ConstantAccelerationEnd

Event name : ConstantAccelerationEnd

Event parameter : 0

Event Description : This event triggers an action when the constant acceleration is finished

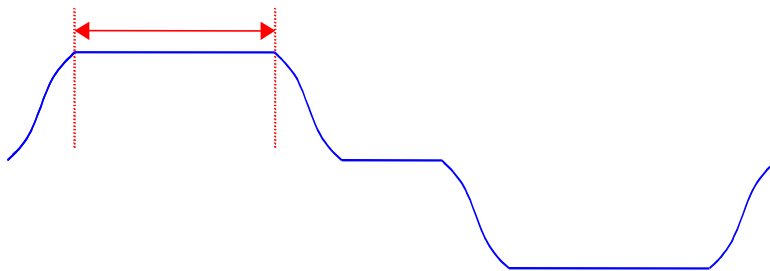


2.11.2.2.10 ConstantAccelerationState

Event name : ConstantAccelerationState

Event parameter : 0

Event Description : This event triggers an action during the constant acceleration

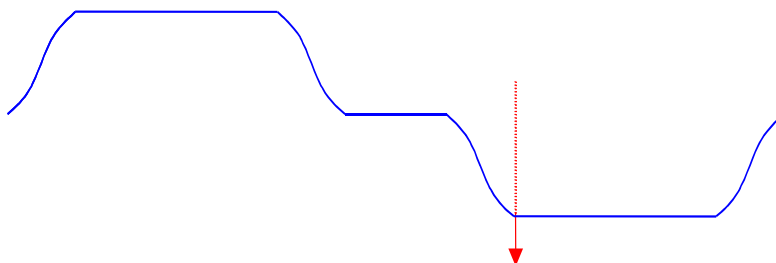


2.11.2.2.11 ConstantDecelerationStart

Event name : ConstantDecelerationStart

Event parameter : 0

Event Description : This event triggers an action when the constant deceleration is reached

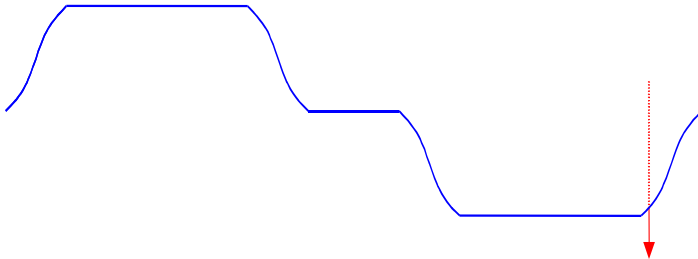


2.11.2.2.12 ConstantDecelerationEnd

Event name : ConstantDecelerationEnd

Event parameter : 0

Event Description : This event triggers an action when the constant deceleration is finished

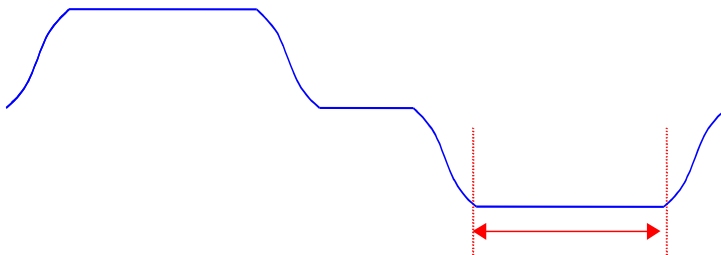


2.11.2.2.13 ConstantDecelerationState

Event name : ConstantDecelerationState

Event parameter : 0

Event Description : This event triggers an action during the constant deceleration

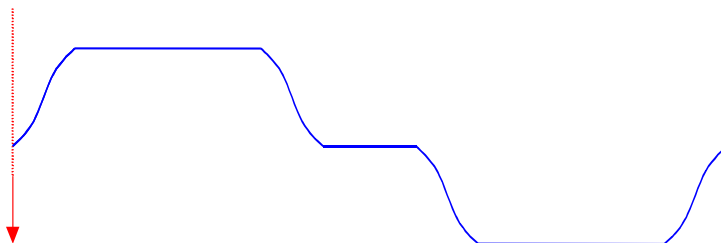


2.11.2.2.14 MotionStart

Event name : MotionStart

Event parameter : 0

Event Description : This event triggers an action when the motion is started

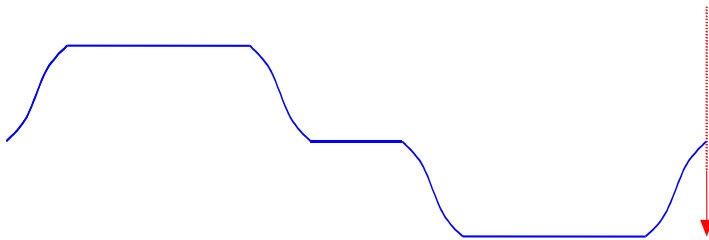


2.11.2.2.15 MotionEnd

Event name : MotionEnd

Event parameter : 0

Event Description : This event triggers an action when the motion is stopped

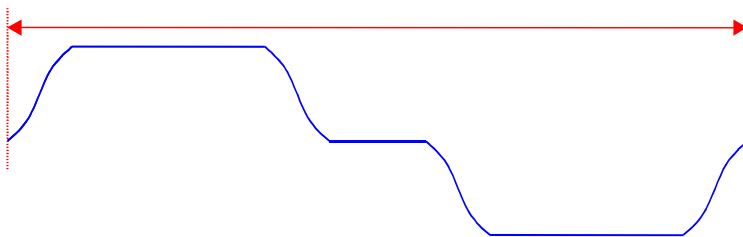


2.11.2.2.16 MotionState

Event name : MotionState

Event parameter : 0

Event Description : This event triggers an action during the motion

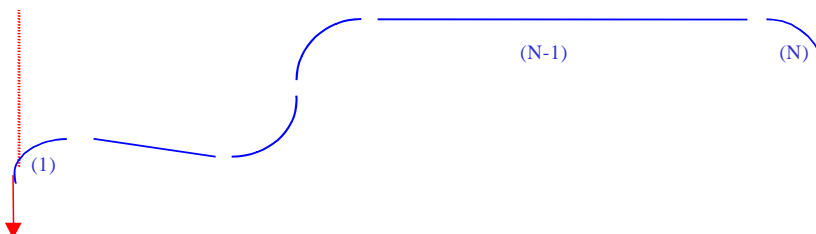


2.11.2.2.17 TrajectoryStart

Event name : TrajectoryStart

Event parameter : 0

Event Description : This event triggers an action when the LineArc trajectory is started

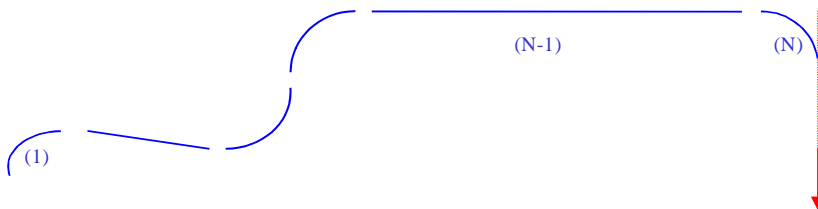


2.11.2.2.18 TrajectoryEnd

Event name : TrajectoryEnd

Event parameter : 0

Event Description : This event triggers an action when the LineArc trajectory is finished

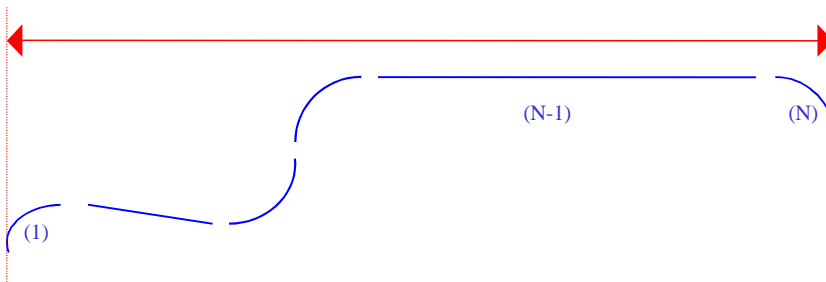


2.11.2.2.19 TrajectoryState

Event name : TrajectoryState

Event parameter : 0

Event Description : This event triggers an action during the LineArc trajectory

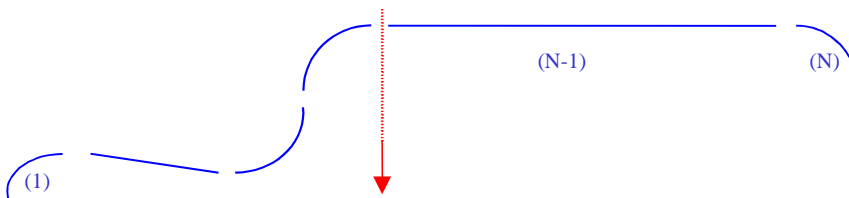


2.11.2.2.20 ElementNumberStart

Event name : ElementNumberStart

Event parameter : 0

Event Description : This event triggers an action when the LineArc element number is started

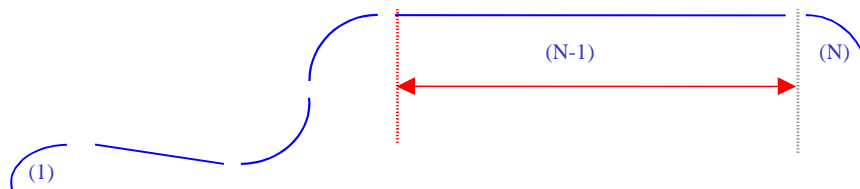


2.11.2.2.21 ElementNumberState

Event name : ElementNumberState

Event parameter : 0

Event Description : This event triggers an action during the LineArc element number



2.11.3 Actions

2.11.3.1 List of actions

Action Name	Action Parameter 1	Action Parameter 2	Action Parameter 3
• DOToggle	GPIO name (DO)	Mask	0
• DOPulse	GPIO name (DO)	Mask	0
• DOSet	GPIO name (DO)	Mask	Value
• DACSet.SetpointPosition	GPIO name (DAC)	Gain	Offset
• DACSet.SetpointVelocity	GPIO name (DAC)	Gain	Offset
• DACSet.SetpointAcceleration	GPIO name (DAC)	Gain	Offset
• ExecuteTCLScript	TCL file name	Task name	Arguments list
• GatheringRun	Number of points	Divisor	0
• ExternalGatheringRun	Number of points	Divisor	0

CAUTION :

When an action is accosiated with an event, it's only when the event is occurred that the action is trigged ... For example, if you want set a digital output to a value only when the constant velocity state ... you must select the "DoSet" action associated with the "ConstantVelocityStart" event and the "ConstantVelocityEnd" event.

This action can NOT do with the "ConstantVelocityState" event because the digital output is setting as soon as and every "ConstantVelocityState" event is occurred and so, the digital output stays in the same state, even if the constant velocity state is finished.

2.11.3.2 DOToggle

Action name : **DOToggle**

Action parameter # : **GPIOName**
The GPIOName must be a digital output name. See the list of all the GPIO Names for the Analog and Digital I/O.

Action parameter # : **Mask**
The mask defines which bits on the GPIO output will be toggled (change their value). For example, if the GPIO output is an 8 bit output and the mask is set to 4 then the equivalent binary number is 00000100. So as an action, the bit #3 will be toggled.

Action parameter # : **0**
This parameter setting is 0 by default.

Description : This action is used to reverse the value of one or many bits on the Digital Output. When using this action with an event that has some duration (for example motion state) the value of the bits will be toggled each servo cycle as long as the event occurs.

2.11.3.3 DOPulse

- Action name* : **DOPulse**
- Action parameter #* : **GPIOName**
The GPIOName must be a digital output name. See the list of all the GPIO Names for the Analog and Digital I/O.
- Action parameter #* : **Mask**
The mask defines on which bits on the GPIO output the pulse will be generated. For example, if the GPIO output is an 8 bit output and the mask is set to 6 then the equivalent binary number is 00000110. So as an action, a 1 μ s pulse will be generated on bit #2 and #3 of the GPIO output.
- Action parameter #* : **0**
This parameter setting is 0 by default.
- Description* : This action is used to generate a positive pulse on the Digital Output. The duration of the pulse is 1 microsecond. To function properly, the bits on which the pulse is generated should be set to zero before. When using this action with an event that has some duration (for example motion state) a 1 μ s pulse will be generated each servo cycle as long as the event occurs. As the servo cycle time is much shorter than 1 μ s (0.1 μ s) the resulting effect is a pulse with a duration as long as the event

2.11.3.4 DOSet

- Action name* : **DOSet**
- Action parameter #* : **GPIOName**
The GPIOName must be a digital output name. See the list of all the GPIO Names for the Analog and Digital I/O.
- Action parameter #* : **Mask**
The mask defines which bits on the GPIO output are being addressed. For example, if the GPIO output is an 8 bit output and the mask is set to 26 then the equivalent binary number is 00011010. Therefore with a Mask setting of 26, only the bits #2, #4 and #5 are being addressed on the GPIO output.
- Action parameter #* : **Value**
This parameter sets the value of the bits that are being addressed according to the Mask setting. So for example since a Mask setting of 26, bits #2, #4 and #5 can be modified, a value of 8 (00001000) will set the bits #2 and #5 to 0 and the bit #4 to 1.
- Description* : The value of bits(s) is modified on a digital output. The GPIOName must be a digital output name.

2.11.3.5 **DACSet.SetpointPosition**

Action name : **DACSet.SetpointPosition**

Action parameter # : **GPIOName**

The GPIOName must be an analog output name. See the list of all the GPIO Names for the Analog and Digital I/O.

Action parameter # : **Gain**

The SetpointPosition is multiplied by the gain value. For example if the gain is set to 10 and the corrected SetpointPosition is 1mm then the output voltage will be 10 V.

Action parameter # : **Offset**

The offset value is used to correct for any voltage that may be present on the Analog output.

Description : This action is used to output a voltage on the Analog output to form an image of the theoretical position. The gain and the offset are used to calibrate this image. This action makes most sense with events that have some duration (always, MotionState, ElementNumberState, etc.) as the analog output will be updated each servo cycle as long as the event lasts. When used with events that have no duration (like MotionStart or MotionEnd), the analog output gets only updated once and holds this value until its next change which has typically less meaning.
The analog output value is setting with the SetpointPosition value and this is multiplied by a gain and an offset is added :
Analog output = SetpointPosition * gain + offset.

2.11.3.6 **DACSet.SetpointVelocity**

Action name : **DACSet.SetpointVelocity**

Action parameter #1 : **GPIOName**

The GPIOName must be an analog output name. See the list of all the GPIO Names for the Analog and Digital I/O.

Action parameter #2 : **Gain**

The SetpointVelocity is multiplied by the gain value. For example if the gain is set to 10 and the corrected SetpointVelocity is 1mm/s then the output voltage will be 10 V.

Action parameter #3 : **Offset**

The offset value is used to correct for any voltage that may be present on the Analog output.

Description : This action is used to output a voltage on the Analog output to form an image of the theoretical velocity. The gain and the offset are used to calibrate this image. This action makes most sense with events that have some duration (always, MotionState, ElementNumberState, etc.) as the analog output will be updated each servo cycle as long as the event lasts. When used with events that have no duration (like MotionStart or MotionEnd), the analog output gets only updated once and holds this value until its next change which has typically less meaning.
The analog output value is setting with the SetpointVelocity value and this is multiplied by a gain and an offset is added :
Analog output = SetpointVelocity * gain + offset.

2.11.3.7 DACSet.SetpointAcceleration

Action name : **DACSet.SetpointAcceleration**

Action parameter #1 : **GPIOName**

The GPIOName must be an analog output name. See the list of all the GPIO Names for the Analog and Digital I/O.

Action parameter #2 : **Gain**

The SetpointAcceleration is multiplied by the gain value. For example if the gain is set to 10 and the corrected SetpointAcceleration is 1mm/s² then the output voltage will be 10 V.

Action parameter #3 : **Offset**

The offset value is used to correct for any voltage that may be present on the Analog output.

Description

: This action is used to output a voltage on the Analog output to form an image of the theoretical acceleration. The gain and the offset are used to calibrate this image. This action makes most sense with events that have some duration (always, MotionState, ElementNumberState, etc.) as the analog output will be updated each servo cycle as long as the event lasts. When used with events that have no duration (like MotionStart or MotionEnd), the analog output gets only updated once and holds this value until its next change which has typically less meaning.
The analog output value is setting with the SetpointAcceleration value and this is multiplied by a gain and an offset is added :
Analog output = SetpointAcceleration * gain + offset.

2.11.3.8 ExecuteTCLScript

Action name : **ExecuteTCLScript**

Action parameter #1 : **TCL file name**

This parameter defines the file name of the TCL program.

Action parameter #2 : **TCL task name**

Since several different TCL scripts can run simultaneously, the TCL Task Name is used to track the TCL programs so that if needed it can be addressed to separately using the task name. For example the TCL Task Name can be used to stop the particular program without stopping all TCL programs that are running.

Action parameter #3 : **TCL arguments list (separator is comma)**

The Argument list is used to run the TCL scripts with input parameters. For the argument parameter, any input can be given (number, string). These parameters are used inside the script. To get the number of arguments use "\$tcl_argc" inside the script. To get each argument use "\$tcl_argc(\$i)" inside the script. For example, this parameter can be used to input the number of loops that needs to run within the TCL script.

Description

: This action allows to execute a TCL script on an event.

2.11.3.9 GatheringRun

Action name : **GatheringRun**

Action parameter #1 : **NbPoints**

This parameter defines the number of data acquisitions. NbPoints multiplied with the number of data gathered each acquisition must be smaller than 1.000.000

Action parameter #2 : **Divisor**

This parameter defines the frequency for the gathering compared to the servo frequency of the system (10 kHz). This parameter has to be an integer and greater or equal to 1. For instance if the parameter is set to 10 then the gathering will take place at a rate of 1 kHz (10 kHz /10) or at every 1 msec.

Action parameter #3 : **0**

This parameter setting is 0 by default.

Description : This action allows to release a gathering on an event, only if a gathering is configured previously with the “GatheringConfigurationSet” API. The “Divisor” parameter must be ≥ 1 . $Nbpoints * NbType \leq MAX_POINTS_ACQUISITION$.

NOTE : *The gathering must be launched by a PUNCTUAL event and not a repetitive event*

2.11.3.10 ExternalGatheringRun

Action name : **ExternalGatheringRun**

Action parameter #1 : **NbPoints**

This parameter defines the number of data acquisitions. NbPoints multiplied with the number of data gathered each acquisition must be smaller than 1.000.000

Action parameter #2 : **Divisor**

This parameter defines every Nth number of trigger input signal at which the gathering will take place. This parameter has to be an integer and greater or equal to 1. For example if the divisor is set to 5 then gathering will take place every 5th trigger on the trigger input signal

Action parameter #3 : **0**

This parameter setting is 0 by default.

Description : This action allows to release a external gathering on an event, only if a external gathering is configured previously with the “GatheringExternalConfigurationSet” API. The “Divisor” parameter must be ≥ 1 . $Nbpoints * NbType \leq MAX_POINTS_ACQUISITION$.

NOTE : *The external gathering must be launched by a PUNCTUAL event and not a repetitive event*

2.11.4 API Description

2.11.4.1 EventAdd

TCL Prototype	int EventAdd (int SocketID, char FullPositionerName[250], char EventName[250], char EventParameter[250], char ActionName[250], char ActionParameter1[250], char ActionParameter2[250], char ActionParameter3[250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName char [250] : EventName (see § Events) char [250] : EventParameter char [250] : ActionName (see § Actions) char [250] : ActionParameter1 char [250] : ActionParameter2 char [250] : ActionParameter3
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int EventAdd (int SocketID, char FullPositionerName[250], char EventName[250], char EventParameter[250], char ActionName[250], char ActionParameter1[250], char ActionParameter2[250], char ActionParameter3[250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName char [250] : EventName (see § Events) char [250] : EventParameter char [250] : ActionName (see § Actions) char [250] : ActionParameter1 char [250] : ActionParameter2 char [250] : ActionParameter3
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the full positioner name, the event name and the action name. Verify the type of all output parameters. Parameters coherence test.
API Description	Add an event associated to an action for the selected positioner. The events and the actions are defined in § Events and § Actions.
API Errors	0 -7 -8 -9 -13 -39 -40

2.11.4.2 EventRemove

TCL Prototype	int EventRemove (int SocketID, char FullPositionerName[250], char EventName[250] , char EventParameter[250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName char [250] : EventName (see § Events) char [250] : EventParameter
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int EventRemove (int SocketID, char FullPositionerName [250], char EventName[250] , char EventParameter[250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName char [250] : EventName (see § Events) char [250] : EventParameter
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner name and the event name. Verify the event. Verify the type of all output parameters. Parameters coherence test.
API Description	Delete an event associated to an action for the selected positioner. The events and the actions are defined in § Events and § Actions.
API Errors	0 -7 -8 -9 -13 -40

2.11.4.3 EventGet

TCL Prototype	int EventGet (int SocketID, char FullPositionerName [250], char EventList[250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	char [250] : EventList
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int EventGet (int SocketID, char FullPositionerName [250], char EventList[250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName
Output parameters	char [250] : EventList
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner name and the event name. Verify the type of all output parameters. Parameters coherence test.
API Description	Get the list of events and actions in progress for the selected positioner. The events and the actions are defined in § Events and § Actions.
API Errors	0 -7 -8 -9 -13

2.11.4.4 EventWait

TCL Prototype	int EventWait (int SocketID, char FullPositionerName [250], char EventName[250] , char EventParameter[250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName char [250] : EventName (see § Events) char [250] : EventParameter
Output parameters	None
Return	TCL error (0 = success or 1 = syntax error) or API error

DLL Prototype	int EventWait (int SocketID, char FullPositionerName [250], char EventName[250] , char EventParameter[250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : FullPositionerName char [250] : EventName (see § Events) char [250] : EventParameter
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Verify the positioner name and the event name. Verify the event. Verify the type of all output parameters. Parameters coherence test.
API Description	Waits on an event for the selected positioner. The socket is locked. As soon as the event is riched, the socket is unlocked. The events are defined in § Events.
API Errors	0 -7 -8 -9 -13 -40

2.12 Positioner error list

code	Error type	Error description
0	NO ERROR	
0x80000001	ERR_GENERAL_INHIBIT	General inhibition detected
0x80000002	ERR_FATAL_FOLLOWING_ERROR	Fatal following error detected
0x80000004	ERR_HOME_SEARCH_TIMEOUT	Home search time out
0x80000008	ERR_MOTION_DONE_TIMEOUT	Motion done time out
0x80000010	ERR_OVER_POSITION	Requested position exceed travel limits in trajectory or slave mode
0x80000020	ERR_OVER_VELOCITY	Requested velocity exceed maximum value in trajectory or slave mode
0x80000040	ERR_OVER_ACCELERATION	Requested acceleration exceed maximum value in trajectory or slave mode
0x80000100	ERR_END_OF_RUN_MINUS	Minus end of course activated
0x80000200	ERR_END_OF_RUN_PLUS	Plus end of course activated
0x80000400	ERR_GLITCH_END_OF_RUN_MINUS	Minus end of run glitch
0x80000800	ERR_GLITCH_END_OF_RUN_PLUS	Plus end of run glitch
0x80001000	ERR_ENCODER_QUAD	Encoder quadrature error
0x80002000	ERR_ENCODER_FOC	Encoder frequency and coherancy error
0x80010000	ERR_ENCODER_HARD_INTERPOLATOR	Hard interpolator encoder error
0x80020000	ERR_ENCODER_QUAD_HARD_INTERPOLATOR	Hard interpolator encoder quadrature error
0x80100000	ERR_DRIVER1_FAULT	First driver in fault
0x80200000	ERR_DRIVER2_FAULT	Second driver in fault

Notice: The most significant bit is always set to 1 to have a negative value in decimal form

2.13 Positioner hardware status list

code	Error type	Error description
0x00000001	HARD_STATUS_GENERAL_INHIBIT	General inhibition detected
0x00000004	HARD_STATUS_ENCODER_ZM	ZM high level
0x00000100	HARD_STATUS_END_OF_RUN_MINUS	Minus end of run activated
0x00000200	HARD_STATUS_END_OF_RUN_PLUS	Plus end of run activated
0x00000400	HARD_STATUS_GLITCH_END_OF_RUN_MINUS_ERROR	Minus end of run glitch
0x00000800	HARD_STATUS_GLITCH_END_OF_RUN_PLUS_ERROR	Plus end of run glitch
0x00001000	HARD_STATUS_ENCODER_QUAD_ERROR	Encoder quadrature error
0x00002000	HARD_STATUS_ENCODER_FOC_ERROR	Encoder frequency or coherancy error
0x00010000	HARD_STATUS_ENCODER_HARD_INTERPOLATOR_ERROR	Hard interpolator encoder error
0x00020000	HARD_STATUS_ENCODER_QUAD_HARD_INTERPOLATOR_ERROR	Hard interpolator encoder quadrature error
0x00100000	HARD_STATUS_DRIVER1_FAULT	First driver in fault
0x00200000	HARD_STATUS_DRIVER2_FAULT	Second driver in fault
0x00400000	HARD_STATUS_DRIVER1_POWER_ON	First driver powered on
0x00800000	HARD_STATUS_DRIVER2_POWER_ON	Second driver powered on

2.14 Group state list

0	NOTINIT	Not initialized state
1	NOTINIT_FROM_EMERGENCY_BRAKE	Not initialized state due to an emergency brake : see positioner status
2	NOTINIT_FROM_EMERGENCY_STOP	Not initialized state due to an emergency stop : see positioner status
3	NOTINIT_FROM_FOLLOWING_ERROR_HOMING	Not initialized state due to a following error during homing
4	NOTINIT_FROM_FOLLOWING_ERROR_NOTREF	Not initialized state due to a following error
5	NOTINIT_FROM_HOME_TIMEOUT	Not initialized state due to an homing timeout
6	NOTINIT_FROM_MOTION_DONE_TIMEOUT_HOMING	Not initialized state due to a motion done timeout during homing
7	NOTINIT_FROM_KILLALL	Not initialized state due to a KillAll command
8	NOTINIT_FROM_END_OF_RUN_HOMING	Not initialized state due to an end of run after homing
9	NOTINIT_FROM_ENCODER_CALIBRATION_ERROR	Not initialized state due to an encoder calibration error
10	READY_FROM_ABORT_MOVE	Ready state due to an AbortMove command
11	READY_FROM_HOMING_DONE	Ready state from homing
12	READY_FROM_MOVE_DONE	Ready state from motion
13	READY_FROM_SETMOTION_ENABLE	Ready State due to a MotionEnable command
14	READY_FROM_SLAVE	Ready state from slave
15	READY_FROM_JOGGING	Ready state from jogging
16	READY_FROM_ANALOG_TRACKING	Ready state from analog tracking
17	READY_FROM_TRAJECTORY_DONE	Ready state from trajectory
20	DISABLE	Disable state
21	DISABLE_FROM_FOLLOWING_ERROR_READY	Disabled state due to a following error on ready state
22	DISABLE_FROM_FOLLOWING_ERROR_MOVING	Disabled state due to a following error during motion
23	DISABLE_FROM_MOTION_DONE_TIMEOUT_MOVING	Disabled state due to a motion done timeout during moving
24	DISABLE_FROM_MOTION_DONE_TIMEOUT_SLAVE	Disable state due to a motion done timeout on slave state
25	DISABLE_FROM_FOLLOWING_ERROR_JOGGING	Disabled state due to a following error on jogging state
26	DISABLE_FROM_FOLLOWING_ERROR_TRAJECTORY	Disabled state due to a following error during trajectory
27	DISABLE_FROM_MOTION_DONE_TIMEOUT_TRAJECTORY	Disabled state due to a motion done timeout during trajectory
28	DISABLE_FROM_FOLLOWING_ERROR_ANALOG_TRACKING	Disabled state due to a following error during analog tracking
29	DISABLE_FROM_SLAVE_ERROR_MOVING	Disabled state due to a slave error during motion
30	DISABLE_FROM_SLAVE_ERROR_SLAVE	Disabled state due to a slave error on slave state
31	DISABLE_FROM_SLAVE_ERROR_JOGGING	Disabled state due to a slave error on jogging state
32	DISABLE_FROM_SLAVE_ERROR_TRAJECTORY	Disabled state due to a slave error during trajectory
33	DISABLE_FROM_SLAVE_ERROR_ANALOG_TRACKING	Disabled state due to a slave error during analog tracking
34	DISABLE_FROM_SLAVE_ERROR_READY	Disabled state due to a slave error on ready state
40	EMERGENCY BRAKING	Emergency braking
41	MOTOR_INIT	Motor initialization state
42	NOTREF	Not referenced state
43	HOMING	Homing state
44	MOVING	Moving state
45	TRAJECTORY	Trajectory state
46	SLAVE	Slave state due to a SlaveEnable command
47	JOGGING	Jogging state due to a JogEnable command
48	ANALOG_TRACKING	Analog tracking state due to a TrackingEnable command
49	ENCODER_CALIBRATING	Analog interpolated encoder calibrating state

2.15 Error list

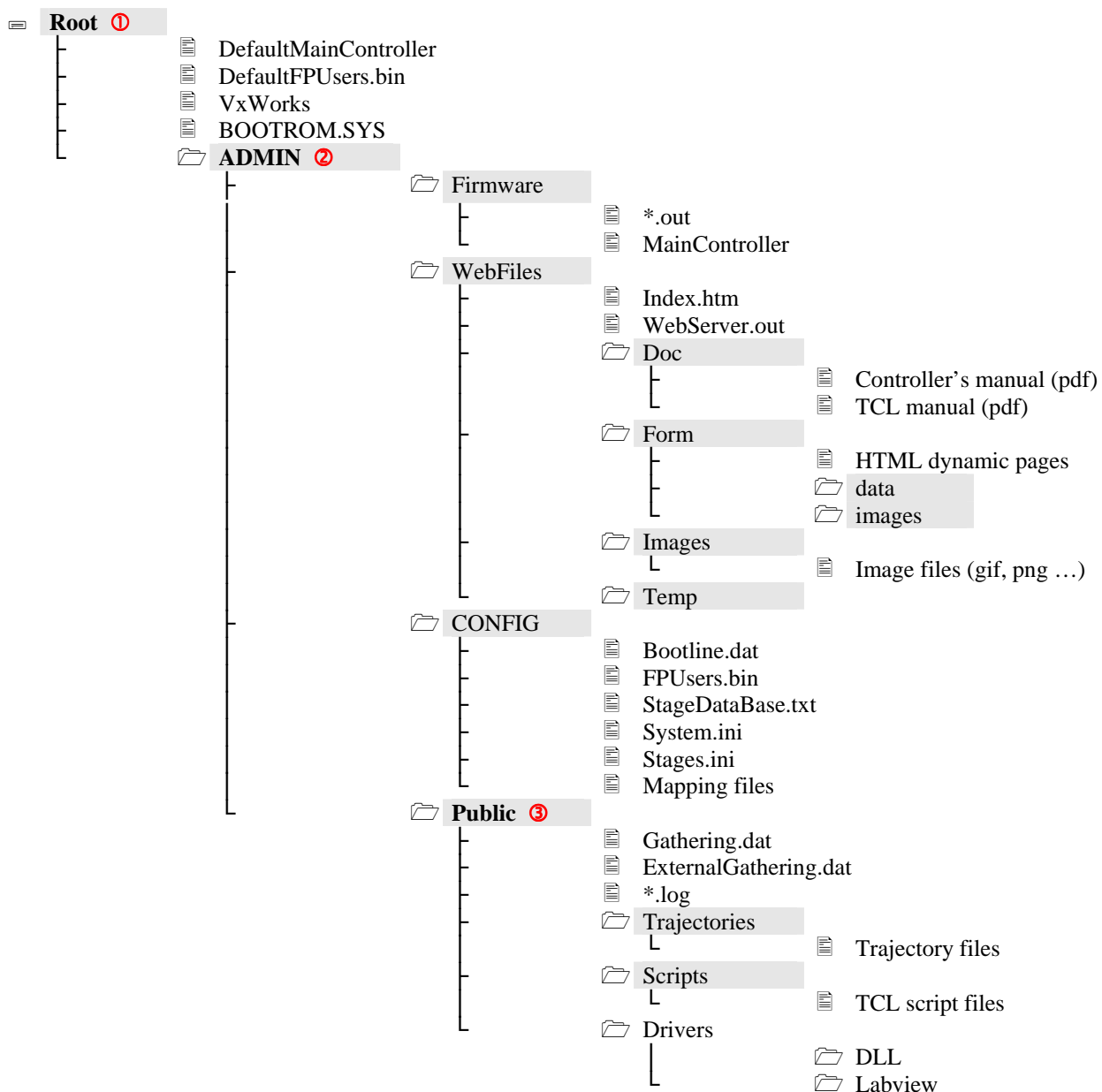
code	Error type	Error description
0	SUCCESS	Successful command
-1	ERR_BUSY_SOCKET	Busy socket : previous command not yet finished
-2	ERR_TCP_TIMEOUT	TCP timeout
-3	ERR_STRING_TOO_LONG	String command too long
-4	ERR_UNKNOWN_COMMAND	Unknown command
-5	ERR_HARDWARE_STATUS	Hardware status error
-7	ERR_WRONG_FORMAT	Wrong format in the command string
-8	ERR_WRONG_OBJECT_TYPE	Wrong object type for this command
-9	ERR_WRONG_PARAMETERS_NUMBER	Wrong number of parameters in the command
-10	ERR_WRONG_TYPE	Wrong parameter type in the command string
-11	ERR_WRONG_TYPE_BIT_WORD	Wrong parameters type in the command string : word or word * expected
-12	ERR_WRONG_TYPE_BOOL	Wrong parameter type in the command string : bool or bool * expected
-13	ERR_WRONG_TYPE_CHAR	Wrong parameter type in the command string : char * expected
-14	ERR_WRONG_TYPE_DOUBLE	Wrong parameter type in the command string : double or double * expected
-15	ERR_WRONG_TYPE_INT	Wrong parameter type in the command string : int, short, int * or short * expected
-16	ERR_WRONG_TYPE_UNSIGNEDINT	Wrong parameter type in the command string : unsigned int, unsigned short, unsigned int * or unsigned short * expected
-17	ERR_PARAMETER_OUT_OF_RANGE	Parameter out of range
-18	ERR_POSITIONER_NAME	Positioner Name doesn't exist
-19	ERR_GROUP_NAME	GroupName doesn't exist or unknown command
-20	ERR_FATAL_INIT	Fatal Error during initialization, read the error.log file for more details
-21	ERR_IN_INITIALIZATION	Controller in initialization
-22	ERR_NOT_ALLOWED_ACTION	Not allowed action
-23	ERR_POSITION_COMPARE_NOT_SET	Position compare not set
-24	ERR_POSITION_COMPARE_NO_ENCODER	Position compare not available without a position encoder
-25	ERR_FOLLOWING_ERROR	Following Error
-26	ERR_EMERGENCY_SIGNAL	Emergency signal
-27	ERR_GROUP_ABORT_MOTION	Move Aborted
-28	ERR_GROUP_HOME_SEARCH_TIMEOUT	Home search timeout
-29	ERR_MNEMONOTYPEGATHERING	Mnemonic gathering type doesn't exist
-30	ERR_GATHERING_NOT_STARTED	Gathering not started
-31	ERR_HOME_OUT_RANGE	Home position is out of user travel limits
-32	ERR_GATHERING_NOT_CONFIGURED	Gathering not configured
-33	ERR_GROUP_MOTION_DONE_TIMEOUT	Motion done timeout
-36	ERR_UNKNOWN_TCL_FILE	Unknown TCL file
-37	ERR_TCL_INTERPRETOR	TCL interpreter doesn't run
-38	ERR_TCL_SCRIPT_KILL	TCL script can't be killed
-39	ERR_MNEMONO_ACTION	Mnemonic action doesn't exist
-40	ERR_MNEMONO_EVENT	Mnemonic event doesn't exist
-41	ERR_SLAVE_CONFIGURATION	Slave-Master mode not configured
-42	ERR_JOG_OUT_OF_RANGE	Jog value out of range
-43	ERR_GATHERING_RUNNING	Gathering running
-44	ERR_SLAVE	Slave error disabling master

-45	ERR_END_OF_RUN	End of run activated
-46	ERR_NOT_ALLOWED_BACKLASH	Not allowed action due to backlash
-47	ERR_WRONG_TCL_TASKNAME	Wrong TCL task name : each TCL task name must be different
-48	ERR_BASE_VELOCITY	BaseVelocity must be null
-60	ERR_WRITE_FILE	Error during file writing or file doesn't exist
-61	ERR_READ_FILE	Error during file reading or file doesn't exist
-62	ERR_TRAJ_ELEM_TYPE	Wrong trajectory element type
-63	ERR_TRAJ_ELEM_RADIUS	Wrong XY trajectory element arc radius
-64	ERR_TRAJ_ELEM_SWEEP	Wrong XY trajectory element sweep angle
-65	ERR_TRAJ_ELEM_LINE	Trajectory line element discontinuity error or new element is too small
-66	ERR_TRAJ_EMPTY	Trajectory doesn't content any element or not loaded
-68	ERR_TRAJ_VEL_LIMIT	Velocity on trajectory is too big
-69	ERR_TRAJ_ACC_LIMIT	Acceleration on trajectory is too big
-70	ERR_TRAJ_FINAL_VELOCITY	Final velocity on trajectory is not zero
-71	ERR_READ_MSG_QUEUE	Error read message queue
-72	ERR_WRITE_MSG_QUEUE	Error write message queue
-73	ERR_END_OF_FILE	End of file
-74	ERR_READ_FILE_PARAMETER_KEY	Error file parameter key not found
-99	ERR_FATAL_EXTERNAL_MODULE_LOAD	Fatal external module load : see error.log

3. Using the Controller

3.1 Directory access rights and privileges

To access to controller data with an FTP client (Microsoft Internet Explorer® or an other tool), you enter the controller ethernet address.



Four privilege types are defined as following:

- ① Super administrator
- ② Administrator
- ③ User
- ④ Guest

Directory access rights in relation to privileges:

<i>Privilege</i>		<i>Directory access</i>
Super administrator	:	Root directory : all directories
Administrator	:	Administrator directory and Administrator sub-directories
User	:	Public directory and Public sub-directories
Guest	:	No directory

3.1.1 Root directory

The main directory contains the vxWorks kernel and the bootrom.

DefaultMainController : Default version of firmware (light version)
 FPUsers.bin : Default logins, passwords and privileges (not editable)
 BOOTROM.SYS : Bootrom to boot the vxWorks system
 VxWorks : Kernel

3.1.2 ADMIN directory

The “ADMIN” directory contains four directories.

Firmware : Controller firmware and modules
 Webfiles : WEB site
 CONFIG : CONFIGuration file and mapping
 Public : Gathering, trajectories and TCL scripts + Documents and tools

3.1.3 Firmware directory

The “Firmware” directory from “ADMIN” directory contains the OUT modules and the “MainController” firmware.

MainController : Firmware
 TCL_API_drivers.out : TCL module
 WatchDog_?.out : Watch Dog module

3.1.4 Webfiles directory

The “Webfiles” directory from “ADMIN” directory contains the files used by the WEB site.

3.1.5 CONFIG directory

The “CONFIG” directory from “ADMIN” directory contains the configuration file and the mapping files.

bootline.dat : Bootline contains the vxWorks boot data : Kernel path, Ethernet address ...

FPUsers.bin : Logins, passwords and privileges (not editable)

StageDataBase.txt : Stages database

System.ini : System configuration file

Stages.ini : Stages configuration file

3.1.6 Public directory

The “Public” directory contains the user configuration, the user data and the error files.

Gathering.dat	: Data file created from the last gathering execution
ExternalGathering.dat	: Data file created from the last external gathering execution
Error.log	: Error file
LastError.log	: Last Error.log file saving
Log.log	: Log error file

3.1.7 Scripts directory

The “Scripts” directory from “Public” directory contains all user TCL scripts (*.tcl).

3.1.8 Trajectories directory

The “Trajectory” directory from “Public” directory contains all trajectory files (*.txt).

3.1.9 Drivers

The “Drivers” directory from “Public” directory contains drivers for Visual C++ / Visual Basic and VI for Labview.

3.2 Configuration

Two configuration files are used during the controller initialization : “System.ini” and “Stages.ini”

The system and groups configuration parameters from System.ini file.
The positioner configuration parameters from Stages.ini file (units = mm).

3.2.1 System.ini file

```
[GENERAL]
FirmwareName = MainController
ExternalModuleNames = TCL_API_drivers.out
CorrectorISRPeriod = 100e-6 ; seconds
IRQDelay = 10e-6 ; seconds
ProfileGeneratorISRRatio = 4
ServitudesISRRatio = 10
BootScriptFileName =
BootScriptArguments = ; the separator is the comma

[GROUPS]
SingleAxisInUse =
XYInUse =
XYZInUse =
MultipleAxesInUse =

[GROUPNAME]
PositionerInUse = ; the separator is the comma

[GROUPNAME.POSITIONERNAME]
PlugNumber =
StageName = MYSTAGE ; see “ stages.ini” file
```

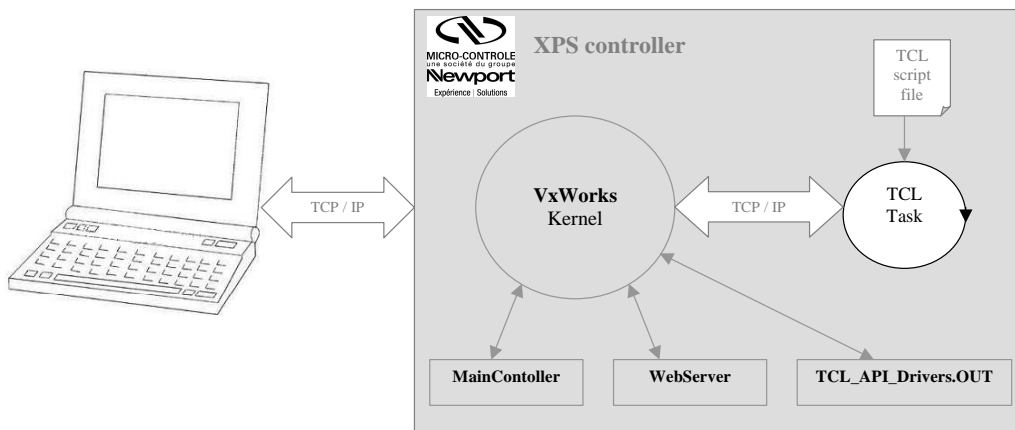
3.2.2 Stages.ini file

```
[MYSTAGE]
; see § Positioner configuration file
```

3.3 TCL programming

3.3.1 TCL using

All controller APIs are available in “TCL_API_drivers.out” module and can be used by a TCL script from “Scripts” directory.



3.3.2 TCL arguments

tcl_argc is an integer that contains the count of arguments that follow in **tcl_argv**. The **tcl_argc** parameter is always greater than or equal to 1.

tcl_argv[NB_ARG] is an array of null-terminated strings representing command-line arguments entered by the user of the TCL script. **tcl_argv(0)** is the first command-line argument, and so on, until **tcl_argv(tcl_argc)**, which is always NULL.

3.3.3 TCL errors

Error code	Description
0	No TCL syntax error
1	TCL syntax error

3.3.4 Boot TCL script

The boot TCL script is a program, defined in the “System.ini” configuration file in the GENERAL section, that will be automatically run after controller start-up.

[GENERAL]

BootScriptFileName = testarg.tcl

BootScriptArguments = arg1, arg2, arg3, arg4, arg5

The “BootScriptFileName” is the file name that contains the TCL script. This file must be stored in the “Public\scripts” controller directory.

The “BootScriptArguments” is the argument line. The argument separator is the comma.

3.3.5 TCL APIs driver

3.3.5.1 TCLScriptExecute

TCL Prototype	int TCLScriptExecute (int SocketID, char TCLFileName[50], char TaskName[250] , char Arguments[250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [50] : TCLFileName char [250] : TaskName char [250] : Arguments string (separator is comma)
Output parameters	None
Return	TCL error or API error

DLL Prototype	int TCLScriptExecute (int SocketID, char TCLFileName[50], char TaskName[250] , char Arguments[250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [50] : TCLFileName char [250] : TaskName char [250] : Arguments string (separator is comma)
Output parameters	None
Return	API error

Input tests	Verify the number of parameters. Parameters coherence test.
Description	Executes a TCL program under the controller TCL interpreter. The TCL script file must be saved in the “\Public\Scripts” controller directory. The execution result is displayed in the controller screen.
Errors	0 -7 -9 -13 -36 -37

3.3.5.2 TCLScriptKill

TCL Prototype	int TCLScriptKill (int SocketID, char TaskName[250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : TaskName
Output parameters	None
Return	TCL error or API error

DLL Prototype	int TCLScriptKill (int SocketID, char TaskName[250])
Input parameters	int : SocketID (Socket identifiant gets by the “TCP_ConnectToServer” API) char [250] : TaskName
Output parameters	None
Return	API error

API Input tests	Verify the number of parameters. Parameters coherence test.
API Description	Kill a TCL script that running.
API Errors	0 -7 -9 -13 -36 -37

3.4 Version

3.4.1.1 **GetLibraryVersion**

TCL Prototype	int GetLibraryVersion (char * LibVersion)
Input parameters	None
Output parameters	char * : LibVersion
Return	TCL error (0 or 1)
Description	Gets the version of TCL library from “TCL_API_drivers.out” module.

DLL Prototype	char * GetLibraryVersion (void)
Input parameters	None
Output parameters	None
Return	Library version
Description	Gets the version of DLL library.

**Newport Corporation
Worldwide Headquarters**

1791 Deere Avenue
Irvine, CA 92606

(In U.S.): 800-222-6440

Tel: 949-863-3144

Fax: 949-253-1680

Email: sales@newport.com
tech@newport.com



Newport

Visit Newport Online at: www.newport.com



Newport Corporation, Irvine, California, has been certified compliant with ISO 9001 by the British Standards Institution.